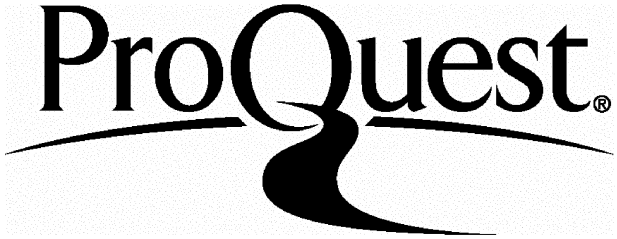
Investigation of the maintenance of integrity in telecommunication networks using formal and heuristic methodologies

by

Maria Victoria Montón Ortega

Thesis submitted for the degree of Doctor of Philosophy of the University of London

University College London



Abstract

Modern telecommunication networks are experiencing a dramatic evolution and increase in complexity. Major technological advances have transformed networks into software controlled systems. An increasing degree of intelligence is being introduced in telecommunication networks, and new paradigms are being developed to satisfy customer demand for rapid deployment of increasingly sophisticated services. Such a fusion of telecommunications and computing as a distributed processing environment gives rise to a dramatic increase in control complexity and fragility. Services have to co-exist and co-operate across platforms and may give rise to undesirable feature interactions. Unexpected perturbations may also ripple through the network to give widespread outages. These problems are exacerbated by regulatory policies demanding open interconnection of networks of competing operators and service providers together with internetworking of services. Threat to network integrity is high in such an environment and its preservation is of paramount importance. But the traditional manual approach to the maintenance of network integrity which relies on the knowledge of individual experts is no longer valid. There is a need to develop formal and systematic methodologies to preserve network integrity; frameworks to pre-empt, detect and quickly resolve failures in telecommunication networks and the services they support. This thesis addresses these issues. A definition of network integrity is proposed and the main problems that jeopardise network integrity in modern telecommunication systems are discussed. An integrity framework to categorise integrity levels and assist in the pre-emption of severe integrity degradation is presented. Methodologies to aid with the provision and maintenance of systems with high integrity levels are introduced. The work focuses on the need to use and improve modelling techniques. The use of formal description languages as a basis for the modelling activity is investigated, in particular the ITU standard SDL (Specification and Description Language), and findings from modelling a case study are presented.

Table of contents

List of figures

List of tables

Acknowledgements

I would like to thank Prof. Keith Ward for his supervision and useful comments.

Thanks to Kevin Woollard, Mel Bale and Gordon Henderson for their support and encouragement during the last and hardest stage.

Also a big thank you to Mark, Céline and Fiona for putting up with me during the last few months.

And apologies to all the friends I haven't called, seen, written or spoken to during the same last months.

A Mark, Zacarías, Victoria, Zaca y Andrés

Chapter 1

Introduction

1.1. Motivation

Once upon a time, there was a telephone network. This network provided basic voice communication to a large number of people, and it had been around for many years, and it was very well known and reliable. Each country had one state owned telephone company managing a telephone network. Then digital technology arrived, and with it, a number of great advances, like fast and powerful computers and later, computer networks. Other revolutionary technological innovations also took place, and the telephone networks benefited from them. These advances were gradually introduced in the telephone networks, providing them with more and more flexibility, processing power and functionality. In addition, most of the formally state owned telephone companies were privatised, and policies fostering competition and open network interconnection emerged. This takes us to today, when that old and well-known telephone network has undergone so many changes that it is hardly recognisable.

The modern networks are designed to provide a whole range of new services to increasingly more demanding customers. They are dynamic, subject to continuous evolution and upgrading. The former telephone companies are now called Telecommunication Organisations, since voice telephony is only one of the many services on offer. Emphasis is

put on the services that these modern networks can provide, and new paradigms are being developed for the fast and efficient provision of these services. The result is extremely complex software controlled networks, which are in fact huge distributed systems.

Maintaining the integrity of such complex systems is a very difficult issue. Changes are happening at an accelerating speed, and hence expertise is limited and telecommunication organisations still rely, to a great extent, in past working practises and methodologies, which are not necessarily suited for the modern networks. Legacy systems, relying heavily on human expertise and manual intervention are still largely in place. But the immense complexity of modern networks and services is well beyond the comprehension of single individuals. Expertise is fragmented, and automated procedures and techniques become increasingly necessary in order to manage the vast amounts of data and knowledge needed to understand the operations of modern telecommunication systems.

The vulnerability of modern telecommunication networks is causing increasing concern in telecommunication organisations, standards bodies and other organisations. A major milestone occurred in the early nineties, when public attention in the USA was brought to the vulnerability of the new network technologies by the severity of the signalling failures that afflicted several carriers, and the difficulty to re-establish service. This originated a number of initiatives from government and non-government organisations to investigate methods to measure and maintain network integrity. Public awareness also developed in Europe, and in 1994, the Commission of the European Union sponsored a study to investigate network integrity under the Open Network Provision environments (ONP)[1]. In the study, carried out by University College London, attention was brought to a number of areas deserving consideration, and to the need for the development of pre-emptive measures to minimise the threat of integrity degradation.

Currently, the usual method employed to minimise the threat to network integrity is to carry out exhaustive, expensive and time consuming testing prior to the introduction of new technology, software and/or services. Even this cannot possibly test the myriad permutations and combinations of signal messages and associated processing that take place across networks with a large base of customers and services. The position is exacerbated when networks are interconnected.

This is the context where this work was born, motivated by the recognised need to:

a) Develop a formal and quantitative definition of network integrity that can be commonly understood and used to identify the degree of integrity within which a network operates.

b) Investigate ways to improve the design and maintenance of modern telecommunication networks and services, minimising the risk of large scale integrity degradation.

The objective of this project is to assist this process by developing an initial screening methodology that can be used in the design stages.

The subject is important because:

a) Customers and national economies are ever more reliant on good, reliable communications; the financial consequences of network outages both to customers and network/service providers (rebates, compensation and litigation) can be very large.

b) Increasing control complexity of networks, particularly with introduction of intelligent network architecture, driven by market needs for ever more sophisticated services, increases risk to integrity.

c) The competitive environment results in new scenarios, e.g. EU regulators forcing incumbent Telecommunication Operators to allow ever more complex interconnect to the networks of competitors and service provider equipment thereby increasing the risk to integrity.

1.2. Project description

The motivation for this project was to investigate methodologies for the preservation of network integrity in telecommunication networks. The reader will appreciate the extremely large size and complexity of the problem space we were facing. Firstly, the meaning of network integrity must be clarified, as many different interpretations exist depending on the context in which the term is employed.

---

[1] The concept of ONP has been developed by the European Commission for the implementation of competition, to facilitate access to and use of telecommunication networks and services – nationally and throughout Europe

The focus here is on developing a pre-emptive approach to maintain network integrity. An initial definition of network integrity was "the ability of a network to retain its specified attributes in terms of performance and functionality" [Ward 95]. This definition has been taken as the basic starting point for this work. A principal aspect of the definition is that integrity refers to the ability to maintain certain properties. Network integrity is then the ability of a network to maintain a safe state in which it is invulnerable to external perturbations. If the network is in a state of high integrity, these perturbations can be easily 'absorbed' by the system without having undue effect. Alternatively, if the network is in a state of low integrity, the same perturbations might have more serious consequences, and may cause failure. Hence, we regard network integrity as related to robustness, invulnerability and incorruptibility. The proposed definition of network integrity will be discussed in detail in chapter 3.

It must be clarified that the term 'network' does not refer only to a collection of physical components connected together, but also to the functionality they provide. Hence, network integrity refers to both the integrity of the physical network and, more importantly, the integrity of the services that the network provides. It is this second aspect that this work is concerned with, i.e. network integrity is viewed here from a services perspective.

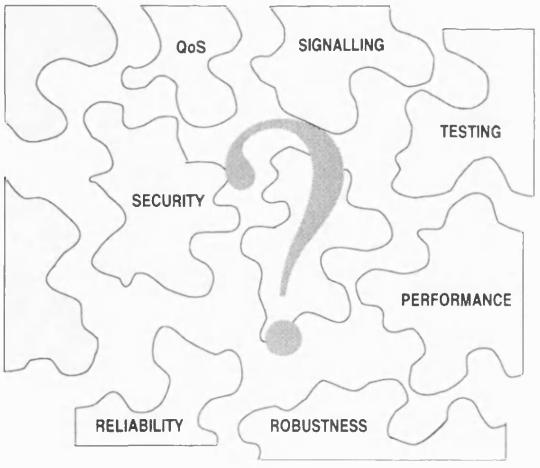


Figure 1. The integrity jigsaw puzzle

Network integrity is a relatively new topic and there is limited literature on the subject. However, it has parallels with many different areas, e.g. signalling, testing, network management, security, quality of service, etc., where there is expertise. But, as shown in Figure 1, this expertise is isolated and does not integrate easily into a coherent approach. Effort must be made to integrate these isolated branches of expertise under the umbrella requirements of integrity. Such an expertise base will be well beyond the comprehension of one individual. Therefore, it is of paramount importance that it is incorporated into a meaningful and manageable structure, easily accessible by a variety of specialists. This was the theme of the first part of the project. The initial stages consisted of research and investigation work, to identify areas of relevance to the problem, and the connections between them. From those findings, a general 'integrity framework' was developed containing key actions that should be undertaken to improve the integrity of modern networks and services. At this point, it became necessary to reduce the problem domain and take a narrower and more specific route. This led to the second major phase of the work, which consisted of the application of modelling techniques for the integrity analysis.

1.3. Methodology

The methodology employed in this work comprises a number of different activities. These include:

a) Literature research. This was the main activity during the first stages of the project, in order to obtain the necessary background information for the work. This provided the general picture from which specific routes of research were followed. In these later stages, further literature research was also undertaken in those specific areas.

b) In order to obtain a realistic view of the state of the art and an awareness of the real issues in commercial organisations, a study was carried out in one major telecommunications organisation in the UK. This study included field trips to operational sites and interviews with key experts in a variety of areas. The study also included an analysis of findings and a proposal of recommendations to the telecommunications organisation to overcome the problems identified.

c) Modelling and simulation work constitute the practical side of the project, the objective being to understand the operation of advanced services, identify where complexity resides and potential causes of failures, and how these aspects effect the degree of integrity.

1.4. Structure of thesis

The main body of this thesis is structured into eight chapters. The first two chapters contain the introduction to the work and the background material, including a review of exiting approaches to the integrity problem. The remaining chapters constitute the author's contribution to the problem.

Chapter 2 sets the scene and provides the background information for the work. A general overview of the main characteristics of modern telecommunication networks and markets is given, focusing on those areas that have a direct effect in the complexity and integrity of the networks. A brief overview of Intelligent Networks and Signalling System Number 7 is included in appendices A and B respectively. A review of concepts related to integrity and existing approaches is also included in this chapter.

Chapter 3 describes the approaches proposed by the author for the design and support of high integrity networks and services. The contribution contains three main areas, namely a pre-emptive network integrity definition, and overall integrity framework and a modelling framework. The chapter finishes with a discussion of alternatives for further work and the arguments for the selected alternative.

The remainder of the work focuses on the use of modelling and simulation techniques for integrity preservation, and the selected approach was to focus on modelling a case study. The description of this case study is included in chapter 4. Several models of the case study were built. Chapter 5 contains a description of a C++ model, the relevant findings and results. An analytical model was developed for validation, and this is included in appendix C.

Following the limitations identified in the use of C++, an investigation of alternative languages and tools was carried out. As a result, the work focused on the study of SDL[2] and its suitability for integrity modelling. An SDL model of the case study was built. Chapter 6 includes the description of this model, and an analysis of SDL capabilities, limitations encountered for this work and proposed solutions.

Chapter 7 returns the focus to the integrity frameworks and methodologies proposed in earlier chapters, by discussing the application of the simulation work and results with respect to the

[2] SDL is the CCITT Specification and Description Language

construction and testing of such frameworks. A review of the proposed modelling methodologies is also undertaken in the light of the SDL modelling experience.

The thesis ends with the concluding remarks in chapter 8.

Chapter 2

The integrity problem in telecommunication systems

2.1. Introduction

This chapter consists of two parts. Firstly, section 2.2 defines the problem of integrity in telecommunication systems and examines what are the characteristics of these systems that make integrity such a big issue. Secondly, in section 2.3 existing approaches related to the integrity problem are reviewed and evaluated in relation to the approaches pursued by this work. The chapter ends with the conclusions in section 2.4.

2.2. Defining the problem

Telecommunication services and networks are experiencing dramatic changes. New technologies are being used with the objective of providing frameworks that allow faster and easier development of services. Telecommunication networks are now distributed processing systems with many different makes of processor, e.g. exchanges, and difficult to re-boot if they fall over. There is a climate of change and evolution, where the historical provision of POTS is becoming something of the past. Operators are adopting new technologies and

architectures to respond to the demand for new and more advanced services. The regulatory frameworks are also changing. New regulatory policies, fostering competition, demand the open interconnection of networks of competing operators and service providers together with internetworking of services. The result is a dramatic increase in complexity in the telecommunication environments. This extreme complexity increases the risk of failures or errors that may jeopardise the integrity of the networks. Not only the risk of outages is higher, but also their effects become more damning.

In such a new and increasingly complex environment traditional manual approaches to detect and fix faults become unfeasible and the need to adopt systematic and formalised methods to maintain the integrity of the networks and the services supported arises. This requires understanding of the current and forthcoming telecommunication networks and identification of high complexity areas and potential sources of problems. This section contains an analysis of the most representative characteristics, changes and trends in telecommunications and their impact in network integrity. Following the ideas introduced in [McDonald 94] the evolution in telecommunications is determined by changes in technology, new government regulations and increased customers reliance on telecommunications services. The following sections discuss the changes on each of these three areas and their effect in the integrity of the networks.

2.2.1. Customer demand

Customer demands are driven by the need to:

a) tailor services to better suit an individual or particular business purpose;
b) find the appropriate service providers who can provide the optimum commercial arrangements, e.g. price;
c) obtain reliable and safe telecommunication capabilities;
d) achieve particular qualities of service which may not always be 'the best' but rather would be appropriate to a particular business purpose.

These increasingly sophisticated customer demands impose the need for fast development and rapid introduction of a large number of complex services, both global and domestic. This has a great effect in network integrity for several reasons, such as:

a) global services require inter working between different networks, each of them with their own infrastructure and architecture, this being the source of potential threats to network integrity; it can therefore be concluded that there is an increasing risk of network integrity

being compromised by problems arising from interconnected networks and equipment connected to the network; this is dealt with in section 2.2.3.1.

b) digitalisation of communications permits to represent all information and entertainment as a stream of bits, and this has been the main driver in the multimedia revolution [Willis and Dufour 95]; the different media (voice, images and data) have different characteristics and requirements in terms of quality of service, increasing the management complexity, with added complications due to correlation amongst the different types of traffic streams;

c) mobility, of both people and information, introduces a higher degree of complexity in the areas of signalling, addressing and management of data;

d) a large number of services, each one consisting of a particular set of features, must coexist in the same platforms; this raises the feature interaction problem, dealt with in section 2.2.2.4, when the behaviour of a particular feature interferes with the behaviour of others.

A major driver for the development of the telecommunications industry is the influence of large and multinational corporations, increasingly looking for telecommunications solutions to improve efficiency and responsiveness, often on a global basis [Leakey 94]. This group tends to provide the major driving force for the introduction of new services. At the same time, the demand for more sophisticated communication facilities such as tele-working, tele-shopping and personalised service offers is increasing in the domestic market sector, this being the driver for most of the innovative services [Valdar et al. 92].

2.2.2. Technology

The main features of a telecommunication network can be categorised into three groups, namely transmission, switching and signalling and control. All three of these areas have evolved significantly, particularly during the last two decades. A major milestone was the introduction of digital systems, which permit the evolution from analogue transmission and switching techniques towards the digital software controlled networks of nowadays. Digital transmission in telephony was first invented in the 1930s, when Pulse Code Modulation techniques were patented. However, this technique together with time division multiplexing techniques were not technically viable until the 1960s. Since then, digital transmission gradually became more cost effective and it has been progressively adopted.

In the same way that digital techniques were adopted in the transmission field, digital exchanges were also introduced. The first digital toll switch, the No. 4 EES, was introduced in

the United States in 1976 and it has a capacity of 47000 Erlang, almost 8 times the capacity of its analogue predecessor, the 4A Crossbar [McDonald 94]. In the UK, the first widely used digital exchanges, System X, were put into the main BT network in 1980. In a similar way, digital technology has also been introduced in local exchanges. Analogue switching systems have progressively been substituted by digital switching during the last two decades, so that nowadays very few analogue exchanges remain. Digital technology has transformed old analogue telecommunication networks into software controlled systems. Stored program control (SPC) now exists in virtually every network element, from digital exchanges to user terminals [Redmill and Valdar 94]. This has had two major effects. On the one hand, the dramatic increase of capacity; on the other, increased network flexibility and functionality for new services.

Finally, in the signalling arena a major milestone has been the evolution from channel associated signalling (CAS) systems to Common-channel signalling (CCS). Details of these signalling methods will not be discussed here. Only a brief overview is presented in section 2.2.2.3, but the reader may refer to [Manterfield 91] for more details. The main difference is that, whereas in CAS systems the transfer of signalling information is conducted over signalling capability that is specific to a particular circuit, in CCS systems the physical tie between the signalling path and the traffic circuit is removed. One major advantage of CCS systems is that the signalling information is not restricted to the control of traffic circuits. CCS systems provide higher speed and increased flexibility, and they become most beneficial when adopted in parallel with digital exchanges and digital transmission techniques.

All the above mentioned advances in transmission, switching and signalling and control have converged to result into networks with much higher capacity, higher speed and, even more importantly, high functionality which permits to use the telecommunication networks for the provision of a large variety of services beyond the traditional voice telephony. This requires new technologies and much more sophisticated signalling systems and protocols. Telecommunication networks and the services they support have become dramatically more complex, and this high complexity and lack of experience with these new approaches is one of the reasons that makes the preservation of network integrity so difficult.

The following sections expand on some of the most relevant technological advances that characterise modern telecommunication networks, with emphasis on their connection to network integrity. In particular, focus is on the signalling aspects, because the role of signalling and associated protocols has become increasingly important in the provision of

network capabilities. It is also an area of fast growth in complexity, and hence likely to result in integrity issues. There is a section on Intelligent Networks (IN), because it is a very representative example of where things are heading. IN capabilities have started to be introduced in some networks, but it is fair to say that they are still mostly a conceptual architecture, and there is a general lack of practical experience and understanding of the implications of IN architectures on the integrity of the networks; particularly in the interconnect scenario. IN is not the only option, and it indeed has its detractors. After the introduction of IN, other approaches are being developed. This is the case of the TINA (Telecommunications Information Networking Architecture) initiative, which attempts to propose an integral platform to support the introduction of new services and the service and network management functions. The TINA approach represents a shift from protocol-based telecommunication engineering principles to software engineering techniques, based on objet orientation and distributed systems.

Regardless of the specific architecture adopted, telecommunication networks are becoming huge distributed interactive systems, and hence they are subject to problems characteristic of distributed computing environments, such as concurrency and real-time issues. These types of problems and their effect on the integrity of the networks need addressing. We have considered IN because, at the time of commencement of this work, it was one of the paradigms that generated most interest and activity within the telecommunications industry and research communities. With technologies based on distributed computing the concept of signalling as it is understood now disappears. In such environments, there is not a distinct separation between signalling and data because every type of information is transferred in the same way. Traditional signalling protocols are replaced by more flexible Distributed Processing Environment (DPE) mechanisms. This is however predominantly in a theoretical stage, and current telecommunication networks still operate using specific signalling protocols, mostly the Signalling system Number 7 (SS7) [Manterfield 91]. This is the reason why, although we mostly refer to signalling in a broad sense of the word, meaning any exchange of control and management information, a brief overview of SS7 has been included in section 2.2.2.3.

2.2.2.1. Software based network control

Modern networks are software controlled and the services they provide are established by the interaction of application programs in appropriate network elements via messaging over the signalling network. Interactions also occur between support systems and embedded network intelligence for the purpose of service and network management. In the future, it is

increasingly likely that IT (Information Technology) applications in customer terminal equipment for multimedia services will require heavy interaction with network control. Should invalid messages be exchanged between application programs then miss-operation can disrupt the normal operation of the network, closing it down in extreme situations. When constituent parts of the networks of different TOs and SPs are interconnected, the probability of such perturbations occurring arises, causing threats to network integrity. Moreover, the behaviour of such interconnected software is non-linear and possibly chaotic and hence most impossible to predict. Network integrity is therefore a control engineering problem and one of the more important aspects of modern Performance Engineering.

Modern networks are subject to problems characteristic of distributed systems. Functionality is provided by a number of processes that operate separately and communicate between each other. This raises concurrency and real time issues such as deadlocks and live-locks. Deadlocks occur when two communicating processes are stalled because they both are waiting for a message from one another. Live-locks are more complex situations in which both processes remain active but they enter a 'vicious circle' where they keep exchanging the same set of messages indefinitely. An example of a live-lock situation between two processes can be described as follows. Suppose that process B receives a message M_{AB} from another process A. This instigates a number of actions in process B, and as a result a message M_{BA} is sent from process B to process A. If the receipt of message M_{BA} causes process A to send the message M_{AB} to process B, the livelock situation occurs. Processes A and B will continue exchanging the same messages M_{AB} and M_{BA} indefinitely.

Threats to network integrity are predominantly due to software faults and issues related to distributed processing. These problems are extremely complex, and massive effort is required to understand and detect them in order to protect the integrity of the networks and services.

2.2.2.2. Intelligent Networks and Service Creation

Intelligent Networks (IN) represent a major milestone in the recent evolution of telecommunications. The IN is gradually being adopted by major operators and it is believed that they will be the predominant technology across the world within the next few years. Hence, understanding service provision under the IN paradigm and emerging issues is of great relevance to identify integrity threats.

A general view of an Intelligent Network (IN) is that it is simply about the convergence of telecommunications and computing. Quoting Willis and Dufour in [Willis and Dufour 95],

"the simplest way to imagine this, is to think of the manual switchboard operator of a hundred years ago, i.e. the intelligence, knowledge and information held by the operator is replaced by data and logic embedded in computer programmes". Perhaps the characteristic that most distinctly defines an IN is that the control of services is functionally separated from basic call processing. This separation makes possible the rapid creation and deployment of services in switched networks. The IN also defines an architecture of modular and reusable network functions that provide service/network independence.

The International Telecommunications Union -Telecommunications Technical Committee (ITU-T) has proposed an evolving sequence of standards that support IN. The starting point is a core structure of capabilities called Capabilities Set 1 (CS-1), which will be advanced by the addition of extra capabilities and services to form the next definition, CS-2. The process is proposed to continue to produce CS-x, each new set built on the last, with ever expanding capabilities. A brief introduction to IN is included in Appendix A.

The IN concept would not be possible without the introduction and development of common channel signalling. This is dealt with in the next section.

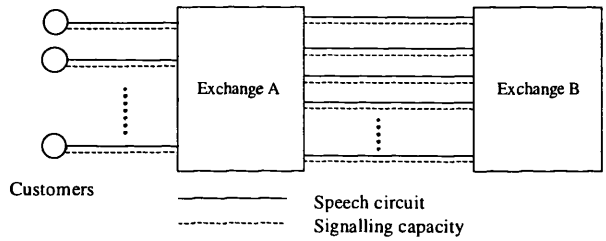
2.2.2.3. Signalling

Signalling has been defined as 'the life-blood, the vitalising influence, of telecommunication networks' [Manterfield 91]. Signalling and the associated signalling protocols is the language by which the different network elements communicate. Due to its critical nature, signalling is experiencing rapid evolution and growth in complexity. A major advance has been the introduction of common-channel signalling in conjunction with digital systems.

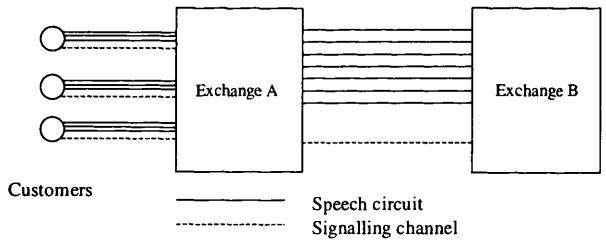
Signalling information can be transferred in two ways. The first way are the so called *channel-associated signalling (CAS)* systems, where the signalling information is transferred over the circuit itself or dedicated signalling capacity is provided for each circuit within the transmission link. The second type of signalling systems are *common-channel signalling (CCS)* systems, where a common signalling path is provided for a number of circuits. Figure 2 illustrates the principles of both signalling systems for both access and inter-exchange signalling. In general, CAS systems are used in 'old-technology' networks in which exchanges use analogue techniques and transmission systems are primarily analogue. Modern CCS systems, on the other hand, are optimised for 'modern-technology' networks, where both exchanges and transmission systems adopt digital techniques.

CCS is being widely adopted throughout the world, and the role of common channel signalling has become more and more important with the evolution of digital networks and intelligent network capabilities. The principal advantages of CCS systems are as follows:

a) In CCS systems, the transfer of signalling information is achieved by sending messages down the common signalling path. These messages are composed of a number of fields, each containing a certain type of information, such as the circuit identifier, the service identifier, etc. The structure of the messages is defined by the specification of the signalling system. The use of messages provides a high degree of flexibility that is not present in CAS systems, where the number of signals is limited.



a) Concept of channel associated signalling



b) Concept of common-channel signalling

Figure 2. Concepts of channel-associated signalling and common-channel signalling

b) An immediate consequence of having a message based signalling system is that the signalling information is not restricted to specific services. This flexibility provides the ability to add new features and respond to new network requirements.

c) Because the signalling information is sent over a dedicated signalling channel, separated from the speech circuit, this signalling information is not restricted to the control of traffic

circuits. Effectively, the signalling channel can be used for the transfer of any type of information.

d) CCS systems make use of the intermittent nature of signalling for voice traffic circuits. The signalling activity when setting up and releasing a circuit is high; however, on average the signalling activity for a circuit is low, because there is no signalling when calls are not being made and during the conversation phase of a call. Hence, a CCS channel can be used to handle numerous traffic circuits.

e) The transfer of signals in CCS systems is much faster than in CAS systems. A greater amount of information can be transferred at one time by one message.

In summary, the use of messages to transfer signalling information in CCS systems provides a high degree of flexibility. The repertoire of signalling information is much larger than in CAS systems, and messages can be transferred at any stage of a call without affecting the calling and called customers. The signals are transferred very quickly, and this permits the inclusion of far more information without an increase in post-dialling delay. The use of a dedicated signalling channel for the transfer of the signalling information provides the ability to send general data, not just information related to the control of traffic circuits. Hence, CCS systems can be used for non-circuit related applications, such as provided in Transaction Capabilities in SS7 (see appendix B). Consequently, CCS systems can easily include new features, hence permitting the incorporation of new services in a flexible and comprehensive manner.

The role of CCS systems has become increasingly important with the evolution of digital networks, intelligent networks and mobile services. Initially, CCS was mostly used for call control, i.e. establishment and release of a call and the associated circuit-switch connection. Then the ability to provide non-circuit associated signalling capabilities started to be used. With the progress of Intelligent Network (IN) it has been used for access to service control and data systems from telephone exchanges to provide various advance services. The flexibility provided by the use of non-circuit related signalling results in an increased complexity to the signalling scenarios. This complexity is exacerbated by the need of providing global services and the interworking scenarios between networks and between service providers and networks. The interconnect issues are dealt with in section 2.2.3.1.

The CCITT Signalling System No. 7

The trend in service provision is towards the distribution of network functions and service control. Distribution brings the need to communicate, and this is the function of signalling. Modern intelligent network principles demand flexible, reliable and efficient signalling systems. It is essential that the protocols used for signalling between the network functions are service independent to allow new services and features to be implemented without upgrades to the signalling systems. Standardisation of the signalling systems is also important, particularly in a multi-vendor network or where interconnect to other networks is required. These two situations will be present in Europe, the latter as a result of the European Commission ONP policies and the ETSI mandatory standards. The standard signalling system widely used in today's digital networks is the ITU-T Signalling System number 7 (SS7). This is a digital, message-based common channel signalling system. An overview of the SS7 protocol is included in Appendix B.

As stated in [ONP-UCL 94], para. 26-30, problems with signalling can compromise the integrity of the networks, particularly in an interconnection environment, where problems may migrate between different networks. The situation becomes more critical as services become ever more sophisticated, needing ever increasing and more complex signalling activity. In [ONP-UCL 94], some examples of the types of problem that can compromise network integrity are stated, e.g. signalling feedback loop, SCP (Service Control Point, see Appendix A) failure, incorrect message treatment, etc. This is only a partial list, and further work needs to be undertaken to identify and categorise different types of problems.

CCS Vulnerabilities. The USA experience.

CCS technology provides increased capability and flexibility to incorporate new services. But CCS networks also present a very high complexity and a number of vulnerabilities that may result in network integrity violations. The integrity of the signalling network is a major factor in the overall integrity of networks and services. The inherent complexity of CCS networks is aggravated by a number of factors, such us the following [Hoang et al. 93]:

a) The diversity of manufactures, which results in a disparity of implementations due to different interpretations of requirements or standards.
b) The CCS networks tend to have very heavily concentrated links and a limited number of Signal Transfer Points – STPs - (refer to Figure 3), hence failures in the CCS network can severely and rapidly impact services.

c) CCS networks have to be robust and flexible enough to accommodate future traffic volumes, which are experiencing a fast and steady growth.

d) The interconnection of carriers provides technical and administrative challenges to assure interoperability between networks.

e) Adequate surveillance capability is required to be provided by each CCS node and by a separate independent entity that monitors the entire network, records unusual events and provides a timely and accurate global status from a network performance point of view. In addition there is need for surveillance across networks.

f) The need for effective signalling network recovery procedures and global network controls for re-initialisation in order to minimise the disruption or loss of service to customers.

Public attention in the USA was brought to the vulnerability of the new network technologies by the severity of the SS7 related failures that afflicted several carriers. In January 1990 AT&T suffered a nine-hour breakdown which had nation-wide impact and affected 65 million calls. At least six separate outage incidents occurred in service territories controlled by Pacific Bell and Bell Atlantic, during summer 1991, impacting 10 million customers. More outages were reported the following autumn. Details about the USA experience regarding these failures and the subsequent measures taken by regulatory bodies and other forums and committees can be found in [ONP-UCL 94], appendix B, and in [Hoang et al. 93]. A brief summary is described here.

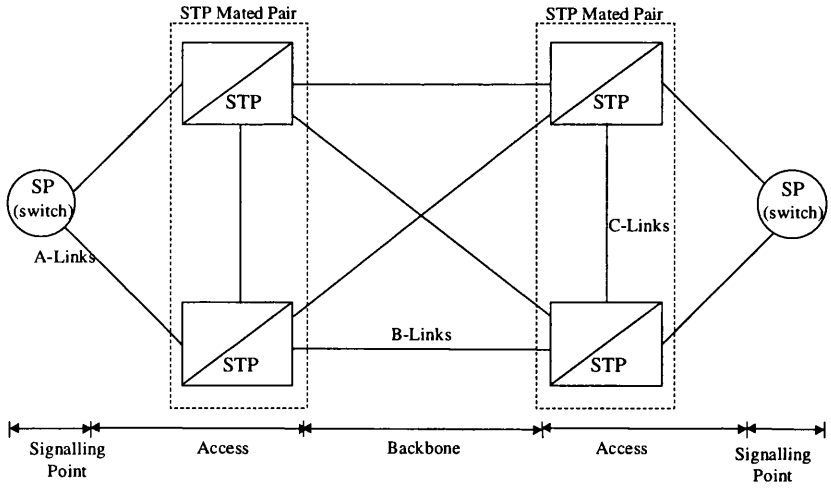


Figure 3. Typical SS7 architecture

Further investigations after the summer 1991 outages revealed that these outages were caused by CCS vulnerabilities. Figure 3 illustrates a typical SS7 network architecture employed in the USA [ONP-UCL 94], appendix B. This is a separate overlay network above the bearer network. The signalling network structure is a mesh, or quad, with the Signal Transfer Points (STPs) being the key components. Each STP is duplicated by a mated pair. The function of the STPs is to relay messages from the originating Signalling Points (SP) to their destination.

The common characteristic in all the events that occurred in the summer 1991 was the spread of congestion between multiple STPs as well as switches. Abnormal events occurred in stages, the combination of which resulted in substantial loss of signalling and, therefore, service capability. The first event that was observed was internal congestion within an individual STP. This started with a triggering event, such as massive simultaneous link failures, which resulted in excessive accumulation of messages queued by an STP for transmission on at least one signalling link. When the transmit buffer of the signalling link filled up, it lead to additional buffering in other internal buffers. Due to a software bug, back up propagated through the internal buffers, leading to accumulation of messages in receive buffers. This caused SIB (status indication busy) messages to be sent to the connected nodes. After receiving SIB messages continuously for an excessive amount of time, the connected nodes took the congested link out of service, as specified by the protocol. This resulted in isolation of some nodes from the CCS network, thereby impacting service at these nodes. The software bug mentioned above prevented activation of standard SS7 congestion control procedures: i) discarding of messages, based of priority, when the number of queued messages exceeds certain thresholds, and ii) notification to adjacent nodes, via TFC (transfer controlled) messages, that congestion exists. As a result, congestion spread from the initially affected STP throughout the network. If TFCs messages had been sent, the adjacent nodes would have slowed down traffic to the congested STP, allowing congestion to abate as the initial congested STP cleared its backlog of messages.

These outages alerted the telecommunications world to the threats to network integrity and reaffirmed the need for industry-wide initiatives to decrease network integrity risks. They helped to identify a number of integrity issues, and as a result a number of initiatives and efforts were undertaken by suppliers, carriers and regulators to decrease the risks to network integrity. Some of these issues and initiatives are as follows:

a) The need for enhancements to SS7, especially regarding its performance during severe congestion and ability to recover became apparent. A working group, consisting of members from Bellcore, Bell Atlantic, Pacific Bell, AT&T, DSC and Northern Telecom, Inc, identified several specific SS7 areas of improvement to enhance recovery and minimise congestion. Additional work with the industry and ANSI (American National Standards Institute) standards committee T1 led to the publication of a Bellcore Technical Advisory on congestion control and enhancements.

b) Software diversity must be implemented in the STPs in order to reduce the risk of having the same defective software present in all four STPs that belong to the same local quad. In [Hung et al. 94] four alternatives to achieve software diversity in common channel signalling networks are discussed, namely: i) multiple software developments in STPs; ii) different software generics for back up; iii) mixed supplier STP pairs; and iv) E-links sets to different supplier STPs. Although these alternatives could improve network reliability, there is a trade-off for increased costs and more complex operational procedures.

c) The outages also raised the need for more exhaustive interoperability testing. Two major programmes were put into place. One effort is specific to testing between interconnected network elements and is achieved through Bellcore's Interoperability Analysis Programme (IAP) whereby suppliers interconnect their lab facilities through Bellcore to facilitate product-to-product testing. Another effort is specific to testing between interconnected networks. In October 1991, Bellcore provided to the industry a proposal for additional interoperability testing that would be performed in a non-live or lab environment. This initiative is known as the Internetwork Interoperability Test Plan (IITP). The IITP is an industry-wide group focussing interoperability testing for assuring network integrity of the interconnected networks while inducing adverse conditions. Co-operative internetwork testing for integrity of CCS networks is done through the IITP.

d) The issue of sharing information about outages was raised, and regulatory authorities released new rules requiring U.S. carriers to promptly report outages that exceed certain values of service disruption. In addition, the Network Reliability Council (NRC) was established by the FCC (Federal Communications Commission) in January 1992 to obtain technical advice and public input on a variety of telecommunication issues [T1A1.2/97-001R3]. The NRC provided the FCC and the Telecommunications industry with recommendations for prevention of service outages in public telecommunications networks and minimisation of the impact of such service outages. The charter of the NRC was renewed in January 1994 to address issues concerning changes in telecommunications networks. Another chartering was also announced in April 1996, but

with a modified name, the Network Reliability and Interoperability Council (NRIC). NRIC is charged with developing recommendations dealing with the applicable areas of the Telecommunications Act of 1996. The Network Reliability Steering Committee (NRSC) was formed by ATIS (Alliance for Telecommunications Industry Solutions) under the recommendation of the NRC in May 1993. The mission of the NRSC is to ensure a continued high level of network reliability. In particular, the NRSC produces quarterly and annual reports analysing network outages. These reports are for the FCC and the telecommunications industry. The NRSC also refers network outage issues to other industry forums and groups.

The studies and initiatives undertaken by these study groups received considerable co-operation from the industry. The industry players are fully aware that it is mutually beneficial to co-operate on such ventures, since any service outages are detrimental to the public image for the industry as a whole.

2.2.2.4. Feature interactions

Technology is expected to support rapid and flexible introduction of increasingly complex sets of services. Thus, the main goal of the Intelligent Network (IN) is to provide a framework for the fast introduction and expansion of telecommunication services in a multi-supplier, competitive environment. The network provides a collection of services which, on their own, are a collection of features that co-operate to provide an overall service structure. One of the major obstacles in this environment is the feature interaction problem; i.e. a service feature may interact with other existing features in some undesirable way, resulting in adverse behaviour.

The problem of unintended feature interactions is not new. It already exists within the scope of current network services. These are usually dealt with on a case by case basis using manual investigation techniques. The usual approach to feature interaction is to exhaustively test all possible feature interactions when a feature is first designed [Brothers et al. 93]. When a feature interaction is detected or predicted, either the two features are not permitted to be active together, or methods for resolving it are determined. This exhaustive and manual approach has been feasible, so far, in the public network because the number of features supported was very limited. However this approach is no longer valid in new IN-type environments characterised by the availability of a large number of features, together with the need for interconnect across several separately controlled administrative regions and the increasing number of vendors supplying network systems and services to the public telecommunications network. Hence, substantial importance is now being placed upon

developing rigorous and automated techniques to resolve feature interactions. Extensive literature can be found on the subject of feature interaction, mostly for IN services. The USA experience and approaches for feature interaction analysis, detection and resolutions can be found in [Cameron et al. 93] [Brothers et al. 93] [Cameron and Velthuijsen 93] [McConnell et al. 93]. An approach to feature interaction detection using SDL[1] models is described in [Kelly et al. 94]. The issue of feature interaction in multimedia services is analysed in [Tsang et al. 95].

Examples

A large number of feature interaction problems have been identified already. These are a few classical example identified in the USA [Cameron et al. 93]:

a) Originating Call Screening and Call Forwarding: a screened number, X, can be obtained if a user, A, calls a user B and B forwards the call to X.

b) Calling Number Delivery (CND) and Unlisted Number (UN): CND delivers the calling party's directory number to the called party; UN is a directory service feature that prevents a subscriber's number from being released; these two services are by their nature incompatible; in most cases it is not a problem, but failure to provide CND can cause some systems to block the call on security grounds.

c) Multiway Calling and Emergency Calling: in Multiway Calling a user must put the second party on hold before contacting the third party; if an emergency call is made, call control goes to the operator and Multiway calling is impossible.

Causes

Feature interaction problems arise for a number of reasons. The main cause is that new services are developed in isolation from existing services and others under development. Standards do not always evolve at the speed that providers and operators would wish, and hence often different proprietary network solutions are adopted by the telecommunications industry. As a result there are many, and varied, protocols in both the access and core network, and a number of similar but incompatible services.

A further case of interworking problems is associated with the methods used to define both existing and future systems. Because of the informality of the current methods, it is not

---

[1] SDL is the ITU Specification and Description Language.

possible to understand the precise behaviour and the implications associated with new product definitions.

The need for new services is forcing the pace, with technology expected to deliver ever-faster service provision with an increasingly complex set of services. Two specific scenarios need to be resolved. For current systems, there is the task of identifying problems and solving these by modifying the service in operation. For coming scenarios, there is a need to change working practices to avoid the problems of yesterday by, for example, developing all networks and services in a common context, and technology and methods are needed to help.

In [Cameron et al. 93] a classification of feature interactions is provided based on their causes. The following three main types of causes are differentiated:

a) Violation of feature assumptions: a feature's specification makes assumptions about its environment; such assumptions are likely to be violated in long-lived systems, resulting in various interactions.

b) Limited network support, e.g. limited network equipment signalling, limited functionalities for communications among network components, etc.

c) Intrinsic problems in distributed systems: telecommunication systems are huge, real-time, reactive systems; the distribution of feature support in the network and the customisation of features by each individual can create interactions that require co-ordination.

Effects

The effects of feature interaction can be viewed from two viewpoints. The first perspective focuses on the user perception, since a feature interaction can cause services to behave in an unintended manner. In this sense, feature interactions are related to the quality of service, and the associated costs are in the form of loss of revenue.

The second perspective focuses on the effect that the feature interaction has on the performance or integrity of the network itself. Here, the costs associated with the problem of feature interactions are related to the equipment, personal and resources need to be made available in order to resolve the interaction conflict.

The two approaches, i.e. user perception and effect on the network, are linked, since a severe degradation in the performance of the network is obviously of concern of the users as well.

Approaches to manage them

The concept of a multi paradigm approach to solve the feature interaction problem seems to be becoming generally accepted, on the basis that it will not be feasible to resolve all possible feature interactions at any one stage in the lifecycle or with any one technique. This must be combined with a 'divide and conquer' philosophy, as proposed in [Cameron et al. 93], whereby the problem is split into a number of sub problems which can be dealt with using specific techniques. However, some people also believe that effort should concentrate in resolving as many feature interactions as possible at the beginning of the products lifecycle, i.e. in techniques of avoidance and specially detection of interaction during the specification of features. In this way, it is possible to highlight the potential interworking problems long before they are introduced into the network where they degrade the quality of service and are costly to rectify. For example using formal specifications and descriptions of services may help in an early detection of feature interaction problems [Kelly et al. 94].

There are two possible approaches to managing feature interaction [Cameron et al. 93]. The first approach consists of removing causes of interactions in order to avoid the occurrence of interaction by carefully designing an infrastructure to support the deployment of features and impose feature design rules. This approach presents two major disadvantages. One is that it may be too limiting given the diverse needs of customers. The other is that proper infrastructures can eliminate certain causes of interactions, but it will not be possible to predict and avoid all of them. Hence, the second approach is needed. This second approach consists of detecting and resolving the remaining interactions, developing software tools to aid in the detection and assist in the design.

Once detected, interactions can be resolved by prioritisation, feature alternation, or mutual exclusion. Alternatively, they could be left unresolved. There are two reasons why feature interactions would be left unresolved. One is the trade-off between costs associated with the feature interaction problem, and the cost of fixing it. The other is that fixing a particular interaction problem can itself cause other problems that did not exist before.

2.2.3. Regulatory environment

The telecommunications market has changed drastically over the last few years. As stated in [Avendaño and Hamelberg 93], historically, in most countries telecommunication operators were government services, or if they were formally independent from the State they were still

protected by some kind of monopoly regime for most of the services they offered. The situation is now changing drastically. New technologies have been widely adopted and as a consequence, a host of new services are being introduced, some of them offered by the traditional telecom operators, but others by new competing players in the field. From the 1980s, increasing competition made it necessary to give telecom operators more flexibility, and that included the separation from the State and changing them to private companies. The telecommunications industry is changing from monopolies to privatisation and competition, and this has important implications in the provision of telecommunication services [Valdar 89]. A major milestone in Europe was the publication of the EU's Green Paper on Telecommunications in 1987. This called for, among other things, more competition in services and in terminal equipment, as well as for the complete separation of regulatory functions and operational activities. It also introduced the concept of Open Network Provision (ONP).

The concept of ONP has been developed by the European Commission for the implementation of competition, to facilitate access to and use of telecom networks and services – nationally and throughout Europe. One of the essential points of ONP is the interconnection of networks and inter-networking of services. Viable competition can only be expected if new competitors and services can access the functionality and capacity of existing networks, together with their customer base. A consistent interconnect policy has therefore to be developed to facilitate the process towards full liberalisation of the telecommunications market. The ONP Voice Telephony Directive represents the first stage of a European interconnect policy. In it, interconnectivity between public networks for the voice telephony service is mandated and national regulatory authorities are called upon to ensure that reasonable requests for interconnection from authorised providers of voice telephony services are met. This Directive is only a first step, since it applies exclusively to voice telephony and leaves open what 'reasonable requests' are. After 1998, when full liberalisation has become a reality, it is likely that interconnection agreements concerning many different telecommunication networks and services will be negotiated between the parties involved. The regulatory authorities will have to contribute to this process by facilitating a framework for negotiations between the parties in which as many parameters as necessary are specified.

The ONP aims to open the networks of dominant Telecom Organisations (TOs) to new competitors - service providers (SPs) or other licensed operators (OLOs) - to gain access to the TO's customer base or use the resources/services of TOs to offer their competitive services. Therefore, an environment of extensive interconnected networks and SPs could develop.

Network Integrity can be compromised by problems migrating between interconnected networks and equipment connected to the network, particularly for interconnected networks comprising equipment from multiple suppliers, where interface standards may have been interpreted differently.

Policies to enhance competition are also being applied in the USA, i.e. the ONA (Open Network Architecture) policy [ONP-UCL 94], p. B3-B4. The Federal Communications Commission (FCC) is an independent government agency whose mission is to encourage competition in all communications markets and to protect the public interest. In response to direction from the Congress, the FCC develops and implements policy concerning interstate and international communications by radio, television, wire, satellite and cable.

In such competitive environments, where time to market is a key factor, there is a need to deliver very quickly, and this sometimes results in integrity affairs being given low priorities. However, measures need to be taken to avoid severe outages and to construct new frameworks for the development of networks and provision of service where attention is paid to potential integrity risk at all stages.

2.2.3.1. Interconnect

As a result of regulatory policies that foster competition and open markets, interconnect of networks of competing operators and service provides becomes a major issue. Initially, new competitive operators and service providers need to interconnect to the networks of established telecommunications operators. In the future, as these now new operators and service providers become well established, the interconnect scenarios are likely to experience a great increase in complexity.

The risk of breach of network integrity is increased when networks are interconnected. The interconnect point may become a point of weakness and possible source of problems that can jeopardise the integrity of the interconnected networks. One major cause of this can be incompatibilities in the software and protocols used at either side of the interconnect and wrong translations at the boundary. Thus, if two interconnected networks use different signalling systems, or even a slightly different implementation of the same standard, this can lead to misinterpretation of signalling messages with associated detrimental consequences. For example, in the SS7 protocol certain aspects are not explicitly specified hence allowing some variations in the implementation. Two examples are described in [Hoang et al. 93]. One is the fact that the protocol does not state the internetwork routing schemes, and as a result routing

may be implemented in a different way across the interconnected networks. Specifically, in the event of link failures the use of different routing schemes by each of the interconnected networks may impact traffic loadings on individual links or interpretation of messaging. The other example is regarding implementation of timers. Because the protocol allows for variations in timer settings, one network may, unknowingly, time-out on some function, which could effectively send a false indication to the interconnected network.

There appears to be a strong relationship between the complexity of an interconnect, in terms of such aspects as signalling activity and the number and range of network elements that are accessed to provide a particular service, and the degree of risk to integrity [ONP-UCL 94], para 29. In this sense, POTs would be low complexity but IN based services can present high degrees of complexity. For example, they require significant volume of transaction signalling to various databases. Conditions of interconnect will be influenced by risk and hence complexity - more complex internetworked services will require more demanding interconnect conditions. Therefore, to assist assessment of request for interconnect it is desirable that a broad measure of complexity be developed. This work is essential both from the industrial and regulatory points of view. Industry needs this type of quantitative definition to construct meaningful service level agreements or contracts between network operators for network interconnect, or between operators and service providers for service provision. From the regulatory point of view, this work is needed as a basis to impose regulatory conditions that can be arbitrated in quantitative terms. In the past, dominant operators have dictated the rules and developed their own criteria by which they would allow or reject interconnect to OLOs. Now, these major operators are being forced interconnect by the regulatory bodies, and hence the need to have clear, fair and well-understood criteria to referee interconnect agreements.

Interconnect scenarios will become increasingly complex for two main reasons. The first reason is that the number of competing telecommunication organisations that require interconnect is growing dramatically. Increasing number of OLOs (Other Licensed Operators) and SPs (Service Providers) require interconnect to the networks of established Telecommunications Organisations (TOs). Regulatory policies aim to open the networks of dominant TOs to new competitors – SPs or OLOs – to gain access to the TO's customer base or use the resources of TOs to offer their competitive services. An environment of extensive interconnected networks and SP equipment could therefore develop. The networks of dominant TOs are interconnected to those of competing or specialised network operators (OLOs) to obtain total connectivity between customers. Initially, OLOs gain access to the PTO's customer base to offer cheap trunk calls, sometimes known as trunk by-pass, but OLOs

are now competing by direct access to customers, e.g. cable TV operators. Mobile operators connect to obtain connectivity to customers on fixed networks. Increasingly sophisticated private networks are also interconnected to the public networks. In the future environment of highly sophisticated IN-like services, an increasing number of Service Providers will also require access to fix and mobile networks to offer specialised services. A complex mesh of networks and management systems will therefore arise, where the preservation of network integrity becomes extremely difficult.

The second reason why interconnect scenarios are becoming so complicated is the need for new advanced types of interconnect. Up until now the interconnect has been limited to POTS (Plain Old Telephony Service) and interconnect between fixed and mobile operators. However, nowadays, and increasingly in the near future, new and more sophisticated types of interconnect will be required for advanced services based on modern service provision paradigms, e.g. ISDN and Intelligent Networks. As a consequence, the approaches currently taken to interconnect testing need to be reviewed, and new policies that embrace these new interconnected scenarios must be developed. The future information infrastructure, with its proliferation of Service and Information Providers etc. and open (plug and play) interfaces sill exacerbate the situation.

Major TOs have a good deal of expertise in testing of 'simple' network interconnect, i.e. POTS with CCITT SS No 7. The interconnect agreements with OLOs are based on the results of these testing procedures. Interconnect policy requirements are set with the aim of protecting the integrity of the networks, e.g. by prohibiting unauthorised use of the TO network as a Signalling Transfer Point (STP), which may decrease the capacity for genuine signalling and possibly compromise the integrity of the TO network. Interconnect policing involves two main areas, namely the design of the physical architecture of the interconnect, and the definition of the policing requirements for interconnect. A common criteria for the design of the physical architecture of the interconnect is to reduce complexity, by for example restricting interconnect to a limited number of points. Regarding policing, the study carried out in the major TO revealed that, at the time of the study, most of the policing is embedded in the switches an it is undertaken at the message Transfer Part (MTP) level of the SS7. The policing consists of discarding invalid messages that arrive in the exchanges. It is recognised that the introduction of new IN-like services requiring non-circuit related signalling will require interconnect at the SCCP (Service Connection Control Part) level. Appropriate action needs to be taken for SCCP interconnect testing and policing at the SCCP level and higher levels in the SS7 structure. This will introduce a dramatic increase in complexity of interconnect testing

and policing, but the impact of future IN approaches in interconnect is still unknown. For this reason TOs are still reluctant to full IN interconnect.

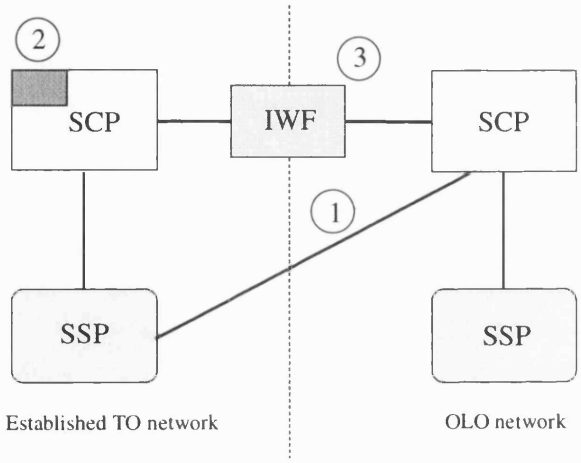


Figure 4. Alternatives for IN interconnect

Figure 4 illustrates different alternatives for IN interconnect, namely:

1. The OLO or SP connect their SCP to an SSP in the established TO network. This alternative introduces a high risk for the TO, because the OLO/SP has direct access to their switches, e.g. contention problems may arise due to several SCPs trying to access the switches, high risk of feature interaction, etc.

2. The OLO/SP has a partition in the established TO's SCP. This alternative seems to represent a lower risk for the integrity of the TO's network.

3. Direct connection between the OLO/SP and the TO's SCPs, by means of an interworking function (IWF) that performs the necessary security and integrity checks. With this approach, the SCPs would be able to resolve the feature interactions, because the SCP is where the intelligence resides. There are two options to avoid problems, namely, to impose requirements to the service providers as to what they can put in the SCP (testing and guidance to SPs), or real time management to resolve conflicts (prioritising in real time and stopping processes when necessary).

New emerging interconnect scenarios will have a great impact in the integrity of the networks. TOs will need to define their architectures and policing for these types of interconnect that best guarantee protection of their network integrity.

2.3. A review of related approaches to the problem

The maintenance of network integrity is being recognised as an issue of paramount importance in the current and forthcoming telecommunications scenarios. Regulators are showing their concern regarding the preservation of network integrity within the open regulatory framework envisaged by the Open Network Provision policies (ONP). An example is the study [ONP-UCL 94]. However, this is still a relatively new topic, and there are not unified concepts, agreed definitions or recommendations. Hence, an initial requirement towards developing a framework for network integrity is to clarify what is understood by network integrity. The meaning of network integrity is not clearly specified, and a number of different interpretations arise depending on the context in which the term is employed or on the person who uses it. In the context of data systems, integrity usually refers to the incorruptibility and completeness of the data. In other contexts, integrity is associated to the security of the data flowing in the networks, in terms of privacy and secrecy. Conversely, integrity may be associated to hardware failures of equipment.

The following sections review existing definitions and concepts related to integrity and failure quantification measures, and provide an analysis of these approaches in comparison to the network integrity definition proposed here.

2.3.1. Concepts related to integrity

The interpretation of network integrity can vary considerably depending on the context in which the term is used. There are a number of other concepts and measures related to it. This section contains an overview and literature review of some of these concepts.

In [Kühn et al. 94], section IV, reliability measures for the case of SS7 networks are grouped into three categories: *availability*, *dependability* and *robustness*. The notion of network availability is related to the performance quality of the network and continuity of the services they provide. The notion of availability is defined for components or systems that can be in either a working state or a failed state. Availability has been defined by CCITT as "the ability of an item to be in a state to perform its required function a given instant of time or at any instant of time within a given time interval, assuming that the external resources, if required, are available" [CCITT Red Book 85]. Instantaneous availability and steady state availability (or simply availability) are defined as the probability that the system or component is in a working state at a time instant (instantaneous availability) and at a random point in time for a

system in statistical equilibrium (steady state availability). Availability, A, can be expressed as A=MTTF/(MTTF+MTTR), where MTTF is the Mean-Time-To-Failure and MTTR is the Mean-Time-To-Repair. Another frequent measure of unavailability of a system is downtime. A common objective for the downtime of a system is that "the average downtime should be less that M minutes per year". The different interpretations of this statement can be found in [Kühn et al. 94].

Although sometimes integrity is understood in terms of availability, we share the view of McDonald in [McDonald 94] that integrity is a higher level measure, which includes a customer's perception of end to end quality. In addition, in our view an essential aspect of integrity is the ability of the networks to withstand adverse conditions, and hence integrity is related to concepts such as robustness, described below.

Dependability measures relate to the ability of a signalling network to reliably transport messages and not cause malfunctions. For the MTP part of SS7, the dependability objectives are defined in terms of four measures: undetected errors, lost messages, messages out-of-sequence and transmission error rate. For the ISDN-UP, the objectives are in terms of probability of false operation and probability of signalling malfunction [Kühn et al. 94], sect. IV.

The notion of robustness refers to the ability of a network to withstand large or catastrophic failures. Another term commonly used to describe this notion is *survivability*. Survivability refers to the behaviour of the networks when failure occurs. In [T1A1.2/93-001R3] network survivability is defined as two components: "(i) the ability of a network to maintain or restore an acceptable level of performance during network failures by applying various restoration techniques, and (ii) the mitigation or prevention of service outages from network failures by applying preventative techniques. Preventative techniques can reduce the need for restoration techniques". It is this latter preventative aspect of the problem that this work is mostly concerned with.

A four-layer framework that classifies survivability techniques in telecommunication networks is presented in [T1A1.2/93-001R3] and summarised in [Zolfaghari and Kaudel 94]. The four layers of the framework are the physical, system, logical and service layer. Network survivability techniques can be deployed in each layer. Failures within each layer can be guarded against by techniques either in that layer or at a higher layer. A summary of various generic network survivability techniques based on the four layers is included in the cited

documents, and some examples of network survivability measurements for the four survivability layers are given, e.g.:

- Physical layer: connectivity, number of surviving cables
- System layer: number of surviving transmission systems
- Logical layer: number of surviving lower-rate transmission channels, connectivity
- Service layer: end-to-end grade of service, number of calls, traffic volume, carried load, etc.

In order to evaluate different survivability techniques or restoration methods, two basic approaches to survivability performance analysis are discussed in [T1A1.2/93-001R3], section 7, namely the Given Occurrence of Failure (GOF) and the Random Occurrence of Failure (ROF) survivability analysis models. The GOF model uses a conditional approach and defines measures for a network assuming that given failure(s) has occurred. The GOF model may either use probabilistic weighting of the resulting states of the network and resulting network restoration and/or repair after the failure or it may use deterministic analysis of these states. Both approaches can be used to evaluate different restoration, repair of preventative methods depending on which type of comparison characteristics are critical. Users often phrase their survivability requirements in terms of which types of failure they want their traffic protected from, and what proportion of the traffic should survive. The general procedure for evaluating GOF measures is as follows:

1. define a survivability measurement,
2. identify the sample space (i.e. the failures that can occur),
3. choose the failures of interest, and
4. calculate the network survivability measures.

The ROF model uses probability of network failures(s) and, possibly, rates of repair and/or restoration to calculate various probabilistic measures of network unservability or loss (e.g. the expected amount of time a network is unservable). This model is based on the assumption that failures can be characterised by random variables with given probability distribution functions. The general procedure for evaluating availability based network survivability measures is as follows:

1. obtain observed rates of failure and repair/restoration,
2. define network survivability measurements of interest,

3. identify the network state space (i.e., the various states in which a network can reside concerning whether its components are working or failed),
4. determine the survivability measurement for each network state,
5. determine or assign the transitional probability from each state to another, and
6. calculate the network survivability measures (e.g. the expected units lost or unservable over time). (It is recognised that step 6 is often a very complex task and sometimes impossible to evaluate exactly in networks).

The GOF and ROF survivability can be applied to develop performance measures for the four survivability layers. In both models it is important to first specify what measurements of the networks to capture, then obtain the survivability measures using the procedures summarised above.

Apart from the GOF and ROF techniques described above, qualitative assessment of the restoration methods is also important. Amongst other criteria proposed in [T1A1.2/93-001R3], section 7.5, the *stability* of the method is defined as "the response or ability to predict the response of the restoration method to variations of "perturbations" in the network parameters, such as type of failure, speed of data links, speed of hardware, database inconsistencies, data communication errors, or size of network".

There are no standard or explicit quantitative measures of robustness, and this is recognised as an area of considerable interest. There is a need to examine performance measurements, e.g. for signalling, that permit a better characterisation of the notions of robustness of the networks, and to develop a framework where these measures can be used to prevent the occurrence of failures. This is one of the main themes of this thesis.

2.3.2. Failure quantification

Measure of integrity is often associated to the quantification of the impact of failure. In this area, several measures have been proposed to quantify and categorise network outages. The Federal Communications Commission (FCC) in the USA define an outage as "a significant degradation in the ability of a customer to establish and maintain a channel of communications as a result of failure in a carrier's network". In [T1A1.2/97-001R3] a service outage is defined as "the state of a service when network failure(s) impair the initiation of new requests for service and/or the continued use of the service, and the service outage parameters exceed their

corresponding thresholds". Three existing definitions to quantify and categorise network outages are reviewed in the following sections.

2.3.2.1. Cochrane Richter Scale

This definition, proposed by Prof. Peter Cochrane (BT), suggests that network outages can be categorised like earthquakes, in terms of the Richter scale. Total network information capacity outage (loss of traffic) in customer effected terms is thus:

$$D = \log_{10}(NT) \tag{1}$$

where N is the number of customer lines effected and T is the total downtime.

2.3.2.2. User Lost Erlang

J.C. MacDonald has proposed in [McDonald 94] as a measure of network integrity the "ULE" (User Lost Erlang), defined as follows

$$ULE = \log_{10}(ExH) \quad \text{for } ExH \text{ greater than 1,} \tag{2}$$

where E is the estimated average user traffic lost during the time of the outage in Erlang taken from historical records. The quantity H is outage in hours. Thus 1 ULE=10 user lost erlangs, 2 ULE=100 and so on.

Although loss of network integrity is commonly assumed to be related to total unavailability, it can also be the result of degradation of other performance parameters that effect large numbers of customers, e.g. poor error performance or the unavailability of a service or services. In such circumstances the ULE parameter could still be used as a measure of lost integrity by re-defining E as "the estimated average user traffic lost or of unacceptable quality of service during the time of the service disruption, in erlangs" and H "the service disruption in hours".

2.3.2.3. ATIS T1-Committee Outage Index

In the USA, the Federal Communications Commission (FCC) in their February 92 Report and Order challenged the telecommunications industry to develop an analytical method for quantifying outages. The Network Reliability Steering Committee (NRSC) passed the remit to

the T1 Standards committee - both are committees of ATIS (Alliance for Telecommunications Industry Solutions).

In an effort to address the FCC challenge, work in the area was undertaken by the ATIS T1A1 Working Group on Network Survivability Performance, resulting in the technical reports No. 24 [T1A1.2/93-001R3], supplemented by No. 24A [T1A1.2/97-001R3], and No. 42 [T1A1.2/95-001R4] (which replaces report No. 38). Report No. 24 contains an analysis of survivability techniques, as explained in section 2.3.1. Reports No. 24A and No. 42 introduce a general framework for quantifying service outage from a user's perspective. The methods consist of guidelines and algorithms to compute an outage index, based on FCC data items and NRSC outage categories, for those outages meeting the 30,000 potentially affected customers and 30-minute duration criteria.

The T1A1 proposed that an index to quantify outages should be based on a combination of service[s] effected, duration and magnitude (customers effected). In particular it should:

a) reflect the relative importance of outages for different services,

b) be able to be aggregated to allow comparisons over time, and

c) reflect small and large outages similarly to their perception by the public.

The ATIS framework is composed of three parameters:

a) The **Unservability (U)**, a measure representing the fraction of service lost as a result of the failure.

b) The **Duration (D),** a measure representing the length of time in which a substantial fraction of service was lost.

c) The **Extent (E)**, a measure representing the breadth of the failure across the network.

The service outages can be categorised by sets of values of the triple (U,D,E), as illustrated in Figure 5, where the triples are categorised in three regions of minor, major and catastrophic.

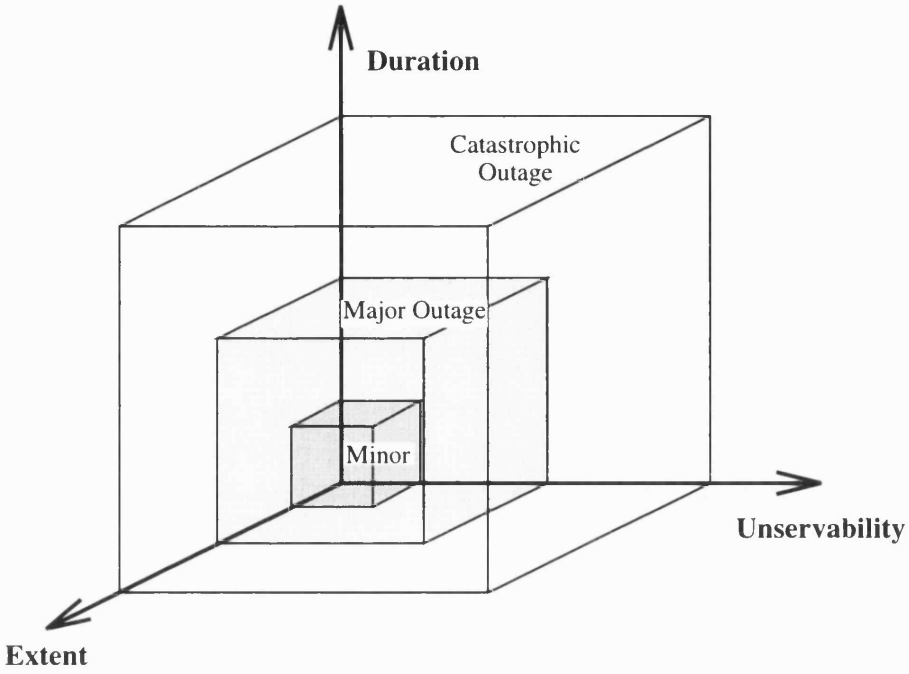


Figure 5. [U,D,E] Quantifying regions

An important objective of the exercise is to enable the summation of individual outage index values over time periods and related to the FCC reporting requirements, for comparison purposes. Aggregation should reflect the importance of outages to customers, such that the aggregate index for multiple small outages is less than the index for one large outage, where its duration (or magnitude) equals the sum of the small outages. Likewise, the aggregate index for multiple large outages should be greater than the index for one very large outage over the same measurement time scale (one year). To account for this behaviour, duration and a magnitude weighting functions in the form of "S" curves are introduced in the construction of the index (Figure 6 and Figure 7). In addition to the duration and magnitude weight, there is a service weight that takes into account the importance of the service.

The index of an outage is the sum of the service outage index values for each service affected. The service outage of each service affected is the product of the:

◊ Service Weight (Ws)

◊ Duration Weight (Wd)

◊ Magnitude Weight (Wm)

Thus, the outage index I(O) for an outage O has the following form:

$$I(O) = \sum_{j=1}^{N} W_{s[j]} W_{d[j]} W_{m[j]} \quad , \tag{3}$$

where $j=1,...N$ are the services.

The weights for a particular service are calculated as follows.

Duration Weight (Wd)

The Duration Weight measures the impact of outage duration on customers. The Duration Weight follows an S shaped curve, with an inflection point at 30 minutes (see Figure 6). Thus, the curve raises rapidly for duration up to 30 minutes (inclusive) and then slowly rises to the asymptote of 2.5. The following equations are to be used for calculating W_D:

If the duration is less or equal to 30 minutes:

$$W_D = (Duration\_Minutes / 30)^2 \tag{4}$$

If the duration is greater than 30 minutes:

$$W_D = 2.5 - 1.5\left(\frac{153.54}{123.54 + Duration\_Minutes}\right)^{1.327} \tag{5}$$

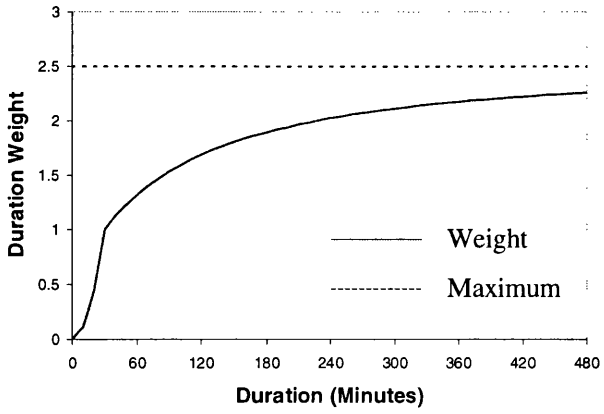


Figure 6. S-shaped curve for duration weights

In the general case, an outage may have different duration for each service effected, but according to ATIS, it will be often difficult to determine this. Therefore, a single duration may have to be used for all services.

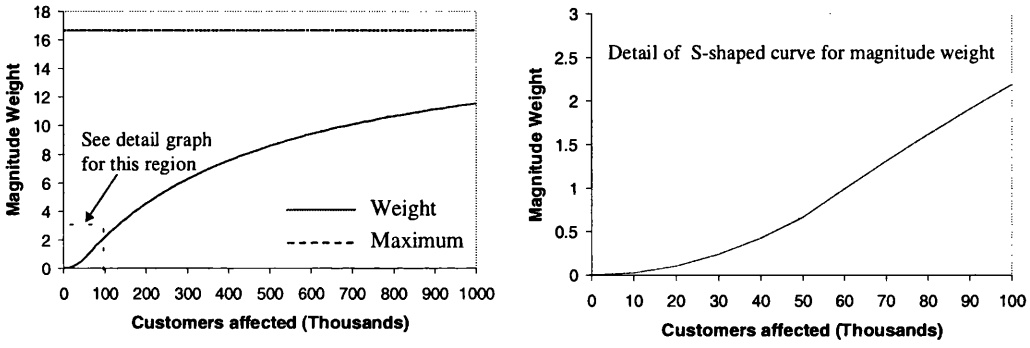


Figure 7. S-shaped curve for magnitude weights

Magnitude Weight (Wm)

The Magnitude Weight measures the impact of outage with respect to the customer base affected. The Magnitude Weight follows an S-shaped curve, with an inflection point at 50,000 customers affected. Thus, the curve raises rapidly for duration up to 50,000 customers (inclusive), and then slowly rises to the asymptote of 16.67 (see Figure 7). For outage calculations, the Magnitude Weight is given by the following equations:

If the number of customers affected is less than or equal to 50,000:

$$W_M = \left(\frac{NumberOfCustomersAffected}{1000} \right)^2 / 3750 \tag{6}$$

If the number of customers affected is greater than 50,000:

$$W_M = \frac{50}{3} - 16 \left(\frac{532.2}{482.2 + \frac{NumberOfCustomersAffected}{1000}} \right)^{1.114} \tag{7}$$

In order to calculate the number of customers effected, two methods are used, depending on the outage category. Two main types of outages are identified. The first type are dedicated outages, where the lost of traffic is generated by service requests that do not have alternative routing capabilities (100% blocking). The second type are diversified outages, where the lost of traffic is generated by service requests that have alternative routing capabilities (less than 100 % blocking). For dedicated outages, diversified outages in which blocked call data is not available, and emergency service outages, the "Lines" method is used to calculate the number of customers affected. For diversified outages in which "real-time blocked calls" or "historical carried call count" are available, the "Blocked Calls" method is used. Both methods are explained below.

a) Lines Method

a-1) For all services except the emergency service

$$Number\ of\ Customers\ Affected = Number\ of\ Lines\ X\ Time\ factor$$

The time factor accounts for the calling pattern and four cases are differentiated. Table 1 summarises the Time Factor values for each of these cases.

Outage Type	Spanned Time Period	Time Factor
Day	8:00 am to 4:59 pm, Mon-Fri	1.0
Evening	5:00 pm to 10:59 pm, Mon-Fri	0.3
Night	11:00 pm to 7:59 am, Mon-Sun	0.1
Weekend	8:00 am to 10:59 pm, Sat & Sun	0.2

Table 1. Time Factors for different types of outages

The time factor of an outage is the maximum of the time factors for each of the time periods it spans.

a-2) For the emergency service

$$Number\ of\ Customers\ Affected = Number\ of\ Lines$$

b) Blocked Calls Method

$$Number\ of\ Customers\ Affected = Number\ of\ Blocked\ Calls/3$$

In the blocked calls method it is assumed that a customer will make two re-attempts when call blocking occurs.

For each service, the number of customers affected is obtained applying either the lines or the blocked calls method, and with this data the magnitude weight can be obtained from equations (6) or (7).

Service Weight (Ws)

The impact of a service outage can only be quantified with respect to the context in which it occurred. In [T1A1.2/97-001R3] this context is provided by (i) the architecture in which the network element failed, and (ii) the services affected by the network failure. A separate methodology is described for each combination of service group and network architecture. Three service groups are identified, namely

1. Voiceband Telephony (including emergency service)
2. Point-to-multipoint
3. Multimedia

The network architectures are grouped in four segments of industry to which a specific network architecture can be applied. These are included in Table 2.

Segment of Industry (Network Architecture)			
Wireline (PSTN)	Wireless (cellular, PCS)	Cable TV (FTTC, HFC)	Satellite (FSS, BSS, MSS)

Table 2. Industry Segments and Network Architectures

Methodologies for calculating outage index values have been developed for the Voiceband Telephony service, and the four segments of industry. The Point-to-multipoint and Multimedia cases are left for further study.

The service weight is different for each type of service. For example, for the case of wireline industry segments providing fixed voice band telephony service, the following cases have been identified [T1A1.2/95-001R4]:

a) Lines method outages

a-1) IntraLATA Intraoffice: Ws = 1

a-2) IntraLATA Interoffice: Ws = 2

a-3) InterLATA Interoffice: Ws = 2

a-4) Emergency: Ws = 3

Service types a-1) to a-4) are shown in Figure 8.

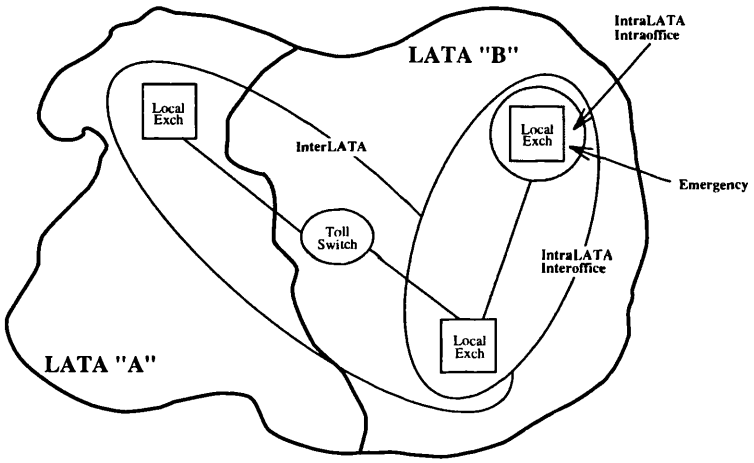


Figure 8. The four types of services identified for the ATIS index

IntraLATA intraoffice services only involve one local exchange. They have the lowest service weight, Ws=1, because the area that they cover is the smallest compared to the other types of services. IntraLATA interoffice are services within the domain of one network, but that extend to more than one local exchange. InterLATA services extend to more than one domain. Both IntraLATA interoffice and InterLATA services have been assigned the same services weight, Ws=2. The highest service weight is for the emergency services (Ws=3), since the consequences of an outage affecting this type of services can be very grave.

b) ***Blocked Calls method outages:*** Ws = 2

2.3.3. Evaluation of the existing approaches

A number of concepts associated to network integrity have been discussed in section 2.3.1. Some of them, e.g. availability, consider probabilities or failure rates based on individual elements of the systems. This is a contributing factor to network integrity, but network integrity needs to look at the systems as a whole and include overall performance and quality of service measures that reflect the impact of failures in customers. In addition, here integrity is regarded as the ability of the network to withstand adverse conditions and recover from initial performance degradations and possible error situations before they result in catastrophic failures. Hence, integrity is intimately related to robustness, otherwise referred to as survivability. This is often associated to restoration and recovery techniques in the presence of failure. A number of these techniques exist and can be applied at different functional levels in the networks, such as discussed in [T1A1.2/93-001R3]. But these techniques are concerned with failures in the transport network i.e. the aim is to guarantee that traffic can be routed to the destination even in presence of some link or node failures. This is a very complex area and considerable amount of work is devoted to it, but it is only one aspect of network integrity. Conversely this thesis is concerned with integrity threats at the intelligence levels in modern networks, which is a new area where little research has been carried out.

Another area related to survivability is the quantification of failures, and three measures proposed in the literature have been presented in section 2.3.2. The first two measures - the Cochrane Richter scale and User Lost Erlang (ULE) - possess the advantage of being easy to calculate and comprehend. They use a logarithmic scale, which reduces the numbers to manageable sizes. However, they are too simplistic since they only consider two parameters, the duration of the outage and the number of customers effected (in the Cochrane Richter scale) or the traffic lost (in the ULE). These methods do not contain some important features like discrimination between different services and information about the context in which the outage occurs. Another disadvantage of both measures is that they do not reflect the effect of outages of important individual services with low traffic volumes.

When defining a measure to quantify the effect of network outages, a key issue is to determine the criteria by which outages are categorised. The Cochrane Richter definition takes the number of customer lines affected as a measure of the magnitude of the outage. This is not a suitable representation of the significance of the outage, because it does not take into account the network usage by those customers - e.g. the outage could happen at midnight or during the busy hour - nor does it reflect the type of the customers effected, e.g. business or residential.

The ULE presents an improvement in this sense, since it takes account of the predicted traffic loss at the time of the outage. However, to consider only traffic loss to quantify the outage is insufficient. Think for example about important individual services that carry low traffic volumes. An outage that affects these services would not cause a great lost of traffic, but the consequences might be very serious if the customers effected are high value business customers, or in the case of emergency services.

A key issue is to understand the impact of potential failure on different customer types. It must be possible to evaluate the number of customers effected, their importance to the operator and the impact on their services - which could be lost calls (difficult to quantify, and may be complicated by some lost calls being repeated later) or loss of access to specific services, or extent of unavailability. Another effect on the operator is the loss of revenue from the affected services and for the financial consequences of rebates or litigation. All the approaches lead to the economic point of view, that is to use measure of outage quantification to evaluate costs, not only direct cost related to, for example, damaged equipment, but also indirect cost as a consequence of the effect on customers, e.g. loss of business by the customer (again difficult to quantify). Therefore, it is very important to identify not only the number of customers affected, but also the type of customers. With the exception of emergency services, an outage that renders the network inaccessible to residential customers may not be as grave for the operator or service provider as an outage that denies service to business customers. The latter are the category that provides the high revenues to the providers, and therefore the most delicate to treat, since many business customers rely on good communications to conduct their business.

It becomes apparent that it is not sufficient to consider only traffic loss as the indicator of the outage magnitude The third measure described (ATIS index) presents a deeper insight into the problem and it is the most developed measure of network outage of the three considered. It offers a basis for a framework to quantify service outages that, carefully developed, could be used as a transportable and agreed definition and which permits expansion to encompass new services. It is relatively simple and easy to understand and operate, and could be adapted and defined for use in a European environment. However, the methods rely on a number of coarse assumptions and on broad averages. Considerable work is needed for example to obtain sensible and meaningful weighting functions that can be adapted to any environment.

One positive aspect of the ATIS index is that it introduces some degree of decomposition of the problem in a systematic way - identified here as a key requirement for an integrity

definition, as discussed in section 3.2.1. The top level is given by the expression in equation (3). All that exists at this top level are three factors, the three different weights -service, magnitude and duration weight. However, there is a whole process behind to obtain the actual values of these weights, where a set of different parameters take part - i.e. time of outage, duration, lines effected or blocked calls, etc. These parameters are not visible at the top level, but they contribute to the index at some point. This type of hierarchical decomposition is necessary when dealing with complex and diverse problems. The degree of decomposition in the ATIS T1A1 index is however very limited, and only a reduced number of factors are considered.

The ATIS index introduces some ability to discriminate between different scenarios, but this distinction is still rather crude. The methodology is heavily based on POTS, and there is no account for new or emerging technologies, such as IN or ISDN. The magnitude of the outages is calculated based on the lines affected or the number of block calls, i.e. it is a very call orientated approach.

There is some discrimination based on the type of services. Each type of service contributes, on its own, to the total index, i.e. a different factor is calculated for each type of service, to take part in the final summation. Nevertheless, the classification of services is very coarse and there is, in fact, very little distinction in the way different services account for the final index. A distinguishing attribute is the service weight, which is specific for each type of service, but the magnitude and duration weight functions are assumed to be independent of the service type. For the case of wireline fixed voice band telephony service four types of services are distinguished. The lowest service weight is for those services that are constrained to an area and within a certain administrative domain (i.e. IntraLATA Intraoffice). These are followed by two other types of services, which have been assigned the same service weight. These are services that span between different exchanges within the same geographical region (IntraLATA Interoffice), and services that cross different regions (InterLATA). A special case is the emergency services, which are given the highest priority. With the exception of the emergency service, the classification and weight assignment is mainly based on the geographical coverage of the services. This may be a sensible criteria for POTS, where the wider the coverage, the larger the number of sites that can be effected, but it is not the only possible criteria. Think, for example, about an outage that effects the communication services of high security bodies or critical political sites. We could argue that these services should be given a higher index than those services that only affect small business companies or residential customers. In other words, we can not assume that the impact of outages increases

with the size of the geographical area. Other factors, such us the type of service and customers involved, need to be taken into account - in the ATIS index, this is only done for the Emergency services. Furthermore, so far, we have only looked at the problem of categorising services from the customers point of view, i.e. how many and what type of customers are effected and where. However, there are other aspects that should be considered in order to categorise services. These include, for example, relationships and interactions of the service with the network elements and with other services, which may make the service critical if it fails, e.g. management services that control the performance of the network and other services. Thus, the characterisation of services for the purpose of analysing the effects of failure needs to be thoughtfully studied.

In the ATIS measure, some consideration is given to the architecture of the networks. Four industry segments are identified but this classification is still rather superficial, and more detail is needed to examine each of these four categories. For example, within the category of wireline telephony a number of different architectures and types of platforms for provision of services can be identified, e.g. network services, ISDN, IN, and the issues relevant to each of them need to be included in the analysis. The ability to embrace different environments becomes extremely important in the diverse and changing networks of nowadays.

The ATIS index represents a good starting point to provide a framework for characterisation of network outages. However, like the other two measures previously discussed, it presents the major disadvantage of being post-mortem measures. They are conceived to quantify the impact of outages and categorise them but they do not provide any means to prevent or at least reduce the probability outages occurring. A further development becomes necessary, and that is to obtain some measure to quantify *threat* of outage as a consequence of some action, e.g. due to a specific type of interconnect or internetworked service. Rather than a measure to quantify the impact of outages, we advocate a definition that can be used to identify the *risk* and magnitude of outage, so that measures can be taken to avoid or quickly correct failures. This risk management consists of two different cases: pre-emptive methods that permit the identification potential threats and allow critical situations to be avoided, and real time monitoring and restoration procedures to quickly resolve faults.

The quantification of the impact of failures can be regarded as a component of a pre-emptive integrity framework. There are several ways in which measures for outage characterisation may contribute to a definition of network integrity to be used as a pre-emptive tool. Recording data about occurring outages may be a very valuable source of information for future

reference. One way is to make use of recorded information about outages to trace back the failure process, identify the cause and build a framework to relate causes to effects. Relationships between causes and effects may not be easy to obtain in our complex environments. Assuming that some relationships can be found, they will be dependent on the context in which the failure occurred, i.e. the same cause may produce completely different results depending on the system and circumstances in which they occur. The measure should embrace these contextual parameters. Therefore, two types of factors must be included and measured, namely:

a) Post-event factors, i.e. measures related to the effects, such as duration, type of service, traffic loss, failure of network components, etc.

b) Pre-event factors, i.e. measures related to what was the state of the system and what was the context in which the outage occurred, e.g. signalling activity, information about usage, time of the day, etc.

The ATIS index incorporates some contextual information, i.e. the time of outage, which is embedded in the magnitude factor. However, this contextual component is very limited, and more factors must be incorporated in order to relate the fault to its causes.

Another possibility is to collect information about failures, accumulating data about past outages and study the trends. This approach presents two disadvantages. On the one hand, statistical analysis of trends in outages requires time, and can only be done after outages have happened. There is a need for a framework that can be used now. It is not sufficient having to wait for several years and outages to happen, to then use that data to predict the trends. On the other hand, it can be questioned if this would be a valid approach to the problem; i.e. can you predict the failure trends based on the previous experience, in such a changing environments as telecommunications? We are dealing with dynamic systems where big changes can occur in a considerable short time, i.e. the transition from basic POTS networks to complex IN systems. In such a quickly changing scenario, the data collected about outages occurred in previous years, or even months, may not be applicable to the present environment. This raises the question of whether such a technique to predict trends based in historic information can be valid in future network scenarios.

2.4. Conclusions

This chapter has provided an analysis of current and future trends in telecommunications, identifying the areas of high complexity that make the maintenance of network and service integrity a major issue. The evolution of telecommunications can be described by significant changes in three areas, namely customer demands, technology and regulations. The result from these changes are increasingly complex telecommunication networks and services, dramatically different to the networks of past years, and continuously evolving. Threat to integrity is high in these environments, and this chapter has provided an analysis of past experiences and of those areas where these threats are more likely to come from. The maintenance of the integrity of networks and systems becomes crucial and ever harder, and the need for more formalised and systematic approaches arises.

A review of concepts and techniques related to network integrity has also been included. It has been discussed that most of the related work focuses on the lower levels of the systems, i.e. the robustness of the transport network. Extensive work exists on survivalility techniques against link or node failures, or against traffic congestion levels, in order to guarantee that traffic is not lost. However, there is little understanding of the integrity problem at higher (intelligent) levels. Additionally, examples can be found in the literature of measures for the quantification of failure. These types of measures are used to identify the impact of failures in customers, but they do not assist with the prevention of failure.

This investigation work has identified two main gaps in the literature, namely:

a) the need for research on the integrity issues at the intelligence levels of modern telecommunication networks, and

b) the need for of pre-emptive measures that can be used to diagnose and avoid of failure in such networks.

These are the topic of this thesis. The remaining chapters contain the author's contribution to the problem and the proposed approaches to improve the integrity of modern intelligent networks.

Chapter 3

Proposed approach for the design and support of high integrity systems

3.1. Introduction

This chapter describes the proposed approaches for the design of high integrity systems, and the preservation of integrity of operating systems. These consist of three main contributions, as follows:

1. A network integrity definition to assist the identification of integrity levels and assist on the detection of integrity degradation.
2. An overall integrity framework comprising essential actions – both static and real time - for the preservation of network integrity.

3. A modelling framework for the use of modelling and simulation techniques to assist in the design of high integral systems.

The proposed integrity definition is described in section 3.2, the integrity framework in section 3.3 and the modelling framework in section 3.4. Next, two work alternatives identified in the project, their advantages and disadvantages, and the selected option are discussed in section 3.5. The chapter ends with the conclusions in section 3.6.

3.2. Proposed network integrity definition

As discussed in previous sections, here it is believed that there is a need to develop pre-emptive frameworks to maintain the integrity of the networks and services and reduce the risk of severe failure. A first step is the establishment of a definition of network integrity, identification of the requirements for an integrity measure and of the factors and areas that have an impact in the integrity of the networks. This is dealt with in the remaining of this section

An initial definition of network integrity was "the ability of a network to retain its specified attributes in terms of performance and functionality" [Ward 95]. This definition has been taken as the basic starting point for this work. A principal aspect of the definition is that integrity refers to the ability to maintain certain properties. When the term integrity is applied to people, it is understood that a person with high integrity is someone with strong believes and principles, who will maintain a clear position, and who is not easily bribed or corrupted. Similarly, a system with a high degree of integrity will be robust and secure against adverse conditions, and it will not be susceptible to external perturbations. Network integrity is then the ability of a network to maintain a safe state in which it is invulnerable to external perturbations. If the network is in a state of high integrity, these perturbations can be easily 'absorbed' by the system without having undue effect. Alternatively, if the network is in a state of low integrity, the same perturbations might have more serious consequences, and may even cause failure. Hence, network integrity is related to robustness, invulnerability and incorruptibility.

Another important aspect of the definition of network integrity is its connection with failure. The ultimate goal of preserving the integrity of a system is to avoid the occurrence of failure. Hence, there is a need to develop a formal and systematic methodology to assess the risk to network

integrity and build a framework to pre-empt failures in telecommunication networks and the services they support. The pre-emptive character of the integrity framework is a key matter, because it differentiates this work from other existing approaches to quantify the effects of failure, such as those described in chapter 3. Those methods are designed to quantify the extent of network failure after failure has occurred. Conversely, this work is motivated by the need to develop pre-emptive approaches to reduce the risk of failure.

The definition of failure differs for each system and is open to the criteria of the operators or service providers who will decide when a certain degradation of performance or functionality of the system is to be treated as failure. An extreme case is catastrophic failure, or complete system breakdown. But different categories of failure may be identified for each problem according to their effects, e.g. minor, medium, major and catastrophic failures.

The integrity of a network is not a fixed attribute, but it will vary depending on the conditions under which the network is operating. Changes to the degree of integrity may occur due to a large number of reasons, and identification of factors that may cause integrity degradation is an essential issue, and it has been addressed as part of this work. An initial analysis of key factors is presented in section 3.2.2. An integrity framework must be able to assist in the detection of this integrity degradation, so that appropriate action can be quickly taken. Here a framework is proposed in which the degree of integrity is divided in a collection of regions [Montón et al. 97]. A formalised measure of integrity must be used to identify in which integrity region the network is operating at any time. Such a framework to measure and categorise the degree of integrity of a network must:

a) discriminate between different states of integrity, or regions of risk, and permit the identification of the region in which the network is operating;

b) assist in the assessment of the probability of moving from one region of risk to another;

c) be formal and well defined to aid understanding.

The shape and size of these integrity regions must be defined for each system, and it is not a trivial exercise since they will be governed by complex relationships between different integrity parameters. Figure 9 represents a simplified case based on a one-dimensional definition of integrity, where the degree of integrity can vary between 0% and 100%. These two extremes are

only conceptual. A system operating with 100% integrity would be absolutely robust and would not be affected by any unexpected perturbations. At the opposite end, 0% integrity means that any perturbation would cause catastrophic failure. In between these two extremes, there is a collection of integrity bands. As we move towards the 100% integrity limit the robustness, stability and invulnerability of the network increases. The system may experience some minor failures while operating in high integrity regions, and these failures will become more severe as it approaches the 0% integrity region. A network operating near the 0% integrity limit would be in a completely unstable state where the risk of catastrophic failure would be very high.

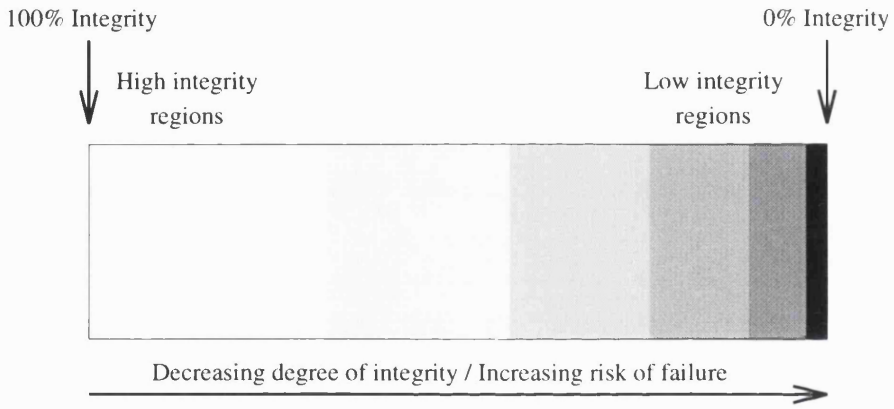


Figure 9. Integrity definition example

The definition of the integrity regions will be different for each particular system, i.e. each operator will define their own criteria. What is necessary is a general framework that is commonly understood and can be customised for different scenarios. These criteria can be variable, hence the boundaries between integrity regions are not be static; it must be possible to re-define them according to the current criteria. In the example illustrated in Figure 9 the size of the integrity bands decreases as the degree of integrity increases. This is only a simple assumption due to the fact that the risk of failure increases as the degree of integrity decreases. Making the integrity bands narrower in the low integrity regions allows for the detection of small integrity fluctuations. Conversely, in the high integrity regions, small changes in the degree of integrity will not have distinct effects, and therefore the integrity bands can be wider. This approach encourages a rational prioritisation in the allocation of resources and investment in the maintenance of the network integrity. If a network is operating in a high integrity band, the effort and cost of detecting

minor integrity deviations might be much bigger than their effects. On the other hand, the costs associated with the restoration of integrity, e.g. human resources, equipment, inaccessibility of services during repairing time, etc., will be larger if the network is operating with a very low degree of integrity. Hence the need to avoid large breach of network integrity in the low integrity regions.

The following activities are necessary for the construction of the proposed integrity framework:

a) specifying a definition of network integrity, examining potential causes of loss of network integrity and how this loss would manifest, i.e. what parameters must be considered as indicators of the degree of integrity of a network;

b) classification of the degree of integrity in regions, distinguishing between safe regions, regions of considerable danger, very dangerous regions, etc.; this involves the identification of threshold values for the integrity parameters that would indicate a significant loss of network integrity; it may not be possible to strictly define the borders between these integrity regions, but it should be possible to identify ranges within which the different parameters can vary.

c) use of monitoring systems to obtain the values of the integrity parameters at any time, in order to detect possible violations of thresholds

d) use of efficient documentation techniques to allow easy and rapid access to the information in order to permit quick actions when necessary.

Each of these activities is a key piece in a methodology to investigate integrity, and they will be expanded throughout this thesis.

3.2.1. Requirements of a network integrity definition

This section discusses the requirements that a network integrity definition must meet in order to assist in the assessment of risk and diagnosis of potential failure due to changes in the networks. Example of these changes are new interconnect, new service, changes in the way existing services are provided, or changes in the characteristics or the environment in which a service operates (such as increase in the number of users). Such integrity definition must be:

a) Flexible

The definition must be applicable to a wide diversity of the telecommunications scenarios, where a large variety of elements have to inter-work, e.g. interconnected networks of different operators, equipment and services from different providers, and multiple interacting and internetworked services. The definition must be flexible, allowing differentiation between common areas and specific situations.

b) Formalised

The definition must be *formal and well defined* so that it can be commonly understood, *and numerical or semi-quantitative*, to permit the introduction of comparative measurements. It must permit to discriminate between different regions of risk, place a network in one of these regions, and assess the probability of moving from one band to another. Rather than providing absolute values, it must permit the identification of an increase in risk, based on comparison between the measured values and the defined thresholds. Thus, it is necessary to define the thresholds that separate different regions and identify the critical values that imply high risk.

c) Structured

Because of the complexity and diversity of networks, the threats to integrity are large and varied. Hence, there is a need to decompose problems. This requires a hierarchical definition where the top level contains parameter groups, describing different aspects of the problem, which can be progressively decomposed into more detail in the lower levels. Relevant paths through the hierarchy can be followed for particular applications.

There is also a need to decompose the integrity of a whole system into different levels of integrity, i.e. unit level, integration level, system level, service level, etc. In addition, for each level, it should be possible to examine the integrity of various components, e.g. integrity of the data, integrity of signalling messages, etc.

d) Adaptable and expandable

To cater for the continuous and rapid change in telecommunications, the network integrity definition must be adaptable and expandable to permit the incorporation of new features. Adaptability and expandability are also required to provide for a dynamic and progressive model

development, and to allow upgrades to account for new parameters and relationships identified by "trial and error".

e) Focused on services

The evolving telecommunications market demands the rapid deployment of advanced and increasingly complex services that need to co-operate across different platforms. Loss of integrity may effect the various services differently and discrimination is required to enable risk to individual services to be separately evaluated. This requires an understanding of how services/features operate and interact with network entities and other services/features.

The following needs consideration:

a) internal behaviour of services, e.g. databases accessed, types of signalling transactions, concentrations of signalling activity, etc., to identify the critical points in the network for particular services;

b) interactions between:

   i) different services/features
   ii) services and network entities
   iii) different network entities;

c) usage characteristics, interactions of the users with the services;

d) performance of the systems, that can impose constrains and limitations to the service behaviour and even cause failure.

3.2.2. Key integrity factors

Network integrity is a relatively new topic and there is limited literature relating to the subject. However, it has parallels with many different areas, e.g. signalling, testing, management, etc., where there is expertise. But it is isolated and does not integrate easily into a coherent approach. Effort must be made to integrate these isolated branches of expertise under the umbrella requirements of integrity. The key of the problem is to find how each of these areas contribute to integrity and how they relate to one another. An analogy is to view integrity as a jigsaw puzzle. All the pieces are there, but it is necessary to find out how they fit together.

An initial stage must be the identification of the pieces that compose the jigsaw, i.e. the factors that are relevant to network integrity. This provides an initial decomposition of the problem into smaller manageable areas, essential due to the large complexity and diversity of the puzzle. Several of these areas have been identified through the analysis of current telecommunication environments and future trends included in chapter 2. A summary of these integrity areas is included next – a deeper analysis of some of these areas, e.g. signalling and interconnect, was included in chapter 2.

3.2.2.1. Population and user models

The behaviour and operation of a service is strongly effected by the way the service is used by the customers. These factors can be embraced in so called *customer models*. These customer models can be decomposed in two main areas, here referred to as user models and population models. *User models* represent how an individual user would interact with the service, e.g. what actions they are likely to take, possible errors they would make and how they would react in different situations. *Population models* represent the characteristics of the whole collection of users of the service. This includes for example number of customers, their geographical distribution, their degree of mobility, and their usage patterns e.g. call holding times, average and peak calling rates, etc. All these factors need to be carefully considered when a service is designed and implemented. A service may be conceived to have two or more implementations, e.g. a demo version, used when the service is being introduced to capture market share, and a more realistic version to be used as service popularity arises.

3.2.2.2. Quality of service

Quality of service (QoS) represents how the service is perceived by the customers. In a market environment such as telecommunications, where customer satisfaction is a primary objective, special care must be taken to understand what parameters specify customers' perception, and to maintain the appropriate values for them. A major problem is how to map the users QoS requirements into network parameters, and a great deal of work is being done in this area, e.g. the RACE project TOMQAT [RACE 2116 95], and the ACTS project MISA [ACTS AC080 96].

3.2.2.3. Performance

The behaviour of a service can be heavily influenced by network performance characteristics. Therefore, the study of possible erroneous behaviour must include the relationship to performance. The performance objectives of the design should also map on to the QoS requirements. Performance aspects are dealt with in detail in chapter 8.

3.2.2.4. Signalling

This factor refers to the general information flows within the system and the associated processing, as opposed to specific signalling protocols or architectures. Signalling is the area of most growth in complexity, and hence highest risk to integrity, in the new telecommunication scenarios. For this reason, it is a factor that must always be carefully analysed, and this is a key topic in this work.

3.2.2.5. Network congestion control

As the outages that occurred in the USA in 1991 showed, congestion in the networks may result in severe degradation of network integrity. The network management systems must be able to detect congestion and to avoid escalation of congestion. This is typically achieved by stopping traffic from entering the congested area, and re-routing calls from and to unaffected areas to circumnavigate the congested area. A number of issues arise regarding the congestion control mechanism, as discussed in [Chung 94], sect. II. Some of these issues are as follows:

a) The procedures to alleviate congestion require new management messages to be exchanged between the nodes; these management messages contribute to additional traffic, and are also subjected to the congestion conditions, which means that they may not reach their destinations, hence not being able to do their jobs. In addition, delays or loss of management messages may instigate the generation of further messages, hence worsening the congestion. The situation is aggravated by an increase in call re-attempts due to connections on first attempts failing.

b) Information about the areas of congestion, type of messages involved, etc. needs to be obtained as soon as possible to allow for speedy recovery of the system. This raises the need for powerful monitoring tools that make this information available at any time.

c) A major issue is high speed of recovery. This requires the design of robust systems that can recover effectively. This can be improved by careful analysis of potential failure situations,

automated systems that help tracing the source of errors and expert systems that help to quickly identify the appropriate actions for fast recovery. This will be discussed in more detail in chapter 4.

3.2.2.6. Interconnect

The new telecommunication environments are characterised by increasing interconnects between different network operators, and between service providers and network operators. New types of interconnect need to be established and the implications on the integrity and security of the interconnected networks must be accounted for. The integrity of a network may be in jeopardy due to things such as:

a) incompatibilities between the protocols and software used at either side of the interconnect point,

b) illegal messages coming from an interconnected network or service provider,

c) congestion due to increasing traffic across the interconnect point, etc.

3.3. Proposed integrity framework

The pre-emptive integrity definition proposed in the previous section is an essential part of this work, but it must not be regarded in isolation. It is necessary to understand what are the threats to integrity in the current and forthcoming telecommunication networks and services, and to identify the actions required in order to minimise these threats. Some key areas affecting network integrity were discussed in section 3.2. The required actions and how they fit into an overall framework are discussed in this section.

An important source of information for this project has been a study carried out on a major telecommunications operator in the UK. Information obtained from interviews with key personal in this Telecommunications Organisation (TO) was used to produce a coherent picture of the wide and disparate activities carried out within this organisation and investigate how to integrate these isolated branches under the umbrella requirements of integrity.

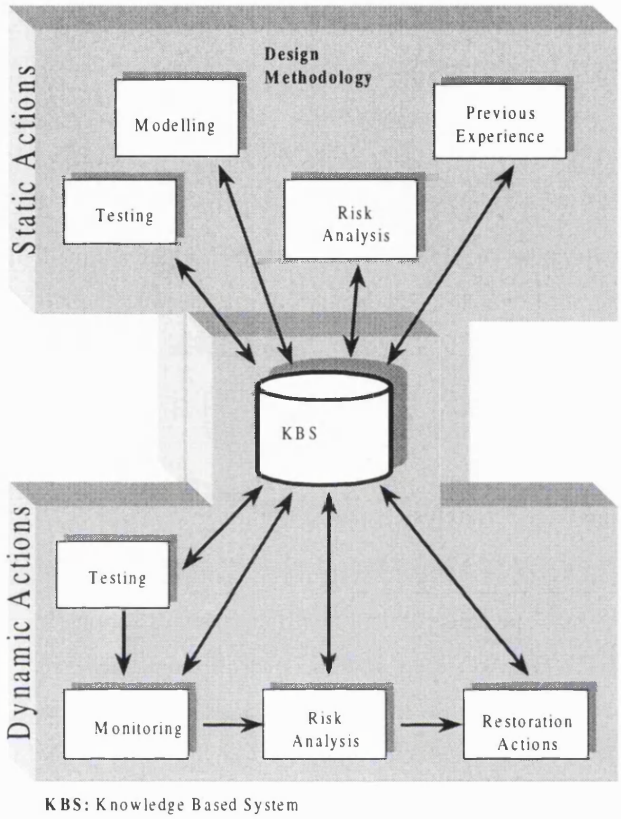


Figure 10. Static and dynamic actions that constitute the network integrity framework

The dramatic evolution that telecommunication networks and services have experienced during the last decades results in an ever increasing complexity, exacerbated by regulatory policies that foster interconnection and competition. In such a changing and increasingly complex environment, the traditional working practises need to be revised. The historical manual approach to the maintenance of network integrity that relies on the knowledge of individual experts is no longer valid. New paradigms are required, involving more formal and structured approaches and better automated documentation techniques.

A methodology to control the degree of integrity of a network requires the use of various disciplines and must be applied coherently through the whole development cycles of the services operating in the network. The methodology framework proposed here is divided into two

categories, namely static and dynamic actions. Static actions are previous to the launch of a service, i.e. actions taken during the design and implementation stage. Dynamic actions are those actions taken in real time, i.e. while the services are in operation.

Figure 10 illustrates the overall framework showing the relationships between static and dynamic actions. Static actions involve improvement of the methodologies for the design and implementation of services. The five main areas identified here are modelling, risk analysis, use of previous experience, testing and documentation of information (the KBS in Figure 10). Dynamic actions include monitoring, risk analysis, testing, restoration actions and documentation of information. The framework is described in detail in the remainder of this section.

3.3.1. Knowledge Based Systems

A Knowledge Based System or Expert System is "a software system that represents and reasons with knowledge in some specialist domain, with the view to solving problems or giving advice" [González and Danke 93]. An expert system is an artificial intelligence application that uses a knowledge base of human expertise to aid in solving problems. The degree of problem solving is based on the quality of the data and rules obtained from the human expert.

A knowledge-based system contains a knowledge base, or collection of data, and a set of algorithms or rules that infer new facts from knowledge and from incoming data. The acquisition of knowledge involves actions such as:

a) interviewing the human expert,
b) collecting facts (data) and rules (heuristics), and
c) building a knowledge map, i.e. create objects and object hierarchies, and group rules into contexts.

The expert systems derives it rules by running the knowledge base through an inference engine, a software program that interacts with the user and processes the results from the rules and data in the knowledge base.

If and expert system is to be used, the technical requirements of such a system, such as time response, capacity, data structure, knowledge paradigm, etc. must be carefully analysed and

chosen. Human issues such as how easy it is to use, how long it would take to train people to use it and more subtle issues related to the change in mentality and habit required need also be addressed. Based on this analysis a choice has to be made between buying one of the systems available in the market or building one of their own. The latter option involves more time and requires people who are experts in the field of knowledge based systems, but it would permit to design a system tailored to the needs.

Expert systems are used in a large number of applications such as medical diagnosis, equipment repair, investment analysis and production control and training. There is a strong research activity in the area. Current migration trends are from stand-alone expert system applications to the integration of expert systems, i. e. of knowledge and inference, into conventional software applications. Here it is believed that they have a big role to play in the area of telecommunications systems, as explained in the subsequent sections in this chapter. For an introduction to knowledge based systems the user may refer to [González and Danke 93] and [Harris-Jones 95].

3.3.2. Static actions

This part of the methodology includes those activities identified as essential for the improvement of service design and development. Figure 10 shows the five main components of the static part of the framework. At the core of the methodology, there is the need for efficient and powerful information systems, represented in Figure 10 by a Knowledge Based System (KBS). The other four main activities are modelling, testing, previous experiences and risk analysis. Results from these activities are fed into the KBS. The information contained in the KBS can then be used as inputs in future modelling, testing and risk analysis activities.

3.3.2.1. Design cycles

Systems that are robust to loss of integrity require design methodologies and principles that take account of factors effecting integrity. An integrated approach is required where all the activities involved in the design and specification of new systems are carried out in a coherent way.

A study of current practices identified a number of issues that need to be reconsidered, introduced or modified in the way services are designed at present. Essentially, there is an increasing need for

more formal approaches and better automated documentation techniques to facilitate handling and communication of information. In addition, new service provision paradigms require modelling techniques to assist understanding and identification of problems early in the design cycles. At each decision making stage, careful risk analysis must be carried out, and known limitations must be documented in order to prevent future errors. Models will also assist in the identification of areas for testing, and risk analysis must be undertaken due to the impossibility of testing everything exhaustively.

Figure 11 shows the different stages involved in the development of a system, namely analysis, design, implementation and testing. The arrows in the diagram represent the feedback loops between the different phases. Ideally, a thorough and complete analysis followed by a correct design would lead to the implementation of a system that works as expected. In reality, it is virtually impossible to implement a system that works perfectly at the first attempt; hence the importance of testing [Bale 95]. In Figure 11 testing has been included as a separate activity at the end of the development cycle for convenience, but in fact, it spans along all the steps of the development cycle.

Errors may occur at any stage, and their detection may require going back to previous phases and introducing modifications. Feedback between the separate phases is inevitable, but a good methodology should help to reduce the long jumps, e.g. to avoid modifying the analysis or the design after the implementation phase has been completed. It is very important for the development of an efficient design methodology to have available information concerning where and how things can go wrong. In order to do this, every time a new system or service is implemented, information from the cycle described above must be extracted and documented in an efficient way to make it available for future applications, e.g. some form of a knowledge based system.

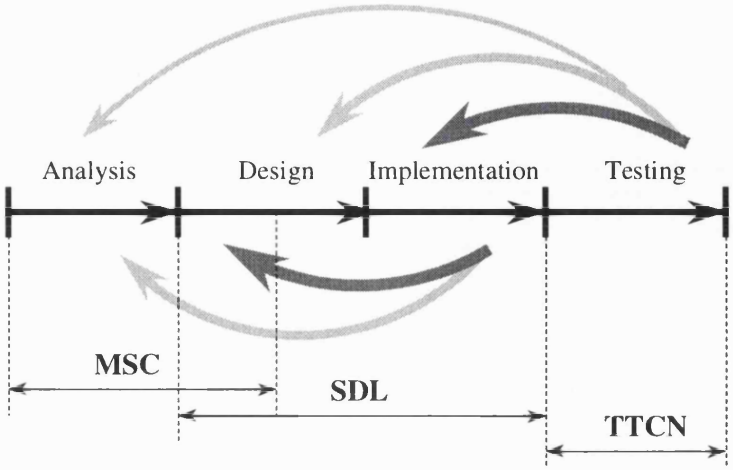


Figure 11. The design cycle and some languages used at each stage

Figure 11 shows the design cycle for a single system, in our case a telecom service, and represents the fact that services have traditionally been designed in isolation. This practice has proven to be a reason for failures because, in practice, services have to interwork and share resources in the network. Compatibility and correct interworking between services can not be guaranteed if they have been designed in isolation, hence the feature interaction problems (see section 2.3.4). Therefore, it is of vital importance to analyse the services within the context of their environments, i.e. multi-service and multi-network scenarios.

It is probably widely accepted that the key for the success of a system is a thorough system specification and design. The specification methods employed by operators and other organisations such as the ITU have for many years relied on informal techniques such as English descriptions and message sequence charts. While these methods have been accepted in the past, the problems that can arise from such a free format are beginning to be understood. The difficulties of specifying OSI (Open Systems Interconnection) standards in natural language were recognised early, motivating CCITT (now ITU-T) and ISO to work on international standards for Formal Description Techniques (FDTs). This led to the development of the languages SDL, Estelle and LOTOS [Turner 93] for producing formal specifications that are unambiguous, clear, complete, and consistent. FDTs are also intended to help in analysis, implementation and testing.

In [Belina and Hogrefe 89], Belina and Hogrefe enumerate the benefits of formal specification languages:

a) a well defined set of concepts;
b) unambiguous, clear, precise and concise specifications;
c) a basis for verifying specifications with respect to completeness and correctness;
d) a basis for determining whether or not an implementation conforms to the specifications;
e) a basis for determining the consistency of specifications relative to each other;
f) use of computer-based tools to create, maintain, verify, simulate and validate specifications;
g) computer support for generating applications without the need of the traditional coding phase.

Given the importance and increasing use of formal languages, it was decided to study their application to network integrity as part of this work. In particular, three techniques were explored namely SDL [Olsen et al. 94], MSC [ITU-T Z.120 94] and TTCN [Probert and Monkewich 92]. Each of these languages is conceived for use at different stages in the development process as illustrated in Figure 10. They were chosen for a number of reasons. The first one is that are currently within the most widely used languages in the telecommunication industries and research communities. Another reason is that they are international standards, SDL and MSC have been specified by ITU, and TTCN by ISO, which makes them well defined and referenced languages. This has the advantage that they are under ongoing investigation and continuous evolution to incorporate new needs. Also, there are commercial software available to build and test systems based on these languages and that integrate the three of them. As this work focuses on design methodologies, only SDL and MSC were employed. A brief description of SDL and MSC is included next. A more detailed description of SDL is included in appendix D.

SDL (Specification and Description Language)

SDL is the Specification and Description Language standardised by the former CCITT (International Telegraph and Telephone Consultative Committee), currently ITU-T [ITU-T Z.100]. The development of SDL began in 1972 after a period of investigations. The first version of the language was issued in 1976. The first CCITT Recommendation (Z.100 Recommendation) was in 1980, followed by updates in 1984, 1988, 1992 and 1996. During this period, the language has experienced considerable evolution. Object oriented features were included in 1992 (SDL-92), and

in 1996, a few updates were made to the language in an addendum to the standard. Current SDL is now defined in the 1992 Z.100 standard with the 1996 addendum.

For systems engineering SDL is frequently used in combination with other languages, such as OMT (Object Modelling Technique) [Rumbaugh et al. 91], MSC (Message Sequence Charts), ASN.1 (Abstract Syntax Notation) and TTCN (Tree and Tabular Combined Notation). Some of these languages have also been studied within the same group within the ITU. The ITU Z.105 standard defines the use of SDL with ASN.1, and the Z.120 standard defines MSC. There has been work relating the SDL and TTCN semantic models, and the TTCN is used for testing standards and systems written using SDL. The use of OMT object model notation in conjunction with MSC and SDL is a powerful combination that covers most aspects of system engineering.

One of the advantages of SDL is the availability of commercial software tools for the specification and design of systems using SDL, and simulation of these systems, hence aiding in the validation process. These software tools have also experienced considerable advance within the past few years, incorporating the upgrades in the language and the integration with some of the other languages mentioned above.

Today, SDL is widely used in the telecommunications field, but it also being applied to a diverse number of other areas ranging over aircraft, train control, medical and packaging systems. In general, the suitability of the language is for real-time, stimulus-response systems.

SDL has two concrete types of syntax, a graphic representation - SDL/GR- and a textual representation - SDL/PR. The graphic form is the most widely used, because it is more intuitive and displays relationships more clearly than the textual form, which looks like a programming language. Both types of syntax can be used in conjunction, embedding fragments of the textual form in the graphic form where text is more suitable, e.g. for representation of data and operations.

SDL is based on structural decomposition of the systems. The top level is the *system* level, and this is decomposed in a top-down fashion into *blocks*, which in its turn can be broken down in *processes*. Figure 12 shows an example of how this structural decomposition is achieved. Blocks/processes communicate via the exchange of signals, carried by channels/signal routes. This structural decomposition of the system into blocks and processes is primarily a logical

decomposition made in order to show the abstract behaviour of the system, and therefore does not necessarily correspond to a physical decomposition of the real system.

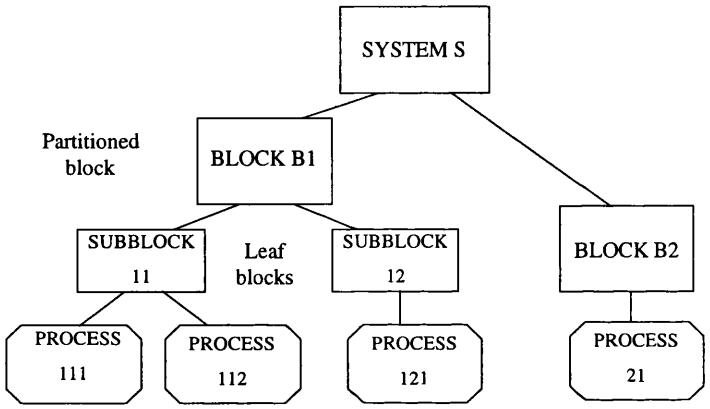


Figure 12. Structuring of a system into blocks, sub-blocks and processes

The problem domain in SDL is reactive behaviour. The behaviour of a system is constituted by the behaviour of a number of processes in the system. In SDL, each process is an extended Finite State Machine that works autonomously and concurrently with other processes.

Processes co-operate asynchronously by exchanging discrete messages called signals. Processes can also send and receive signals from the environment. In SDL, each system is viewed as a black box that exchanges messages with the environment. Information and details about the environment are not included. It is assumed that the environment acts in an SDL-like fashion, and must obey the constraints given by the system description.

Since a formal specification has to be as implementation independent as possible, specification languages do not cover performance issues. However, if SDL is to be used for simulation of networks and services in the context of network integrity, performance characteristics are essential, and ways to incorporate performance information into SDL models must be found. This issue is dealt with in chapter 8.

MSC (Message Sequence Charts)

MSC is a graphical and textual language for the description and specification of the interactions between system components. The main area of application for Message Sequence Charts is as an overview specification of the communication behaviour of real-time systems, in particular telecommunication switching systems. MSCs are a means of visualising selected system runs (traces) within communication systems. Message Sequence Charts may be used for requirement specification, simulation and validation, test-case specification and documentation of real-time systems. They have been informally used for a long time, and in 1990, their standardisation was agreed by ITU in the language called MSC. This standard was approved in 1992 in recommendation Z.120 [ITU-T Z.120]. Since 1992, recommendation Z.120 has been updated with enhancements and modifications to MSC. The current version is known as MSC'96.

MSC can be viewed as a special trace language which mainly concentrates on message interchanging by communicating entities such as SDL services, processes, blocks and their environment. Like SDL, MSC has two syntactical forms, a pure textual one MSC/PR and a graphical one MSC/GR. A main advantage of MSC is that its graphical notation, in the form of graphical layouts, provides a very intuitive understanding of the described system behaviour.

An example of an MSC is depicted in Figure 13, showing three processes in an SDL defined system plus the environment, and the flow of messages between them. The vertical lines represent the passage of time of time, the hexagons symbols represent a state in the processes, and the arrow headed lines are the messages exchanged between the processes – signals, in the SDL notation. Each signal has an associated name and can carry some parameters.

ITU recommendation Z.120 contains the syntax and informal explanations of the semantics. One of the goals of the process of standardisation is the definition of a formal semantics of the language, using formal algebra. For more information about the formalisation of MSC, the reader can refer to [Mauw and Reniers 94].

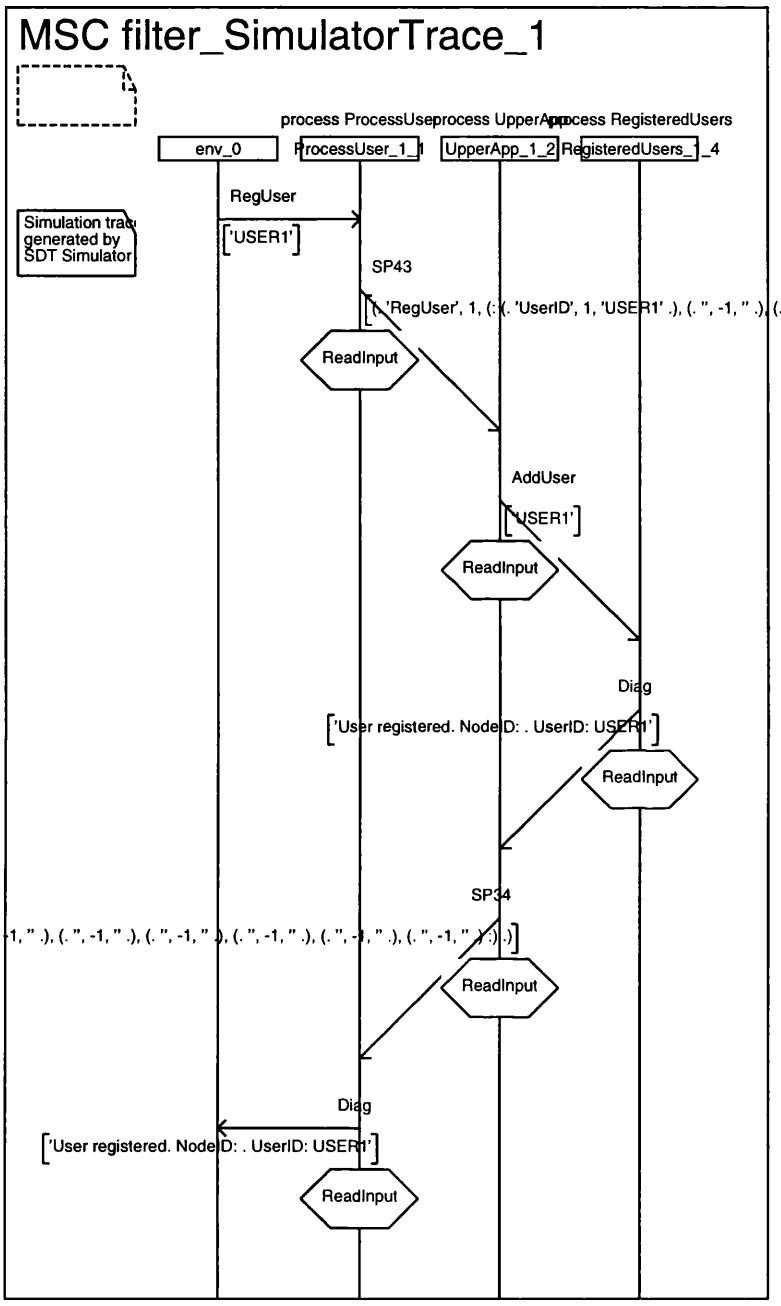


Figure 13. Example of a Message Sequence Chart

MSCs are often used in conjunction with SDL, since both languages complement one another in many respects. Whereas SDL provides a comprehensive description within individual processes,

MSC focus on the communication behaviour of system components and their environment by means of message exchange. One MSC represents only one scenario of the problem, a selected system trace. Hence, a set of specified MSCs covers partial system behaviour only. In contrast, SDL describes the overall system behaviour.

3.3.2.2. Modelling

Telecommunication services are becoming increasingly advanced and complex. New technologies and paradigms, such as intelligent networks, permit the fast provision of services and a large increase in the number of different services offered. Consequently, the comprehension and understanding of the behaviour and operation of services by single individuals becomes unfeasible. In this complex scenario, the role of modelling and simulation is of paramount importance. Modelling and simulation techniques present the following advantages:

a) Reduction in costs, i.e. the cost of the resources required for modelling are considerably lower than those needed to replicate the real network.

b) An adequate modelling exercise helps to detect and resolve problems before the launching of the services, when those errors could jeopardise the integrity of the real network.

c) Telecommunication markets are moving towards new services where there is no expertise. There is a general lack of understanding, and this can be provided by appropriate modelling. It is worth remarking that the value of a model is not only the results it produces, but also the expertise and knowledge acquired during the construction of the models themselves.

Modelling activities are considered here an essential component in the proposed overall integrity framework, and the rest of the work focuses on this area. The topic is discussed in depth in subsequent chapters.

3.3.2.3. Testing

Service testing

The design cycle of a new service involves a number of stages, from the feasibility study to the detailed design. At each stage, appropriate validation and verification of the obtained results must be accomplished. Then, different levels of testing of the product must be carried out.

Besides the testing against specifications, it is necessary to carry out end-to-end testing of services in the way the customers would use them. Many of the intranetwork and internetwork problems are in fact identified by testing, and hence the specification of test scripts is a vital activity. Clearly, it is not possible or desirable from a cost and time perspective, to test every combination of circumstances. The problem is exacerbated by the increasing variety of systems and equipment combinations, the complexity of the services and the fact that the development cycles tend to be very different between different services, which makes it difficult to develop general testing scripts. There is therefore a need for risk analysis to determine the costs/benefits of how much testing to carry out (see section 3.3.2.4)

There are two traditional approaches to testing. The first approach consists of trying to identify problems as early as possible in the development cycles. The second approach concentrates on time to market to gain market share, launch the services as soon as possible, and fix the problems afterwards. Hence there is a trade-off between the advantages of end-to-end testing to minimise in-service problems at increasing cost with time, against the delay to launching the services, as illustrated in Figure 14.

Due to the increasing complexity of the telecommunication systems, it is not possible to test everything, and the number of test cases generated has to be reduced to a manageable number. An important issue in the construction of test scripts is to identify those areas of greatest complexity where problems are more likely to arise. At present, the exercise of producing adequate test suites is performed manually. The selection of test suite structures is done by the test engineer who has a deep knowledge and understanding of the system being tested. There is a strong requirement for experience and imagination from the people designing the tests to determine aspects that are most likely to reveal problems. Although this human component will always be necessary, there is a need to develop more automated techniques and procedures to capture the expertise of these individuals and make it widely available [Woollard 93]. The study carried out in a major UK operator revealed the use of automated testers, which can reproduce testing sequences automatically over a period of time. This type of testers are useful to identify those type of problems which only occur once in many times, and can not easily be found through manual testing. However, the specification of the testing sequences that the tester must perform remains a complex task, still left to the human expert. As networks and services sophistication increases, the single expert who has a global knowledge of the whole system is replaced by many experts, each

specialising in specific parts of the problem. Expertise is fragmented and this distributed knowledge must be unified in a coherent way that can be accessible and useful to a variety of people. Hence the need for powerful and efficient documentation techniques.

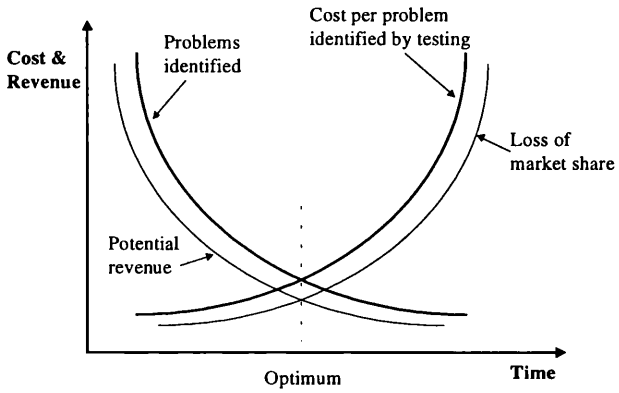


Figure 14. Testing compromise

Systems testing. The USA experience.

The increased heterogeneity within modern telecommunication networks stems from the incorporation of new network technologies and multi-supplier systems in a multi-network environment, supporting an ever-increasing range of services. For the assurance of network integrity in such a heterogeneous arena, the approach adopted by the US telecommunication industry, and most notably Bellcore, has been to undertake considerable testing analysis to achieve seamless operation. Testing programs have been organised which encompass stand-alone technical analysis and auditing of individual systems, multi-supplier interoperability, as well as internetwork interoperability. Details about the USA test environment can be found in [ONP-UCL 94]. A programme deserving special mention is the Internetwork Interoperability Test Plan (IITP) [Ward 95] [Lewin and Branflick 94]. The IITP is an industry-wide group focussing on interoperability testing for assuring network integrity of the interconnected CCS networks. The objective of the IITP is to identify and address consequential internetwork Common Channel Signalling (CCS) interoperability issues. IITP allows for early detection of interoperability problems between interconnected CCS networks, and it is widely accepted as being essential to maintaining national CCS network integrity. The identification and resolution of internetwork interoperability problems in a test environment makes it less likely that significant problems will occur in the live network environment. In fact, over 80 issues were discovered in a recent IITP test phase.

Here it is believed that co-operation between competing operators, service providers and suppliers to build a common ground for testing and sharing information and experiences is a key step to identify and prevent integrity problems related to interconnect, and that the creation in Europe of programs such as the IITP should be pursued, as recommended to the European Commission in [ONP-UCL 94].

3.3.2.4. Static risk analysis

Risk analysis must be integrated in the development process, in order to understand the limitations imposed in the specification, design, implementation and testing of systems and the risk associated to these limitations. Building a system that is absolutely robust and designed to work under any circumstances is not feasible for two reasons. On the one hand, conscious limitations must be imposed due to the trade off between the robustness and flexibility of a system and the associated cost of building it. On the other hand, it is not possible to predict all the conditions under which the system will have to work in the future, particularly in the rapidly changing telecommunication environments. Even if a system is designed to work for a wide spectrum of conditions, unexpected situations may rise in the future for which the system has not been designed. Understanding the potential consequences of both conscious and unexpected limitations becomes of paramount importance. For this reason, a risk assessment methodology that helps to assess the probability, impact and consequences of actions must be developed and integrated in the development of systems.

Risks also exist in the testing arena. As stated in section 3.3.2.3, the high complexity of telecommunication systems and services impedes to test everything. Testing must focus in those areas where problems are more likely to arise, whereas other aspects that can be treated as low risk and will not be included in the testing. The level of testing carried out must also be subjected to risk assessment, and the testing limitations must be understood and documented, together with foreseeable consequences of these limitations. Risk analysis should be applied at the different levels of systems development, in order to determine the amount of test that needs to be performed for different levels of risk. Increasing the amount of testing decreases the risks, but increases the costs and delays. These risks should be analysed, categorised and documented.

A great part of the risk assessment activity would consist in keeping records of any constraints decisions taken during the development of the systems, and an assessment of the effects that these limitations would have in adverse conditions. Such an activity would:

a) provide a better understanding of the systems operations;
b) make the information accessible to a variety of people, other than those involved in the design process;
c) help to identify weak points in the systems and where problems are likely to arise;
d) help in the diagnosis of failure and to a more rapid identification of the actions required to fix the problems;
e) help to evaluate consequences of failure, for example, in terms of lost calls, damage to equipment, cost, etc.

The development of an efficient risk assessment methodology relies in the use of efficient documentation techniques and powerful information tools (e.g. KBS).

3.3.2.5. Documentation and information processing

The traditional approach to service design has been to design new services in isolation, without taking into account the operating environment. This has worked well in the past in the public network because the number or services supported were very limited. Conversely, current and forthcoming telecommunications are characterised by the fast provision of an increasing number of emerging services. New technologies and architectures are being developed, e.g. Intelligent Networks, TINA, TMN, etc. to respond to the demand for new and more advanced services. The increasing number of available services that have to co-exist in the networks raises the problem of undesired service or feature interactions (section 2.2.2.4). As the number of services and interactions between them increases, the need to design services taking into consideration other services already in operation becomes apparent. This requires an integrated approach where an essential component is the availability of data and information about the different existing services. It is believed that Knowledge Based Systems can play an important role in this area, and this is represented as KBS (knowledge base systems) in the diagram of Figure 10.

Another characteristic of historic approaches to service design is the strong reliance on human expertise and imagination. Areas such as the design of test scripts and the identification of

potential problem scenarios rely heavily on the expertise, imagination and ability of the people who have been working with the systems for a considerable amount of time. This reliance on single experts presents two problems. On the one hand, the increasing complexity and large amount of information necessary to understand new telecommunication systems require an expertise base well beyond the comprehension of single individual. On the other hand, a major issue arises when these experts leave their jobs, taking all their knowledge and expertise with them. Thus, the traditional manual approach to the maintenance of network integrity, which relies on the knowledge of individual experts, is no longer valid. Hence the need to develop efficient electronic documentation techniques and mechanisms to store and process information in an formalised and automated way to make it accessible, meaningful and useful to different people. The human factor will always be necessary, i.e. it will always be necessary to have people who understand the systems. However, it would be convenient to reduce the dependency on the expertise of these individuals and find a way to merge the expertise of people in different areas, gather their knowledge and document it in an intelligent way to make it available to other people. We believe that knowledge based systems will play a key role in this area.

3.3.3. Dynamic actions

This part of the methodology refers to those activities that must be undertaken in 'live' systems while services are in operation, in order to ensure that the appropriate integrity levels are maintained. Figure 10 shows that, as in the static part, a key component of the framework is a KBS. Testing in this context includes testing of the monitoring systems themselves, hence the link between the testing and monitoring boxes in Figure 10. The other essential activities are monitoring, in order to detect violations of integrity levels; risk analysis, to assist in the decisions about the actions to take when problems are detected; and finally the restoration mechanisms to solve the problems. Information and results from these activities are fed into the KBS. This information can be very valuable when new tests have to be designed in the future, and as an input to the risk assessment in order to decide about the appropriate restoration actions to be taken.

3.3.3.1. Monitoring and control

Monitoring and control of the integrity parameters is required in order to identify in what integrity region the network is operating at any time. This will permit the assessment of the network to

determine whether it is in a safe state or, conversely, in a situation of high risk, where unexpected actions or changes, such as the introduction of a new service or network interconnect, incurs a high probability of catastrophic failure. The integrity framework must assist in predicting the loss of integrity due to such actions and providing a criteria to take appropriate measures, e.g. not to allow the interconnect in the terms proposed.

Powerful monitoring systems that check specified parameters are required. Some sophisticated monitoring tools are available in the market. In the area of signalling, for example, there are commercial tools that permit checking of all the incoming and outgoing messages in the STPs (Signalling Transfer Points), and provide statistics about types of messages in the link, link failures, etc. They also permit the analysis of the contents of the signalling messages in the links, giving complete information about types of message, dialled numbers, content of the message fields, etc. and allow the definition of thresholds and introduction of alarms when thresholds are exceeded. As in the testing arena, an essential issue is to decide what things must be monitored, and how to extract useful information from the vast amounts of data that these monitoring systems can produce. Again, an information processing problem. Finally, the monitoring systems are also exposed to potential integrity degradations, and hence the need to subject them to appropriate testing.

3.3.3.2. Restoration mechanisms

Due to the high complexity of telecommunication networks, the possibility of failure cannot be completely eliminated, even with a rigorous design methodology and exhaustive testing. Hence, another essential component of dynamic actions is the selection of efficient recovery mechanisms to restore the functionality of the networks when integrity degradation is detected. This requires a good knowledge of the system behaviour together with information that helps to diagnose the cause of failure. Monitoring the integrity parameters in real time will permit rapid detection of threshold violation. Information about possible error states and their probabilities can be used to rapidly identify the source of failure and trigger the pertinent actions. These actions will depend on the criteria set by "operators", depending on the band of integrity in which the system is operating.

The design and selection of appropriate restoration mechanisms requires:

1. Monitoring the parameters that can be considered as indicators of the degree of integrity, in order to quickly detect threshold violations that indicate a degradation of the integrity levels (i.e. monitoring and control, section 3.3.3.1).

2. To build criteria for taking actions, depending on the degree of loss of integrity. This includes identification of different regions of risk, and definition of the threshold values of the measures that define these regions. This requires the development of some risk analysis methodology to assess the severity of the failures (i.e. dynamic risk analysis, section 3.3.3.3).

3. To use information from previous experiences to assist in deciding what actions should be taken according to the type of failure detected. This information from previous experiences must be maintained in an information system (i.e. use of a knowledge-based system, section 3.3.1), that must be updated every time new situations arise and new criteria or restoration actions are defined.

A methodology for identifying immediate actions that should be taken in the event of SS7 failure is proposed in the EURESCOM Project 307 Task 3: Methods for Handling and Prevention of Failures in Signalling System No. 7 [EURESCOM 307]. The objective of this project considering SS No. 7 failure reports was to give methods and guidelines to European PNOs on how to avoid SS No. 7 failures in the future. The flowchart illustrating the proposed methodology is reproduced in Figure 15.

The proposed methodology is a good starting point towards an efficient recovery methodology. It is however a very high level description, and there are various points that require further consideration. In particular:

a) How is the seriousness of a failure assessed? In [EURESCOM 307] it is suggested that the seriousness of a failure is related to the size of the failure, in terms of number of customers effected, how long the failure could continue for, which services are effected, etc. This approach is rather simplistic, and this is in fact a point that requires much more analysis. A large collection of factors needs to be considered to assess the severity of a failure. For example, the number of customers effected is not indicative on its own, the type of customers that are effected is also important, e.g. residential or business customers. The analysis of seriousness or severity of a failure must be done following a decomposition approach,

dividing the problem into areas or categories and applying severity criteria to each of them. This task links to the risk analysis activity proposed.

b) An essential piece in the EURESCOM proposed methodology is an information system containing records of previous failure experiences. This relates to the expert system in the integrity framework proposed here.

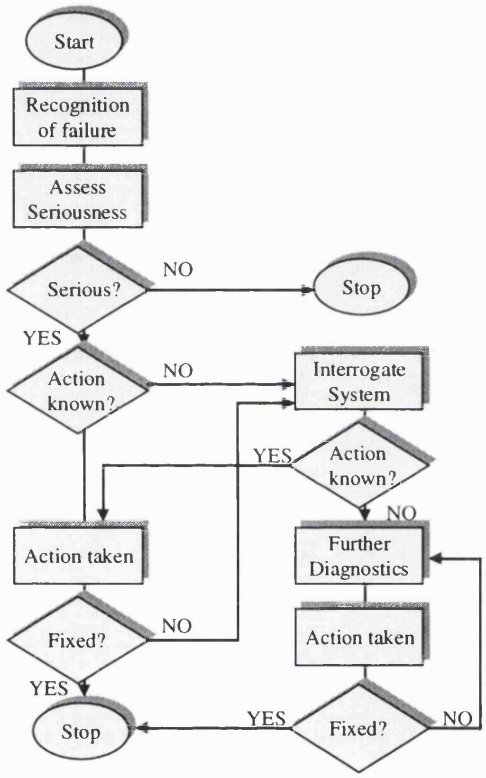


Figure 15. Example of a flow-chart for identifying what immediate action should be taken in case of failure

One of the main obstacles mentioned in the EURESCOM report is the difficulty in obtaining information from operators and suppliers as regards failures. Two of the points underlined in this report are as follows:

1. The relevance of analysing information from US operators, because of US operator's experience and confidence in planning, implementing and operating their SS No. 7 networks and their "open" reporting environment.

2. The difficulties in obtaining information from European operators and manufactures, because they are possibly worried that they could show defects in the hardware and software provided by suppliers and/or errors in planning, operating and maintaining their SS No. 7 networks. The importance of sharing information about failures and outages among PNOs and between PNOs and hardware and/or software suppliers is outlined as an essential way of improving network effectiveness and service quality. This is in accordance with the views defended here, as stated in section 3.3.2.3. As proposed in [EURESCOM 307], in order to grant confidentiality, reports could be made through mediating bodies (standardisation organisations like ITU an ETSI or EURESCOM are suggested).

3.3.3.3. Dynamic risk analysis

Risk analysis is also necessary during the operation of the services, in order to assess the severity of detected integrity violations and to decide the appropriate restoration policy. For high integrity violations, the recovery actions might involve a complete reset of the system, whereas minor integrity deviations might be acceptable if the cost associated in restoring integrity exceed the cost due to the integrity loss. These decisions must be taken based on a rigorous understanding of the risks taken, and hence the need of risk dynamic analysis.

3.3.3.4. Testing

Testing is necessary due to the changing nature of the networks and services. As described in section 3.3.2.3, testing must be undertaken prior to the launch of a new service to detect potential failures and feature interactions, and when new equipment is to be introduced in the network. This type of testing has been included in the static part of the methodology, because it occurs before the new service or piece of equipment is introduced in the real network. Dynamic testing is also necessary to ensure the correct functioning of the live systems.

Testing is essential to assure that a high level of network integrity is maintained under the introduction of changes in the networks, e.g. introduction of new equipment, new product features, new network services, new architectures, etc. In particular, testing plays a major role between interconnected networks. Agreed industry documents that dictate the terms of this interconnect testing must exist. In past years, when the telecommunications market was dominated by large operators, and these dominating organisations specified the interconnect testing scripts and

conditions to allow interconnect. As the telecommunications market becomes increasingly open to competing organisations, the role of the dominant operator becomes less clear, and hence interconnect testing agreements will need to be produced by neutral bodies or by consensus between the affected operators or service providers, possibly with mediation of neutral regulatory bodies.

Interoperability testing across multi-product and multi-network environments becomes very difficult to achieve, due to technical, regulatory and time-constrained reasons. When changes are introduced in the network, a first step is usually supplier stand-alone testing and lab testing (using network provider and/or supplier lab facilities) to reflect the unique environment for the changed network. Interoperability testing is then typically done in a non-intrusive mode, as a final verification of the changes environment. Live network testing, although more realistic, is difficult and risky, because it is limited to "sunny-day" scenarios, i.e. to test the ability of the system to perform under normal conditions, and might jeopardise "live" calls. Provocative testing, where the normal conditions of operation are deliberately exceeded to observe what causes failure, will also need to be done in an off-line or lab environment. Even in this case such type of testing is extremely difficult due to the large number of network and supplier products interconnecting to the changing network element. However, provocative testing is a necessity in order to achieve some level of assurance that interoperability across products or network is not affected by changes to existing products.

3.3.3.5. Documentation and information processing

The different types of information involved in the dynamic part of the integrity methodology have been discussed in previous sections. As discussed in previous sections, a key piece in the dynamic actions part of the integrity framework is the efficient documentation of relevant information obtained from different activities. This information must be used to increase the knowledge base of the expert system, and it includes:

1. Integrity parameters and threshold values that define different regions of risk.
2. Information about possible restoration mechanisms for particular failure conditions and the associated risk of each the possible options.
3. Information from previous experience in integrity violations.
4. Testing results.

3.4. Proposed modelling framework

The integrity framework discussed in the previous section is a very large and complex framework proposed as a guide for telecommunication organisations, highlighting key areas and activities. For the purpose of this project, the scope needed to be narrowed down, and hence it was decided to concentrate in one of the components of the framework. Thus, from all the activities involved this project focuses on the predictive modelling part. There are two main reasons for this choice. The first one is that modelling is the most feasible activity to understand the systems and their potential behaviour with the resources available in the academic environment where this work was developed. Certain activities, such as monitoring or documenting previous experiences, require access to the real network systems and data, which in many cases may be regarded as confidential. The other reason for focusing the work on modelling is that, due to the theoretical, analytical and long-term nature of the work, these activities are better suited for academic environments than for industrial organisations, where practical, shorter-term activities have a higher priority.

The remainder of the work focuses in the investigation of the use of modelling and simulation techniques to identify integrity threats and enhance the integrity of networks and services. It is believed here that modelling has a key role to play in commercial telecommunication organisations, and this becomes imperative with the dramatic increase in the complexity of modern services and networks. An additional advantage of modelling arises because building a model is actually as useful as its possible applications. The use of models is not only the numerical results obtained from them, but also the expertise, understanding and knowledge acquired from building the models. In real environments, modelling and analysis must work in conjunction with experimentation, so that predictive capabilities of the models can be tested and validated. A deeper discussion on the benefits of using modelling techniques to assist with the design of services is included in section 3.4.1. The details of the modelling framework proposed here are discussed in section 3.4.2.

3.4.1. Modelling rationale

In the changing and innovative telecommunication environments, modelling and simulation techniques must play an essential role. There is a need to understand and analyse how new services operate and the effects of the introduction of new services in the networks, e.g. interaction

with network resources, with the users, with other services, etc. This is particularly important in the context of preserving network integrity, where the identification of potential errors or undesirable interactions before the services go live, is of paramount of importance.

Here it is believed that modelling and simulation will become increasingly important within commercial organisations such as network operators, equipment suppliers and service providers. Hence the importance of developing appropriate and well structured modelling methodologies that result in a coherent modelling activity within an organisation. A major consideration must be to aim for re-usability of models. The development and use of models is an activity that requires a great deal of investment in terms of time, money and effort. Models designed in isolation, for specific scenarios, and tailored to solve an individual problem may provide effective solutions for this particular problem. However, these models are likely to be discarded afterwards, and it is very inefficient to ignore all the effort and expertise put into the construction of such models.

Models are often used to reproduce existing systems or systems under development, with the aim of identifying problems. The view sustained here goes beyond this. In the proposed approach, models are a component of a larger framework where all the integrating activities have to be co-ordinated (as described in section 3.3). This is regarded as a long-term framework, and aims to support future scenarios as well as current ones. This is why major requirements of an integrity framework are flexibility, adaptability and scalability to adapt to diverse situations (as discussed in section 3.2.1). Hence, it is believed that it is essential that telecommunication organisations focus on the investigation and development of long-term methodologies, rather than on the provision of short-term solutions to specific problems. The proposed approach to maintaining integrity is from a pre-emptive perspective, where identification of risk is an essential area. It is believed that improvement of design methodologies will reduce the risk of problems in the future; and in case problems occur, it will be easier to detect them and correct them. Risk analysis implies accounting for future possible situations or combinations of events that could cause integrity degradation, the study of their probability and the assessment of their effects. Since future can not be exactly predicted, models must not be strictly limited to precisely defined situations; hence, the need for adaptability and flexibility of the models.

The benefits of modelling activities are not always well appreciated in commercial environments. In industrial organisations, where time and money constraints are usually very limiting and speed

to market is a key issue, modelling activities tend to be regarded as low priority. Effort usually concentrates on activities that are more practical and in providing short-term solutions to existing problems. One reason why modelling activities are not always well supported is that the results obtained from them often appear too late to allow effective action to be taken. This is mostly due to the lack of a well-structured and maintained modelling activity within the organisation. Therefore, for every problem under study, models are developed from scratch to solve that specific problem. This approach presents two disadvantages. One is the inevitable delay due to the time consuming phase of building the models. The other is that, because these models are developed in isolation and tailored to solve specific problems, they are often discarded after the analysis of that problem is completed.

To minimise the issues mentioned above, well-structured modelling activities are required where one key aim should be the future re-use of the models. A team of model developers should be responsible for the construction of libraries of model components that can then be used by other people to analyse specific scenarios. This separation between model developers and model users permits a more efficient use of the expertise within a company. The construction of models requires highly technically skilled people with a strong knowledge of modelling techniques and programming languages. On the other hand, the potential users of models are people with a good understanding of other areas or sets of problems, i.e. personnel responsible for the provision of a particular service, but they do not necessarily have significant expertise in modelling or programming. Thus, libraries of models built by the model developers may be used as tools by a variety of people. This approach permits easier and faster generation of results from the modelling activity, because the basic components are readily available for use. The user only needs to concentrate in the details specific to the particular scenario under study, in the simulation of that scenario, and in the acquisition and analysis of results.

Building generic model components is a very complex task. A major issue is to achieve a satisfactory degree of flexibility. The model developer must account for possible future extensions to the models, which are not always obvious. Aiming for a high degree of flexibility adds complexity to building the models, since often what seems the best solution for a particular problem is not satisfactory for another one, and these trade-offs must be solved. Designing flexible models is also a much longer-term activity than building models tailored to specific problems. As previously stated, this is a big obstacle in an industrial organisation, where the time factor always

plays a key role, and where people are generally required to provide short term solutions rather than spending long periods of time building something whose use might not be apparent until much later. Besides, a model to evaluate a particular service or network solution needs to be developed and used within the development time of the network or service, which generally has specified dates to meet the service launch date.

The construction of models needs to adhere to well-defined rules and guidelines. Before any modelling activity starts, a methodology must be developed to produce these rules, guidelines and instructions. This activity is essential in order to guarantee compatibility between different model components, which are likely to be developed by a variety of people, as well as re-usability of these components. The modelling methodology must cover issues such as:

a) Notation - A consistent notation must be used by all model developers in order to achieve compatibility between components and to provide clarity and uniformity of model descriptions.

b) Specification of interfaces - Each model component should be regarded as a black box, whose internal details only need to be known by the developers of that component. To guarantee compatible interconnection of different model components, the interfaces of these black boxes need to conform to predefined rules. For example, in SDL, the sets of acceptable inputs and possible outputs need to be specified.

c) Construction of libraries - The model components must be organised in the form of libraries where components are grouped by a meaningful criteria, e.g. network components, service components, population components, etc. (see section 3.4.2.4)

d) Documentation - The model libraries must be documented both for the model developers and for the users of the models. The former should include information about how the components are constructed, with special emphasis in the detailed specification of the interfaces. The latter should be a higher level description of the functional aspects of the components, avoiding details about their internal implementation, and focusing on guidelines on how to use them, their limitations, connectivity to other components, etc.

e) Reference - All the modelling components must be referenced in a common framework, in order to facilitate easy identification of existing components, and their location.

The type of modelling required to investigate threats to network integrity presents a high degree of complexity. This is due to several reasons, namely:

a) the large variety of services,

b) the diversity of network equipment and software, often provided by different vendors,

c) the fact that integrity threats may come from a multitude of areas, i.e. signalling, performance, incorrect service specifications, etc. which on their own embrace a multitude of factors,

d) complexity is increased in the network interconnect scenario.

The diversity of environments makes re-usability of models difficult because building generic model components that can be used to simulate different scenarios is a complex issue. An appropriate level of abstraction must be achieved, where essential common features are identified and included in basic modelling components. Added to these base components, other more specific components will embrace features specific to particular scenarios. This can be achieved by a process of customisation or refinement of base components where possible, or by designing completely separate specific components. In the former case, for model components to be built as refinement of base components, the languages used for modelling must support some form of inheritance[1]. In the latter case, when specific components cannot be derived from base components and hence must be constructed from scratch, these specific components must still satisfy the requirements in terms of compatibility and connectivity specified in the methodology.

The investigation and treatment of network integrity degradation requires input from a number of activities, e.g. previous experiences, test results, people's expertise and modelling activity. As discussed in section 3.3, this raises the need for powerful automated information systems, where information from all these sources can be incorporated, processed and accessed easily and efficiently. These requirements apply to the modelling activity itself. The modelling tools and methodologies should facilitate the extrapolation of information from the modelling and simulation results. Extremely complex models and extensive simulations that produce large amounts of data are not useful unless there are mechanisms in place to process all this data

[1] Inheritance is a concept in object oriented languages by which an object type (subclass) can inherit some of the properties of another (superclass).

efficiently. It is believed that visualisation techniques should play an essential role in this respect. They would provide the users of the models with global views of the system, as well as the ability to select specific parts of the system to be observed in greater detail. It must be emphasised that the work does not finish with the construction of the models and the production of quantitative results from the simulations. This on its own is not useful unless it is followed by efficient processing and meaningful presentation of those simulation results.

3.4.2. Key components in the proposed integrity modelling framework

In order to use modelling and simulation techniques a careful planning of the modelling strategy to be employed must be carried out. The proposed modelling framework identifies four main areas for analysis . These are:

1. Aims of models, i.e. expected outputs, and modelling methodology,
2. Level of abstraction of the models,
3. Inputs required to the models,
4. Language and tools employed.

The next subsections deal with these points in detail.

3.4.2.1. Aim of the models and modelling methodology

Before a model is started, it is essential to define the objectives of the model and the modelling methodology. Due to the constraints in time and resources, here it was decided to focus on the signalling aspect, where signalling refers to control information flows within the system and the associated processing and data. Signalling is not the only factor relevant to network integrity, and threats to integrity may occur due to a large number of reasons and from a variety of domains. Signalling was chosen because it is an area of very fast complexity growth in modern telecommunication services and hence it requires deeper understanding and study. As discussed in chapter 2, the trend in telecommunications is the introduction of new complex services, many of them working across the domain of more than one network and based on the principles of intelligent networks (IN). In IN the control of services is separated from the call control and specific network functions can be built on physically distributed platforms [Bale 95]. Telecommunication networks are becoming distributed computing environments, where

functionality is physically distributed in the networks. This implies the need for different entities to inter-operate and communicate in real time via signalling, which makes the issue of concurrency a key problem in this type of environment. Signalling becomes increasingly complex as services evolve, and because of this increasing complexity, it is likely to be one of the main sources of failures.

The problem is approached from a services perspective. In this context, a service is viewed as a collection of distributed objects or entities performing different tasks and interacting with each other to provide the service functionality. It is necessary to gain a good understanding of how services operate, i.e. the interactions between different service components, the use of resources and signalling activity, etc. This requires formal service descriptions that represent the behaviour of services (see section 3.4.2.4).

As discussed in [ONP-UCL 94], appendix H, a number of different approaches can be taken to build a model:

1) The first approach looks at the service as one of many in a system, i.e. a service is viewed in the context of other services.

2) The second, views a service in terms of its internal structure, where all the functions provided by the service can be observed.

3) The third, views a service in terms of a logical distribution of functionality among the various players involved.

For the initial phase of the work the second approach was selected in order to model the internal structure of services. This approach provides an insight into aspects such as:

a) Service features involved,

b) Operations that take place,

c) Signalling messaging involved,

d) Type of data involved,

e) Use of network resources.

The study of these aspects permits identification of the areas of greater complexity, where integrity problems are most likely to arise for a single service.

Further stages require analysis of integrity issues in more complex scenarios. Approach 1) is needed in order to study the service in the context of a multi-service scenario, e.g. to identify potential feature interaction problems. In addition, the network interconnect scenario must be included, and this may require approach 3), as functionality may reside at either side of the interconnect point. However these are more advanced scenarios to model. The initial phase of the study must be kept as simple as possible, allowing for complexity to be introduced gradually. Therefore, it was decided to focus on a one-service one-network scenario.

Regarding the use of the models, it was decided to focus our modelling work primarily on the identification of potential errors in services, e.g. perturbations such us invalid messages that may trigger a violation of integrity. Hence, an initial stage was the development of a methodology for this purpose. The pursued aim was to identify the possible states of the system, the transitions between them and their probabilities, both for normal behaviour and possible error states. The behaviour of a system can be represented as an extended finite state machine, as is done in SDL. A system run, or trace, is a specific path of behaviour, i.e. a subset of states and the transition between them. MSCs (Message Sequence Charts) can be used to represent system traces. A path of correct behaviour follows a sequence of acceptable states. Under adverse conditions, the system may leave this path of correct behaviour and move to an erroneous state. Some errors will not produce a strong deviation from the path of correct behaviour, i.e. the system leaves the normal path but still remains within an area of acceptable behaviour. Such minor deviations may also be pulled back to the normal path by, for example, error correction mechanisms. These can therefore be considered minor errors. Conversely, major errors or combination of errors can cause "control mutations", i.e. can make the network follow a path divergent from that of the correct behaviour.

The aim of the pursued framework is to obtain the overall probabilities for complete paths of behaviour, i.e. the probability of moving from one initial state to any of the possible final states. It must provide the probability of erroneous behaviour and should permit identification of the risk of control mutations that can take the system out of its stable condition towards catastrophic breakdown. There is a need to detect deviations and their size, classifying regions of risk and categorising errors according to their consequences. Thus, minor errors can be treated differently

to those leading to catastrophic situation. The actions to be taken will be different in each case. For example, in extreme situations grave errors may impose the need to reset the system, whereas minor errors may even be ignored if their effect is only temporary service unavailability.

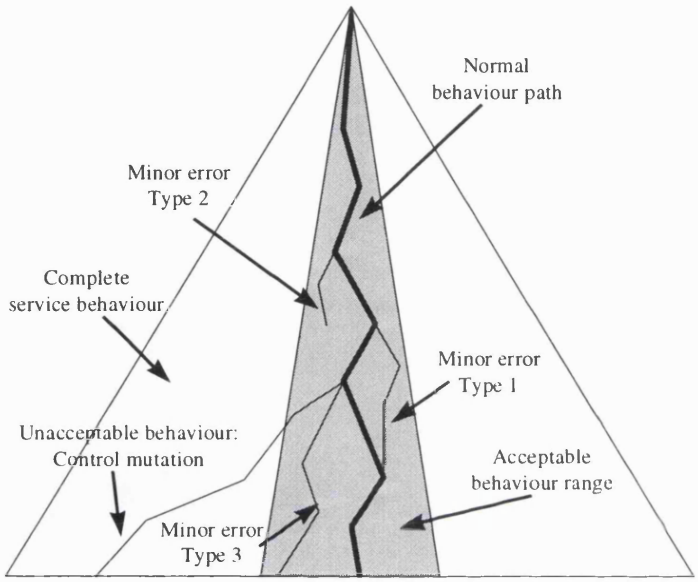


Figure 16. Path of normal behaviour and deviations from it

Figure 16 shows a symbolic representation of the ideas discussed above. The large triangle represents the whole problem domain, i.e. all the possible paths that the system can follow when the service is running. The top vertex represents the initial state and the surface of the large triangle all the different possible trajectories, the shaded area indicating the region of acceptable behaviour. The thick line represents one specific path of behaviour, i.e. a particular trace or system run. Suppose that this is a path of correct behaviour, where all the states in the path are acceptable states. In a real system there is often more than one acceptable path of behaviour, depending for example on the actions taken by the user, the state of the called line, etc. For the sake of simplicity in this example, let us assume that there is only one. A control mutation that takes the behaviour path out of the safe region is represented by the long diverging trajectory. Three types of minor errors are shown, namely:

a) Type 1: minor deviations that return to the correct path of correct behaviour.

b) Type 2: the errors are detected and the process is stopped, by e.g. correction mechanisms.

c) Type 3: the process terminates in a final state out of the normal path but still within the safe region.

The initially proposed methodology is as follows. Given a formal description of a particular service, the first step is to identify one possible path of desired behaviour. Then, concentrate on this path of behaviour and analyse the transactions involved, e.g. signalling, state transitions, etc. Once this path of correct behaviour has been described and understood, the next stage is to identify possible errors that can occur, and study how the introduction of these errors affects the behaviour of the service. This requires observation of the possible final states that the system could reach when a particular error occurs and their probabilities. It is necessary to detect deviations from the normal behaviour due to errors and how big these deviations are, and classify regions of risk, i.e. categorise the possible sources of errors according to the consequences they might introduce. A necessary final stage is the adequate compilation and organisation of the information, so that it can be used efficiently.

As shown in Figure 17, the methodology, for the simplified case of one service, can be decomposed in seven stages. These stages are described next.

Stage 1: modelling and understanding service behaviour

A formal service description that defines the behaviour and operation of the service is needed. SDL can be used to describe the overall behaviour, whereas MSC can be used to visualise selected traces and identify message interchanging between communicating entities.

Stage 2: find path of normal behaviour

Given an SDL description of a particular service, focus on one path of desired behaviour and analyse the transactions involved, e.g. signalling, state transitions, etc. from the MSC description.

Stage 3: identification of possible sources of error

After describing the behaviour, it is necessary to identify possible error events, and study how the introduction of these errors affects the behaviour of the service. Two inputs are needed in order to identify possible sources of errors, namely:

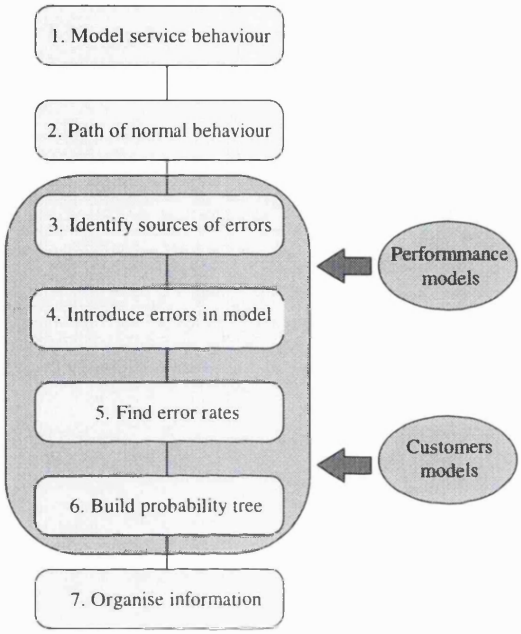


Figure 17. Stages of the proposed methodology

a) Performance models of the system, in order to identify dynamic errors due to poor performance.

b) Customer models representing the characteristics of the population of users and the usage patterns, i.e. customer demand, how services are accessed and utilised by the customers, etc.

These inputs are discussed in more detail in section 3.4.2.4.

A model is built using formal service descriptions, performance characteristics of the system and customer models as inputs. The output at the end of this stage will be the identification of a set of errors that can occur within the given environment.

Stage 4: introduction of errors in the model and identification of error states

The MSCs that describe a normal path of behaviour need to be modified to introduce the error events identified in previous stages. Analysis of this information will permit the identification of error states that can be gradually incorporated into the model in order to build paths of erroneous behaviour.

Stage 5: find probabilities for the different error states

After a list of possible error events and states has been identified, they must be matched to rates, i.e. the probability of errors occurring per time unit. Performance characteristics and customer models need to be taken into account to obtain the probabilities of each particular type of error.

Stage 6: construct overall probability tree

In previous steps, errors were treated in an isolated manner. However, the worst problems are likely to arise when a chain of errors occurs. Hence the need to identify possible combination of errors or sequences of error states that will progressively take the system away from the normal behaviour. We must find the correlation between different error states, i.e. the probability that one error state leads to another. This forms a large and complex decision tree where each node represents a state and the interconnecting branches represent transitions between states and their probability. The aim of this stage is to construct this tree and find the probabilities associated with each transition.

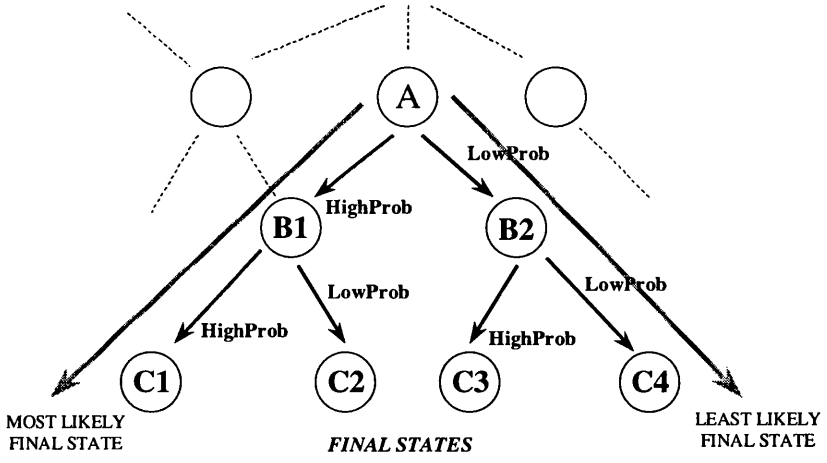


Figure 18. Example of state transitions and probabilities

A simple example is demonstrated in Figure 18, which illustrates part of a tree of state transitions and probabilities. Consider A is the initial state. Suppose that from A the system can go to state B1 with a high probability (HighProb), and to state B2 with a low probability (LowProb). From state B1 the system can move to state C1 with a high probability or to state C2 with low probability. From state B2 there is a high probability to reach state C3 and a low probability to reach state C4. Let's assume that C1, C2, C3 and C4 are final states. We need to find what is the probability of the transition from the initial state A to any of these final states. Clearly, from this example the most likely final state is C1, because the path that leads to it is a full high probability one. Alternatively, state C4 is very unlikely to be reached because both transitions from A to B2 and from B2 to C4 are very improbable. States C2 and C3 fall into a different category of possible states, because the paths to reach them both consist of a high probability transition plus a low probability one.

Ideally, it should be possible to start at any point in the structure and from there identify all the possible paths and their probabilities. This will provide the overall probability of moving from the initial state to each of the possible final states. However, realistically, in complex systems, where the number of possible paths and final states is extremely large, it would be unfeasible to find the probabilities to reach each of the states. Therefore, the work must concentrate in two cases, namely:

a) the most likely states (such as state C1), which describe the normal operation of the system, and, predominantly,

b) the most dangerous states, i.e. those that fall out of the safe region, even if their occurrence is very unlikely, because they need to be quickly identified in order to prevent catastrophic failures.

Regardless of the extreme improbability of entering a dangerous control sequence, identification of such sequences is extremely important. This is because a system operating for a significant length of time will explore a considerable proportion of all the possible sequences. Hence, it will eventually enter this dangerous region with catastrophic consequences. In summary, unlikely events are inevitable, provided enough time elapses, and if such unlikely events lead to a threat to

integrity, the threat will manifest itself. The key point is to rapidly identify them and take the appropriate actions, before the system fails.

Stage 7: organise information

The last stage of the methodology is to extract useful information from the data obtained in the previous stages, so that in can be easily accessed and utilised as a pre-emptive framework. There is the need for adequate documentation of all the acquired information, possibly in the form of a new kind of expert system where it can be extracted in an intelligent way.

3.4.2.2. Level of abstraction

As Woollard states in [Woollard 95], building a sensible model requires adequate levels of abstraction, completeness and simplicity. There is a trade-off between the accuracy with which a model represents reality and the simplicity of the model. Some features of the real system may be essential for studying a particular problem, whereas they may not have any effect in others. Models must capture the aspects of interest for the problem under analysis, whilst maintaining a reasonable level of simplicity to facilitate understanding of the models and interpretation of results. This trade-off between simplicity and accuracy of the models requires identification of the areas relevant to the problem, and exclusion of other aspects that would not add any useful information. Careful planning of the modelling methodology where these issues are addressed is needed before the modelling activity starts.

The large complexity of modern networks and services introduces the need for decomposition of the problems into smaller areas. In order to provide this decomposition, a top-down layered approach was designed [ONP-UCL 94], based on the IN conceptual model (see Appendix A). This top-down layered modelling approach permits the models to be started with a high level of abstraction, increasing the degree of detail gradually and only in the relevant aspects requiring finer decomposition. Each layer in the structure represents a different level of detail, increasing from top to bottom. For a particular service, the components that can have a relevant effect on integrity must be identified and broken down to the necessary level of detail. Other components that do not affect the problem under study can be modelled only at the top level.

The methodology, illustrated in Figure 19, consists of the following layers:

1. The **Service Layer**, where a service is contemplated as a set of features, that can be modelled as a set of interacting objects. At this level we are not interested in how these features are implemented in the system or what resources they use.

2. The **Functional Layer** is concerned with the functionality that supports the service. This service functionality is supported by a set of functions provided by certain entities in the network. The functional layer looks at how these functions are constructed within the network and how they interwork to provide the required behaviour. At this layer, the functional entities are still distributed.

3. The **Network Layer** looks at the network resources that every functional entity uses, and how these resources are distributed around the network. At this level, we start to consider what is the physical network structure that supports the functionality. This not only refers to hardware, but also to the mode of operation of each network entity.

4. The **Operational Layer** is where each network entity is broken down into its operating components. It must be understood what these operating components are and the interaction between them.

5. The **Customer Layer** is concerned with how customers would use the system, and it is placed in the vertical plane because it is present at all the other layers. The customer layer represents the interaction of the users with the service. Information such as population models, usage patterns, demand characteristics etc. will be extracted from this level and introduced as inputs in the other layers.

The important point of this decomposition is that different aspects of the models would access only those layers that are required, thus providing a framework that allows the treatment of different aspects of the problem with the level of detail needed. The structure of a model will generally make use of several degrees of complexity simultaneously. For a given service, certain functions can be treated on a gross scale whilst others will need to be expanded in great detail. An example of how this decomposition can be applied is included in [ONP-UCL 94] for the case of a basic Virtual Private Network (VPN) service [Scott and Stansell 92]. In chapter 6, this decomposition is shown for the service modelled in our case study.

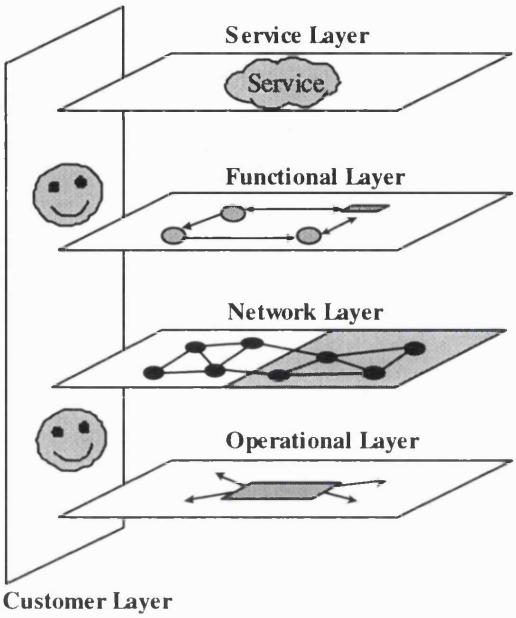


Figure 19. Layered modelling framework

3.4.2.3. Modelling languages and tools

Another important step previous to the modelling activity is the selection of the language and tools to be employed for the construction of the models. This decision depends on a number of factors such as:

a) capabilities offered by the language,
b) resources available,
c) human factors.

Firstly, the aims and requirements of the models must be carefully thought about. The selected language and tools must provide the capabilities needed to meet these requirements. Secondly, consideration must be given to the availability of resources, which may impose limitations on the range of tools to use. This includes human resources, e.g. people's expertise, as well as physical resources, e.g. hardware and software available, and time constraints. Finally, another important

aspect when choosing the languages and tools to be employed for the modelling activity are human factors. The context for the application of this work is commercial organisations, i.e. network operators and service providers. This imposes certain requirements related to the applicability, usability and user-friendliness of the proposed tools. A main driver is to provide tools that can be easily understood and used by a variety of people, who do not necessarily have a large background and expertise in programming and simulation techniques. Hence, the tools need to be commercially orientated, as opposed to purely academic approaches that would require specially trained people. And they must be easy to use, in order to minimise the learning time.

As a result of these considerations, two languages were chosen. The first one is the programming language C++ [Stroustrup 91]. The second is the specification and description language SDL [ITU-T Z.100 94]. C++ was viewed as a powerful, flexible and highly technical language. The object-oriented approach on which C++ is based is suitable for our modelling methodologies, where models are broken down into components that provide specific functionality. The tools required for modelling in C++ were readily available, e.g. workstations, compilers and debuggers. In addition, the author had a background in programming languages, which reduced the learning time. However, C++ does not provide good ways of visualising interactions between components, which is one of the key capabilities we need. Moreover, we realised that C++ is not a very user-friendly language and that, in the context of commercial organisations, tools that are more familiar and easier to understand are required. This is one reason why it was decided to use SDL. The behaviour-orientated nature of SDL made it a suitable candidate for the description of service behaviour, as described in section 3.4.2.1. In addition, used in conjunction with MSCs, it provides a better way of visualising message flows and interaction in the system. Other advantages of SDL are that it is an international standard, it is easy to use, it has proven to be the most used method in telecommunications applications (currently used by a large number of operators and manufactures), and there are advanced software tools available to build and test SDL systems. Moreover, there is a strong international research activity in SDL, and modifications are introduced to the language to respond to new demands. Finally, as part of the research carried out in a major UK operator it was discovered that SDL was being widely used, and we could have some access to that expertise and resources. In summary, C++ was chosen for its powerful and object orientated capabilities, and SDL for its visualisation capabilities and commercial applicability.

Since this work is orientated towards commercial environments, as opposed to purely academic, emphasis was placed on the use of SDL. A major part of the work consisted of exploring the capabilities of SDL and the available tools in the context of integrity modelling, to identify their limitations and propose solutions. This is dealt with in chapter 6.

Finally, as previously stated, the modelling work does not finish with the production of simulation results. This must be followed by processing, analysis and interpretation of these results. This requires the use of data manipulation tools that permit the processing and manipulation of the data and graphical representation of results, as well as analytical characterisation of the problem. The tools we used for this purpose are MATLAB[2] (The MathWorks, Inc.) and Microsoft® Excell spreadsheet.

3.4.2.4. Inputs to the models

The discussions included in previous sections have identified three main inputs needed to build a modelling framework to assist in the design and understanding of advance services and help in the identification of integrity problems. These three main inputs are service descriptions, performance characteristics and customers models (see Figure 20).

Formal service descriptions

Formal functional descriptions of the services under study are an essential part of the models. The behaviour of a service can then be modelled by a combination of service components, each representing one or several selected features, in a similar way as SIBs in IN (see Appendix A). Ideally, service descriptions must be independent of the underlying network, and hence service components must not contain any network-related information. This permits analysis and comparison of results of the same service for different types or network, by using the same service components with different network components.

The service descriptions should permit identification of signalling and information flows between the collection of service components that provide the service. They must also include the points of

---

[2] MATLAB is an integrated technical computing environment that combines numeric computation, advanced graphics and visualisation, and a high-level programming language.

interaction between service components and other elements in the models, e.g. network and population components.

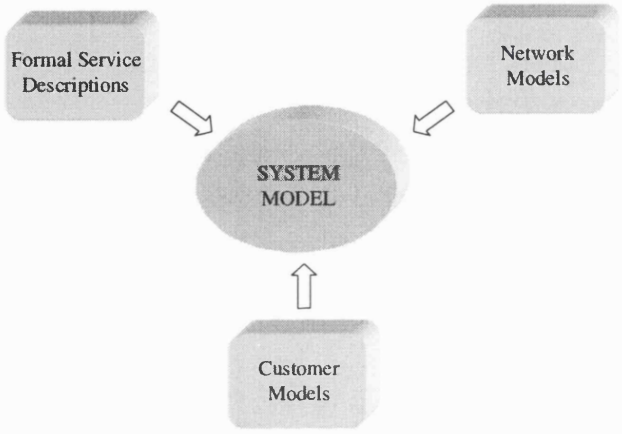


Figure 20. Inputs needed to build the system model

Network models

Network model components are needed to model the architecture of the network where the services under study are provided. Two main aspects need to be modelled. One is the topology of the network, in order to identify flows of communication messages between different entities, thus enabling the identification of integrity degradation due to, for example, signalling problems. The other important aspect that must be included in a network model is the performance characteristics of the network.

From the integrity perspective, a major use of the models is to identify errors and analyse their effects. The performance characteristics of the network can have a great impact in the behaviour of services and in the presence of errors. Two types of errors can be differentiated in services, namely *static errors* and *dynamic errors*. Static errors occur due to bad service specification, poor logic or incorrect programming. They are inherent to the services and will be present regardless the characteristics of the system or the environment in which they operate. An example of this type of errors can be found in the feature interaction problem, where a significant number of the feature interactions are due to incomplete and imprecise specifications of the features [Kelly et al. 94]. This results in a high degree of ambiguity that leads to misinterpretation of the feature behaviour

or interaction with other features. In [Woollard 95], one solution proposed and applied to solve this particular problem is to use formal languages, such as SDL, for the service specifications. In this paper, it is stated that the main cause of the shortcoming in product development and testing is the lack of a sufficiently clear and complete specification. The use of formal methods increases the clarity and completeness of the specifications and provides a better understanding of the behaviour of a service.

The other types of error are dynamic errors. Even when a system is carefully designed, *dynamic* errors can occur due to performance limitations from delays, signalling overloads, bandwidth use, etc. These errors are more difficult to detect and correct because they may be caused by numerous factors outside the service itself, and by a large number of combinations of events. In order to identify sources of dynamic errors performance models need to be included in the overall system model.

An essential aspect in performance evaluation is modelling of time, in order to detect errors due to increased delays in, for example, signalling transactions. Consider a situation where a process is expecting a particular message, e.g. a response from another process. Suppose that the process has a timeout of Δ time units, so that the process keeps dispatching requests for the expected message every Δ time units until the message is received. If the timer is not designed properly, for example if the value of Δ is too low, the requests will be reissued too often, and these requests messages may introduce a significant increase in the system load. This may not have severe consequences if there is only one process involved, but if there are many processes in the same situation, i.e. all of them reissuing request messages, the load increase could be significant. Furthermore, in a real system, the picture can be much more complex since there will be several timeouts and several processes that have to inter-operate. It is believed that timing issues play an important role when modelling in the search for errors due to performance.

One major limitation of SDL in this context is that it does not include performance evaluation. A great deal of research is being carried out on how to incorporate performance information into SDL models. An overview of the existing literature, and the solutions adopted for the models built here are discussed in chapter 6.

Customer models

Customer model components provide information about how a service is used by the customers. Customer models must embrace two different aspects of the information about customers. The first aspect refers to information about the population of users as a whole and their patterns, and it is referred to as *population models*. These population characteristics include information such as number of customers, their geographical distribution, their mobility patterns and average service usage characteristics. These factors are major drivers in the service operation. A service operating correctly may cease to function properly if, for example, the number of customers increases very rapidly. This demonstrates the importance of understanding the limits in the population conditions beyond which the service fails to function as specified.

The other aspect in customer models is description of behaviour of individual users when accessing the service. These are included in so called *user models*, representing how an individual user would interact with the service, e.g. what actions a user is likely to take, possible errors he/she would make and how he/she would react to different situations.

A collection of different customer models must be built and stored in the form of library components. When a particular service or collection of services is to be modelled and studied, the service components need to be coupled to a variety of customer models. The behaviour of the service can then be studied for different customer models, the objective being to identify limitations and possible errors due to stressful conditions of service usage, i.e. surges of high usage. This permits identification of the limits for which the service is valid and the type of dynamic errors that may occur if these limits are exceeded.

Sensitivity analysis

The three types of modelling components - formal description of services, performance models and customer models - must be incorporated into an overall system model to study integrity problems. Each model part needs to be carefully constructed, and the limits for correct use of each component when they are put together must be identified. In particular, for performance and customer models, sensitivity analysis must be carried out in order to find the margins of applicability. This requires modelling the behaviour of the system under stressful conditions, such as a large increase in the number of customers or long processor delays.

The modelling activity must be able to assist with the identification of integrity threats, and in defining and measuring the degree of integrity of the system under analysis, as discussed in section 3.2. The initial stages must concentrate in standard or normal conditions, where the values of the parameters in the models do not impose any limitations. This will provide:

a) an understanding of the system and the model,

b) a basis for validation of the models,

c) identification of key areas of risk and potential failure in the system under study,

d) identification of measurements that need to be included in the system as indicators of the degree of integrity, namely integrity measurements, and

e) identification of variables whose value have a relevant effect in integrity indicators, namely control parameters.

After this phase is completed, "rainy day" scenarios must be modelled, where the system is forced to work under stressful conditions. This is achieved by modifying the values of the control parameters and observing the effects that these changes have in the integrity measurements. This will allow for identification of integrity regions, and the threshold values of the control parameters that define the boundaries between regions. These ideas are discussed in depth in the context of a case study in chapter 7.

3.5. Alternatives for further work

The above discussions regarding modelling have identified the following outcomes and key issues:

a) Modelling is an essential activity in the preservation of network integrity.

b) Importance of building generic, re-usable models.

c) Lack of tools and expertise in that area.

d) Proposal of a modelling strategy, including identification of candidate languages, required inputs and modelling methodology outlining expected outcomes from the models.

As a result of this analysis, two different avenues of research opened up, namely:

a) To focus the work on the design of generic, re-usable modelling systems, following the principles outlined in section 3.4.1.

b) To focus on modelling and simulating a specific service and investigate the uses and applicability of the modelling framework proposed in section 3.4.2.

These two alternatives are discussed next.

3.5.1. Alternative A: design of generic, re-usable model architectures

This option was considered very relevant because of the increasing role that we believe modelling techniques must play in industrial telecommunication organisations. The research carried out by the author in this area showed that there is a lack of expertise and infrastructure in using consistent, well-structured, long-term frameworks for the development of reusable models for integrity analysis. Further research is therefore necessary to identify generic requirements for the construction of such frameworks.

This approach should specify the basis for a systematic and structured modelling methodology that allows study of integrity in a common environment, where models are built in a structured way and model components can be re-used for different scenarios. The need to construct a library of models that can be re-used, i.e. a basis for a coherent and efficient modelling activity has been discussed section 3.4.1. This identifies a market need for advanced modelling tools providing the basic functionality and necessary elements, and with an adequate degree of flexibility to permit customisation for specific situations. Such tools should provide libraries of elements that can be incorporated into the models. This activity is probably out of the scope of companies such as network operators, and is more in the area of software providers. Nevertheless, an initial analysis phase needs to be carried out to identify the requirements and set the basic principles to achieve this modelling framework. The outcome of this work would be:

a) a study of available modelling language and tools, their limitations and investigation of alternatives,

b) identification of the set of requirements to build reusable models and construct libraries of model components,

c) detailed description of basic components needed for integrity study,

d) proposal of a modelling architecture for the development of the models.

3.5.2. Alternative B: application of the modelling strategy to a case study

This line of work would focus on testing and exploring the benefits of the proposed modelling strategies for integrity analysis by modelling a specific scenario. This would provide:

a) a practical understanding and evaluation of the proposed modelling approaches,

b) a better awareness of the suitability and limitations of the available tools,

c) identification of key areas of study for integrity analysis, and how to extrapolate them for generic systems,

d) illustration of how modelling techniques can be used for measuring and categorising the degree of integrity of a system.

3.5.3. Selected alternative

The two alternatives were considered and discussed, and pros and cons were evaluated. Alternative A is regarded as a very important area for research and it is believed that there is a great deal of scope for it in commercial organisations. However, this is a rather long-term proposition, well beyond the time scales for this project. We also felt that the work carried out so far and the theories proposed needed to be tested. This requires some experimental results to support the theoretical arguments. Hence, it was decided to take alternative B, and model a particular case study.

It is important to remark that the modelling work would suffer from some limitations that we were aware of at this stage. Time constraints would force us to focus on one specific case, and the models have to be tailored to this particular scenario. In addition, due to a limitation in the resources available, it is not feasible to build the model based on generic and reusable components, as should ideally be done. These limitations will become apparent in subsequent chapters, when the modelled case study is described. In order to minimise the impact of some of these limitations, the case study to be modelled must be carefully designed, so that although specific, it contains relevant features that make it good representative of real problems. The description of the selected case study is included in chapter 4, followed by detailed discussions of the models in chapters 5 and 6. The main outcome of the modelling work and its connection to the ideas put forward in

early chapters are included in chapter 7. Additionally, as a consequence of this modelling exercise we also developed an insight into the construction of generic modelling frameworks, i.e. alternative A. A discussion of some ideas in this area is also included in chapter 7.

3.6. Conclusions

This chapter contains the author's proposed approach to the problem of maintain integrity in modern telecommunication networks. Firstly, an integrity definition to categorise the integrity of a system in different regions has been proposed. The use of such a framework is to permit identification of the degree of integrity at which a network is operating at any time, and to quickly detect degradation of integrity so that necessary action can be taken to avoid failure. The construction of such a framework requires a quantitative representation of network integrity. The requirements of such a measure have been discussed.

Telecommunication organisations need to improve their working practices in order to cope with the dynamic, diverse and complex modern networks and services and the increasing difficulty of preserving the integrity of these systems. A proposed overall integrity framework containing a number of key activities that must be undertaken to ensure the integrity of networks and services has been described in this chapter. The framework has been divided in two categories, namely static and dynamic. Dynamic actions are taken in live systems, whereas static actions occur 'off-line', i.e. in a testing or laboratory environment, and mainly before the introduction of a change in the network.

The improvement of integrity practises in an organisation would require an initial analysis of current practises and comparison with the activities proposed in the framework described earlier, and the integrity factors described in section 3.2.2. The cross-relationships can then be documented and used to identify further actions that need to be taken [Montón et al. 97 b]. An example with typical results of such an analysis for a new service process is illustrated in Table 3. The content of each table co-ordinate indicates whether the integrity factors on the top of the table are being applied to the process areas on the left. To facilitate the investigation, the development of a new service is considered in terms of five sub-processes, namely: requirements capture, service specification, design, implementation and testing. Application of the analysis methodology

to the design process of a 'Telco' revealed a number of areas requiring action. In particular, the lack of predictive modelling to identify areas of potential risk to network integrity was identified.

	Population models	User models	QoS	Performance	Signalling	Monitoring	Automated records	Risk analysis	Modelling
Req.	!	✓	?	?	_	✓	?	!	_
Spec.	!	✓	?	?	_	_	?	!	✓
Design	!	✓	?	?	✓	_	?	!	✓
Implem.	!	✓	?	?	✓	_	?	!	!
Testing	?	✓	?	✓	✓	✓	!	✓	!

✓ Activity carried out ? Lack of information

! Activity not carried out _ Not applicable

Table 3. New service process

From the discussions in this chapter, the following conclusions can be highlighted:

a) The need for automated techniques to prevent severe degradation of network integrity, and permit rapid recovery from failure situations;

b) The need to use formal techniques for the specification, design and testing of services, i.e. FDTs.

c) New working practises are required, involving more formal and structured approaches and better automated documentation techniques, i.e. KBSs

d) New service provision paradigms require modelling techniques to aid understanding and identify problems early in the design cycles.

e) Increasing need for efficient modelling and simulation techniques for integrity preservation.

Modelling is one key area in the proposed overall integrity framework. The importance of developing consistent, structured and generic modelling methodologies in telecommunication organisations has been discussed. This is an extremely complex task, and identification of requirements for such generic methodologies becomes a necessary and a very interesting topic of research.

A good modelling activity requires careful planning. This includes developing a modelling methodology, achieving an appropriate level of abstraction, identifying the inputs required to the models, identifying the basic components in the models and choosing the language and tools to implement the models. These factors constitute the proposed modelling framework and they have been discussed in this chapter.

As a result of the analysis undertaken, two possible directions of research opened up, namely investigate requirements and propose an architecture for the construction of generic, re-usable models, or to put into practise the proposed modelling strategy by modelling a particular service. These two alternatives, their pros and cons, have been discussed, and it was decided to focus on modelling a specific case study. This is the topic of the following chapters.

Chapter 4

Modelling a case study

4.1. Introduction

This chapter contains a description of the case study developed for the modelling work. In general terms, the example service rearranges the location of distributed files following the location of mobile users. Metaphorically speaking, it is as if the users had some sort of magnetism that attracts the files they need towards their current locations. Hence, we decided to call our example service the Magnet Service. A detailed description of the Magnet service is included in section 4.2, followed by the conclusions in section 4.3.

4.2. Description of the system

The object of the modelling work is a generic representation of management of location of resources within a network, called the Magnet service. It is used as a simple exploratory exercise, and the system does not represent any particular existing service. It is not a service offered as such to the end users, but rather a mechanism to provide capabilities needed for the introduction of end services. The Magnet service contains several essential features that make it a relevant example, representative of current issues and trends in telecommunication services, in particular:

a) Mobility of users and demand for seamless services,

b) Customer demand for personalised flexible services,

c) Distribution of data and service logic across network elements

The service view is presented from the perspective of the user of the system. This is done for two reasons. Firstly, it places no constraints on the service capabilities from the network's limitations. Secondly, the logical construction of a service is from the capabilities offered to the user, rather than expecting the user to adapt to functions presented by the network. This approach is followed because our work is oriented towards the provision of a framework to aid the design of systems. In other words, instead of constraining the requirements to specific capabilities, we build the requirements and study how they would be best implemented, given the constrains of current systems.

The Magnet Service can be regarded as a very simplified version of a universal personal mobility service. Each user is given a personalised environment. This allows the customisation of the telephone system, so that the users can have their personal numbering scheme, and can apply call blocking to specific numbers, etc. Mobility of the users is assumed, so that all these features are available at any point where the user accesses the service.

Although the Magnet service is designed to operate within a telecommunications environment, there are strong parallels with distributed computing. Specifically, it can be treated as file serving an individual user account throughout the network. When a user logs on, a file server exports the information to the remote machine, part of which consists of a configuration script, which is used to personalise the environment. The server represents a bottleneck in the system, since it has to deal with a population of users, all of whom have their files stored there. This solution has worked well in the distributed computing environments because the population of users has so far been small and basically static. These circumstances are definitely changing nowadays, as typified by the astounding growth of the Internet and the associated services, and the limitations of centralised approaches have become apparent. In contrast to this, a telecommunications service would ideally deal with a large number of users, with relatively high mobility. Hence, a static addressing scheme is not desirable in this context, because the location of the users can be changing frequently.

The users of the Magnet service have full mobility capability, and different mobility patterns can be considered. The list of resources available for each user is described in a *user profile*

file, which is always moved to the user's local node. The resources are *data files* whose location is not fixed. These data files contain the service logic for the services that are offered to the customer. The user profiles are only a model of the data space and hence do not contain any service data. They are a reference to the data files, i.e. a list of the data files to which a particular user may require access.

The system allows users to register at any location in the network, and users change their location according to certain population patterns. When a user changes location, his/her customer profile is moved to the new location. Data files, containing service logic and data, are distributed around the network and can be relocated upon request. For the sake of simplicity, it is assumed that the data files are read only files. This avoids issues such as ensuring that the users obtain the most up to date information. The current locations of the data files are stored in a central database.

The simplest case scenario would be that in which each user is assigned a unique user profile, without any information common to more than one user. This is a simplification of the problem for several reasons. Firstly, having only one user related to each personalised file means that only one person can modify it. Secondly, since the user is always at only one place at any one time, the criteria for placement and location of files is greatly simplified since there are not contention issues. A more complex scenario arises when several uses require access to the same data files, thus competing for them. This case opens a whole range of possibilities for file relocation strategies. The problem of optimising the location of resources in distributed systems is not new, and there is extensive literature on the subject, e.g. [Papadimitriou et al. 94] [Mahmoud and Riordon 76] [Kurose and Simha 89] [Jain 87]. Some strategies have been explored, but detailed analysis of this area is outside the scope of this work.

A number of data files exist in the system, and they are initially randomly located throughout the network. When a user accesses the network at a certain location, his/her user profile file is moved to this local node. Then the user proceeds with a request for a service, and at a certain point some of the data contained in the data files will need to be accessed. These data files can be located at any node in the network. The addresses of the necessary data files are obtained from the addresses database.

When the location of the data files is obtained, two actions are possible, namely: the information can be extracted from the data files at their current location, or the whole data file can be moved to the user's current locations. There are many alternatives for the location and

management of these data files and an identification of the optimum mechanism is one of the fundamental requirements of service design. One possibility is to store them at a fixed location. This approach could be efficient if the users always access the service from the same location, because all the user data could be kept in the user's local database. However, in our system full mobility of the users is assumed, and a full distributed solution for storing the data files has been chosen, i.e. the data files may be stored at any location in the network. A criterion must be found by which files should be moved, or left, depending on relative positions and sizes. The idea behind this approach is to avoid the use of network resources unnecessarily. It is assumed that only a small percentage of all the users data is needed each time the user accesses the service, and therefore there is no need to transfer all the data files to the local node. Several file relocation strategies can be studied in order to optimise certain parameters, e.g. bandwidth usage (this is dealt with in chapter 5).

The environment that was considered for the test implementation is a square grid network. In order to simplify the analysis, periodic boundary conditions are imposed, i.e. there are no edges in the grid. All the nodes have a database where data files can be kept. The size of the databases is not considered a limitation.

In the context of the layered framework presented in section 3.4.2.2, at the service level, the Magnet service can be viewed as composed of three main features:

a) an addressing function, that deals with searching for the requested data files within the network;

b) database consultation, or file access function, to access the requested information in the data files, and

c) a file transfer function, that performs the relocation of the data files according to the selected relocation criteria.

Table 4 illustrates how the different service features require different levels of detail, and hence access different levels in the layered structure. The decomposition is done assuming a scenario where the Magnet service is offered across two interconnected networks, and where the aim of the model would be to study integrity threats due to interconnect. Hence, high levels of detail are required for those functions that cross the interconnect point.

SERVICE LAYER	ADDRESSING FUNCTION	FILE ACCESS FUNCTION	FILE TRANSFER FUNCTION
FUNCTIONAL LAYER	Might cross interconnect. More detail required.	Local function, but response time may be critical	Treated as reliable. May need to include failure rates
NETWORK LAYER	Need to identify addressing entities and their interactions.		
OPERATIONAL LAYER	Type of addressing scheme.	Type of file access system, response times, etc.	
CUSTOMER LAYER	Number and mobility of users determine addressing scheme.		Mobility of users and size of files determine relocation strategy

Table 4. Decomposition of the example service

Let us take the addressing function as an example. At the functional level, we can identify that the addressing function might cross the interconnect, if the resource requested from one network is present at the other network. Therefore, a higher degree of detail is required. At the network layer, we need to look at what entities in the system provide the addressing function and how they interact. In the Magnet service there is a central database containing the list of resources and their current locations. At this point, the customer layer needs to be accessed to understand possible limitations to the service. For example, if the number of customers is very large, or they have a high degree of mobility, a centralised structure is not appropriate, because a very large number of updates to this data structure would be required. In this case, a hierarchical addressing scheme would be more appropriate. Conversely, if the users have a very low degree of mobility, searching procedures based on flood filling algorithms could be used, because the number of searching messages would not be too large. If the two interconnected networks have different addressing functions, integrity problems may arise. Hence the need to model the addressing function at the operational layer, to understand how the addressing scheme is implemented. A similar decomposition process for the file access and file relocation functions is shown in Table 4.

The Magnet service will be described in more detail in the context of the C++ and SDL models.

4.3. Conclusions

This chapter contains a description of the modelling case study, the so-called Magnet service. Although the Magnet service does not represent a specific existing service, it has been designed to contain essential features in modern telecommunication services. These features are mobility, distribution of resources and personalised service capabilities. The Magnet service has been described in the context of the layered modelling structure described in the previous chapter. The next step is to model and simulate the Magnet service in order to identify key integrity issues, and explore the use of models for identification of potential errors and measure of system integrity levels. Three models were developed: an analytical model, a C++ model and an SDL model. The C++ model, described in chapter 5, was the initial one, and it was used to build the ideas that resulted in the frameworks and methodologies described in the previous chapter. The analytical model is included in appendix C, and it was used for validation of he C++ model. In order to overcome the limitations of C++, particularly for visualisation of interactions within the system, a third model was built using SDL. The SDL model was used as a proof of concept for the ideas proposed in chapter 3. Details of the SDL model are included in chapters 6 and 7.

Chapter 5

The C++ model

5.1. Introduction

The first stage of the practical modelling work was the construction of a C++ model of the Magnet service described in chapter 4. This C++ model was the first practical exploratory exercise to understand the problem and identify key areas, and resulted in many of the ideas proposed in chapter 3.

As discussed in section 3.4.2.3, C++ was chosen as the initial modelling language for the following reasons:

a) C++ was considered a powerful, flexible and highly technical language.

b) The object oriented approach on which C++ is based is suitable for our modelling methodologies, where models are broken down into components providing specific functionality.

c) Availability of the tools required for modelling in C++, e.g. workstations, compilers and debuggers,

d) Existing background in programming languages, which reduced the learning time.

The objectives of this first part of the modelling work were as follows:

a) To develop and put into practice appropriate modelling methodologies for the network integrity problem,

b) to use it as a first approach to understand the service operation and identify service features,

c) to investigate which service features identified for the Magnet service are most relevant in the context of network integrity,

d) to begin to understand the influence of population models in the response and way of operation of the Magnet service,

e) to evaluate the suitability and identify limitations of C++ for this type of integrity modelling.

This chapter contains the description of the C++ model of the Magnet service, the simulation results and the concluding remarks that led to further modelling work. Section 5.2 describes the simulation environment. A first version of the model and the obtained results are included in section 5.3. A subsequent, more advanced version was developed to embrace the key issues identified from the first version. This second version is described in section 5.4. A discussion of the applications of the C++ modelling work is included in section 5.5, followed by the conclusions in section 5.6. An analytical model was also developed and used for validation of the first version of the C++ model. This analytical model is included as Appendix C.

5.2. Simulation environment

A simulation program was written in C++ to investigate the operation of the Magnet service on a square grid network with periodic boundary conditions. The initial phase of the simulation program is to distribute data files around the network. The data files are assigned a unique identifier, a random size between zero and a maximum value, and a random location in the network. For each data file, an entry containing the file identifier and its present location is created in a database. In addition, the user profile files are created and they are updated with a reference to several of the data files available. When this initial stage is completed, there are a number of data files randomly distributed around the network. The addresses database contains a list of all these files and their locations, and the user profiles contain a reference to a number of data files that the user may request later. After this set-up phase, the simulation clock starts to tick and the system starts to simulate the behaviour of the Magnet service.

The following is a list of the parameters involved in the problem:

a) Size of the network: square lattice of size X x X (X rows by X columns).

b) Total number of data files: N.

c) Maximum size of the data files: S_{max}.

d) Minimum size of the data files: S_{min}. By default, unless otherwise specified, S_{min} =0.

e) The size of a file is a fixed random number between S_{min} and S_{max}, allocated initially.

5.3. First version of the model

The first version of the C++ model considers the simplified case where the data files associated with a given user profile file are unique to that user. Therefore, there is no contention between user profile files for access to resources. This is examined first because it is possible to study this situation analytically and therefore verify the simulation before further work is considered.

The aims of this first version of the model service were as follows:

a) verification of the model with analytical results,

b) start to examine simple population models,

c) find about average bandwidth usage when moving files around based on the demands given by the population model,

d) start to identify the relevant parts that have an effect in the integrity of the system and that need modelling in more detail.

In this first version of the model, the aim of the simulation was to determine the average bandwidth usage when moving data files around, based on the demands driven by the population model. The actual measurement taken as an indicator of the bandwidth usage when a file is moved from one location to another, is the product of the size of the file by the distance is travels. The distance is measured in number of hops that the file travels through, where a hop is the distance between two neighbouring nodes. This bandwidth usage was obtained and compared for different system conditions. In the second version of the model, other measurements such as delay times and signalling load were also included.

In chapter 4, the three main features of the Magnet service were identified as addressing function, database consultation and file transfer function. In this first version of the model, these service features are modelled at a high level. It is assumed that they are provided by the network and they are reliable, and at this point, the details of how the functionality is provided are not specified. Referring to the layered modelling framework described in section 3.4.2.2, we are at the functional layer.

Two types of data files are introduced. The difference between them is the probability of the files being requested. *'Priority 1'* data files have a higher probability of being requested than *'Priority 2'* data files. The effect of this classification of files was studied running different simulations changing the percentage of files belonging to each class. The following is a list of the relevant parameters:

◊ Total number of data files: N.

◊ Number of priority one files: $N_1 = n_1 \times N$.

◊ Number of priority two files: $N_2 = n_2 \times N = (1-n_1) \times N$.

◊ Probability of accessing priority one files: p_1.

◊ Probability of accessing priority two files: p_2.

Figure 21 illustrates the first version of the Magnet C++ model. A simulation clock provides the following events for every clock cycle. First, the user profile file is moved to a new location, representing the user access to the service at that location. The algorithm controlling this movement is referred to as the population model, and two alternatives are considered (see section 5.3.2). The user profile contains a list of data file identifiers and a pointer to a data structure where the current location of the data file is kept. This is referred to as the addresses database and it is located in one of the nodes in the network. Its specific location is not important, since periodic boundary conditions are being imposed in the network lattice.

When the user registers in a new location, a selection of the data files included in the user profile are selected to be requested. Each data file is assigned a fixed probability of being requested - *p1* for 'Priority 1' files and *p2* for 'Priority 2' files. The decision to request a data file or not is made by comparing this probability with a random number generated each time for each data file.

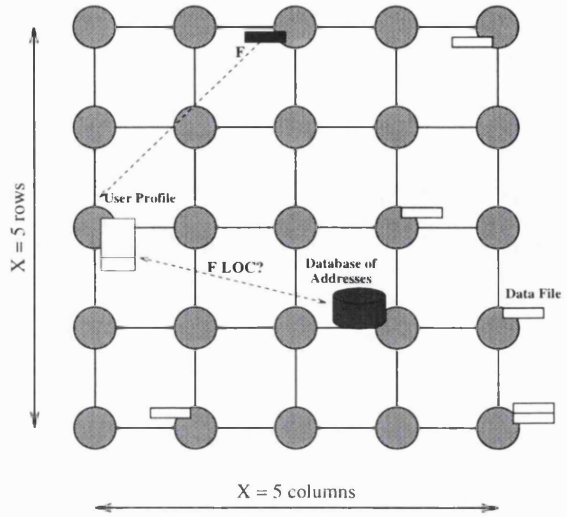


Figure 21. First version of the service, for a 5x5 square lattice[1]. The user's local node requests the data file F. The location of F is obtained from the addresses database

For each of the selected data files, a request for the file address is sent to the database of addresses. A response containing the address of the file will be sent back to the user's local node. This is the addressing function, which at this stage is still modelled at the functional level (see section 3.4.2.2). In other words, it is assumed that the user somehow obtains the correct data file address, but the process by which this happens is not detailed until the second version of the model (described in section 5.4).

When the address of a data file is received, a request message is sent to the file's location, and the file is transferred to the user's local node. Every time a data file changes location the corresponding entry in the addresses database is updated. For every file that is moved, the product number of hops times size of the file is computed as an indication of the bandwidth usage, and the cumulative result is stored. At the end of each cycle, the total bandwidth usage for all transactions is obtained. At the end of the simulation, the average bandwidth usage per cycle is calculated.

The routing scheme implemented in the C++ model of the Magnet service is a static scheme based on shortest path between nodes. Each node in the lattice has a fixed routing table containing the output link that must be used to send information to each of the other nodes.

---

[1] This figure shows the case of non-periodic boundary conditions

For the totality of the work it is assumed that the local access times to the data in the files, i.e. the time to access the relevant information within a file, are fixed compared to communication times, i.e. delays in finding the files and transferring them. For this reason, local access times are not considered when computing the measurements. Another assumption is that there is only one user in the system. This simplification imposes no significant additional constraints, since at this stage we are considering that data files are unique for each user.

5.3.1. Relocation criteria

The simple model so far described has no capability to minimise the bandwidth usage. Every time a file is requested, it is transferred to the users local node. In order to impose some localised management on the use of network resources the concept of *threshold is* introduced, which is used to decide whether a file is relocated or not. Only those files that satisfy a certain condition would be moved, whereas the rest would remain at their present location. The condition to move a file or not is to compare the product of the file size and the distance it would have to travel - which is the quantity used as indicator of the bandwidth usage - with a threshold T. Two opposite cases were studied:

a) moving files when the product is less than threshold

b) moving files when the product is greater than threshold

The three cases, i.e. no threshold, greater than threshold and lower than threshold, are discussed in subsequent sections.

5.3.2. Population models

The importance of introducing information about the population of users of the service and their behaviour patterns in a model has been discussed in the course of this thesis. This type of information constitutes the customer layer in the modelling framework described in section 3.4.2.2, and it is introduced in the overall model in the form of population models.

In this case, the analysis focuses on the complex relationship between user behaviour patterns and the structure and control procedure that is best suited to support expansion of the service to cater for the growth of users. It is a complex task and requires a great deal of input from many diverse disciplines, not least human factors studies, which are not addressed here. Simple population models are examined and used to drive Magnet service. This defines the operational behaviour of the system demonstrating the differences of requirement.

Two population models have been simulated. They are described below.

5.3.2.1. Random population relocation

In this population model, the location of the users in the network is random. This means that when a user, i.e. the user profile file, changes location there is no correlation between the new site and the previous one. This results in a uniform distribution of data files in the network. When a user is relocated, a random selection of data files will be moved to this new location. Because the current location has no association to any previous location, the moved files on the whole can be at any location. Hence, the probability of a file being present at a given node is the same for every node, i.e. $1/X^2$, where X^2 is the total number of nodes in the square grid (see Appendix C). On a subsequent movement of the user, the previously moved files are at a location that has no relationship to the new one, and the data files moved were selected at random. Hence, their location relative to the new position is equally likely to be any site. The net effect is that the probability of file location remains constant in time. This uniform and temporally static distribution, together with the fact that we are applying periodic boundary conditions to the network, make the calculation of average quantities such as bandwidth usage very straight forward. The analytical calculations for this case are included Appendix C.

5.3.2.2. Diffusion based relocation

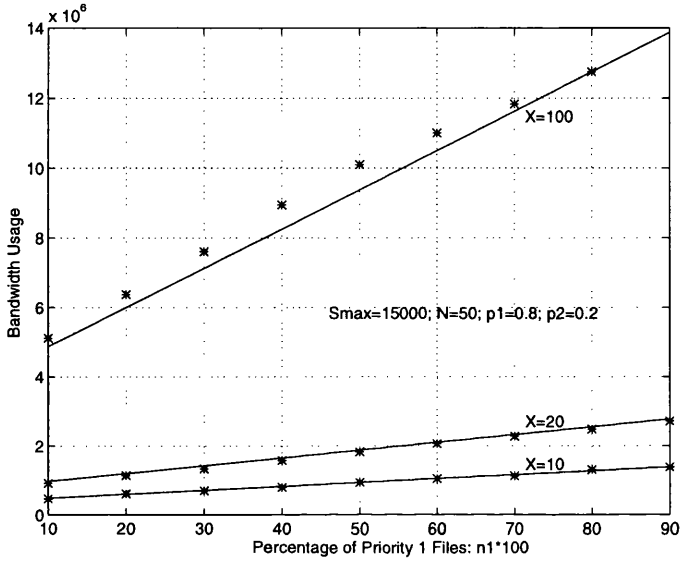
In this case, a user migration model based on simple diffusion is introduced. This means that when a user profile file is relocated it can only be moved to one of the nodes neighbouring its current location. Although this is still a simple description, a great deal of correlated behaviour is introduced. The correlation makes the analysis difficult and therefore in this case we must rely purely on simulation techniques.

5.3.3. Results for the random population relocation Verification of simulation

5.3.3.1. No threshold case

The average bandwidth usage for the case of moving all the files that are requested (no threshold case) was measured for different network sizes and different percentages of priority 1 and priority 2 files. Figure 22 shows the comparison between the theoretical calculation and the simulation results using the appropriate parameters. The theoretical analysis is included in appendix C, where the bandwidth usage for this case is giving in equation 1.12. The results are shown as a function of n_1 - proportion of priority 1 files - and for three separate values of X -

number of rows in the lattice. Since priority 1 files are more likely to be accessed than priority 2 files, the larger the proportion of priority 1 files, the larger the number of files that are relocated, and hence the larger the bandwidth usage. Figure 22 shows that there is a linear dependency between the percentage of priority 1 files and the bandwidth usage. In addition, the bandwidth usage increases for large grid sizes, due to files travelling longer distances.



——	Theoretical calculation	*	Simulation results
p_1	Probability of requesting a priority 1 file	p_2	Probability of requesting a priority 2 file
X	Number of rows in the square grid network	S_{max}	Maximum size of data files
N	Total number of data files		

Figure 22. The bandwidth usage as a function of priority 1 files. The solid lines are the theoretical values and the stars are the results of the simulation

The correspondence between the simulation and the analytical results of course is not perfect. The minor deviation can be attributed to inadequate averaging and hence the results can be regarded as validation of the simulation.

5.3.3.2. Lower than threshold Case

With this condition files are moved only if the necessary bandwidth is below the threshold. This includes small files, even if they are far away from the local node; and local files, even if they are very large. That is, very small files and files very close to the user's local node will always be moved. The argument for doing this is that if the necessary bandwidth is very small it may be more effective to transfer the file to the local node and then access the information

locally, hence avoiding all the communication transactions that would be needed if the data is accessed remotely.

The usual test case was considered where the parameters are set such that

- N=50
- X=20
- $S_{max} = 15000$
- $p_1 = 0.8; p_2 = 0.2$

where N is the total number of data files, X is the number of rows in the grid, S_{max} is the maximum size of the data files, p_1 is the probability of accessing a priority 1 data file and p_2 is the probability of accessing a priority 2 data file.

The value of n_1 (proportion of priority 1 files) is considered through the range 0.1 to 0.9, with each value being a separate line on the graph. The labels in the graphs are percentages, i.e. $n_1 x 100$, therefore ranging from 10% to 90%. The values of the threshold were chosen as the average size of files, 7500 (equation 1.3 in appendix C), times a distance factor ranging between 0 and twice the maximum distance in the grid, X. Thus, the threshold is $T=7500\sigma$, where σ ranges from 0 to 2X and is used to label the x axis.

The simulation results are averaged over 100 separate runs for each set of parameters. This gives reasonable level of averaging for the quantities monitored. However, the sensitivity of a particular output variable can vary considerably.

Figure 23 shows the comparison between the theoretical and simulation results of bandwidth usage for different values of the threshold. The theoretical calculations are included in appendix C. Since a requested file is moved only if the necessary bandwidth is below the threshold, the higher this threshold, the larger the number of files that are relocated, and hence the larger the bandwidth usage. The graphs in Figure 23 show this growth in bandwidth usage as the threshold increases.

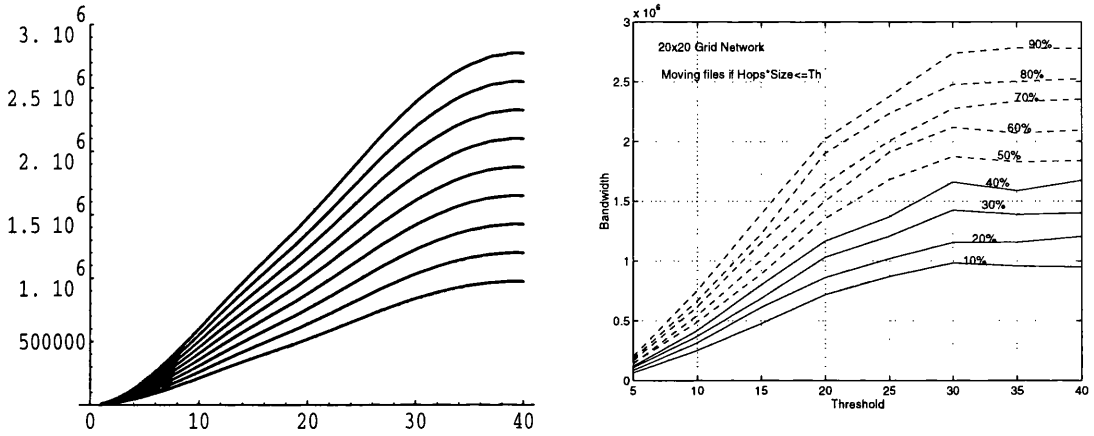


Figure 23. A comparison of theoretical (on the left) and simulation (on the right) results of bandwidth usage against σ, for the case of file relocation if the file size and distance product is less than the threshold.

NB: there is no difference between solid and dotted lines in the graph.

The comparison shows excellent agreement between theory and simulation. The remarkable result is that, although the process is governed by complex non-linearities, the growth of bandwidth usage with increasing threshold is predominantly linear. Only at very small thresholds and near saturation[2] does it deviate from this. There appears to be a slight disparity on the point at which the bandwidth usage saturates, the simulation saturating earlier than predicted theoretically. This is only slight and is hard to quantify as it is within the errors of the measured results. If the effect is real, it indicates that there are slightly more files moved, or they are on average slightly larger than predicted. This is perfectly possible, as the exact way in which real numbers were truncated to integers in the simulation was not treated in the analysis.

The linearity of bandwidth with threshold would have interesting consequences on the choice of control parameters, if it were to be applied to a more complex population of users. In the case considered, the range of activity for a user is the entire network. If, as is more realistic, users are expected to be confined to a particular region with high probability and only range beyond this rarely, the threshold should be chosen to suit that range.

It is worth noting here that we are examining only one parameter in isolation, i.e. bandwidth. To make a judgement as to which value is most applicable we need to consider the trade-offs and dependencies on other performance characteristics. In particular, it is necessary to

[2] Saturation occurs when changes in the values of the threshold do not affect the bandwidth usage.

consider parameters related to the Quality of Service (QoS), i.e. measures related to how the service is perceived by the users. This is not done in this first version of the model, but in the second version the factor "delay" is introduced as a QoS parameter (see section 5.4.2.1).

The performance can be demonstrated by considering the same system as above, but changing the value of S_{max}. In this case we look at values either side of that already chosen, i.e. S_{max}=10000 and S_{max} = 20000. The theoretical results of Figure 24 show that if the threshold is chosen to be too large, i.e. the files are small relative to the threshold, saturation occurs and there are no gains to be made. On the other hand making the threshold too small does not utilise its full range of flexibility, i.e. as shown in the right graph saturation is not reached within this range. In a real case, careful choice of the control strategy as well as the parameters should be considered, but at such preliminary stages only possible directions can be suggested.

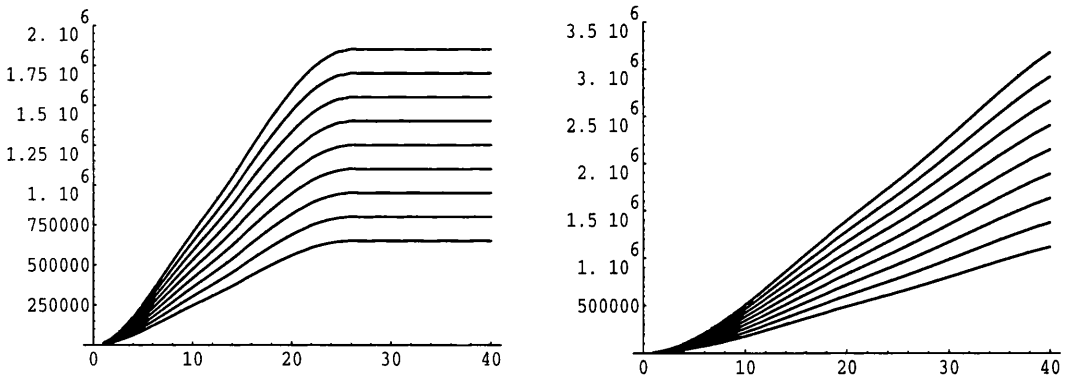


Figure 24. Theoretical bandwidth usage against σ for the case of file relocation if the file size times distance product is less than the threshold. The data on the left shows the bandwidth usage if S_{max} = 10000 and that on the right shows the results for S_{max}=20000. As previously, T=7500σ

5.3.3.3. Greater than threshold Case

In this case files are moved only if the necessary bandwidth exceeds the threshold. This includes large files even at relatively close distance from the local node; and relatively distant files, even if they are small. This scheme tries to maintain large files in the vicinity of the model file and disregards the location of small files. The rational for considering this alternative is that it could be assumed that for large files the amounts of data that need to be

accessed is larger than for small files. The advantage of bringing large distant files to the local site is that once the file is moved, all the data is available locally, avoiding all the communication overhead necessary if the data was to be accessed remotely.

Figure 25 shows the simulation results under exactly the same conditions as those used for the lower than threshold case (see 5.3.3.2), except that the threshold condition is reversed.

Direct comparison of the results shows that, as is to be expected, they are simply the reverse of one another. The degree of averaging is less than in the lower than threshold case by a factor of two, which explains the noisier data.

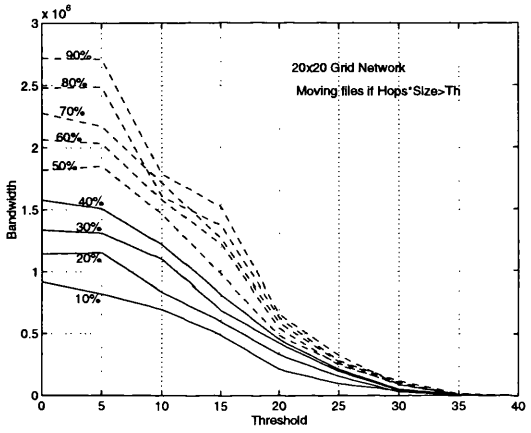


Figure 25. The simulation results of bandwidth usage against σ, for the case of file relocation if the file size and distance product is greater than the threshold

5.3.4. Results for the diffusion based population relocation

The use of random relocation in the above model is the basic device that allows the simple theoretical calculations to be performed. If we consider the more realistic, but still simple population diffusion model, then it becomes much too difficult to obtain analytic results, and we must rely purely on simulation. Simulation results for this population model are discussed below.

We proceed by using the same meaning and value for the parameters p_1 and p_2 as we used previously, i.e. p_1=0.8 and p_2=0.2. We also make a comparison to previous results, but this is slightly misleading. In the previous example relocation of the user could be to any node in the network, hence at each time step the user profile moves an average distance of X/2 (appendix

C, equation 1.6). In the case of this diffusion population model, it moves only one unit. It is therefore fair to say that the speed of the users is vastly different in the two models. Compensation of p_1 and p_2 such that the average distance travelled is comparable is possible. However, it was considered no more useful than direct comparison without modification. This is a matter of interpretation. The basis for our argument is that users are to move from one site to another, where it is assumed they stay for a reasonable period of time. The time required for movement of the user is considered insignificant in comparison to this residence time; therefore, a direct comparison of the two situations is made.

The fact that there is a strong correlation effect gives rise to significant differences in the behaviour in the presence of the threshold mechanisms described previously.

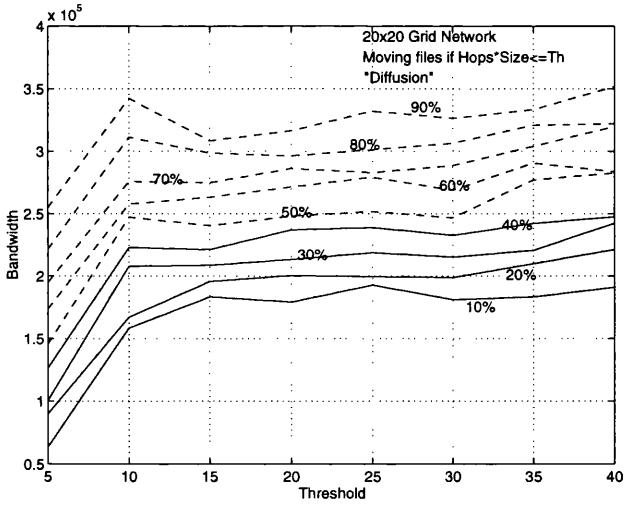


Figure 26. The simulation results of bandwidth usage against σ, for the case of diffusive file relocation for the lower than the threshold case

5.3.4.1. Results for threshold model

The Magnet service was simulated for the diffusion based population model, using the same parameter set as in previous cases:

- N=50
- X=20
- $S_{max} = 15000$
- $p_1 = 0.8$; $p_2 = 0.2$

The results for the case of file relocation if the needed bandwidth is the lower than the threshold are shown in Figure 26.

The obvious difference with the random population model is that the bandwidth usage, even for maximum threshold, is drastically reduced - by approximately one order of magnitude. This is because most of the data files will be in the vicinity of the user's local node when requested. The model file travels one hop at a time, taking some of the data files with it. The next time it moves, a significant number of the requested data files will be present in the model file's previous few locations, which by the nature of the motion are only a few hops away. In other words, the user profile file propagates dragging a cloud of data files with it. Hence, most of the data files are kept in the immediate vicinity, being the used bandwidth consequently much lower.

The other difference is that saturation[2] is reached much more quickly than in the random case. The effect is reminiscent of the case where S_{max} is large. The reason for the effect is simple; the diffusion process allows the files to be moved to within the vicinity of the users location. Once this has happened the effective size of the network is much smaller, as data files are more likely to be close to the user, than distant from it. Increasing the threshold only increases the effective distance over which files can be moved. The diffusion process defines a range of its own through the correlation gain, if the threshold exceeds this distance then there can be no effect by increasing the threshold still further.

The converse case where the files are moved if they exceed the threshold is shown in Figure 27. Again, there is the dramatic reduction in bandwidth usage. However the most striking feature is that from $\sigma = 5$ and beyond there is little appreciable difference between the cases for different percentages of high priority files. This is because although a high priority file is accessed quite often, it is only moved when the distance is sufficient to satisfy the threshold condition. This tends to happen infrequently, as once a file has been moved it remains near the model file for some time. Therefore, files are not moved often, and when they are moved they travel over minimal distances. In contrast, low priority files are accessed less frequently. This gives time for the user to move longer distances away from them; thus, they contribute a large component to the bandwidth usage. The two effects tend to cancel out the difference in priority.

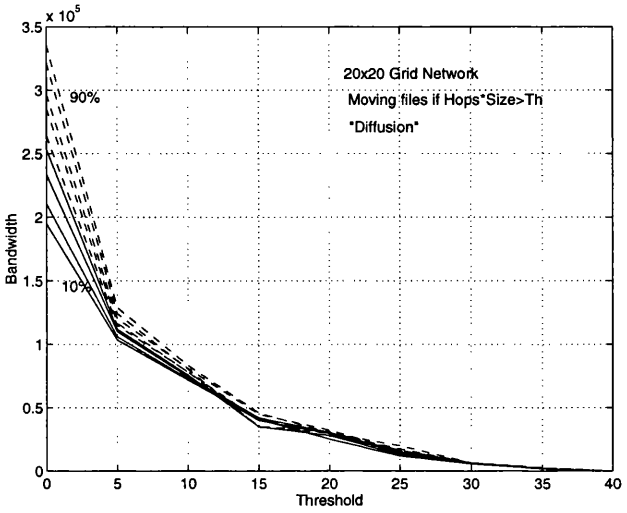


Figure 27. The simulation results of bandwidth usage against σ, for the case of diffusive file relocation for the greater than the threshold case

Possibly the most important overall effect is that there is now no inverse symmetry between the two threshold criteria. Much more complex considerations need to be made when trying to optimise a control strategy for this model of population dynamics. This clearly shows that it is unwise to separate usage and physical structure from the operation and control architectures of even a simple service.

5.3.5. Comparison between the random and diffusion patterns

The results in the previous sections have shown that with the diffusion pattern:

- The bandwidth usage is lower than with the random pattern,
- Saturation occurs earlier, therefore different threshold values should be chosen in order to exploit the file management mechanism to its full extent.
- Results for the greater than threshold case are not the opposite to the lower than threshold case (no symmetry)
- For the greater than threshold case the behaviour for different % of priority 1 and priority 2 files is not that different. I.e. the mobility pattern changes the impact of the amount of data accessed (higher P1 files => more data moved).

5.3.6. Relevant aspects identified

The modelling and simulation work and the results obtained with the first version of the model permitted the identification of a number of areas requiring careful analysis. Some of these areas have already been considered in this first version of the Magnet service. These include the effects of population models and management strategies for the relocation of the data files. The impact of changing them in the performance of the Magnet service has started to be identified. The results obtained from the simulation identified other areas requiring further study, and this was the basis for the development of a second version of the model. These areas are the addressing function, the need for a measure of quality of service (QoS) and the issues about contention for the files.

Addressing function

The addressing function in the first model of the Magnet service is modelled at the functional level, i.e. it is assumed to be provided by the network and it is reliable. The details of the actual mechanism by which the addresses are obtained are not implemented. In a real context, this function would be provided by exchange of signalling messages between the local node and the addresses database. The introduction of this signalling adds complexity to the system. It represents an increase of the signalling load and if any error occurs, such as corruption of these signalling messages, it may result in a degradation of the performance and even in complete service breakdown, being thus an integrity problem. Hence, it was decided to model the addressing function with a greater level of detail. Conversely, the other two features of the Magnet service, i.e. the file transfer function and the database consultation, are kept at the functional level. The former can be regarded as a well-known and reliable function once the destination address is known and it was assumed that modelling it in more detail would not introduce any relevant new information. The database consultation function could, on the contrary, have important consequences in our problem, because if this function fails it could render some of the data files inaccessible, and this might have severe consequences in our service. However, for the sake of simplicity it was decided to maintain this function at the functional level and focus only on the addressing function.

Contention issues

The first model of the Magnet service assumes that the data files are unique for each user. A further development is to introduce a more realistic situation in which there are several users in the system sharing a certain number of data files. This new situation complicates the problem because it brings contention issues. Having more than one user moving around the

network and competing for the same files raises new questions such as where the data files should be stored, what happens if more than one user requesting the same file at the same time, etc.

Quality of service

Finally, another aspect deserving consideration is the analysis of Quality of Service (QoS) for the Magnet service. The only measure taken in the first version of the service is the bandwidth usage. The different file management control strategies and the simulations were developed with the objective of minimising this parameter. A more realistic approach requires consideration of other trade-offs and dependencies between different parameters. For example, from the network perspective the bandwidth usage should ideally be as low as possible, so that the available capacity can be allocated efficiently. In an extreme case, this would imply to move as little a number of files as possible. On the other hand, if we consider the problem from the user's point of view, they want the service to perform in a fast an efficient way for themselves. Ideally, each user would like to have all their files locally, so that they can be quickly accessed. This shows a trade-off between the users point of view and the 'network' point of view. The network point of view was considered in the first version of the Magnet service, but in order to introduce the users point of view some measure of QoS was needed.

The considerations discussed above led to the development of a second model of the Magnet service, in which these new aspects were included. This new version is described in the following section.

5.4. Second version of the model

Based on the findings from the first model of the Magnet service a second version was developed in which three major enhancements were introduced, namely

a) The addressing function is modelled in more detail, including signalling messages sent across the network in the form of packets. It is assumed that each message is sent in one single packet. There are two types of messages. The first type is request messages that travel from the user's local node to the addresses database, asking for the address of a particular data file. The second type is the response messages containing the requested address and sent from the addresses database to the user's local node. When a request for a file is generated in a node, the file is first searched within the local node, and only if it

is not available locally an address request message is sent to the database. The signalling load is measured in terms of number of packets present in the network at a given time, and it was observed for different values of the parameters in the problem.

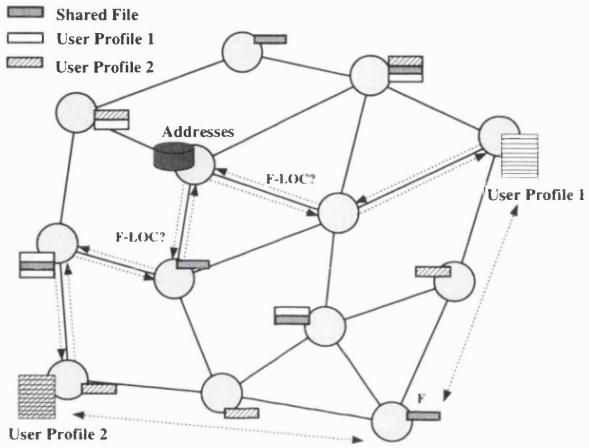


Figure 28. Second version of the service, for a generic network topology

b) In order to examine the effect of the contention for the data files, there are two users in the system who share some of the data files and who may simultaneously require to access them. Both users follow the random relocation mobility pattern, and they move at the same time but to locations independent of one another. It is assumed that data files are only requested on the clock cycle when the users change location. During the time between two consecutive changes of location for one user, no new files are requested. This represents the case of users only changing location at certain times of the day, and only accessing the service when they change location. Other alternatives are possible, but due to time constraints, only this case has been studied. An example of this scenario could be a call forwarding service, where customers request for their calls to be re-directed only when they move between home and office locations, and the whole population of users changes location at roughly the same time, i.e. in the rush hour. Figure 28 illustrates the basic behaviour for the case of two users in the system, when both users request for data file F. The data files can be divided in three categories, namely files unique to user 1, files unique to user 2 and those shared between both users. The selection of data files for each user is done at random. The two user profile files represent the two users in the system. The arrows from the user profile files to the database of addresses represent the signalling messages involved in the addressing function. These

messages have to go through several nodes within the network to reach the database. The routing scheme employed in the model is based on a shortest path algorithm. The arrows between the user profile files and file F represent the data transfer function. The fact that they do not follow a route within the network is used to signify that this function is only dealt with at a functional level.

c) A new measure is introduced in the model as an indicator of QoS. This is the delay in obtaining the data files. Since the local access times and file transport times are regarded as constant, the measurement taken is the elapsed time between an address being requested and its arrival to the source node. From a user's perspective, the aim should be to minimise this delay time. However, things are more complex than that, because of the trade-offs between this and the performance measures, i.e. bandwidth and signalling load. The simulation results showed this trade off.

5.4.1. Contention issues

The competition for files from the users introduces higher complexity with respect to the one user scenario. In particular, it introduces the possibility of obtaining wrong information about the location of data files. Consider the following situation, where the two users in the network are user 1 and user 2. Suppose that user 1 asks for the location of a certain file F, which is not present in user 1's local node. In this case, a request message is sent from users 1's local node to the addresses database, the location of file F is looked up in the data base and a message containing this location is sent back to users 1's local node. Suppose that at the same time that the latter process is going on, a request for the address of file F arrives to the database from user 2. If the current information about F's address were sent to user 2, by the time that user 2 obtains it, it will possibly be out of date, because during that time file F may have been moved to user 1's home location. A strategy has to be designed to deal with this situation.

The solution adopted in the model is as follows. A flag is included in each entry in the database of addresses. Every time that a file is moved to a different location this flag is activated, and remains like that until the database is updated with the new location of the file. If a request for the address of a file arrives when the flag is activated, the address is not sent to the requesting node. Obviously, if this is the case, the requesting node would not obtain the desired address unless a new strategy is introduced. The selected strategy is to introduce a time-out, Δ, in the requesting node, so that if the requested address does not arrive after Δ time units, the local node issues a new request. This situation repeats until the desired address

is obtained. The value of Δ is chosen to be small compared to the speed of movement of the users, i.e. the first re-attempts are made well before the users leave the node, hence it is assumed that the user will eventually obtain the file. In fact, the number of attempts needed to obtain an address was measured, and the simulation results showed that no more than two attempts were needed (see section 5.4.2). This time-out Δ is an example of the importance of the performance characteristics of the system. If Δ is too low, a large number of request messages would be generated to obtain one file, hence increasing the signalling load significantly. If, on the contrary, Δ is too large, the delays in obtaining the requested address can be very long, and this address might even arrive too late, after the user has moved to another location. This is a simple example of the trade-offs that appear in the design of services, and of the importance of studying them carefully and using risk assessment to evaluate the potential consequences of different implementation options. For the sake of simplicity, in the model, the value of Δ is kept constant, but other alternatives are possible, e.g. use random values for each re-attempt, increase the value for consecutive re-attempts to obtain the same file, etc.

5.4.2. Simulation Results

This section contains the simulation results obtained for the second version of the Magnet C++ model. As previously introduced, we can differentiate two types of measures: QoS measures and performance measures.

5.4.2.1. Measurements of QoS

Two QoS related measures were taken in the model. The results are discussed next.

a) Delay time to obtain a requested address. The elapsed time between the generation of a request and the arrival of the requested address to the requesting node was measured. Figure 29 shows the histogram of these delay times. Axis x in the graphs represents time, and for each value of x, axis y represents the number of cases that experienced that delay. Each plot illustrates the results for a different percentage of files shared between the two users, starting with no shared files and finishing with the case of all the files being shared. As it would be expected, the more files are shared the larger the delays are, because the contention for files is higher.

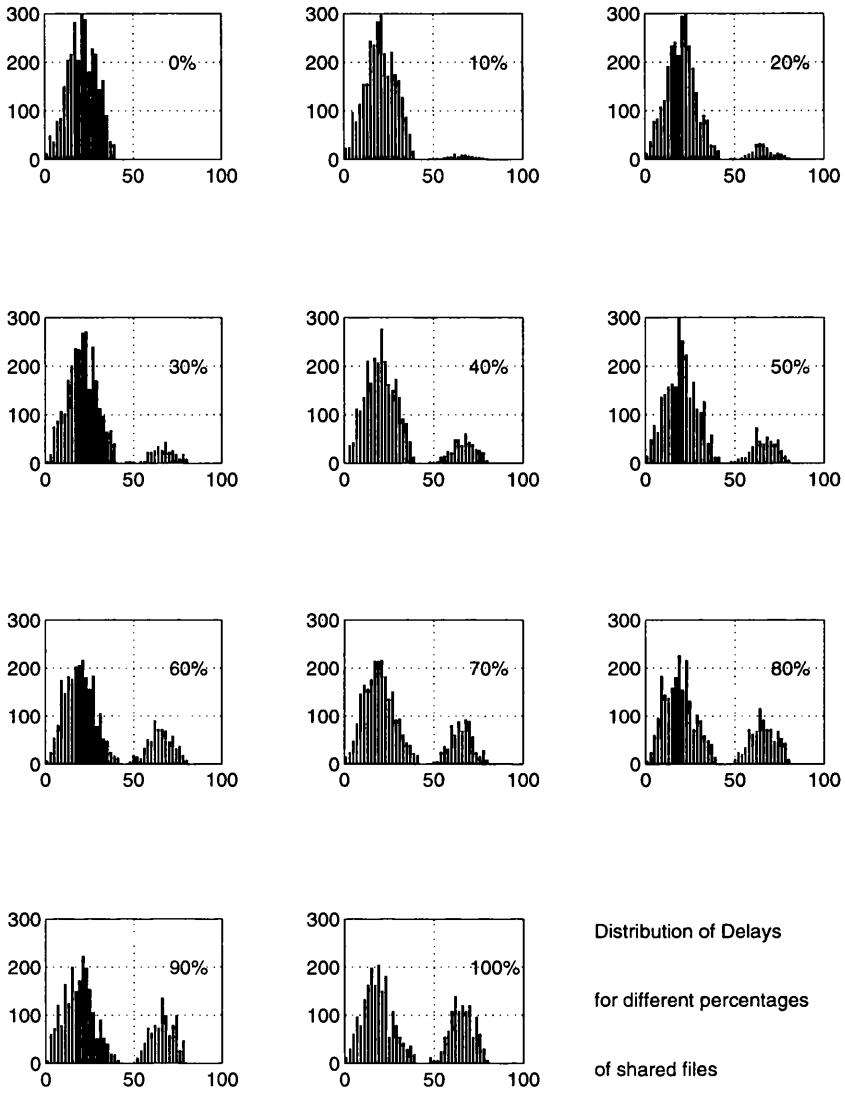


Figure 29. Histogram of delays

The case of no files shared shows a distribution of delays with the maximum at around 20 time units. The explanation to this is as follows. There is a strong relationship between the delays and the size of the network. The request packets travel one hop every clock cycle. Therefore, the time required for a request packet to reach the database of addresses - in time units - equals the distance it has to travel - in number of hops. Since the distribution of the users is random, the arguments developed in Appendix C to obtain the average distance apply here. Therefore, the average distance between a user and the database is X/2, where X is the number of lines in the grid. The measure considered as an indication

of the delays is twice this value, i.e. X, because it includes the time for the information to travel back from the database to the requesting node. The simulations were run for a 20x20 grid, hence the theoretical average delay time is 20 time units. This agrees with the experimental results for no file contention.

As contention is introduced, a new cluster, i.e. group of values in the graph, grows in the histogram of delays. This is because now the two users have to compete for the data files and, as explained in section 5.4.1, more than one attempt may be necessary to obtain a file address. This new component in the graph has its maximum at around 60 time units. This is because the time between two consecutive attempts to obtain one address is set to be Δ= 40. A second attempt will then happen at time=40, so for those cases needing two attempts the average delay is 40 + 20 = 60 time units. The larger the number of shared files, the larger the number of cases needing two attempts, and therefore the bigger the second component becomes. The graphs indicate that the maximum number of attempts is 2 (there are only two clusters in the histogram), which means that a value of Δ=40 time units is enough for the requested file to reach the new location and the database being updated.

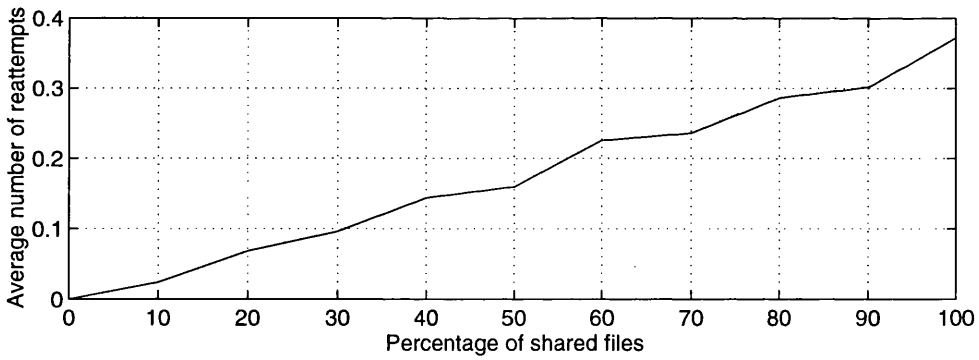


Figure 30. Average number of re-attempts to obtain a file

The case of 100% shared files shows that the two components of the histogram have similar size, but still the one closer to the lower values of delays dominates. This is because re-attempts are only needed if there is contention for a file. This does not happen if:

i) the requested file is available locally, and therefore the addressing function is not performed for one of the users.

ii) a file requested by one of the users is relocated and its new location is updated before the request from the other user reaches the database.

b) Attempts needed to obtain a requested address, i.e. number of times that the user's local node has to re-send the request to the addresses database. The average results are shown in Figure 30. For no files sharing, the average number of re-attempts is 0, i.e. the files are always obtained at the first attempt, because there is no file contention. The average number of re-attempts increases linearly with the number of shared files. For 100% of files shared this value is around 0.37, considerably lower than 1 for the same reason as in ii) above.

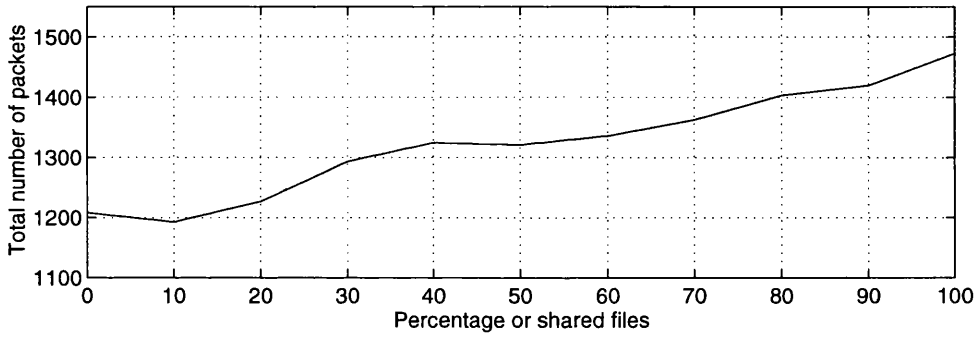


Figure 31. Average number of packets in the network

5.4.2.2. Measurements of performance

a) The signalling load is measured counting the average number of packets present in the network at a given time. The results are plotted in Figure 31 for different percentages of shared files. The graph shows a practically linear growth of the signalling load with the percentage of shared files. This is because the larger the number of shared files, the more likely is there a need for more than one attempt to obtain requested files, and hence the larger the number of signalling packets that are created. This measure is therefore related to the average number of re-attempts (Figure 30), but it is different to it because the number of packets present in the system is related to the distances that these packets have to travel. In other words, if the users are close to the database, the address request and the address response packets have to travel shorter distances, and disappear earlier.

Alternatively, if the local node is far from the database the packets remain in the system for a longer time, contributing to the measure of total number of packets present in the network.

b) The bandwidth usage - again considered as the product of the size of the files times the number of hops that they travel - for different percentages of shared files is shown in Figure 32. This graph shows that this measure does not depend on the percentage of shared files (the minor variations in the graph can be attributed to the averaging process). The reason is that the bandwidth is calculated only for the movement of data files, not for the movement of request and response packets. The number of data files that are moved is independent of the percentage of shared files, because even if contention is high, both users will obtain the files they request within the first two attempts (as discussed in section a) in 5.4.2.1).

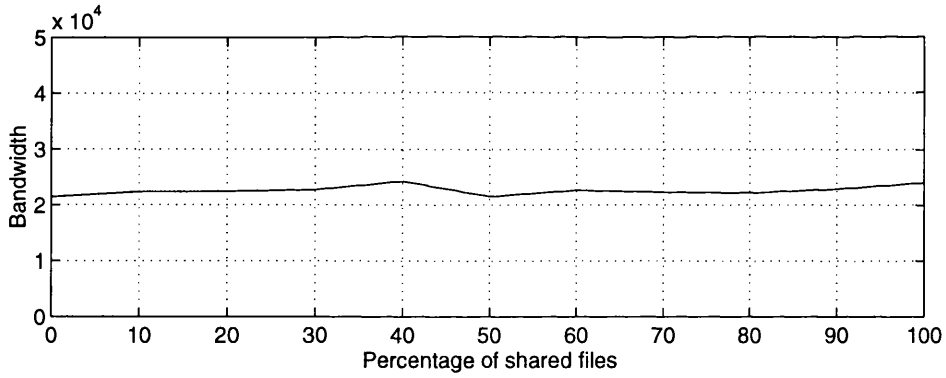


Figure 32. Average bandwidth usage

These results illustrate the importance of analysing the problem from different angles and taking into account different aspects that may be contradictory. The number of shared files does not affect the measure of bandwidth. However, as shown in previous section, it has a great effect in other measures such as signalling load and delays.

5.5. Outcome from the C++ modelling

The principal use of the C++ modelling work was the development and initial testing of the ideas in the modelling framework proposed in chapter 3. This section reviews how the C++ models contributed to each of the four key areas in the framework described in section 3.4.2.

5.5.1. Aim of the models and modelling methodology

In terms of the seven stage methodology proposed in section 3.4.2.1, only stages 1 (model service behaviour) and, 2 (identify path of normal behaviour) have been fully completed with the C++ models. Stages 3 (identify sources of errors) and 4 (introduce errors in models) have been partially developed. Thus, performance and QoS degradation as a result of changes in the customer demands have been demonstrated in section 5.4.2. This information could be used to define integrity levels as discussed in the proposed integrity definition in section 3.2. For example, if limits were defined for delay times, this would result in a number of regions from 'optimum' to 'unacceptable' delays. The results in Figure 29 would then provide the bounds in the percentage of shared information between the users for which the delays would remain within the specified regions.

Other potential failure causes have also been detected through the modelling exercise, e.g. incorrect timer values and loss or corruption of signalling messages. Although these failures have not been introduced in the models, the possibility of them causing catastrophic failure has been identified.

5.5.2. Level of abstraction

One objective of the C++ modelling work was to test the suitability of the top-down layered modelling framework described in section 3.4.2.2. In the first version, the Magnet service is modelled at the highest level - the functional layer. This was very useful in order to understand essential features of the service and begin to identify areas of complexity and potential integrity degradation. Three areas requiring further analysis were identified and included in the second version, namely the addressing function, contention issues and quality of service measures.

The layered framework proved to be a very efficient tool to breakdown the problem and understand the level of detail required for each scenario, providing the ability to expand on the areas of highest interest for the specific issues being investigated. The C++ models also demonstrated the use and importance of the Customer layer in the framework, in the form of the population models examined – random and diffusive.

5.5.3. Modelling languages and tools

One issue that became rapidly obvious with the initial modelling work was the difficulty to keep track of the modelled system as its complexity increases. This identified the need for user friendly visualisation tools to assist understanding of the systems being simulated. Specifically, for visualising information flows between the different components (that is the signalling).

C++, as most high level programming languages, is a powerful language that can be used to build virtually any kind o application. Thus, it would have been possible to build these visualisation capabilities required for the models using C++. However, this would have required considerable time and effort, and it was felt that this would deviate the work from the pursued aim e.g. integrity analysis. Thus, instead of developing these visualisation capabilities it was decided to look at another language with built in visualisation capabilities instead (SDL).

5.5.4. Inputs to the models

Building the C++ models permitted identification of the three main inputs: service, network, and customer (user and population) models.

When the C++ models were created, the importance of maintaining the independence of the three components had not been fully understood. Thus, in these models the *service* and the *population* components are embedded with the network components. This makes these models completely tailored to the Magnet service, and difficult to modify or re-use to model other services. The identification of this limitation led to some of the key ideas discussed in section 3.4.1 regarding re-usability of models.

The *network* architecture has been kept very simple, because the aim of the work is to identify integrity threats in the 'intelligent' levels, as opposed to the network levels. The square grid network topology was maintained in all cases, examining only the effect of the size of the network. More attention has been paid to the identification of dynamic errors due to performance limitations (see section 3.4.2.4, network models). The need for performance models has been identified in the Magnet service as a main component to examine integrity. In particular, the values of the different time outs were identified as a critical design factor.

The C++ model focuses mainly on the effect of the *customer* models in the performance of the Magnet service, by investigating three aspects namely, number of users, their mobility patterns and amount of information shared between them:

a) Number of users: the single user versus the two users cases have been studied and the results showed how increasing the number of uses introduces QoS and performance degradation.

b) Mobility patterns: a study of the random versus the diffusion mobility patterns based on bandwidth usage has been carried out for the one user case. The comparative analysis of both relocation patterns has demonstrated the impact of different population models in the service, and the need to take these factors into account for the service design.

c) Percentage of shared files between the users: the simulation results have shown that increasing the percentage of shared files results in an increased number of packets (Figure 31) and causes degradation in the QoS, specifically increased delay times (Figure 29 and Figure 30). On the contrary, the bandwidth usage does not change with the percentage of shared files (Figure 32), i.e. the same amount of data is moved.

In summary, the exercise has shown how small variations in the number and distribution of users can produce very different results. This demonstrated the importance of examining different customer models in the design of services.

5.6. Conclusions

This chapter contains a description of the C++ model of the Magnet service and the simulation results obtained for it. The C++ modelling exercise was designed in order to start to build and put into practice the ideas in the modelling framework described in chapter 3 and to begin to identify some of the relevant integrity aspects for the Magnet service. The value of the model is not only the numerical results obtained from the simulations, but mainly the knowledge and expertise gained from the modelling work which contributed strongly to the proposed methodologies. The models have been kept very simple, in order to facilitate understanding of the system and identification of relevant areas and key issues. If the models used at these early stages were too complex, this complexity would obscure the essential features, and would make it extremely difficult to interpret the results. The complexity of the models can grow gradually as work progresses, and the results obtained from different stages can be reused and incorporated in later stages to build more advanced models.

The effect of different design strategies has been considered in this exercise. This has been the beginning of a much more complex process, and only the first steps have been taken. For example, in the signalling arena, next stages could include the introduction of a probability of losing packets and the study of the effects that this would have in the service behaviour. In the way the Magnet service has been designed, the loss of a request packet would not have major consequences, because a new request packet would be generated after the time-out expires. A loss of a packet used to update the address of a file, on the contrary, would have catastrophic consequences, because it could render that file inaccessible. A subsequent step would be the design of recovery mechanisms to solve this type of problems.

The possibilities for further development of the model of the Magnet service are numerous, and due to time constraints only a small number of them could be explored. The following areas had to be abandoned:

a) deeper study of the file management strategies and comparison to other schemes,

c) interconnect scenario, where the Magnet service is offered across two interconnected networks,

d) analysis of other population models,

e) introduction of different time-out strategies, i.e. timers of variable value.

With the modelling work presented in this chapter some limitations in the use of C++ as the modelling language became apparent. These limitations are concerned with the representation of information. C++ and the development tools do not provide mechanisms for visualising interactions between components in the model, which is one of the key capabilities needed. This makes it difficult to understand the behaviour of the system and to identify areas of conflict. Moreover, C++ is not a very user-friendly language, and here it is believed that in the context of commercial organisations other more familiar tools, easier to learn and use, are required. Continuing with the modelling work using C++ would have resulted in an interesting academic exercise, but at this stage, the investigation of alternatives was given priority. Hence, it was decided to consider other possibilities for modelling languages and tools, and SDL was chosen as the language to investigate. A new model of the Magnet service was created using SDL. The objectives of this SDL model were twofold. One area was to use it to further investigate the Magnet service, and to assess the value of the modelling work to identify integrity regions and for integrity preservation. The other objective was to investigate the

applicability of SDL and the SDL tools employed, identify limitations and propose solutions. This is dealt with in the following chapters.

Chapter 6

The SDL model

6.1. Introduction

The C++ modelling work identified an important limitation in the use of high level programming languages to model and simulate telecommunication services. This is the lack of powerful visualisation techniques to observe communication flows between the different components in the system. This is one reason why it was decided to investigate the use of SDL and the existing SDL tools for modelling and simulating the Magnet service. The behaviour-orientated nature of SDL made it a very suitable candidate for the description of service behaviour. In addition, used in conjunction with MSCs, SDL provides a better way of visualising message flows and interactions in the system. Other advantages of SDL are that it is an international standard, it is easy to use, it has proven to be the most used method in telecommunications applications (currently used by a large number of operators and manufactures), and there are advanced software tools available to build and test SDL systems. Moreover, there is a strong international research activity in SDL, and modifications are introduced to the language to respond to new demands. Finally, as part of the research work carried out in a major UK operator it was discovered that SDL was being widely used in that organisation, and the author had access to some of their expertise and resources.

A new model of the Magnet service was developed using SDL. The model was built based on the outcome from the C++ model. Both models have some common characteristics, but there are some important differences driven by the intrinsically different nature of the two languages and the things they are best suited for. The objectives of the SDL modelling work are as follows:

a) to explore some of the areas identified with the C++ model in more detail,

b) to identify integrity parameters for the Magnet service and to use the simulation to detect and prevent network integrity degradation,

c) to illustrate how modelling techniques can be used for measuring and categorising the degree of integrity of a system, and

d) to investigate the suitability of SDL as a language for modelling services in the context of integrity analysis, identify limitations and propose solutions.

This chapter is organised as follows. The SDL model built for the Magnet service is described in section 6.2. For the sake of readability of this document, very detailed descriptions of the model are avoided, but the complete model is included in appendix F. The SDL model has been used to simulate the Magnet service, in order to obtain integrity measurements. The limitations found when using SDL for this purpose are described in section 6.3. In particular, a major issue is the lack of performance constructs in SDL. A literature review on the area of performance evaluation, and on the integration of performance evaluation and formal description techniques is included in sections 6.4 and 6.5. The approach taken in this project to introduce performance information and measurements in the Magnet service is described in section 6.6, followed by the conclusions in section 6.7.

6.2. Description of the SDL model

This section contains a detailed description of the SDL model of the Magnet service. This description has been divided in three sections. The first section details the structure of the system, the main SDL components and their connections. The second section outlines the most significant data types used. Finally, the third section describes the behaviour of the SDL model of the Magnet service.

6.2.1. Structure

In order to cope with complexity, the system has been decomposed in a number of structural components in a hierarchical fashion, making use of the SDL concepts of system, blocks, processes and procedures. The concepts of block and process *types*, introduced in SDL 92, have also been used [Olsen et al. 94].

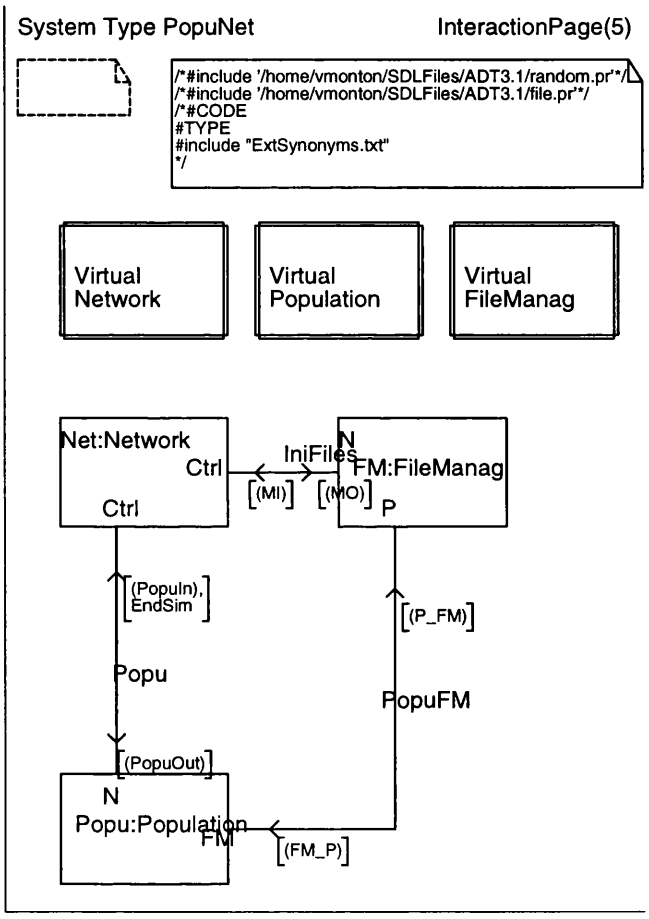


Figure 33. SDL system level: structure of the whole system

The top level, i.e. the system level, is represented in Figure 33. A system is composed of three blocks, which are instantiations of three block types; namely the Network, Population and FileManager block types. The functionality of each of these blocks is as follows:

The Population block contains the population models, i.e. the characteristics of the population of users in the system and the mechanisms for the mobility of users. The File Manager block deals with the functions related to the management and relocation of data files in the Magnet service. It contains the database with all the current addresses of the data files. The Network

block contains the architecture of the network. The structure of the network is given as a collection of blocks representing the nodes in the network. As in the C++ model, these nodes are connected in the form of a square lattice. There is an additional block, named the Network Interface block, connected to all the nodes. The function of this block is to permit the transfer of signals from the outside of the network block to any of the nodes.

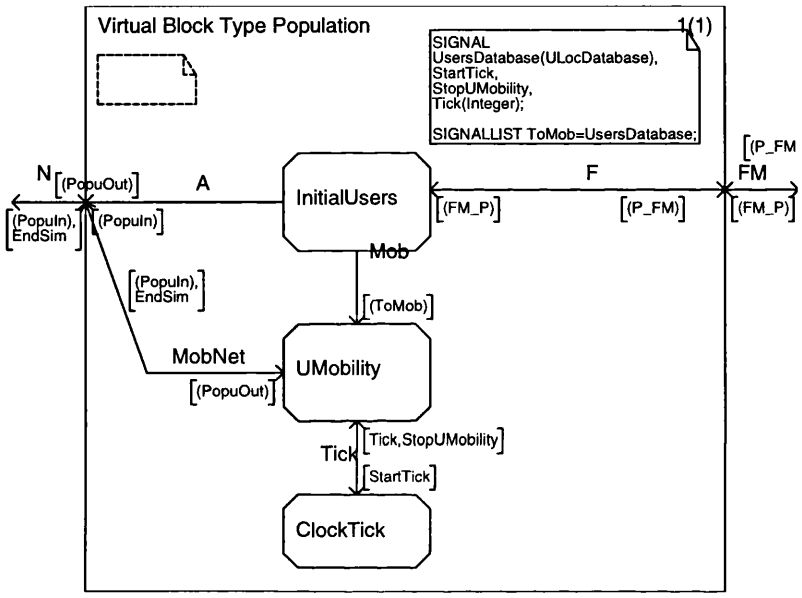


Figure 34. The block type *Population*

6.2.1.1. The Population Block

The *Population* Block, shown in Figure 34, contains the population models. It is composed of three processes. The process *InitialUsers* creates the initial user profiles and distributes them in the network randomly. The process *ClockTick* governs the passage of time, by sending out at regular intervals a signal called *Tick* that initiates the movement of users. Finally, the process *UMobility* deals with the relocation of the users upon receipt of the signal *Tick*.

6.2.1.2. The File Manager Block

The block *FileManag* (Figure 35) contains three processes. The process *NewFiles* distributes the initial data files around the network at random locations. The process *FileLocations*

represents database containing the locations of all the data files. It is connected to the process *DelayProcess*, which models the delay time associated with each database look up (this will be described in more detail in section 6.6.1.1).

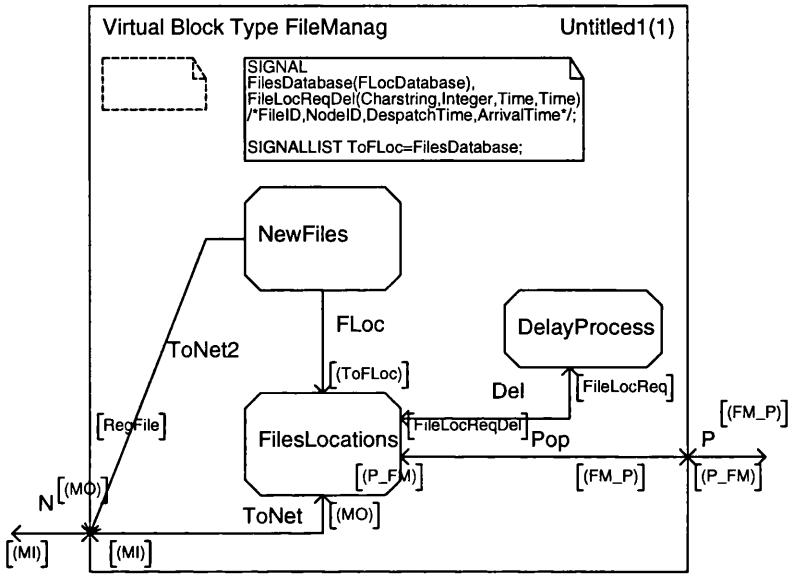


Figure 35. The block type *FileManager*

6.2.1.3. The Network Block

Figure 36 shows the SDL structure representing the architecture of the modelled network. Three block types are defined, namely *BaseNode, FirstNode* and *NetIF* (which stands for network interface). The network consists of eight instantiations of the block type *BaseNode* and one of the type *FirstNode,* each representing one node in the network, plus an instantiation of the block type *NetIF*, named *NIF*, representing the interface between the network block and the outside world. The *NIF* block is connected to all the nodes in the network. Its function is simply to direct the incoming signals to the correct node in the network, and it will not be described in more detailed. The type *FirstNode* is defined as a redefinition of the type *BaseNode.* The only difference between the two types is that the type *FirstNode* can receive the signal *CreateLink* from the *NIF* block. This is needed only for constructing the model, and is not relevant to the behaviour of the simulated system; therefore, it will not be described further.

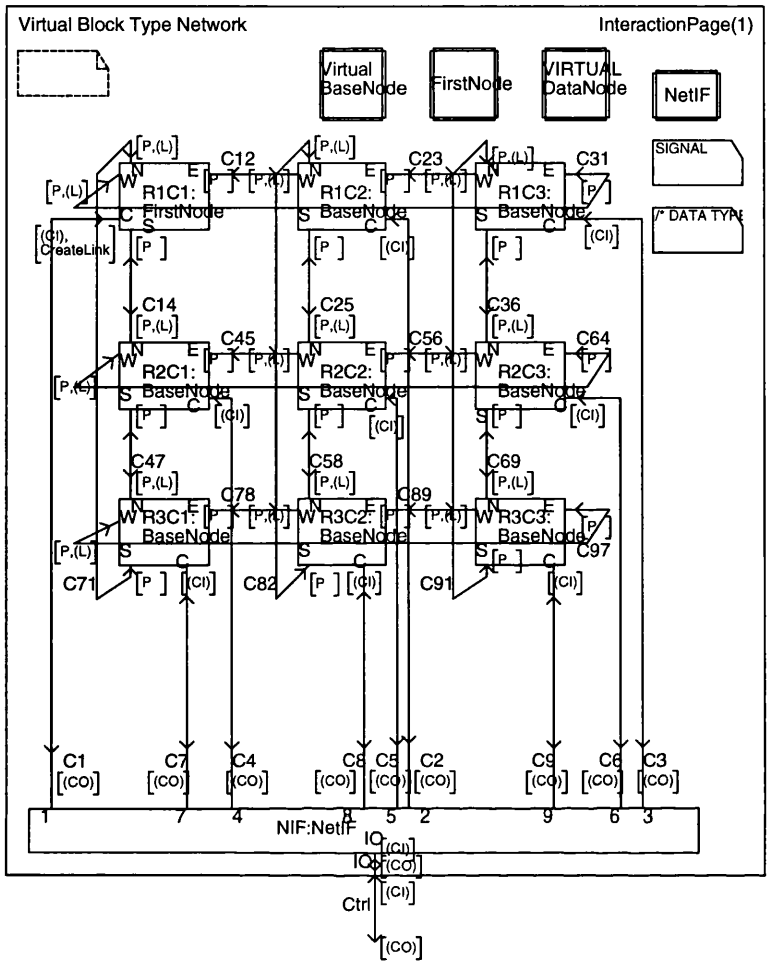


Figure 36. The network block

The nine nodes are connected in the shape of a square lattice, with periodic boundary conditions, i.e. there are no edges in the network and hence all nodes are equal in behaviour. The reason for this is to facilitate the analysis of the results. The naming convention for each node block is based in the spatial representation of the network on the page. Each node block is named by their position in the lattice in terms of row and column. The rows are numbered 1 to 3 from top to bottom, and the columns are numbered 1 to 3 from left to right. For example, the block R1C2 is the block in row 1 and column 2. It must be said that these block names are in fact hardly ever used. For the description of behaviour, each node is assigned a unique node identifier, which is in fact an integer number between 1 and 9. The node identifier is stored in every process within each node block in the form of an SDL variable, and their values are assigned at run time. The correspondence between block names and node identifiers is shown in Figure 37.

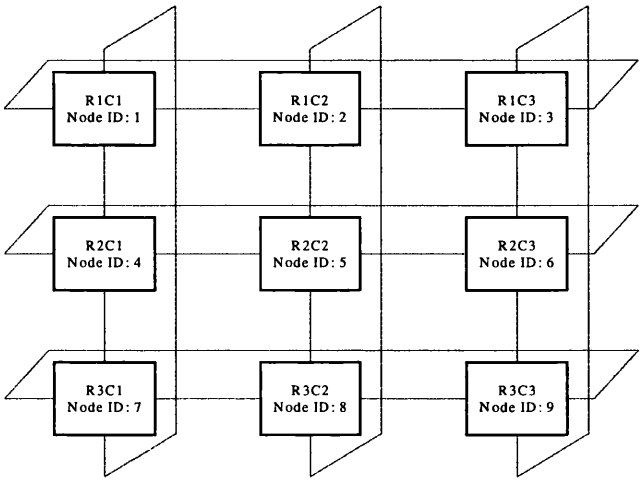


Figure 37. Node block names and node identifiers

Each node block has five gates. The gates *N, S, E, W* (naming convention standing for north, south, east, west) are connection points to each of the four adjacent nodes, and the gate *C* is the connection point to the network interface block.

6.2.1.4. The Node Blocks

A node is modelled as a layered structure consisting of four layers, namely the user layer, the application layer, the network layer and the data link layer. The layered structure resembles the layered architecture of the OSI basic reference model for open systems interconnection [ISO/IEC 7498-1:1994], but for the sake of simplicity it has been simplified to make the model more manageable and tailored to the case study. Each layer is modelled as an SDL block type, as shown in Figure 38. Thus, each node instance consists of an instance of the *UserLayer, ApplicationLayer, NetworkLayer* and *DataLinkLayer* blocks connected in a top down fashion.

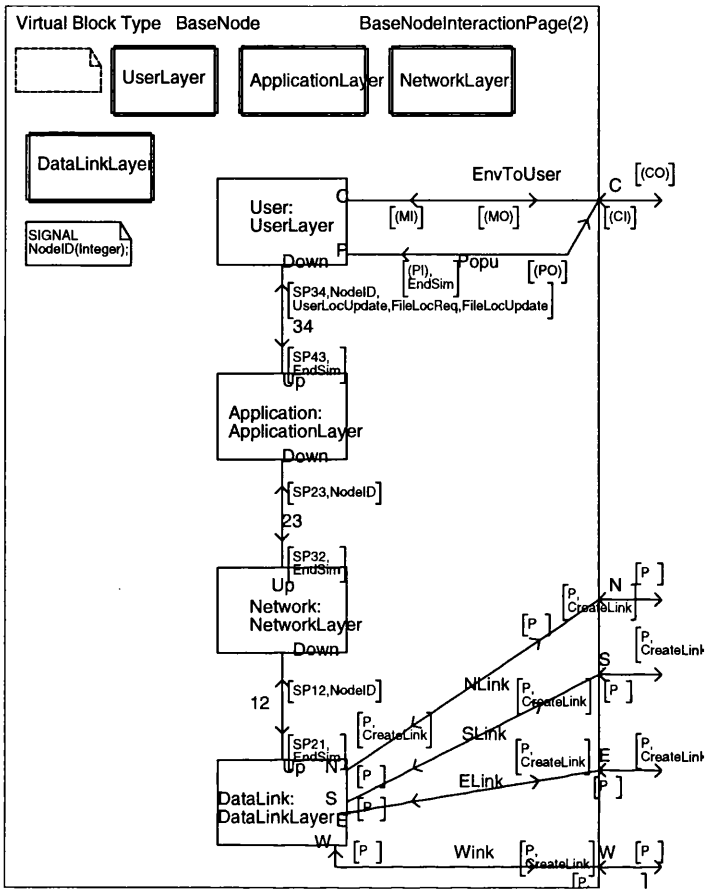


Figure 38. Structure of a node

The ***DataLinkLayer*** block (Figure 39) deals with the transmission of information between two adjacent nodes. The information is sent in the form of SDL signals, called *packets* in the model, which carry one parameter of a type also called *Packet*. The data type *Packet* is defined as a structure in which two of the members are the origin and the destination nodes (see section 6.2.2). The four gates *N, S, E* and *W* correspond to the gates with the same names in the block *BaseNode*, and they represent the links to the four adjacent nodes in the lattice.

The *DataLinkLayer* is modelled as a collection of SDL processes. The process *MakeLink* of type *LinkCreation* exists only for the initialisation phase of the model, and does not have any functionality related to the behaviour of the Magnet service. The process *ProbeIB* is included for the introduction of measurements, and hence it does not describe service behaviour either. The only processes that really represent service behaviour are the *UpperDataLink, InputBuffer*

and *OutputBuffer*. The *UpperDataLink* process is the interface between the *DataLinkLayer* and the *NetworkLayer* above. The process *InputBuffer* receives all the incoming packets from other nodes and processes them in a FIFO (first in-first out) fashion, whereas the *OutputBuffer* deals with the outgoing packets. The process *PacketDelay* is added as part of the scheme to introduce performance in the model, as discussed in section 6.6.

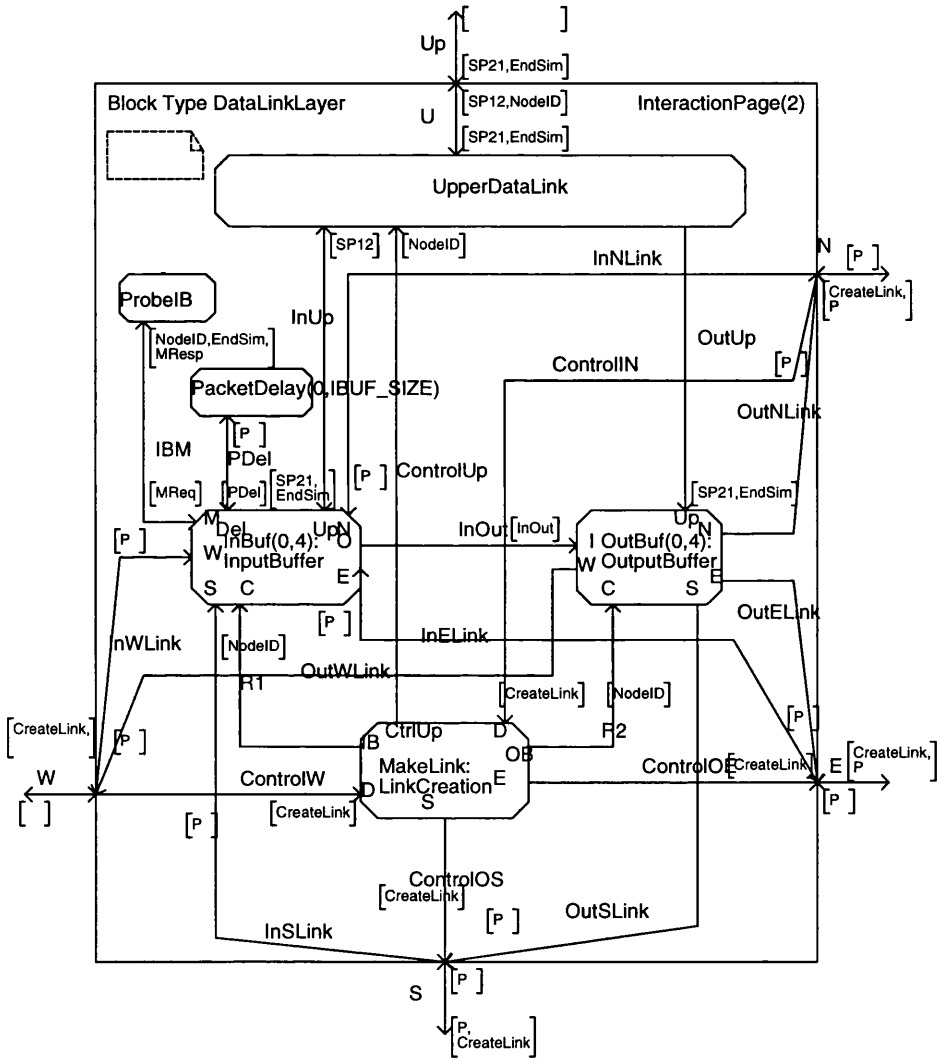


Figure 39. The DataLinkLayer

The next layer up is the ***NetworkLayer*** block (Figure 40), which performs the routing functions in the node. A non-adaptive, static routing algorithm is used. Each node has a routing table containing a list of all the nodes in the network and the link on which the

information must be sent from this node to each of the other nodes. The routing algorithm uses the shortest path for neighbouring nodes, whilst a random link is selected for the rest. For example, a packet from node 1 to nodes 2, 4, 7 or 3 (see Figure 37) will be sent directly to the destination, since these are neighbouring nodes (remember the lattice has periodic boundary conditions). If the destination is any other node, the packet will be sent randomly to any of the neighbours, where the same routing procedure is repeated. The routing functions are dealt with by the process *Routing*. In addition to the process *Routing*, the block *NetworkLayer* also contains the process *UpperNet* and *LowerNet*, which represent the interfaces to the *ApplicationLayer* and *DataLinkLayer* respectively.

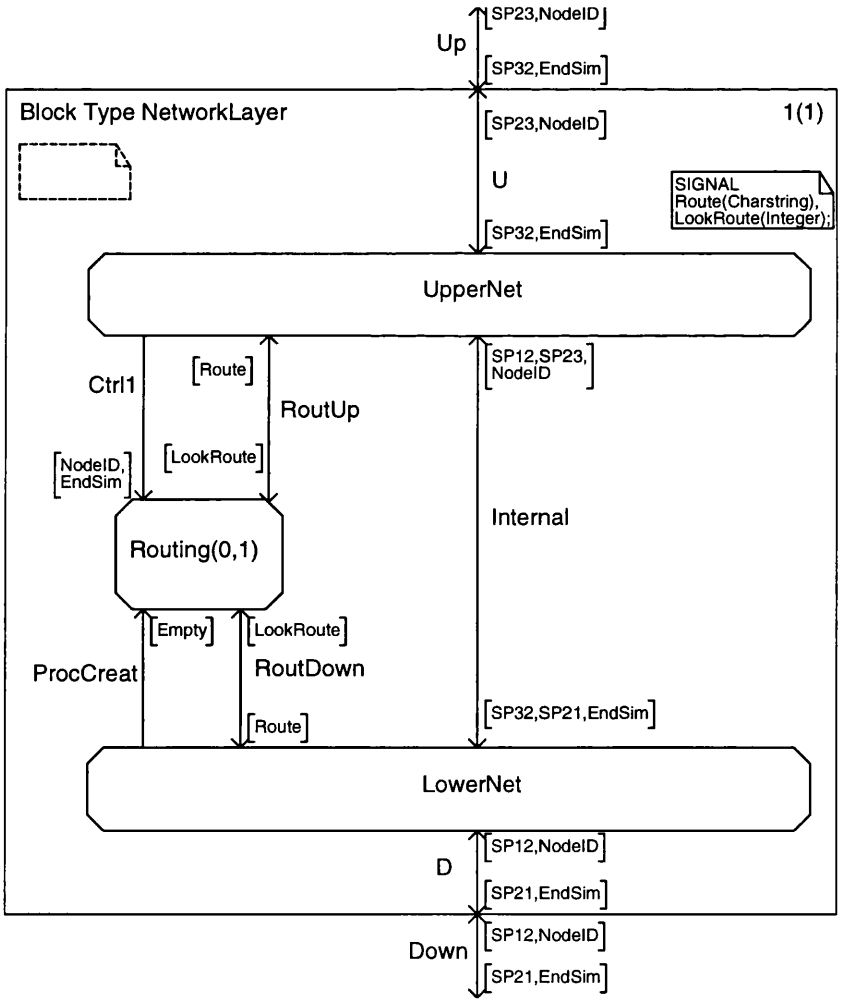


Figure 40. The NetworkLayer

The ***ApplicationLayer*** (Figure 41) deals with the functions related to the management of users and files registered in the node. This is performed by the processes *RegisteredUsers* and *RegisteredFiles* respectively. These two processes operate in a very similar fashion. The process *RegisteredUsers* contains a database with the profiles of the users registered in the node. This users database is implemented by a user defined data type (see section 6.2.2). Similarly, the process *RegisteredFiles* contains a database with all the files present in that node. These processes perform operations such as registration or deletion of a new user/file, in response to requests coming from the layers above or below through the interfaces *UpperApp* an *LowerApp*. Two additional processes have been included for the sake of taking measurements during the simulation, namely the processes *ProbeUsers* and *ProbeFiles*.

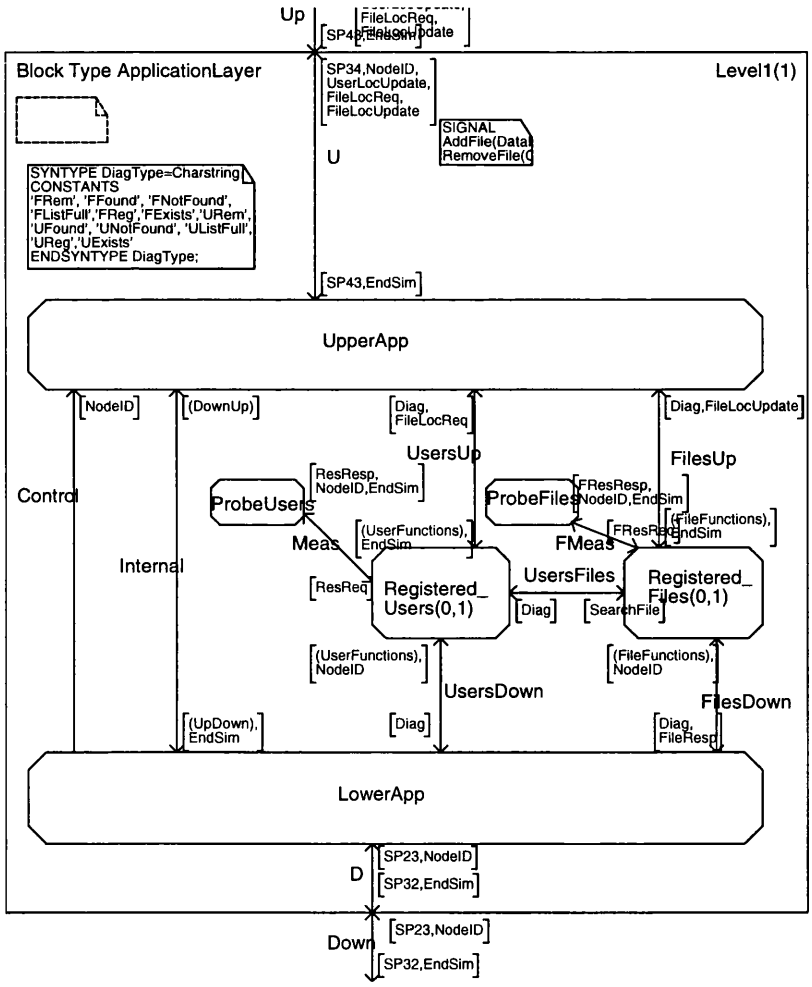


Figure 41. The ApplicationLayer

The top layer within the node structure is the ***UserLayer***, shown in Figure 42. This layer deals with the following functions:

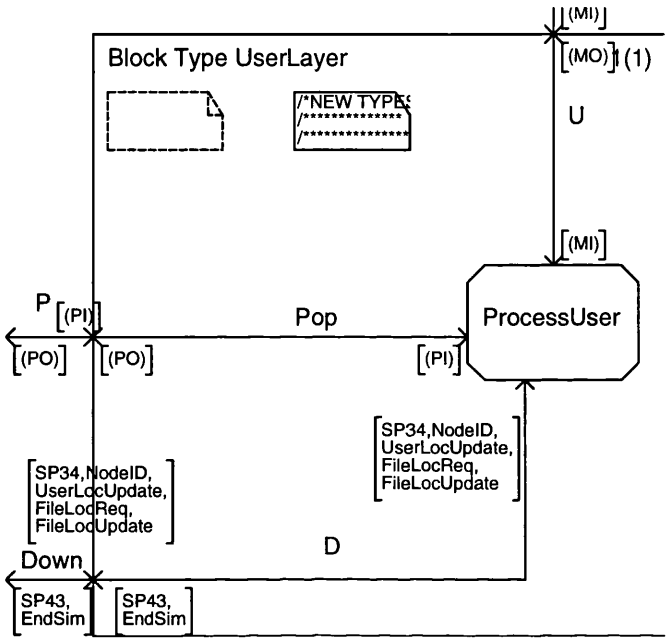


Figure 42. The UserLayer

a) Interaction of the users with the system. All the signals related to the registration of new users in the node and the request for service are sent to the *UserLayer*. In the model, these signals come in fact from the block type *Population* which, as previously stated, contains the population models.

b) Handling of signals related to the management of files. It might be regarded as unusual to do this at the *UserLayer*, since these functions are not driven by the users themselves, but by the file management strategy implemented in the system. Perhaps the most intuitive way of modelling these file management functions would be to place the database of data file addresses inside one of the nodes in the network (as it was done in the C++ model). Instead, for the sake of simplicity in this model the database has been placed externally, in the block *FileManager*, outside the network block. Hence, the signals related to the file management are not treated as inter-node signals (in which case they would have been exchanged in the form of packets at the *DataLinkLayer*). Instead, they are sent directly from the block *FileManager* to the *UserLayer*. This approach simplifies the model, because it avoids extra functionality to transform these signals into packets and let them

be treated as ordinary packets. In terms of the modelling methodology described in section 5.3.2, the addressing function is only modelled at the functional level. No detail on how this is implemented by the network is included.

c) Interaction with the layer below, i.e. the *ApplicationLayer*

6.2.2. The data

For the representation of data in the system, the SDL concept of user defined data types has been used. The operators associated to the data types have been implemented directly in C code, instead of SDL notation[1]. This reduces the complexity of the SDL specification of behaviour and makes the handling of lists more straightforward. In the model, extensive use of lists has been made. For example, the database of users registered in a node, the list of files present in a node, etc. have been implemented as arrays, and the operations such as adding, removing or searching for an element in the array have been implemented in C. A discussion of the most relevant user defined data types is included next.

UserDatabase

This type is used to store a list of users, and it is defined as an array of user profiles. The user profiles are stored in the form of another user defined data type called *UserProfile*, which is a structure with 5 components, namely the user ID, the previous and current locations of the user, the number of data files associated to that user and the list of these data files.

The following operators are defined for the *UserDatabase* type:

i) Search: searches for a user ID in the array and returns *Found* if the user is in the list and *NotFound* otherwise. *Found* and *NotFound* are the two possible values of the user defined type *SearchResType*.

ii) GetUser: takes a user ID as input and returns the user profile if the user ID is in the list.

iii) ListFull: checks whether the array is full, and returns True or False.

iv) Count: returns the number of user profiles in the users database.

---

[1] C is the programming language that our tool permitted to incorporate to the SDL descriptions.

v) Add: adds a user profile to the list. The insertion is made in the first available position in the array.

vi) Remove: removes a user from the users database. If the user ID was not part of the list, nothing happens.

vii) Clear: deletes all the entries in the users database.

A number of other data types similar to the *UserDatabase* type have been defined in the model, namely

a) *UserDatabaseShort*: like *UserDatabase*, but the elements in the array are of type *UserProfileShort*. This type is a simplified version of the type *UserProfile*, where only the names of the data files associated to the user are included, and not the whole file. This is a more realistic type, since a user profile should only contain a reference to the data files, and hence it is the most widely used. The type *UserDatabase* is there for historical reasons.

b) *UserList*: This type is simply defined as an array of *charstrings*. Each *charstring* represents a user ID. This type is used in the block *Population* to handle the movement of users. The operators Search, ListFull, Add and Remove are defined, in a similar way as described above.

Types for handling the communication between layers

An OSI-like approach has been chosen for modelling the nodes (see section 6.2.1.4). The communication between the layers within a node is done via service primitives carrying parameters or service data units. A basic description of the OSI layered model and examples of modelling protocols using SDL can be found in [Belina et al 91], chapter 9.

Service data units are transmitted as parameters in service primitives. These service primitives are modelled as SDL signals. The naming convention used in the model is SPXY, where SP stand for service primitive, X is the originating layer and Y the receiving layer. The layers have been numbered from 1 to 4, in a bottom up fashion: DataLinkLayer = 1; NetworkLayer = 2; ApplicationLayer = 3; UserLayer = 4.

The format of a level N service primitive data type is represented in Figure 43. The length field refers to the number of parameters carried by the service primitive. The list of parameters is an element of the type *ParamList*, defined as an array of elements of the defined type *Parameter*. The *ParamList* type contains the usual operators for list handling described above, e.g. Search, ListFull, Add, Remove, GetParamByName and Clear. The data type Parameter is a structure with three fields, namely the parameter name (charstring), the length field (integer) and the data field or values (charstring).

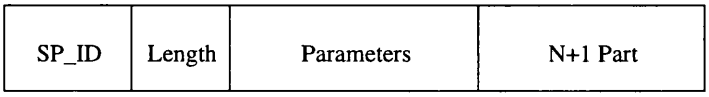


Figure 43. Format of a level N service primitive data type

6.2.3. Behaviour

6.2.3.1. General description

For the sake of simplicity, the random relocation population model has been used (see section 5.3.2.1 for more detail). Therefore there is no correlation between the new and the previous locations of a user. In order to examine the effect of the contention for the data files, the multi-user case is designed so that the different users in the system share a number of data files. All the users follow the random relocation mobility pattern, and they move at the same times but to locations independent from one another. It is assumed that data files are only requested on the same clock cycle when the users change location. During the time intervals between two consecutive changes of location, no new files are requested. This represents the case of users only requesting service when they change location at certain times of the day. An example of this type of service could be a call forwarding service, assuming that the vast majority of users redirect their calls when they move from home to office locations at specific times of they, i.e. rush hours. This is just a simple user relocation scheme, and other alternatives are possible. Figure 44 illustrates the basic behaviour for the case of two users in the system, when both users request for data file F. The data files present in the network can be divided in three categories, namely those unique to user 1, those unique to user 2 and those shared between both users. The selection of data files for each user is done at random.

When a data file is needed in a node, the addressing function is used to obtain the location of the file. As previously stated, the file addressing function is modelled at a high level. The transmission of the addressing messages is not implemented via a routing function across the nodes in the network; i.e. they move directly between the node and the database, without any associated delays. There are two types of addressing messages. The first type is request messages sent from the user's local node to the database of addresses, asking for the address of a data file (*FileLocationRequest*). The second type is the response messages sent from the database to the user's local node (*FileLocationResponse*). This response is sent back some time units after it is received. This delay is given by the parameter called F_LOC_DEL. When a request for a file is generated in a node, the file is first looked for in this local node, and only if the file is not available locally the address request is sent to the database of addresses. If there were no contention issues or other possible error situations, the local node would receive the response after F_LOC_DEL time units. However, things are more complex than that.

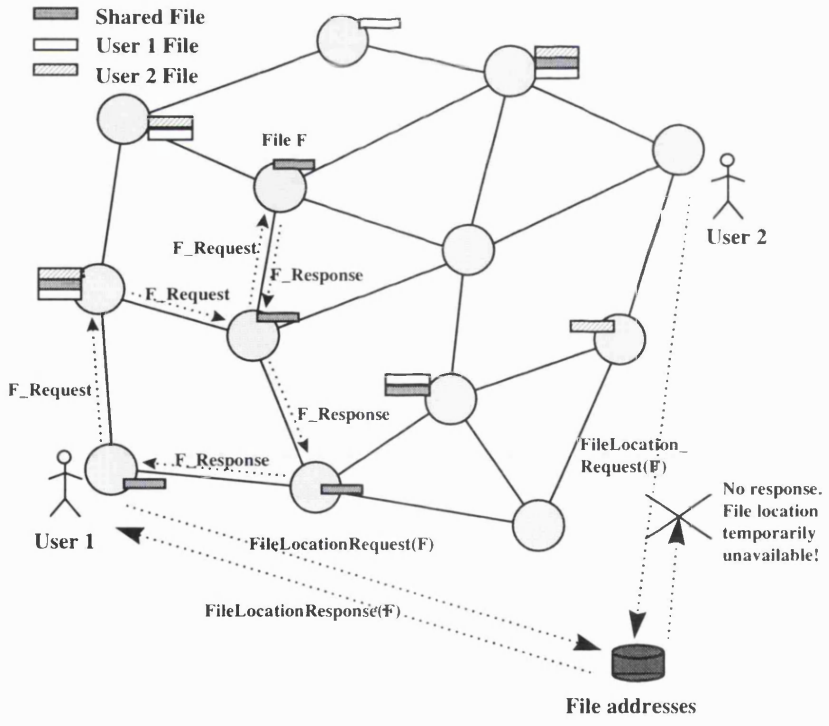


Figure 44. The modelled system for the two users scenario

As discussed in section 5.4.1, errors may occur if the two users request the same file at the same time. The same strategy employed in the C++ model is used in the SDL model: a flag is introduced in the entries of the addresses database. A file address is only dispatched when the flag for this file is not activated. This creates the possibility of the requesting node never obtaining a *FileLocationResponse*. In the SDL model this possibility is increased due to the probability of losing packets (see section 6.6.3).

As in the C++ model, the approach taken is to introduce a time out Δ, so that if the requested information does not arrive after Δ time units, the local node issues another request. This situation repeats until the desired address is obtained. The value of Δ is chosen small compared to the speed of movement of the users. Therefore, the first re-attempts are made well before the users leave the node, so it can be assumed that the user will eventually obtain the address of the file.

Once the location of the needed file is known, a similar process is started in order to obtain the information needed. A *FileRequest* packet is sent to the node where the file is present, and this node will send a *FileResponse* packet back. These packets have to travel to the destination node according to the routing mechanism, and hence they are subjected to the packet transmission delays. They are also subjected to the mechanism for losing packets described in section 6.6.3. Thus, another time out is necessary to restart the whole process in case the file is not obtained within a certain time interval (see section 6.6.1.1).

6.2.3.2. Request for service from the users

Every time a user registers in a new location, the new location needs to be updated in the database of users in the population model, and the user needs to be de-registered from the previous location. To achieve this, when a user registers in the new location a *UserLocUpdate* message is sent to the process *UsersMobility*, in the block *Population*, and a *UserRemReq* is sent to the user's previous location, in order to delete that user from its database. These messages are sent after confirmation that the user has been registered in the new node is received by the *UpperApp* process in the *ApplicationLayer*. This confirmation is the receipt of the *Diag* (diagnostics) message *Ureg* (user registered).

The *UserRemReq* is passed down to the *DataLinkLayer*, and it is transformed into a packet in the *InputBuffer* process. This *URemReq* packet is sent to the user previous location to de-register the user. The *URemReq* messages are real packets that travel around the network according to the routing protocol, and hence may be lost. In that case, the user would not be de-registered from the previous location. The only effect of this would be that the user databases in each node would contain more entries than users there are in fact in the node. Since the size of the users databases is not considered a limitation, this has not effect in the behaviour of the model. Hence, for the sake of simplicity and in order to facilitate the measures about distribution of users, in the model the user de-registration packets are never lost.

The request for files for a user starts after the *UserLocUpdate* arrives in the process *UsersMobility*. The data files to be requested are selected within the process *RegisteredUsers*, in the application layer of the nodes, because it is here where the database of users and their associated files are kept. The requested files are selected at random, using the SDL 'ANY' construct. For each data file, 'ANY' is used to generate either a '0' or a '1'. If '1', the file is requested, i.e. the probability of a file being requested is 0.5. Note that, contrary to the C++ model, there are not different priority files, so the probabilities of requesting a file are calculated in the same way for all the data files.

For a requested file, the first thing is to check if that file is available locally, and this is done sending a *SearchFile* signal to process *RegisteredFiles*. When this signal is sent, the *RegisteredUsers* process waits for a reply in a state called *WaitFileExistsCheck*. In this state, only a *Diag* signal carrying the result of the check can be consumed, but other signals arriving in the process are saved. If the requested file is not available locally, a *FileLocRequest* is sent to the *FileManager* (i.e. database of file locations) and the File ID is added to the list of expected files locations in the node (*ExpectedFLocList)*, in the *UserLayer*. A timer is included at the *UserLayer* to reissue requests if the *FileLocResp* does not arrive. When the *FileLocResp* is received at the *UserLayer*, the file ID is removed from the *ExpectedFLocList*, and an SP43 signal of type *RemoteFileReq* is sent to the *ApplicationLayer*. This propagates all the way down to the output buffer, and the file request is then sent to the corresponding node.

6.2.3.3. Files management

The relocation of files is done using real packets that propagate through the network according to the routing protocol. The originating node sends *FileRequest* packets to the node where the file is stored, and this receiving node sends *FileResponse* type of packets which are suppose to contain the file. Fragmentation of the information contained in the file so that it is sent in several packets is not considered.

The file addresses database is included in the process *FilesLocations* in the block *FileManager*. Each entry in the database is an element of type *FLocEle*, which is a structure with the following fields: File ID, File priority, File location, Flag. The flag is an integer. The default value is -1. The flag is set to 1 to indicate that a file is in transit, and is set back to -1 when the file location update for that file is received.

When a request for a file arrives to the destination node, the request is propagated all the way up to the process Registered Files in the *ApplicationLayer*. Upon receipt of a *FileReq* signal, two things happen in this process. On the one hand, a *FileResp* signal containing the requested file as a parameter is sent to the sender process. On the other hand, a diagnostics message of type *FRem* (file removed) is sent to the *UpperApp* process. This *FRem* message contains the *FileID* and an additional parameter that takes the value -1, to indicate that it corresponds to a file in transit. Upon receipt of this *FRem* message, the *UpperApp* process sends a *FileLocUpdate* signal up to the *UserLayer*, and from here it will be passed on to the *FileManager*. When a *FileLocUpdate* signal arrives in the *FileManager*, a third parameter is checked. This third parameter is named *FPrevLoc*. If the value is -1 it means that the file is in transit, and the flag in the database entry corresponding to this file is set to 1. This makes the file location unavailable until a *FileLocUpdate* signal containing the new location is received. This happens when the file eventually reaches its new destination, i.e. when the *FileResp* signal containing the file arrives in the *LowerApp* process at the *ApplicationLayer* in the destination node. When this happens, an *AddFile* signal is sent to the process *RegisteredFiles*. If the add file operation succeeds, a Diag message of type *FReg* is sent up to the *UpperApp* process. This Diag signal also contains the *FileID* and the ID of this node. Upon receipt of this *FReg* Diag signal, the *UpperApp* process sends a *FileLocUpdate* signal to the *UserLayer*, and this signal propagates to the file manager, causing the update of the file location. Note that if the *FileResp* signals are lost, the locations of the files are not updated.

6.3. Limitations of SDL and the tool employed

This section contains a discussion of some limitations found in the use of SDL as the language for modelling networks in the context of integrity. It is assumed that the reader is familiar with SDL. An overview of SDL is included in appendix D.

A main advantage of SDL is its graphical syntax, the most widely used, since it provides a clear representation of the systems being modelled. However, this advantage is also the cause of a number of limitations and constraints. An obvious problem is the fact that the number of structural components is limited to the size of the drawing area, i.e. a page. SDL provides the *block* construct to break a systems structure down into components in a hierarchical way, but the connections between different blocks at a particular level in the hierarchy need to be included on one single page. Unlike process descriptions, block descriptions and system interaction diagrams can not be split into more than one page. This imposes an important limitation for large complex systems, where system or blocks are decomposed in a large number of sub-components. For example, if a network architecture is to be modelled, where a network is represented by a block containing a large number of sub-blocks representing the different network components. This limitation was encountered when the model for the Magnet service was developed, as it is obvious from Figure 36, and this is for a relatively small system of nine nodes. The main reason why the study was not carried out for a larger system was the increasing complexity of the SDL diagrams as the number of blocks increases, and the difficulty to handle all the required block instances and their connections.

Another limitation arises from the fact that in SDL the dynamic creation of structural components is not possible. All the components of the system modelled have to be included statically; i.e. it is necessary to draw all of them and their connections during the systems specification. This makes it very cumbersome to introduce structural modifications to the systems. For example, consider the case in which an analysis is being carried out to investigate the impact of the size of the network in a particular problem. If a programming language, e.g. C++ were employed, the number of nodes in the network and the way they are connected can be provided as data in a file. This file will be an input to the program, which would take these input parameters and build the desired network at run time. Hence, it is simple to change the size and topology of the network. This approach is not possible in SDL. Suppose that the nodes in the network are modelled as SDL processes. SDL processes can be dynamically created at execution time (although they still need to be declared, i.e. drawn

initially). The number of nodes could then be sent as an input to the system, and the necessary number of nodes could be created dynamically. However, the network topology, i.e. the way the nodes are interconnected, can not be produced dynamically.

The problem mentioned above becomes worse with SDL blocks, since blocks can not be dynamically created. Blocks in SDL are the most natural and elegant way to represent a structure, because they can be hierarchically decomposed into further substructures. However, the use of blocks provides a fixed structure with no flexibility. In the SDL model of the Magnet service, the network is modelled as a collection of SDL blocks connected together (see section 6.2.1). If, for example, the network topology needs to be modified, this would require manual introduction of modifications in the system. SDL processes have the advantage that they can be created dynamically. However, SDL does not support a hierarchy of processes, and this makes them not suitable to represent the structure of complex systems. It seems that the introduction of hierarchy in processes is one of the biggest demands of the SDL users, and there is the possibility that the language will evolve in this direction in the future. However, this can not be guarantied at this point in time.

A major problem in using SDL for integrity modelling is that SDL does not cover performance issues. This is because SDL was designed to specify and describe abstract behaviour. Specifications of logical behaviour must be implementation independent, and hence a specification language does not need to cover implementation issues. In the past few years, however, SDL has experienced major development, and software tools that permit the translation of SDL into an executable language, i.e. C, have arisen. This has encouraged the use of SDL beyond the specification of protocols, as originally conceived. In particular, SDL has started to be used for simulation of telecommunication systems, such as is the case of this project. For this type of applications, the lack of constructs for performance evaluation in the language becomes a major obstacle. The importance of performance measures for integrity analysis has been discussed in previous chapters i.e. performance models are essential to identify dynamic errors. The need to introduce performance evaluation in simulation models is a key issue in this work, and it will be discussed in detail in the following sections.

Other limitations encountered are due to the SDL tool employed (Telelogic SDT). The main one is regarding the impossibility of running simulations in a batch mode. This is important when simulations are used to study the performance of a system. In order to obtain

measurements that are more reliable a number of simulations must be undertaken for the same values of the parameters and the averaged results must be calculated. This must be done for each combination of the value of the parameters. The result is that a very large number of simulations need to be done. The simulation facility is provided by the SDT Simulator, integrated in the SDT package. When generating a simulating application a monitor system is included, whose function is to allow the interaction between the application and the user via a command-line user interface. The monitor system provides functionality similar to high-level language debuggers. The simulator can be run in a graphical mode or in a stand-alone mode directly from the UNIX console, but in any case when the simulator is invoked this interactive monitor system is started and it waits for simulator commands to be introduced. These commands must be introduced manually by the user. There is the option of logging the execution commands in a script that can then be executed. However, the invocation of the script must also be one from the interactive monitor system. This means that it is not possible to run simulations in the background, because the monitor system will only accept commands from the keyboard. This is a rather important constraint when a large number of simulations are needed.

Another problem is to do with the fact that if the value of parameters is changed, it is necessary to recompile the system before it can be executed for these new values. Even if the variables are declared as external and their values specified in an external file, if the values are modified and saved in the same file, re-compilation is necessary. It would be much more efficient if a simulation of a system could be invoked directly from the UNIX console, and the values of the constants could be passed as input parameters to the simulator. This would permit a higher degree of automation of the simulations.

Finally, the visualisation capabilities resulted rather more limited than expected. In particular the use of MSCs was not as effective as anticipated. They are useful for very small systems, or to view the interactions between a reduced number of processes. For large systems, they were found impractical. One reason is the impossibility to fit the whole diagrams in one single area and hence the difficulty to trace the flow of messages, as this requires continues scrolling of the pages. This obscures the global view of the system. Other reasons were the limited ability to manipulate the charts and the obscure labelling used by the tool, which makes interpretation of the graphs difficult. MSCs were used in the early phases of the construction of the model, but they were progressively abandoned as the model grew.

6.4. FDTs and performance evaluation

6.4.1. FDTs and performance

Section 3.3.2.1 contained an introduction to Formal Description Techniques (FDTs) and the advantages of using them. FDTs were originally conceived to describe protocols behaviour and functionality in an implementation-independent fashion. Since a formal specification of behaviour has to be as implementation-independent as possible, specification languages do not cover performance aspects. However, the use of FDTs is being broadened beyond the mere specification of logical behaviour. The formal nature of FDTs makes possible the automatic transformation of FDT specifications into executable formats that can be used as a basis for simulation. And if simulations are to be used to measure real time behaviour, performance characteristics of the systems become necessary. Performance aspects such as response times, effects of processing overheads, packet loss rates, throughput or effect of errors in the QoS cannot be investigated in an implementation-independent way. To obtain performance measurements, in addition to the functional behaviour the quantitative properties of the system have to be specified.

System developers often require performance measures to decide on implementation or design alternatives. Performance models allow for tuning of different parameters in the system (e.g. duration of time-outs, size of buffers, etc.) in order to obtain the best performance within the given constraints. For a given specification, the suitability of the different design and implementation options needs to be investigated and compared. Design and implementation decisions need to be made in order to choose the best option to build the system.

Therefore, in order to undertake performance analysis via simulation from an FDT specification, extra implementation-dependent information has to be supplied. As discussed in [Deck et al. 91] such implementation-dependent information may cover issues involving:

a) parallelism in the execution of the protocol functions

b) scheduling strategies

c) processing speeds

d) size of buffers and queues

e) transmission speed of the medium

f) time-out values

6.4.1.1. Classical performance models

There are two types of approaches to study the behaviour and performance of dynamic systems, namely analytical models and simulation. Analytical models are useful for simple, idealised systems, whereas for real complex systems analytical models are often not possible, and one must resort to simulation. Some of the classical performance models are discussed in [Bochman and Vaucher 88] and briefly described below.

Analytical models

a) Markov models and probabilistic finite state machines.

   In such models, the systems are characterised by a collection of states and probabilistic transitions between the states. Solving the model gives steady state probabilities of being at any given state.

b) Queuing models.

   A system is characterised by a number of resources which process service requests in a sequential fashion. The execution of each service request takes a certain amount of time. When a resource is busy, further requests wait in an associated queue. Arrivals of new requests and service times usually have random distributions.

c) Models with real time constraints.

   These models include timers to guarantee specific response times requests. An example of this type of models are 'timed Petri nets'.

Discrete event simulation

Simulation methods consist of imitating the behaviour of the system by the construction of a simulation model. System performance is then estimated by measurements on the model. The disadvantage of simulation techniques is that they are computationally expensive, and they only provide approximate solutions. The more precise a solution is sought, the more computer time is required. Simulation also presents problems when dealing with rare events, e.g. loss of a message, channel failures, etc. which would require very long execution times.

A possible approach is to combine simulation techniques with analytical models. Simulations can be used only to get a good understanding of the system and to identify bottlenecks. Once this is done, the simulation can be discarded and performance can be obtained from simple analytical models of the identified bottlenecks.

6.4.1.2. Key aspects of performance evaluation

Several formal description techniques, including SDL, are based on finite state machines (FSM). In SDL, the behaviour of a system is constituted by the combined behaviour of a number of processes in the system. Each process is an extended finite state machine, i.e. a finite state machine that can use and manipulate data stored in variables local to the machine. An SDL process communicates autonomously and concurrently with other processes. The co-operation between processes is performed asynchronously by signals. A process can also send signals to and receive signals from the environment of the system. It is assumed that the environment acts in an SDL-like fashion, and must obey the constraints given by the system descriptions. Each process has an input FIFO (first in first out) queue, where the received signals are stored until they are consumed. The length of the queue is unlimited. A process either is in a state (waiting) or performs a transition between two states. A transition is initiated by the first signal in the input queue. When a signal has initiated a transition, it is removed from the input queue (and is said to be consumed). In a transition, variables can be manipulated, decisions can be made, new processes can be created, signals can be sent (to other processes or the process itself), etc.

This section describes some of the key aspects of a performance model for finite state machines (FSM). These aspects include the modelling of time, probabilities and resources.

a) Time

A necessary step to specifying performance is to represent the passage of time. As described in [Bochman and Vaucher 88], there are two main ways in which an FSM model can be extended to do this. One way uses Markov performance models, where transitions are instantaneous, but transitions do not necessarily start as soon as they become possible. The actual time when a transition starts may be defined by a probability distribution. The other possibility is to let the transitions start as soon as possible, but associate an execution time to each transition. In this case, the execution of a transition blocks the execution of others.

Variations of these two basic models are possible. The approach taken in this project is described in section 6.6.

Once a mechanism for modelling time is implemented, it is then possible to introduce performance measurements such as transmission delays, throughput or use of resources.

b) Transmission delays

One essential factor to measure the performance of telecommunication networks is the transmission delays. In models such as SDL, where behaviour is given by a combination of processes, transmission delays can be modelled by modelling the communication between processes. For example, the transmission of a packet from one node in the network to another can be modelled as one process sending a signal to another process. Modelling the delays associated with these transmissions between the nodes is equivalent to modelling delays in the communication between the sender and the receiver processes. More about how to introduce performance in the communication between processes is discussed in section 6.6.

c) Resources

A modelled system will have a number of shared resources that provide service under requests. The performance of these resources needs to be specified. Parameters defining characteristics such as the speed of processors, the degree of parallelism or the scheduling and dispatch disciplines of the resources need to be specified. The time that the jobs spend in the resources will depend on these factors, and is a component to be added to the transmission delays in order to obtain end to end delays.

d) Maximum throughput

The maximum throughputs, together with the delay, are the two basic performance parameters of a transmission medium. In order to model the throughput, the input to the medium can modelled as a resource, and the service time of the resource will limit the number of transmission requests that can be handled.

e) Probability

Systems are subject to unreliable non-deterministic behaviour. This non-deterministic behaviour may result in failure, and hence the importance of being able to model it. SDL provides two constructs to model non-determinism, namely spontaneous transitions and arbitrary decisions [Belina et al 91] [Olsen et al. 1994]. These constructs, however, are not powerful enough, since it is not possible to assign probabilities to them. An alternative is to describe non-deterministic behaviour by means of probability distributions. For example, different probabilities for different branches of execution can be imposed by using a random variable in a decision symbol. The next transition to be executed after the decision symbol is based on the comparison between the value of this random value and a threshold. This is discussed in more detail in section 6.6.2.

Probability distributions are also needed to model aspects other than non-determinism, e.g. in our model of the Magnet service they are used to model the distribution of users in the network.

6.4.2. Integration of formal descriptions and performance evaluation

As stated in previous sections, both formal definitions and performance evaluations are necessary for the construction of models to aid the design and implementation of networks and services. The former are used to describe logical behaviour, whilst the latter are required to analyse real-time behaviour of the implementations.

Traditionally, two separate models are used, one for describing logical behaviour, using some formal description (FD) technique, and another for performance evaluation (PE) using a different formalism more suited to performance analysis. As discussed in [Deck et al. 91] [Bause and Buchholz 90] and [Bochman and Vaucher 88], this approach presents the non-trivial issue of consistency and equivalence between the two models, and validation of the performance model with respect to the formal description. Moreover, part of the effort spent in creating the FD model will have to be invested again for the PE model. For these reasons, it seems preferable to have only one model, i.e. to merge logical behaviour and performance in the same model. This requires tools that allow the automatic or partially automatic implementation, validation and performance analysis from a given FDT description.

The problem of combining formal description techniques with performance evaluation has been researched in the past years, in particular since the development of the FDTs such as Estelle, Lotos and SDL [Turner 93]. The use of a formal description language as a paradigm for performance modelling requires the extensions of the language with temporal and probabilistic specifications. The temporal specifications are necessary to describe the time lapse between consecutive events, and the probabilistic specifications are necessary to describe the selection among different possible behaviours. The integration of performance evaluation with SDL descriptions is not a straightforward issue, and a considerable amount of research is being undertaken in this area (see section 6.5).

The extension of existing formal languages with added features for performance specification must be done with care, to be as consistent as possible with the given standard. A trade off between simplicity and power of the new construct must be achieved. On the one hand, a small set of primitives with simple semantics that can be easily interpreted is desirable. On the other, the new features have to be sufficient to express the real system properties that are to be modelled. In [Bochman and Vaucher 88] some of the requirements to introduce performance aspects into FDTs are discussed, and specific examples are given for Estelle, SDL and Lotos.

FDTs were originally designed to specify OSI protocols. Thus, there is a considerable amount of work done in the area of protocol performance analysis, and in how to combine it with the FDTs descriptions of the protocols. As previously stated, typical problems of performance analysis are the estimation of throughputs and the optimisation of parameter settings (e.g. time-out length or buffer places). Most of the techniques proposed in the past are based on Markovian modelling of the protocol [Bause and Buchholz 90]. The disadvantages of using this approach is that for most realistic protocols the state space becomes too large, and exhaustive analysis is impossible, hence needing to resource to non-exhaustive validation techniques or simulation practises.

Next section contains a review of some of the proposals found in the literature to introduce performance into SDL.

6.5. Introducing performance into SDL-specified systems

As discussed in previous sections, if SDL descriptions are to be used as a basis for performance modelling, it is necessary to introduce some modifications to the descriptions of abstract behaviour. There is a great deal of research on how to extend SDL with additional constructs that allow the introduction of performance characteristics in the systems. A review of some of this work is included in this section.

In [Wohlin 92] a methodology is proposed for performance analysis at an early stage in the development cycles. The methodology is formulated for a general case (independent of description), and its applications for the particular case of transforming SDL system specifications into SDL descriptions of the original system from a performance point of view is discussed. The basis of the proposed methodology is to divide the problem domain into three partially independent parts, namely application software, architecture and environment. Each of these three aspects are modelled in a separate model, and they are then put together to form a simulation model of the system. The advantage of this approach compared to traditional performance simulations is that the actual behaviour of the software is incorporated in the analysis at an early stage. This is important since a great part of the dynamic behaviour of the system is described by the software. In order to obtain a performance model, a number of transformations need to be incorporated to the original SDL description and the systems structure. The authors propose the introduction of execution times for symbols modelled as delays, using delays procedures (see section 6.6), and of probabilities for different paths after a decision. Both execution times and probabilities have to be known or estimated, e.g. from previous experiences. The software description model has to be merged with a simulation model of the architecture, also described with SDL processes. In order to do this the concept of hierarchical processes is introduced. The original SDL software description is transformed into a partial simulation model supporting the handling of hierarchical processes, and this model communicates with the architecture model. Additions to existing tool environments are required. It can be noted that there is a similarity between the ideas presented in this paper and our proposed decomposition of complete models into service descriptions (software), network models (architecture) and population models (environment).

In [Bause and Buchholz 90] a Timed-SDL is presented (TSDL), where a number of modifications to SDL are proposed. These modifications include extensions to the syntax in order to introduce probability and time, changes in the scope and use of identifiers and introduction of constructs for inspecting a process input queue and the state of a process.

Automatic validation and performance evaluation of TSDL models is then achieved by transforming the TSDL description of a system into an equivalent Finite State Machine description, and using non-exhaustive Markovian algorithms.

An integrated tool for the modelling, verification, analysis and automatic simulation for real time networks, based on a variant of SDL is presented in [Yuang et al. 92]. A network is modelled as a system process construct, modelling the topology of the network, and a number of station process constructs, used to model the behaviour of stations (nodes). Networking parameters, such as the channel delay and the station latency delay are physically built into the system. The behaviour of the stations is specified by a variant of SDL, named SDL^N, where a number of new constructs are introduced in order to augment the mechanisms to model time, parallelism of execution, probability and critical sections (i.e. to specify exclusive use of resources).

In [Dou 95] a Timed-SDL is also proposed, where an approach similar to [Wohlin 92] is taken for the introduction of execution times: the execution times for different symbols are modelled as delays, and a 'delay procedure' call is added before the execution of a task. In [Wohlin 92] the specified delay must be a known value; in [Dou 95] random variables are introduced to describe the specified delays. The four main parameters of the model proposed in [Dou 95] are communication delay, input queue delay, scheduling delay, and task execution delay. The communication delay, if the two communicating processes are distributed, can be subdivided into the transmitting delay, the channel propagation delay and the receiving delay. The input queue delay is the lapsed time that a signal is waiting in the input queue of the receiving process, and it depends on whether the receiving process is waiting in the corresponding state or not. The scheduling delay refers to the delay due to access to a common resource. The task execution delay is the time consumed in executing the corresponding task. The four above-mentioned types of delays are introduced into the processes state diagrams, using the 'delay procedures' approach.

In [Dieferenbruch et al. 95] a method for introducing performance evaluation into SDL specifications based on the adjunction of queuing stations to SDL specifications is proposed. New syntactical constructs for load and machines are introduced in SDL, and the resulting specification and performance language, named QSDL provides means for the specification of load, machine and their binding.

The techniques used in our work to introduce performance in the SDL model of the Magnet service, in particular regarding processing times and delays, are discussed in section 6.6.

6.6. Introducing performance in the SDL Magnet model

The basic ideas about performance analysis and some approaches that can be found in the literature to introduce performance into SDL specifications have been discussed in previous sections. This section describes the approach proposed here and how some performance measurements have been introduced in the SDL model of the Magnet service.

It is important to remark that one of the objectives of this work was to obtain experimental results from the SDL models with the tools available, in order to explore the potential of SDL and the existing tools, as opposed to a merely theoretical study on how to improve SDL to introduce performance. The approach followed is therefore driven by the need to make use of the available tools (in particular the SDT tool by Telelogic) and of the standard SDL. The limitations of the tools employed and SDL as currently defined are very restrictive, and hence the solutions adopted to overcome them may be regarded as cumbersome and not very efficient. If SDL is to be used for performance evaluation, new constructs should be incorporated to the language that allow to introduce performance in a more straightforward manner, and tools must evolve to incorporate these developments. But this must be the result of long term joint effort within the research community, and is beyond the resources and time scales of this work.

The main performance aspects to be introduced in a model of a network have been identified as passage of time, transmission delays, use of resources, maximum throughput of the media and probability of transitions. In our model of the Magnet service, the capacity of the links between nodes is considered unlimited; hence, the maximum throughput is not modelled. The ideas to introduce time and probability information in the modelled system are discussed next.

6.6.1. Time and delays

In SDL, the behaviour of a system is given by a combination of processes that communicate in an asynchronous fashion by the exchange of signals. Each process is an extended finite state

machine; i.e. a process consists of a number of states and transitions between states, triggered by the consumption of signals. Some approaches in the literature propose to model time by introducing new syntax constructs to attach time delays to the execution of symbols or transitions (see section 6.5).

Actions with an associated delay time can be divided in two main categories, namely:

a) actions that require exclusive use of the resources, i.e. where the process that performs the action stops any other activity while it is serving the job in hand; this situation will be referred to as a blocking delay.

b) the opposite case where the process involved can proceed with other tasks while the task in hand is being processed; this will be referred to as a non-blocking delay.

The actions that a process performs are triggered by the consumption of signals. In a general case, there are two processes involved in a transaction: the process that sends the signal and the receiver process that executes a transition upon receipt of the signal (the exception is when a process sends a signal to itself). Bearing this in mind, the two types of delayed actions described above lead to the composite cases where two processes are involved. The result is the four following cases:

i) Non-blocking delay in both sender and receiver processes: neither the sender nor the receiving process block, i.e. the sender continues with other actions after sending out the signal, and the receiver can perform other tasks while the signal is being processed.

ii) Blocking delay in the sender, non-blocking delay in the receiver: after sending out a signal the sender process blocks and waits for response; the receiver process can perform other actions while the received signal is being processed.

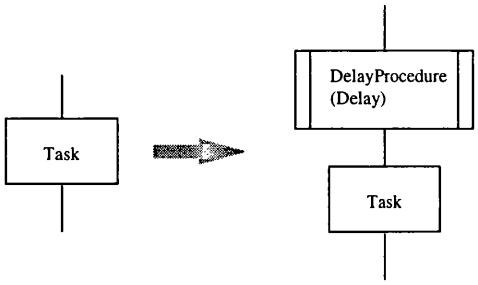


Figure 45. Adding a delay procedure call before the execution of a task

iii) Non-blocking delay in the sender, blocking delay in the receiver: the sender process continues with other actions after sending out the signal; the receiving process stops any other activity while the received signal is being processed.

iv) Blocking delay in both the sender and the receiver: both the sender and the receiver processes block, i.e. the sender process waits for response, and the receiver process stops any other activity while the signal in case is being processed.

The option depends on the chosen implementation and on the type of action being modelled, and the way in which each of these types of delays is introduced in an SDL system needs to be explored.

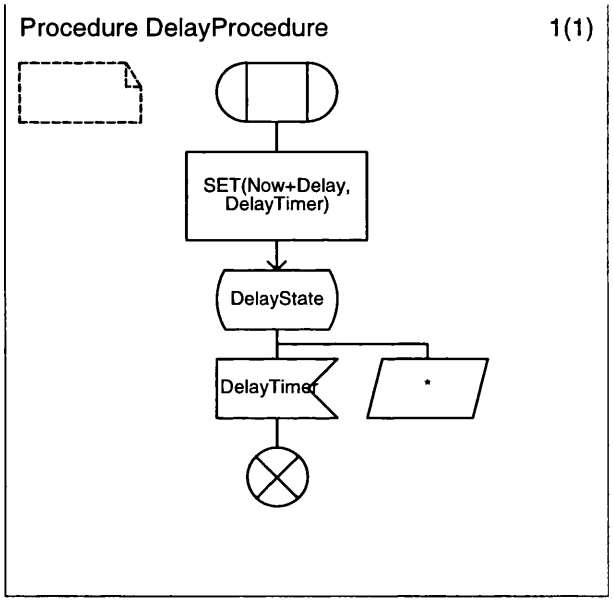


Figure 46. The delay procedure

Let us consider the problem from the perspective of the receiver process. Suppose that we want to associate an execution time to a specific action performed by this process. This action will be triggered by a particular signal. Hence, introducing a processing time for the action can be modelled by introducing a delay in the processing of the signal that triggers this action. In other words, when the signal is consumed, instead of starting the associated transition immediately, the signal is sent to a delay construct, that returns the delayed signal to the receiver process after the delay time requested. In [Dou 95] and [Wohlin 92], this is achieved by adding a procedure call before the execution of the task, as shown in Figure 45. This

procedure is denoted the delay procedure, and a parameter, i.e. the delay for the task (in Figure 45, 'Delay'), is passed to it. The procedure delays the execution of the task by use of the *timer* concept in SDL. All signals that arrive in the process during the execution of the delay procedure are saved, because the addition of temporal information must not alter the functional behaviour. The body of such a delay procedure is represented in Figure 46.

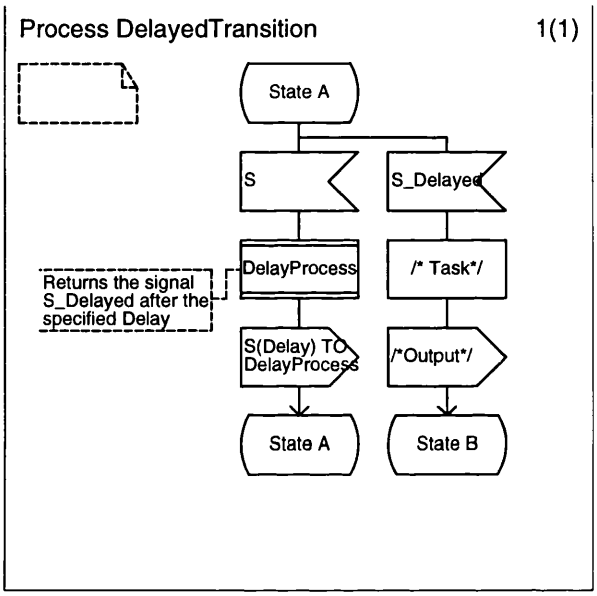


Figure 47. Associating delays to a transition using a delay process

An SDL procedure is executed as a part of the body of the process from which the procedure is called. This means that *the use of delay procedures will result in blocking delays*, because the receiver process stops any other activity while the procedure is being executed. A different strategy needs to be used in order to introduce non-blocking delays. The approach used in our model of the Magnet service is to use delay *processes* instead of delay procedures. A *DelayProcess* is created before the execution of the task, and the signal that triggers the transition that has to be delayed is sent to this *DelayProcess*. After this, the receiver process continues with the normal activity, i.e. it can consume more signals and perform their associated transitions. The *DelayProcess* makes use of the SDL timer construct, and all it does is to set the timer to the value specified for the delay, e.g. 'DelayTime'. When this timer expires, the delayed signal is sent back to the receiver process, and the *DelayProcess* is then terminated. An example of use of a delay process is shown in Figure 47. When the delayed

signal is consumed by the receiver process, the associated transition is triggered; the net result being that the execution of the task has been delayed 'DelayTime' units. Note that for the resulting delay to be as desired, the receiver process must not enter any blocking delays while the delay process is active. This is because if a delay procedure is called, the timer signals in the procedure (blocking delay) will have priority over the delayed signals returned by the delay process (non-blocking delay).

This approach, although rather cumbersome, can be used in any part of the system where processing times need to be introduced. If SDL was to be extended to incorporate performance, new features could be introduced in the language that perform these delay constructs in a more automated manner.

Let us now look at the problem from the perspective of the sender process. The blocking case, i.e. when a process sends a signal and stops any other activity until a response is received, can be represented by letting the sender process move to a waiting state after sending out the signal. In this waiting state, the only signal that the process can consume is the response to the signal previously sent out. This causes the process to stop any other activity until the response is received. The time of arrival of this response will depend on the delays associated to the transmission of the signal from the sending to the receiver process, the processing delays in the receiver process and the transmission delays of the response. The blocking delay in the sender process can be measured by calculating the difference between the arrival time of the response and the time when the signal was sent out. The non blocking case is easily modelled; after the signal is sent the sender process, instead of entering a waiting state, moves to an state from which it can continue receiving other signals. This way the activity in the sender process does not stop after sending the signal out.

6.6.1.1. Timer and delay constructs in the model of the Magnet service

This section describes the time constructs introduced in the Magnet service. For the sake of simplicity, most events are modelled as instantaneous transitions. Time delays have only been introduced in a limited number of events, in order to facilitate the understanding of the model and the analysis of the results. Two basic types of time information are introduced in the model, namely delay constructs, i.e. events with an associated processing time, and time outs to repeat actions if the expected answer is not received within a certain time.

Events that take time

a) Packet transmission

The time it takes a packet to travel one hop[2] is modelled using the delay process construct described in section 6.6.1. For the sake of simplicity, the transmission delays are regarded as constant, and they are the same for all the connections. Let us call this delay PACKET_DELAY. This delay is modelled by introducing a delay process in the input buffer at the data link layer in each node. This way, every time a new packet arrives in the node, it is sent to the delay process. The packet will not be consumed until PACKET_DELAY time units later, thus modelling the transmission time that the packet would have experienced from the previous node. A packet travelling from node A to node B will be exposed to this transmission delay at every intermediate node. Hence, the resulting transmission delay from A to B will be given by D*PACKET_DELAY, where D is the distance between A and B in hops.

b) File address query

The operation of obtaining a file address from the database of addresses also has an associated time. This is modelled by introducing a delay process in the database of addresses, so that when a new request for an address arrives, the request is sent to the delay process before the file address response is issued.

Time-outs

The two time outs introduced in the system were described in section 6.2.3.1. The first one is used to periodically re-issue file location requests from the user's local node to the database. This time out is set to a value called FLOC_TIMEOUT, and if the address of a file does not arrive to the local node FLOC_TIMEOUT time units after it is requested, a new *FileLocationRequest* packet is generated. The second time out is similar, but it is used to periodically request a file, instead of a file address. If a file does not arrive to the requesting node F_TIMEOUT time units after the *FileRequest* message is sent out, the request has to be reissued. Note that in the mean time, the requested file may have changed location. Hence, the process needs to start from the beginning, i.e. by sending a *FileLocationRequest* message to the database. These time outs are important parameters in the Magnet service, and their values should be tuned for best performance of the system. Due to time constraints, this has not been done in our model, and the analysis has been performed for a fixed pair of values of the time outs.

---

[2] A hop is the distance between two adjacent nodes

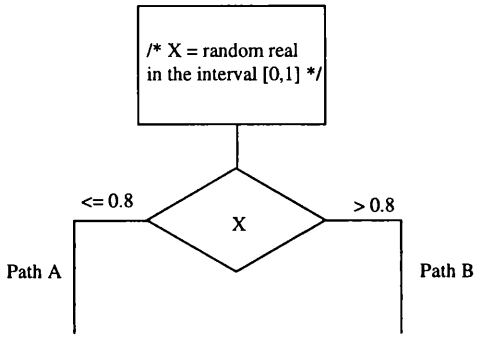


Figure 48. Associating probabilities to SDL transitions

6.6.2. Probability

There are a number of events in the case study governed by a probability function. Amongst them are the population patterns, the selection of data files to be requested and part of the routing algorithm. The introduction of probabilities is done using random variables. The value of a random variable is compared to a threshold and a decision symbol is used to select a specific transition based on the result of the comparison. For example, suppose that at a given point in a process two different paths (A and B) can be taken. Suppose path A must be chosen with probability 0.8 and path B with probability 0.2. This can be achieved by defining a random variable X and assigning a random value between 0 and 1 to it every time that point in the process is reached. The value of X is then compared with the threshold value 0.8. If X<0.8, path A is chosen. If X>0.8, path B is chosen. This gives a probability of 0.8 to chose path A and a probability of 0.2 to choose path B (note that the boundary cases, i.e. X=0.8, must be properly accounted for). Figure 48 illustrates these ideas. For the generation of random numbers the Abstract Data Type library for random numbers provided by Telelogic SDT has been used.

6.6.3. Failure

The integrity of a system is related to the risk of failure. Hence, an essential issue is to identify possible sources of failure in the model and investigate its consequences. Some failure may occur when the system is forced to operate under stressful conditions, i.e. when the external factors take values very different to those for which the system was designed. For example, a dramatic increase in the number of users in the Magnet service may result in a large

deterioration of the performance of the system, due to the contention issues. This and other causes of failure must be identified and introduced in the models in order to understand their effects.

The type of error included in the model of the Magnet service is a probability of losing packets. The packet loss failure has been introduced by limiting the number of packets that a node can keep in the input buffer waiting to be processed at a given time. This is achieved by making use of the delay construct included in the input buffers. As described in section 6.6.1, every packet that arrives in a node is sent to a delay process before it can be processed. These delay processes are instances of a delay process type. They are created dynamically, one for each arriving packet, and they are destroyed after the delayed packet is sent out back to the input buffer. In SDL, the maximum number of processes of a given type that can exist at a given time can be specified. Thus, the number of delay processes that can exist for the input buffer in a node can be limited. This imposes a limit to the number of packets that the node can process. If an incoming packet arrives in the node while the maximum number of delay processes has been reached, the packet will be discarded. New packets will only be accepted when some of the existing delay processes terminate. By using this technique, we are effectively imposing a maximum size to the input buffers in the nodes. If this size is sufficiently large, i.e. greater or equal to the number of incoming packets, no packets will be discarded. However, if the arrival rate of the packets is higher than the speed at which delay processes become available, packet loss would occur in the system. The size of the buffers is hence another example of an implementation factor that would need to be tuned to obtain adequate performance. The Magnet service has been simulated for different values of the size of the input buffers. The results are discussed in chapter 9.

As stated in section 6.2.3.2, one exception has been introduced regarding packet loss: the user de-registration packets that are sent from the user's new location to the previous one are never discarded. This technique is slightly unrealistic, but it has not major effects in the behaviour of the system. If this type of packet could be lost, this would result in users permanently registered in nodes. The only effect would be to fill the database of users in the nodes, but the capacity of this database is considered unlimited so this would not be a restriction. Old entries of users do not generate request for files, so they would not have any active effect. It was decided that the user de-registration packets are never lost for the sake of the measurements, because when the measure of the number of users in a node is taken, we are only interested in the 'active' users.

The errors due to lost packets could have been modelled at a higher level, by simply introducing a probabilistic function to model the packet loss. The approach to link the packet loss to the buffers size was taken in order to illustrate the use of the modelling and simulation techniques to assist with the systems design. The discussions section 7.4 illustrate how the simulation results can be used to identify the buffer size values required for the Magnet service to operate within certain integrity regions.

6.7. Conclusions

This chapter contains a description of the SDL model of the Magnet service. The objective of this modelling work was not to design a real service or obtain the best implementation for it. One main aim of the exercise was to test the suitability of SDL for simulation of networks and services for integrity analysis. This chapter has focussed on these issues, whereas the simulation results and their application to preservation of network integrity are dealt with in chapter 7. An essential area is performance analysis and how to introduce performance information in SDL-specified systems. An overview of literature in the area has been presented, followed by the approach proposed here.

As a result of this modelling work, a number of important limitations of SDL and the tool employed in this project have been identified. Some of these limitations have been overcome e.g. an approach to introduce some performance information in SDL specifications has been proposed and tested. This study of the encountered limitations and approaches to solve them with the resources available is regarded as a key outcome of the work. There are other limitations that would require deeper changes to the language to fix, e.g. hierarchies of processes and dynamic creation of blocks, and they have imposed certain constrains to the work (e.g. fixed network architecture). Nevertheless, a complete model for a specific service has been built, including some performance information, and it has been used for simulation of the system and evaluation of its performance. Some preliminary results have been obtained which can be used to identify and measure integrity issues. The results obtained and the connection to the pre-emptive integrity definition presented in chapter 3 are discussed in the next chapter.

Chapter 7

Outcome from the SDL modelling work

7.1. Introduction

The SDL model of the Magnet service described in chapter 6 was used in a number of different ways. One major application was the investigation to investigate the use of SDL for modelling and simulating services for integrity analysis and to identify necessary enhancements to the language and tool employed for this purpose. These issues were discussed in the previous chapter.

This chapter focuses on the application of the simulation results for integrity analysis. The SDL model of the Magnet service was used as a test case for the integrity definition and modelling methodologies proposed in chapter 3. The SDL tools were used to simulate the behaviour of the Magnet service, and obtain performance and quality of service parameters under different operating conditions. The approach taken for using the simulations to build the proposed integrity definition is described in section 7.2. This is followed by a description of the main parameters and measures taken in the Magnet service in section 7.3. The simulation results are discussed in section 7.4. The modelling work was also used to test the modelling

methodologies proposed in section 3.4. A review of these methodologies based on their application to the Magnet service is included in section 7.5. Another use of the SDL model was to illustrate its application to the integrity definition proposed in section 3.2 for defining, measuring and maintaining network integrity. This is discussed in section 7.6.

Finally, there was another type of outcome from the SDL modelling work. Section 3.4 highlighted the importance of developing well structure modelling frameworks within an organisation, in order to build generic modelling components that can be re-used for different applications. Investigation on the construction of such type of frameworks was considered as one possible subject for research in this project. Although it was decided not to pursue this as the main area of work, some useful insight into the problem was obtained as a result of the SDL modelling work. A brief summary of these ideas is included in section 7.7. This is followed by the conclusions in section 7.8.

7.2. Using simulation to identify integrity threats

As discussed in previous chapters, one main application of modelling is to assist in the design of services. Simulation by the models can be used to obtain performance measures of the system for different values of the design and implementation parameters, and under various external conditions. This can be used to detect potential integrity breaches in the system, i.e. circumstances that would cause significant degradation of performance.

In order to do this in the model of the Magnet service, two categories of parameters must be identified. One is design and environmental parameters whose effect needs to be investigated. These are referred to as the control parameters in the model. The other is the measurements to be taken as indicators of the integrity levels. These include network performance parameters as well as quality of service measures, e.g. end to end delays. Once both control parameters and integrity measures have been identified, the effect of changing the values of the control parameters on the integrity measures must be investigated. Then, threshold values must be defined for these integrity measures. The threshold values will define a number of integrity regions, from the correct behaviour, to the complete system breakdown, with perhaps some regions of incorrect but acceptable behaviour in between. Having defined the threshold values for the integrity measures, the values of the control parameters for which these thresholds occur must be found. This is the sensitivity analysis mentioned in section 3.4.2.4, and the objective is to identify the limits in the design parameters for which the system would operate

within the defined integrity regions. Special attention must be paid to combinations of parameter values that would take the system into catastrophic failure.

It must be emphasised here the importance of keeping the models simple, in order to facilitate the examination and interpretation of results. The analysis must start with the simplest scenarios, adding complexity gradually. Thus, the simplest case scenario is to consider the effect that one control parameter has on one particular measure of integrity. The relationship between the control parameter and the chosen integrity measure can be investigated by simulating the behaviour of the system for a range of values of the control parameter. Identification of threshold values for this integrity measure will define a collection of integrity regions. An example is shown in Figure 49, where a hypothetical generic performance measure has been plotted against a control parameter. For the sake of simplicity, let us assume that only one threshold value is defined for the performance measure, hence resulting in two integrity bands, i.e. a low integrity region and a high integrity region. The simulation must permit identification of the values of the control parameters for which the system operates in one region or the other.

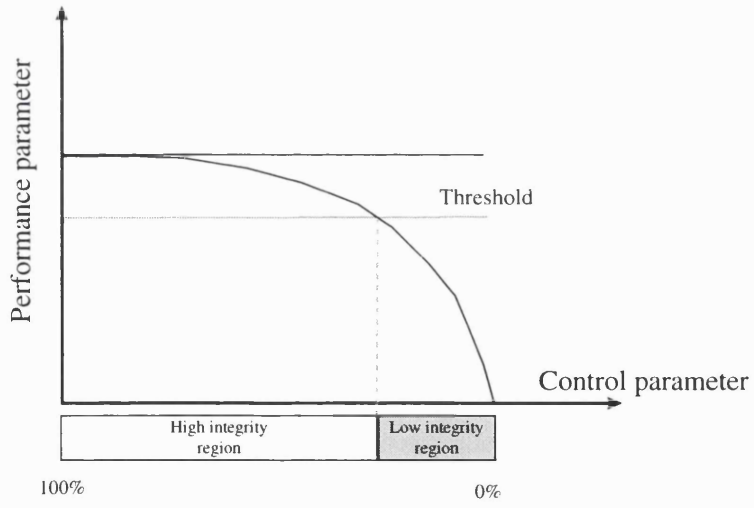


Figure 49. Hypothetical representation of an integrity measure as a function of one control parameter

This case is a coarse simplification of the problem because:

a) Only one performance measure is being observed; in practice, the level of integrity will be given by a combination of measures.

b) The effect of only one control parameter is observed; in practise, there will be a large number of parameters that simultaneously affect the integrity of the system.

c) Only two integrity regions are defined; in practise, it may be more valuable to identify a collection of regions.

The one parameter and one measurement case is only useful as a starting point. To obtain a more realistic view, the interdependencies between different control parameters must be studied. The next step would be to consider the case of two input parameters, and observe how a particular performance measure changes with these two parameters. Graphically, the values of a performance measure plotted against two control parameters would result in a surface. The threshold value for the performance measure would intersect this surface in a curve. This curve represents a collection of the pair of values for the two control parameters that satisfy the threshold. As discussed in previous sections, analytical evaluation is not feasible in practice, due to the large complexity of the systems. Hence, the exact form of these functions can not be exactly obtained. Simulation has to be used to obtain approximate results. An approach may be as follows. Start by identifying one pair of values that satisfy the threshold condition, i.e. one point of the curve. Simulation can be used to search around these values, i.e. the system can be simulated changing the control parameters incrementally, and obtaining further points of the curve. The applicability of these ideas could not be pursued due to time constraints.

7.3. Parameters and measurements in the Magnet service

This section contains a list of the main parameters in the model of the Magnet service, together with the values chosen for these parameters in the simulations. This is followed by a description of the measurements taken in the model.

7.3.1. Control parameters

The behaviour and performance of the Magnet service is distinctly affected by a number of factors or control parameters. Two types of parameters can be distinguished in the model. The first type is the design and implementation parameters for the service. The second type are those parameters that model external influences, i.e. factors that in a real case are not under the control of the service designer.

a) Design and implementation parameters

i) Values of time-outs.

The time-outs introduced in the Magnet service and an indication of their effects were discussed in section 6.6.1.1. These time-outs govern the rate at which request for files or addresses are periodically dispatched until a response is obtained. As indicated in section 5.4.1 the value of these time-outs could have significant effect in the signalling load and the delays in obtaining the requested information.

ii) Maximum number of attempts.

This parameter limits the number of times that the same address or file can be requested. This stops the generation of new signalling messages that may worsen a persistent failure situation because of the increased load in the network. But if the value is too low, it may prevent the local node from obtaining the requested information within a reasonable delay.

iii) Size of buffers.

Each node in the network in the Magnet model has an input buffer where the incoming packets are stored. As described in section 6.6.3, the size of this input buffer can be modified, hence limiting the maximum number of packets that can be waiting to be processed in a node. New packets arriving in a node when the input buffer is full will be discarded. Thus, the size of the input buffers in the nodes is related to the probability of losing packets.

iv) Processing times.

The processing times associated to different event in the model are another control parameter that can be changed. As explained in section 6.6.1.1 there are two types of events with associated times in the Magnet model, namely the transmission of data files and the search for an entry in the database of addresses.

v) Size and topology of the network.

The size and topology of the network on which the services are offered will be another key factor affecting the performance, as they determine the distances that the information must travel, and hence the delays in obtaining the data. As explained in section 6.3 there is not a straightforward way to change these factors in the SDL model built here. Hence, due to time limitations, these factors were kept constant in the model.

b) External factors

These factors refer to characteristics of the environment in which the service operates. For example, in a real network there are a collection of services that may compete for network resources and whose behaviour may interfere with one another (hence the feature interaction problem, discussed in section 2.2.2.4). When modelling a particular service, the potential effect of other services could be accounted for in the models. Another example is the interconnect scenario, where the effects of providing a service across two networks need to be studied. One possible approach would be to construct two complete separate models, one for each network, and interconnect these models together. This may not always be possible, as full details of the interconnecting network may not be known. Moreover, they may not be necessary, as the only characteristics of the interconnecting network that need to be included in the model are those that affect the interconnect. A simpler approach would be to use the full model of the service in the proprietary network, and to enhance this model with additional control variables capturing the relevant characteristics of the interconnect point. Finally, another major type of external factors is given by the characteristics of the customers of the service. These are the so-called customer models, whose importance has been emphasised at various stages in this document (refer, for example, to section 3.4.2.4).

The limitations in the time and resources available for this project did not permit the analysis of the multi-service and the network interconnect scenarios for the Magnet service. As for the customer models, some basic population patterns were investigated with the C++ model. Some of the main factors involved in the customer models are:

i) Number of users in the system.

ii) Mobility patterns of the users, i.e. random and diffusion based population relocation models.

iii) Frequency of service requests: in the model, service is requested only when the users register in a new location.

iv) Degree of contention, i.e. number of files shared amongst the users.

c) Simulation parameters

As previously stated, the study must start for a limited number of parameters. The selection of parameters to consider was partially limited by the constraints imposed by SDL (see section

6.3). Thus, changes to the size and topology of the network are rather difficult to introduce, and for this reason, it was decided to keep them fixed.

There are two main parameters in the SDL model whose effect was observed. These are the number of users and the size of the input buffers in the nodes. Simulations were run for different values of these parameters, whilst the values of the remaining parameters remained fixed. The values of the variables governing the movement of users were chosen to provide slow movement compared to the expected delays in relocating files for the non-failure scenario. This means that for the non-failure scenarios the users stay in the same location long enough to obtain the requested information. The relevant simulation parameters are as follows:

- ◊ Size of the network: 9 nodes (3 rows by 3 columns grid)
- ◊ Total number of data files in the system: 10
- ◊ Number of data files associated to each user: 5
- ◊ Probability of a data file being requested for a user: 0.5
- ◊ Speed of movement of users: users change location every 5000 time units
- ◊ Packet transmission delay: 100 time units per hop
- ◊ Database access delay: 100 time units
- ◊ Time out to obtain a file address: FLOC_TIMEOUT = 400 time units
- ◊ Time out to obtain a file after a *FileRequest* has been sent: F_TIMEOUT= 400 time units
- ◊ Initial simulation time: the first movement of users occurs at t_i=1000 time units
- ◊ Final simulation time: t_f= 1000000
- ◊ Frequency at which measures are taken: every 5000 time units (D=5000)
- ◊ Offset of the measures with respect to the movement of users: measures are taken 500 time units after the movement of users

For each value of the number of users and size of input buffers a total of 10 simulations have been done, and the results have been averaged for the 10 different simulations. The number of simulations run for each case is considerably lower than for the C++ model. This is due to the very large execution times of the SDL simulations.

7.3.2. Measurements

A series of simulations of the Magnet service were run for different values of the number of users and the size of the input buffers in the nodes. Measurements are taken at regular

intervals during the simulation. Let us call D the time interval between two consecutive measurements.

In order to obtain the impact of the control parameters in the signalling and processing load, and in the delays experienced by the users of the Magnet service the following measures were introduced in the system:

a) Average number of data files present on each node, in order to determine how files are distributed in the network.

b) Average number of incoming packets in a node during the time interval D, as an indication of the signalling load in the network.

c) Average number of discarded packets in a node during the time interval D, to obtain an indication of the failure rates.

d) Delay in obtaining a requested file, i.e. time lapsed between the generation of a request and the arrival of the file to the requesting node.

The simulation results are discussed in the following section.

7.4. Simulation results

The simulations have been divided in two different cases. First, the one user case was modelled, and the results are included in section 7.4.1. Second, the multiple users scenario where users require access to a number of common files was simulated, in order to understand the contention issues. The simulation results for the multiple users case are discussed in section 7.4.2.

7.4.1. One user

The first set of simulations was done for the case of only one user in the system. This simplifies the problem because it means that there are no file contention issues. The user follows the random population model. This means that the population pattern follows a uniform distribution, because the user can be at any location with equal probability (refer to section C1 in Appendix C for detailed explanation). Changes of location occur every 5000 time cycles. A number of measures were taken in order to validate the model in terms of the distribution of users, as shown in Figure 50. The top graph illustrates this distribution during the simulation time. For each value of the simulation time, the number of users registered in

each node is plotted. Since there is only one user in the system, this number must be 0 for all the nodes except that where the user if present at the time, where it is 1.

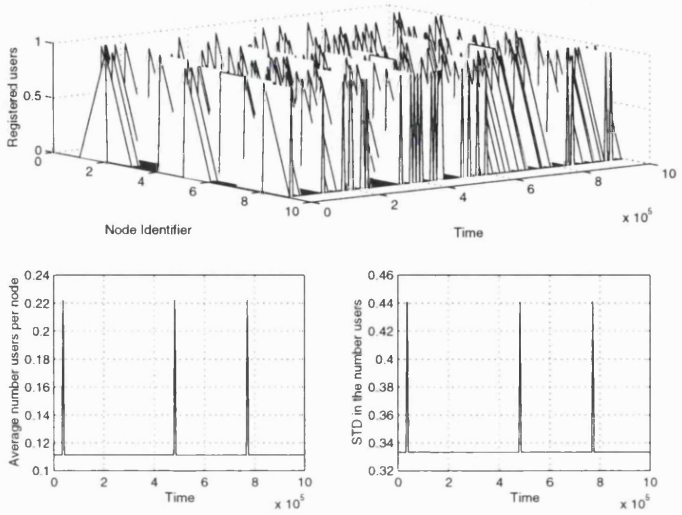


Figure 50. Distribution of users in the network (one user case)

The average number of users per node is given by

$$\eta = \frac{TotalNumberOfUsers}{NumberOfNodes}$$

This is represented in the bottom left graph of Figure 50. Note that, as opposed to expected, the average number of users registered does not remain constant all the time, but it raises at three specific points. This is because when a user is registered in a new location, a user de-registration packet is sent back to de-register the user from his/her previous location. These de-registration packets are subjected to the packet transmission delays. The packet transmission delay is 100 time units per hop, and the longest distance between two nodes in the network is 2 hops (since the test network is composed of 9 nodes with periodic boundary conditions). The measurements are taken 500 time units after the user changes location, which in most cases gives enough time for the de-registration packets to arrive to the user's previous location. Hence, for most cases $\eta = 1/9 = 0.11111$. However, the non-deterministic nature of the routing algorithm (if the destination node is not one of the neighbours, the packet is sent on a link selected at random) may result in long delays of the de-registration packet.

Therefore, there is an interval of time during which the user may be registered in both the new node, and the previous one. If the number of users is counted during this time interval, the result will be two, but in fact it is the same user who has not yet been de-registered from the previous location. In these cases, $\eta = 2/9 = 0.22222$, which is the value of the three spikes in the plot.

The bottom right graph in Figure 50 represents the standard deviation of the number of users, calculated as

$$STD = \sqrt{\sum_{1}^{9} \frac{(x_i - \eta)^2}{9}}$$

where x_i is the number of users at node i, and η is the average number of users in each node.

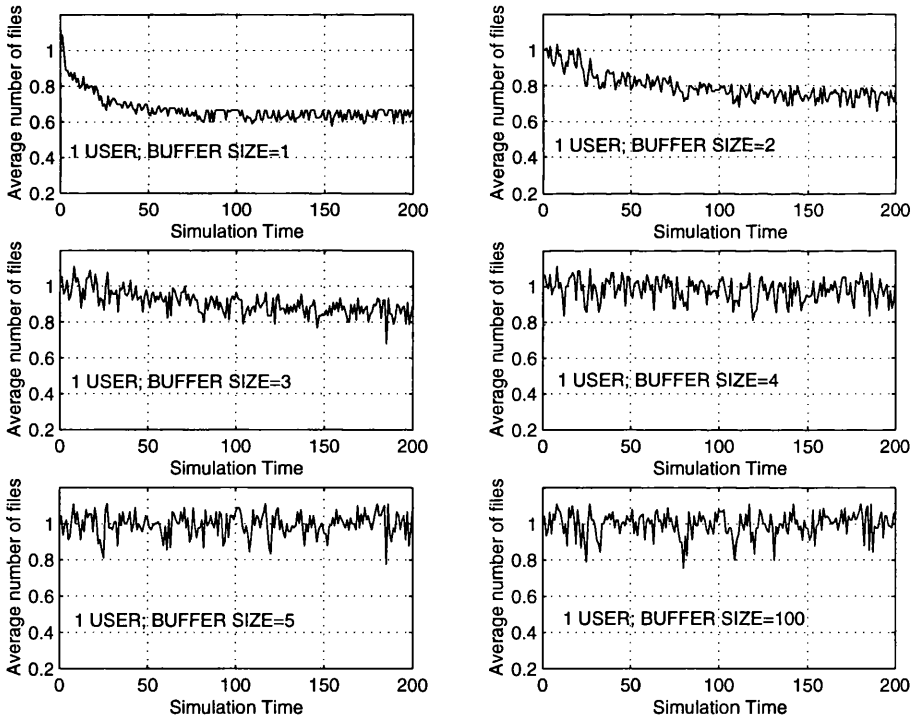


Figure 51. Average number of data files per node for different sizes of the input buffer, and 1 user in the system

7.4.1.1. Number of files

A similar approach can be followed to obtain the number of files in the system. In this case, there are 10 different data files in the network, which gives an average of 10/9=1.1111 files per node. Again, due to the possible delays in deleting files from previous locations, the measure may experience variations around this value. The simulations were run for different values of the size of the input buffers in the nodes. The results are depicted in Figure 51[1].

Some relevant information can be obtained from these figures. The last plot in the graph represents the case of size of the input buffers equals 100, i.e. there can be up to 100 packets waiting to be processed in a node before any packet is discarded. This value was chosen experimentally to be well above the packet arrival rate, and hence the buffer size in this case does not impose any limitations in the system. Thus, the graph shows that the average number of data files for a buffer size of 100 fluctuates around a value slightly lower than the theoretically expected 1.1111. The fluctuations occur because we the measure only accounts for the number of files registered in the nodes, and hence does not include files that are in transit.

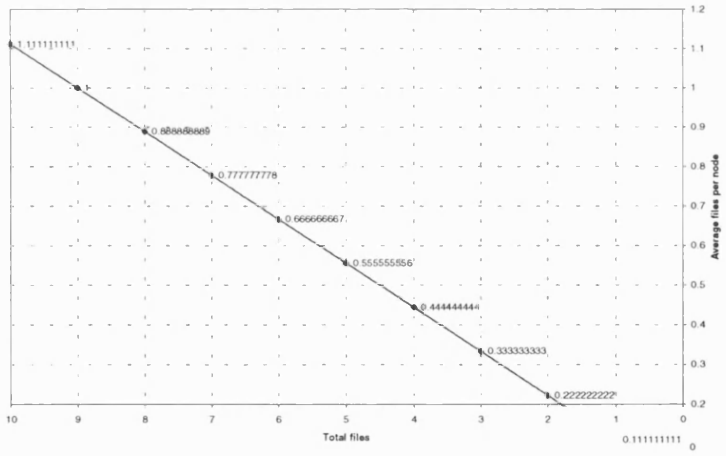


Figure 52. Average number of files versus total number of files

The opposite case is when the buffer size is set to 1. This means that any packet arriving in a node while there is another packet in the delay process will be discarded. The effect of this in the number of files present in the system is shown in the first plot in Figure 51. This plot shows that the number of files decreases from the initial values around 1.1111 down to around

[1] In this and subsequent graphs the simulation time in the axis x of the plots is scaled by the measurements interval, D=5x10^3. Hence, the total simulation time is 10^6.

0.6666 (i.e. to 6 files in the system). For convenience, the correspondence between total number of files and average files per node for the 9-node case is shown in Figure 52.

The decrease in the number of files with time for small buffer sizes is explained as follows. When a request for an address arrives at the database and the address response is sent to the node, a flag is set for that entry in the database to indicate that the file is going to be in transit, and its location will be undetermined for a period of time. While the flag is set, no other requests for the address of that file are served; i.e. no other node can obtain the address of the file. This situation is maintained until the file is registered in the new location (i.e. a *FileResponse* message arrives in the new location), and a file location update is sent to the database. Now, if the *FileResponse* message gets lost, the file will never be registered in the new location, and hence the flag in the addresses database will never be reset. Effectively, the file is lost in the system. This is what causes the decrease in the average number of files with time. Figure 51 shows how this decrease is lower for higher sizes of the input buffers. This is because increasing the sizes of the input buffers decreases the probability of loosing packets. In fact, it can be inferred from Figure 51 that for a buffer size of 4 there is no apparent file loss during the time that the system is left to run.

Another effect that can be observed is that in the cases where file loss occurs, the loss rate is high at the beginning of the simulation, and it becomes much lower with time, with a tendency to stabilisation. This is because one effect of files getting lost is that those files will never be moved. Hence the number of packets generated in the system decreases, and the size of the input buffers can cope better with the lower load. This results in a smaller number of packets being discarded in the nodes, and hence in a lower probability of losing files.

The loss of files, even if it is a slight loss, may have severe consequences. Some of these data files contain service logic, so losing the files may render those services inaccessible. In a real system this type of failure may and should be fixed by the use of appropriate recovery mechanisms. One possible recovery mechanism would be to keep a back-up copy of all the files and to reset all the flags and file addresses in the addresses database when a significant number of lost files has been detected. This is an example of how monitoring (the number of data files) can be used to detect integrity violation (number of lost files beyond a threshold) and to take an appropriate recovery action. This was not done in the model, because the objective was not to optimise the system, but to illustrate how modelling can be used to assist in the service design and to identify potential failures.

7.4.1.2. Generated load and discarded messages

Another measurement has been introduced to obtain an indication of the activity in the network, in terms of number of messages generated. The measure taken is the number of packets that arrive in each node during the interval *D*, and the average number of packets per node is calculated. The results over time are shown in Figure 53 for the different sizes of the input buffer.

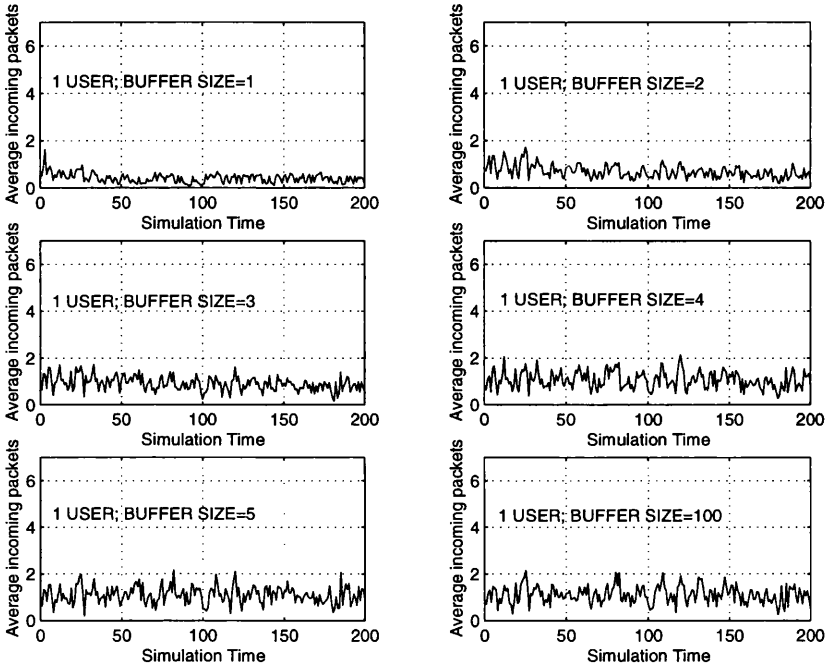


Figure 53. Average number of incoming packets in each node every *D* = 5000 time units for the one user case.

For a buffer size of 100 (non-limiting) the average number of incoming packets per node is around 1.5 packets every 5000 time units. And it can be seen that it remains approximately the same for buffer sizes of 4 and 5. Again, this agrees with the results regarding the number of files; i.e. it appears that a buffer size of 4 would be enough for the system to operate in the same way as if the buffer size was not limited. For a buffer size of 3, there appears to be a slight decrease in the number of incoming packets, and this decrease is bigger for buffer sizes of 2 and 1. In other words, by imposing a limit to the buffer size below a certain threshold, the activity in the network in terms of generated packets decreases. This result may seem unusual, because it could have been expected that introducing packets loss would instigate more re-attempts to obtain files, and hence would increase the load in the network. However, as

previously discussed one critical effect of losing packets in the Magnet service is the progressive loss of data files. Every time that a lost file is requested its location will be unobtainable, and the requesting node will re-issue the location request periodically until the user moves to a new node. But this is the only type of messages that are repeated, at a rate of one request message every 400 time units (the value of the timer FLOC_TIMEOUT). Since the location of the file is never returned to the requesting node, file request and file response messages will not be generated. And these are in fact the types of messages that have the major contribution to the number of packets in the network. It must be recalled that the address request messages are not modelled as packets travelling between nodes, but as signals sent directly to the locations database (refer to end of section 6.2.1.4). Hence, they are not accounted for in the measure of the number of packets. Therefore, the net result is that a small size of the input buffers causes the loss of files, and this has the effect of decreasing the number of packets travelling between the nodes over time.

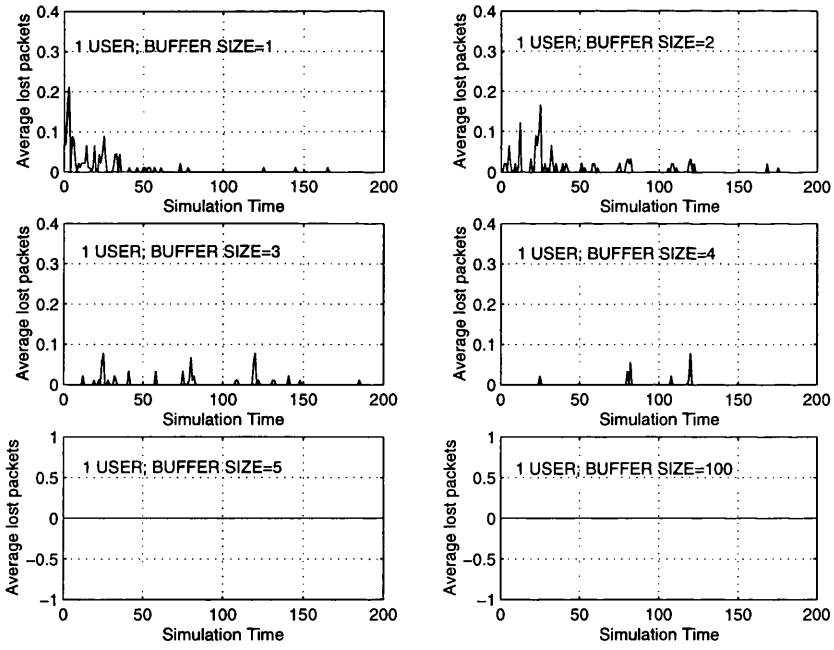


Figure 54. Average number of packets discarded in each node every D = 5000 time units for the 1 user case.

Finally, the decrease in the number of generated packets with time means that the number of packets that a node has to discard also decreases over time, because the rate of arrival of packets in each node is lower. The average number of discarded packets over time is depicted

in Figure 54. The graphs show how with a buffer size of 5, no packets are discarded, i.e. this size guarantees correct performance of the system. Another interesting result is that for a buffer size of 4 there is a small number of packets discarded, but in section 7.4.1.1 we saw that this buffer size is enough to avoid files loss. The explanation to this is that a file is lost when one of the *FileResponse* packets is discarded. If the total number of discarded packets is very small, the probability of loosing a *FileResponse* packet will also be very small. Therefore, for a buffer size of 4, the results show that there is no file loss during the period of time that the system has been simulated. However, there is some packet loss, which means that there is an associated risk that the system will eventually lose data. Empirically, no files are lost during a time interval of D=20000 time units for a buffer size of 4. If restoration techniques were introduced that reset the data at regular intervals (below 20000 time units) the probability of losing data files would be virtually zero. Note that this can not be absolutely guarantied, as this calculation is based on empirical results. Therefore, a buffer size of 4 would cause the system to operate within a region of high integrity, but yet below absolute integrity (this will be discussed further in section 7.6).

7.4.1.3. Delays

The delays in obtaining a requested file were also measured. This is the elapsed time between a node sending out a request for a file address and the file arriving in the requesting node. This delay is calculated for every file being requested during the total simulation time. The histograms in Figure 55 represent the number of cases (axis y) that experienced a particular delay (axis x). It must be remarked that this measure only accounts for those requests that succeed, i.e. when the requested file arrives in the requesting node before the user changes location. The success rate is obviously lower for small size of the buffers and hence the total number of cases is lower for lower buffer sizes. The plots in Figure 55 have been normalised to facilitate comparison. The non-normalised graphs can be found in appendix E. The important feature here is how the delays are distributed. From Figure 55 it is clear that higher delays occur for small size of the buffers. For a buffer size of 3 and above, there are no delays higher than 800 time units.

The delays in obtaining a file are related to the distance that the file has to travel. Due to the periodic boundary conditions in the network lattice, and given that the size of the network is 3 rows x 3 columns, the maximum distance between two nodes is 2 hops. However, due to the non-deterministic nature of the routing algorithm the file request and file response messages

only travel one hop if the destination node is a neighbouring node; but they may travel more than two hops if the destination is not a neighbour.

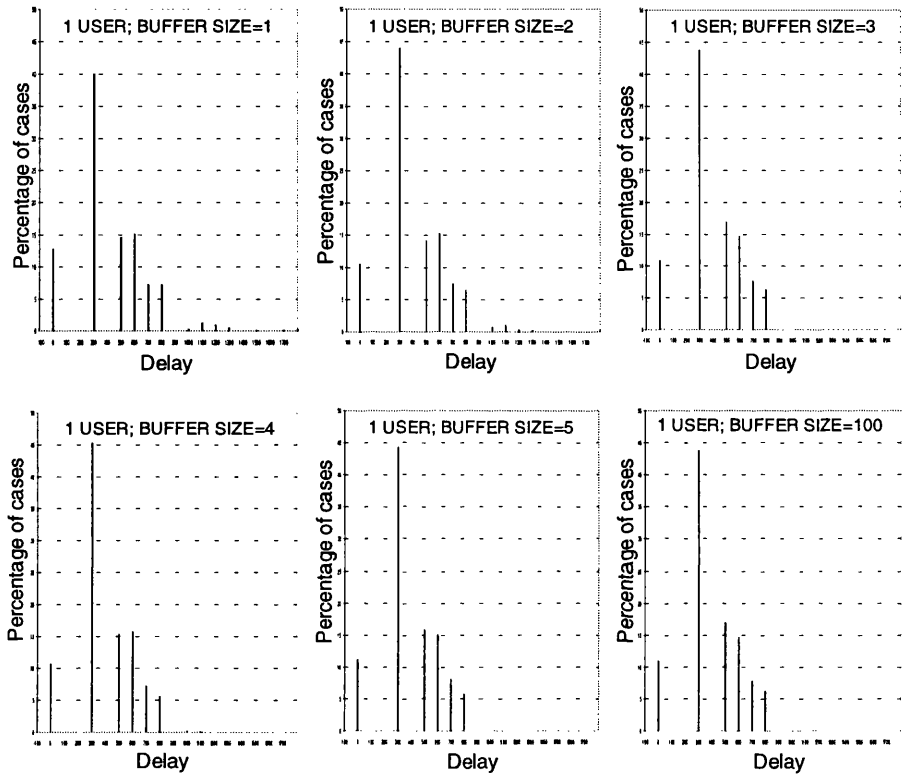


Figure 55. Normalised histogram of the delays in obtaining a file for the 1 user case.

NB: labels in the graph are as follows: a) Axis x: -100 to 1700 at intervals of 100; b) Axis y: 0 to 50 at intervals of 5.

Due to the uniform distribution of users and to the periodic boundary conditions in the network, the distribution of files will also be uniform; i.e. a file can be present at any location with equal probability. The probability of a file being at a certain location is hence 1/N, where N is the total number of nodes (N=9 in the case study). Each node has four neighbouring nodes and four nodes situated at a distance of two hops. This is illustrated in Figure 56.

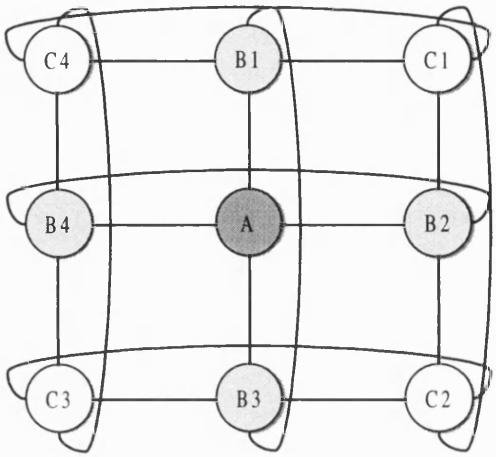


Figure 56. Nine nodes grid network with periodic boundary contitions.

Consider node A. The four neighbouring nodes are B1, B2, B3 and B4, situated one hop away from A. The nodes two hops away are C1, C2, C3 and C4.

Let us consider the non packet loss scenario. The following cases can be differentiated in terms of the distance from the requested file to the requesting node and the delays:

a) The file is present in the user's local node; the probability of this is 1/9. In this case, the delay in obtaining a file is 0.

b) The file is in one of the neighbouring nodes; this happens with a probability of 4*(1/9), i.e. it is four times more likely than the previous case. The total delay in this case is 300 time units, corresponding to 100 time units to consult the addresses database, plus 100 time units for the file request packet to arrive to the destination (1 hop) plus 100 time units for the file response to travel back to the originating node. The histograms Figure 55 show that this is the most frequent case, and that if happens roughly 4 times more often than the 0 delay case.

c) The file is in one of the four non-neighbouring nodes. This also happens with probability 4*(1/9). However, in this case the delay in obtaining the file is not deterministic, due to the nature of the routing algorithm. If a shortest distance routing algorithm was being employed, the delays in this case would be 500 time units, i.e. 100 for database consultation, 200 for the request packet to arrive to the destination and 200 for the file response to travel back. However, with the routing algorithm employed in the model these delays may be longer, because a packet for a non-neighbouring node is sent to any of the neighbours at random. This explains the different components in the delay beyond 500 for

the case of a buffer size of 100 (non-limiting). The graph shows that in fact the maximum empirical delay in this case is 800 time units.

When the size of the input buffers imposes a limitation and causes packet loss, the delays increase due to the need for re-attempts. Figure 55 shows that there are delays of over 1000 time units for buffer sizes of 1 and 2.

In section 7.4.2.3 it will be discussed how the delays increase dramatically when the number of users increases, due to the competition for the files between the users.

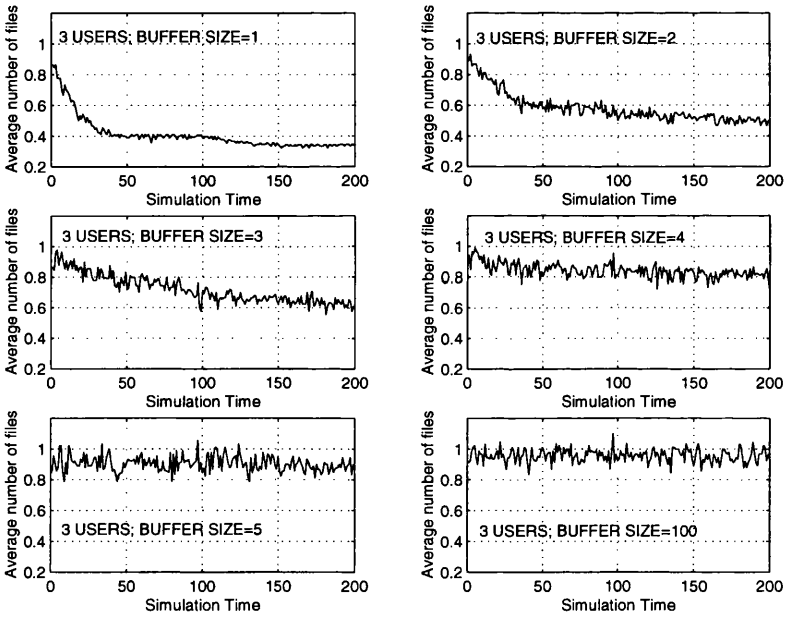


Figure 57. Average number of data files per node for different sizes of the input buffer, and 3 users in the system

7.4.2. Multiple users

Simulations were run for the cases of two and three users in the system, and the same measurements as described before for the case of one user were taken. For the sake of brevity, only some graphs with the results for the three-user case are shown in this section. The full set of results is included in Appendix E. This section focuses on the effects of the increase in the number of users in the different measurements.

7.4.2.1. Number of files

The following effects can be observed by comparison of the results for different number of users (see Figure 57 for the three-user case):

a) For the non-failure case (buffer size of 100) the average number of files in each node is slightly lower than in the one user case. This is because as the demand for data files is higher, more files are relocated and hence there is a larger number of files in transit at the time of taking the measurements.

b) The higher the number of users, the more rapidly the number of files decreases with time. Consider for example the case of buffer size equal to 2. During the interval of time that the system is simulated, the number of files for this case goes down to approximately 0.5 for the 3 users case, 0.6 for the 2 users case and about 0.75 for the one user case. This is because having more users competing for the same files increases the likelihood of these files being moved and hence of them getting lost. Again, for the same reason as in the one user case (in section 7.4.1.1), the graphs show a fast decay at early stages and lower file loss rates with time, with a tendency to stabilisation.

c) The higher the number of users, the larger the size of the buffers must be in order to avoid file loss. For the 1 user case, it was discussed how a buffer size of 4 seems to be enough to maintain all the files in the system. This value is obviously not enough in the 2 and 3 users cases. For the 2 users case it would appear that a size of 5 is enough, whereas in the three users case 5 seems to be still too small. This demonstrates the importance of taking demand characteristics into account in the design and implementation of services.

7.4.2.2. Generated load and discarded messages

As would be expected, the number of packets travelling in the network increases with the number of users, because there are more file requests and hence more files being moved. Thus, for the non error case (buffer size equal to 100), the average number of incoming packets in a measurements interval is around 1.5 for the one user case, 2 for the 2 users case and 2.5 for the 3 users case (Figure 58). Note that the signalling load does not increase by the same factor as the number of users, i.e. doubling the number of users does not double the generated load. This is due to two main reasons. One is that the users are competing for some of the files, and when this happens only one of them will succeed at a time. Only the request that succeeds involves file request and file response messages. So not all the requests initiated result in generated packets. The other reason is that, as discussed in section 7.4.1, the loss of

files results in a decrease of the number of generated packets. And the file loss is higher and faster as the number of users increases. Hence the non-linearity of the messages load with respect to the number of users.

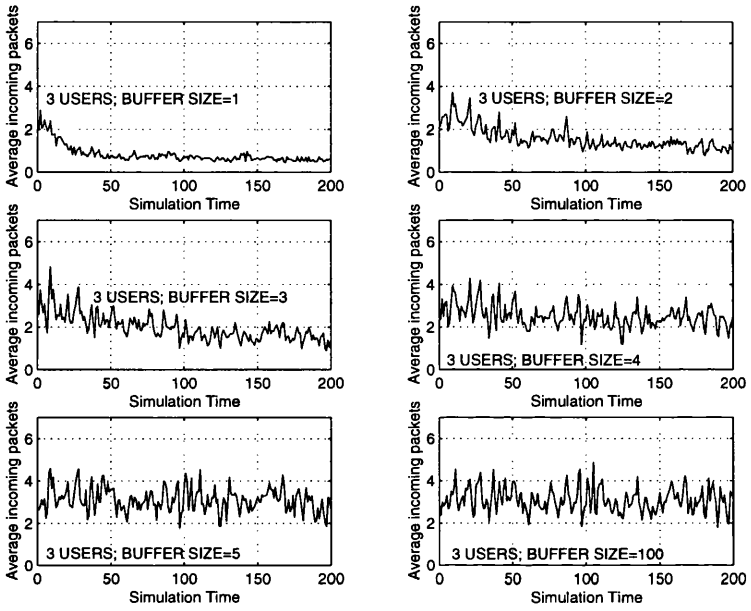


Figure 58. Average number of incoming packets in each node every D = 5000 time units for the three users case.

The results regarding the number of lost packets confirm the observations described previously (see Figure 59 for the three-user case):

a) The larger the number of users, the larger the number of discarded packets, because the total number of generated packets is larger.

b) The larger the number of users, the larger the size of the buffers must be in order to avoid packets loss completely. For the one user case, a buffer size of 5 would guarantee that no packets are lost. The same buffer size of 5 results in a small number of discarded packets in the two users case, and a bigger loss for the three users case. More simulations would be needed for size of the buffers larger than 5 in these two cases, in order to find which is the minimum buffer size that guarantees no packet loss. However, as previously discussed, it appears that the designed system can tolerate some packet loss, i.e. it can operate correctly (no files are lost) even if a small number of packets is discarded. Thus, for the one-user

case, a buffer size of 4 does not result in apparent file loss, and the same happens for a buffer size of 5 in the two users case. As previously stated, for these sizes of the input buffers the system would be operating with an associated risk of failure, because the probability of losing *FileResponse* packets - and therefore the probability of losing files - would not be zero.

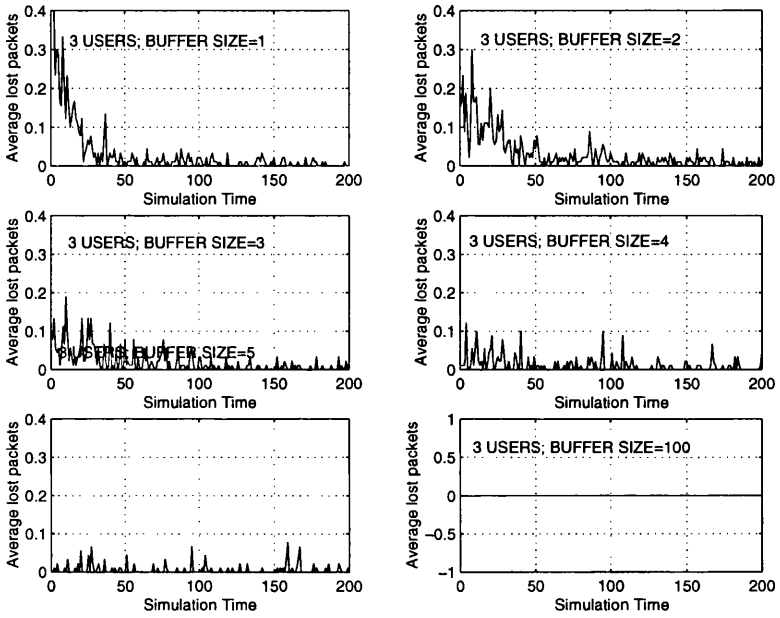


Figure 59. Average number of packets discarded in each node every D = 5000 time units for the 3 users case.

7.4.2.3. Delays

The main effect of increasing the number of users is a major increase in the delays in obtaining a file (see results in appendix E). This is due to the contention issues, i.e. the more users trying to retrieve the same file, the more likely it is that the file is inaccessible and hence a larger number of retries are needed. Again, this effect is not linear. The increase in the delays when changing from one user to two users is much more dramatic than changing between 2 users and three users. This is because in the one-user case, there is not contention for the files, so the files are always available and they can be obtained with only one attempt.

7.5. Review of proposed modelling strategy

The results presented in previous sections provide an insight to the behaviour and performance of the Magnet service. Some interesting characteristics of the system have been revealed. In particular the focus here is on potential failures, and how they are affected by the control parameters.

Before the modelling work started, a modelling strategy had been planned. This was described in section 3.4.2, and involved planning ad defining the following:

a) Aim of the models and modelling methodology
b) Level of abstraction
c) Modelling languages and tools
d) Inputs to the model

One of the objectives of the modelling work was to test the applicability of the proposed methodologies. Let us therefore review the modelling framework in the context of the SDL modelling work carried out for the Magnet service.

Section c) is straightforward. Two different languages, C++ and SDL, have been tested. The C++ model served as an introductory exercise to identify key points in the Magnet service. In the search for a more visual language that permits the observation of the behaviour of the system in a more user-friendly environment, it was decided to continue the modelling work using SDL. It must be recalled here that SDL was not conceived as a simulation language, but as a specification language. The semi-formal nature of SDL and the increasingly advanced commercial tools, permit simulating the behaviour of SDL specified systems, but the possibilities are still rather limited. Here it was decided to investigate the suitability of SDL for integrity modelling, being aware that this would require extending the use of SDL beyond the purpose it was designed for originally. Hence, a major part of the work was this analysis of SDL (and the tools employed), identification of limitations and proposed solutions. All these issues were discussed in chapter 6.

Let us now consider the inputs to the models (d). The three main inputs were identified in section 3.4.2.4. as: formal service descriptions, network models and customer models. These three inputs have been included as components in the SDL model, as discussed in section 6.2.1. It has also been argued how, ideally, these inputs should be independent from one another, in order make them re-usable modelling components. But the construction of generic

modelling components and re-usable models is a very complex problem, aggravated by the lack of suitable tools, and hence a long-term research proposition. Some introductory ideas following the discussions in section 3.4.1 are included in section 7.7. In section 3.4.2.4 it was discussed how sensitivity analysis must be carried out for the performance and customer models in order to identify regions of validity of the parameters. Some sensitivity analysis has been undertaken in the Magnet model, by simulating the behaviour for a range of values of some of the control parameters, i.e. number of users and size of input buffers. The results are included in section 7.4 and discussed further in section 7.6.

Regarding the level of abstraction, a layered framework was introduced in section 3.4.2.2 where different aspects of the systems can be modelled with different degree of detail, depending on what features are relevant in each context. The top level of the framework is the Service level, where a service is decomposed in a number of features. The three main features of the Magnet service have been identified as the addressing function, database consultation and file transfer function. Below the service level, there are the Functional layer, Network layer and Operational layer. The Customers layer can be present in any of the other layers, and it represents the characteristics of the users of the services and the effects of the interactions between the users and the system. These are the population patterns in the Magnet service, and the effect of changing these patterns has started to be addressed. As previously remarked that the use of such layered framework is to provide an appropriate level of abstraction in the models. This is achieved by differentiating those aspects relevant to the problem under study, and hence requiring greater modelling detail (lower layers), from aspects that would not add any useful information and hence can be modelled in less detail (higher layers). It must be noted that this layered framework is not rigid, and does not need to be applied strictly. It should be regarded as a guideline, and taken only as far as it is useful. The three features of the Magnet service have been modelled with different levels of detail. In the first C++ model the three features were modelled at the functional level, i.e. it was assumed that they were provided by the network in a reliable way. In the second C++ model the details of the addressing function were implemented, identifying the network resources needed, i.e. at the 'Network layer'. The database consultation and file transfer function remained at the functional level. In the SDL model, it is the file transfer function that is treated at the Network level, and the details of how files move around the network were modelled. The database consultation and addressing function were kept at a higher level. However, some additional information was introduced by associating transmission times to the address request and address response messages and assigning processing times to the database searches. The latter

can be regarded as beginning to address the Operational layer. Thus, the layered modelling framework has been used in a flexible way, according to the objectives of the model.

Finally, there is the issue of the aim of the models and modelling methodology, discussed in section 3.4.2.1. Regarding the use of the models, it was decided to focus on the identification of potential errors in services. A methodology, comprising seven stages, was designed for this purpose. The seven stages are reproduced in Figure 60, for reference.

Stages 1 to 4 were followed for the Magnet service. The service behaviour was modelled and coupled to models of the network (with performance models) and the characteristics of the users (customer models). Simplest scenarios were modelled initially, to understand the 'normal' behaviour and validate the models. Potential causes of errors were identified, a selection of them was introduced in the models, and the system was simulated for these error conditions.

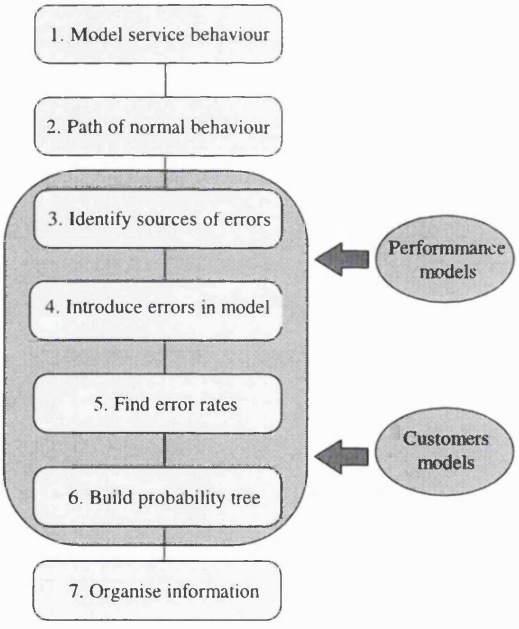


Figure 60. Seven stages of the proposed modelling methodology

As for stages 5 and 6, they were not finally applied for a number of reasons. One is the extreme complexity of this approach, aggravated by the limitations encountered SDL and MSCs. The approach in stages 5 and 6 relies heavily in the capability to identify system traces and obtain states of behaviour –both error states and normal states. Note that these states are

not the states defined in the SDL processes. They are some kind of meta-states that represent the ability of the system to operate in a way in which certain characteristics of the system hold and certain conditions satisfy. This proved to be a very complex problem, and it requires a large number of simulations. The limitations in the tools and language employed (refer to section 6.3) were too restrictive to follow this approach within the available time scales.

Regarding stage 7, this falls more into the area of documentation techniques and information processing, which have identified as one major area for improvement in industrial organisations and a key component of the overall integrity framework presented in section 3.3. As discussed in there, it was decided not to pursue this area.

The discussions above show that most of the modelling strategy proposed in section 3.4 has been successfully applied to the case study. The modelling work also questioned the validity of part of the methodology. Some of this is due to limitations we encountered in the tools and mismatch with expected functionality.

7.6. Application to the integrity definition

This section discusses how the results from the models of the Magnet service can be used in the context of the integrity definition proposed in section 3.2. The main features of this definition are reproduced here for convenience. Figure 61 shows an example where the level of integrity of a system is divided in a collection of regions. The proposed integrity definition can be used to identify the level of integrity of a system, i.e. the integrity region in which the system is operating within the framework. This requires:

a) examining potential causes of loss of network integrity and how this loss would manifest, i.e. what parameters should be considered as indicators of the degree of integrity of a system,

b) identification, for those integrity parameters, of threshold values that would indicate a significant loss of network integrity,

c) definition of the integrity bands for the system, based on the threshold values for the identified integrity parameters,

d) monitoring the integrity parameters in order to detect the band in which the system is operating at any time.

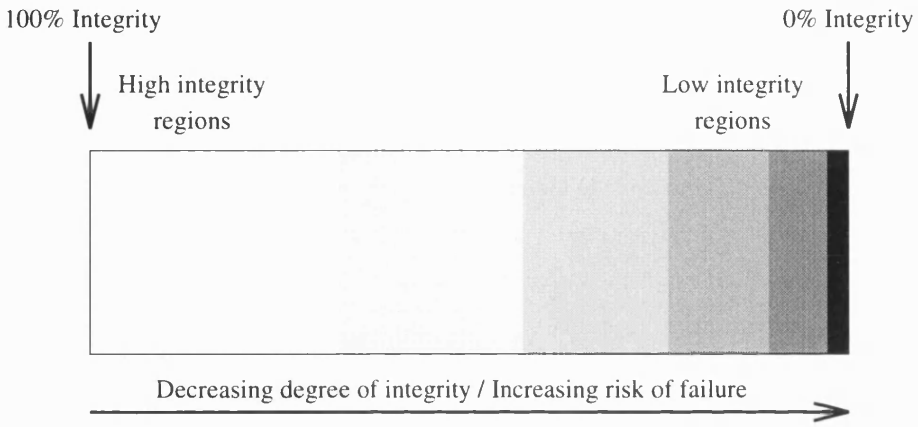


Figure 61. An example of the proposed integrity definition

The points in section a) have been discussed in section 7.3. Loss of integrity in the Magnet model may occur due to packet loss, which on its turn may result in loss of files. Hence the number of discarded packets and the number of lost files can be regarded as integrity parameters. Other parameters that can be considered as integrity indicators are those related to the quality of service. A direct example is the delays in obtaining requested files, but other measures can be derived from the obtained results, e.g. percentage of requests that succeed or fail. The application of steps b) and c) to these integrity parameters is discussed next, for both the single user and multiple users cases. Regarding section d), the monitoring activity that would need to be carried out in a real system to obtain values for the integrity parameters is equivalent to introducing measurement procedures in the model to obtain these values.

An interesting and striking feature of the Magnet model has been identified as a result of the modelling work, namely: the system adjusts itself to errors due to packets being discarded. It could have been expected that high probabilities of packets loss in the system would result in a sustained increase of the signalling load, due to the re-attempts procedures that generate new packets if the requests fail. In fact, it has been demonstrated that the effect is quite the opposite. Increase in the signalling load results in packets being discarded, which may result in files being lost. This on its own results in a decrease of the signalling load, because there are less files being moved. In summary, an initial increase in signalling load results in a significant decrease of activity. This indeed is not a desirable situation because information is being lost in the system, and appropriate correction mechanisms should be introduced, as discussed in section 7.4.1.1. However, a key feature of the Magnet service is that the response to failure depends on the severity of the failure as follows:

a) for small failure rate, i.e. small number of packets being discarded, the system 'self-heals', i.e. the re-attempts mechanisms provide a way to overcome the loss of packets and avoid files loss;

b) for larger failure rates, i.e. size of buffers significantly smaller than would be needed for an error-free situation, a decrease in the activity in the system can be observed, produced by the loss of data files.

Hence, it is necessary to find where the boundary between the two types of behaviour resides, i.e. what are the limits within which the system recovers from errors. This will identify the different integrity regions. This is discussed in the following subsections.

7.6.1. Discarded packets and lost files

One indicator of the degree of integrity is the number of discarded packets in a given time interval T. As shown in Figure 54, for the one user case a buffer size of at least 5 is required in order to guarantee that no packets are lost. Therefore, for a system where the loss of packets is critical a minimum buffer size of 5 would be required.

The threshold criterion of 0 discarded packets is very strict and represents a completely error-free scenario. However, the simulation results have shown that the Magnet service presents certain tolerance to the loss of packets. The packet loss has two effects in the Magnet service. One is to increase the delays that the users would experience to obtain service. But the most critical effect is that if packets are discarded this could result in data files becoming inaccessible. This could have catastrophic consequences, as data files contain service logic and data necessary to perform services . As discussed in section 7.4.1.2, packet loss does not necessarily cause files loss. Thus, for the one user case, a buffer size of 4 incurs a packets loss, whereas the simulations showed no file loss for this value. The loss of data files can be regarded as the major type of failure in the Magnet service, so the number of files lost in a time interval T can be considered as another integrity measure.

A key issue is to understand the dependencies between the integrity measures and the control parameters. The impact of two control parameters is examined next, namely the size of the input buffers and the number of users. The time window for the measures T is chosen as the total simulation time. This value is convenient in this case to show changes in the integrity parameters, but the 'resolution' of the measures can be changed by considering smaller or larger time intervals of analysis.

7.6.1.1. Impact of the buffer size

Figure 62 shows the relationship between the integrity parameters file loss and packet loss in the total simulation interval and the control parameter 'buffers size' for the one user case. The values have been normalised (dividing by the maximum value) to facilitate comparison of results. The number of lost packets was measured directly during the simulations. The number of lost files during the simulation interval is derived from the values in Figure 51 for simulation time = T and from the plot in Figure 52.

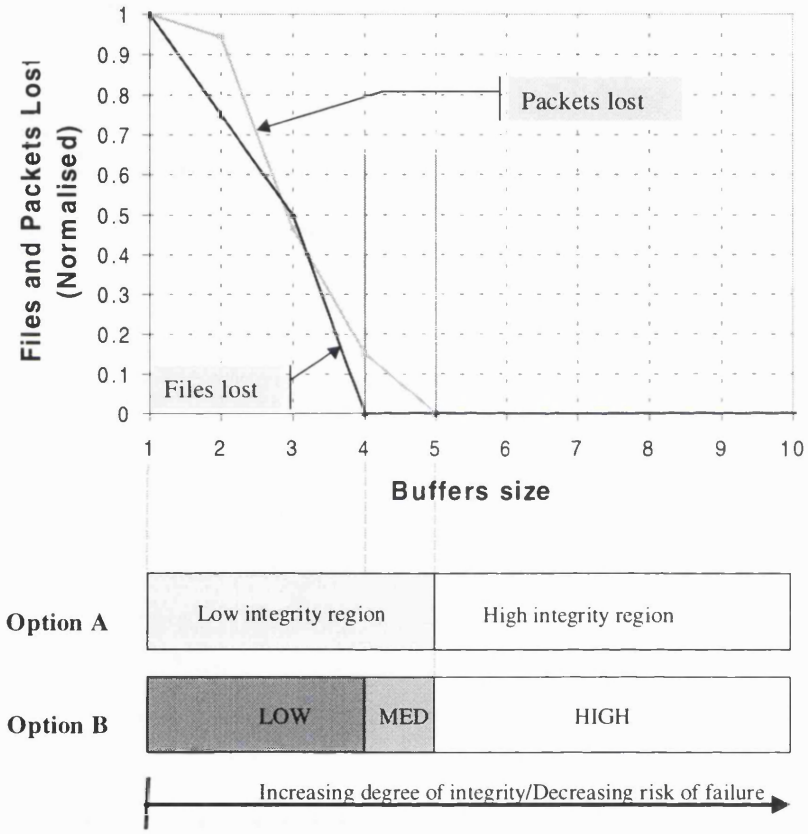


Figure 62. Definition of integrity regions based on the number of lost files for the one user case.

Consider the simple case scenario of Figure 49 where only one threshold value is defined for the integrity parameter, thus resulting in two integrity regions. Let us assume that the threshold is defined to be 0 packets lost. As shown in Figure 62, this results in a threshold value 5 for the control parameter 'buffer size'. The two resulting integrity regions for this case

are depicted in Option A in Figure 62. If the Magnet system operates within the low integrity region in option A, there is a probability of losing packets, and hence data files. If, on the contrary, it operates within the high integrity region, it is guaranteed that no files will be lost.

In a real case, the cost associated with increasing the buffers sizes should be balanced against the risk of losing data files. A deeper analysis would be required to identify the sensitivity of the services governed by the Magnet system to the loss of files. This sensitivity will depend on the type and contents of the data files that are lost and possible alternative ways to obtain the same information. In addition, other threshold criteria may be introduced to account for this sensitivity of the service to loss of files. Thus, as discussed in section 7.4.1.2, three integrity regions may be defined based on the probability of losing data files (see option B in Figure 62). A high integrity region guarantees that no packets will be discarded. This requires a minimum buffer size of 5 for the one user case. A low integrity region can be defined for buffer sizes lower than 4, where the empirical results show that data files are lost in the system. In between (i.e. buffer size of 4), there is a medium integrity band where the probability of discarding packets exists, but the results showed that this does not cause data files loss. There is however no guarantee that file will never be lost in this region, as the system is not completely error free.

The results above could assist in the design of recovery mechanisms against file loss in the system. For example, if a periodic reset of addresses database option is adopted, the frequency with which this reset ought to be performed can be chosen based on the simulation results. For instance, if the size of the buffers were fixed to 4, a periodic reset of the addresses database every 10^6 time units (which is the total simulation time) may suffice, because at that point the probability of losing files is very low. A lower buffer size of 3 may be acceptable, but it may require more frequent resets. This of course depends again on the sensitivity of the services to the loss of files. Risk assessment must be undertaken to understand the potential effects of the risk of losing files.

7.6.1.2. Impact of the number of users

Figure 63 shows the total number of packets discarded during the simulation interval for different sizes of the input buffers for the one, two and three users cases. These results show that a buffer size of 5 suffices to guarantee that no packets are discarded for the one user case, but not for the two and three users case (further simulations would be required to identify the buffer sizes that guarantee no packet loss in these cases). The integrity definition can be

applied here to identify the limits in the external variable 'number of users' for which the system maintains predefined integrity levels. Thus, let us assume a fixed buffer size of 5, and three integrity regions defined as follows:

a) High integrity region, where there is no packet loss,
b) Medium integrity region, where the average number of packets lost is lower than 1,
c) Low integrity region, where the average number of packets lost is greater than 1.

These three integrity regions are depicted in the right hand side of Figure 63. The graph shows that to maintain the system within the low integrity region under these working conditions, a maximum of 1 user can be accepted. Two users would move the system into the medium integrity region, and three users would take it into the low integrity region.

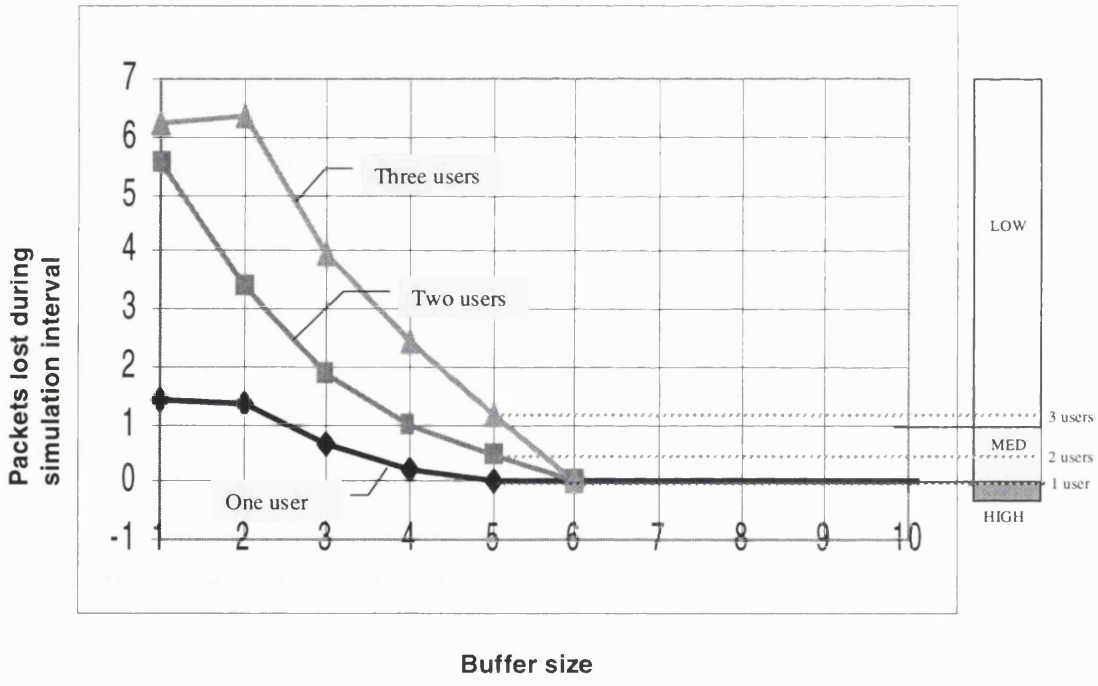


Figure 63. Total number of packets discarded during the simulation interval.

A similar analysis could be undertaken to obtain the impact of the number of users on the 'number of lost files' parameter. This is more difficult to quantify with the available simulation results, because contention for files in the multi-user cases increases the probability

of files being in transit and it is not possible to determine whether a file has been lost or is in transit. However some initial relationships can be derived. For example, for a buffer size of 5 the simulation results showed that:

a) One user in the system does not cause neither packet loss nor file loss
b) Two users in the system cause packet loss, but not file loss
c) Three users in the system cause packet and file loss.

Therefore if three integrity regions are defined as in Option B of Figure 62, e.g.:

1. a high integrity region where there is no packet loss,
2. a medium integrity region where there is packet loss, and hence the probability of losing files exists, but the empirical results showed no file loss,
3. and a low integrity region where there is file loss,

then for buffer sizes of 5 the Magnet service can only accept one user in the high integrity region and two users to operate in the medium integrity region. With three (or more) users the service will be operating in the low integrity region.

This type of information and analysis could be used in real time to maintain the integrity levels by limiting the number of customers that use the service at a given time. If the number of users reaches the threshold, further requests for service should be rejected until some users leave the system.

7.6.2. Quality of Service - impact of the number of users and the buffer sizes

A quality of service measure in the Magnet system is the delays experienced by the users to obtain the requested information. The histograms of delays shown in figures 95, 99 and 103 in appendix E can be used to extract some relevant information regarding quality of service in the Magnet model. An important observation must be made about these results, and this is that the measures only account for the requests that succeed, i.e. when the requested information is obtained. But not all requests succeed because:

a) errors due to size of buffers being too small can result is lost files, which can not be retrieved,

b) an increase in the number of users and contention for files could result in delay times higher than the time that the user stays in one location; if a user moves location before obtaining a requested file, this is a failed request.

Thus, another measure of QoS for the Magnet service is the percentage of file requests that fail or succeed. One approach to obtain these is to introduce new measurements in the system to obtain the success rate. Another option is to calculate a probabilistic estimate, as follows.

From the list of parameters in section 7.3.1 it can be seen that each user requests each of their 5 associated files with a probability of 0.5 every time they move location. This gives an average of 5*0.5=2.5 files requested per user. Users change location every 5000 time units. Movement of users starts at time at t_i=1000 time units, and simulation ends at time t_f= 1000000. Hence, the total number of times that users change location is given by the integer part of $[(t_f - t_i)/5000]$, i.e. 199 times. Thus, the estimated number of file requests for the total simulation time is 2.5*(Number of users)*199. This gives 497.5 for the one user case, 995 for the two users case and 1492.5 for the three users case.

The number of successful requests can be estimated from the histograms of delays, by adding the total number of cases. The unsuccessful requests is then given by:

a) (497.5-Number of successful cases) for the one users case

b) (995-Number of successful cases) for the two users case

c) (1492.5-Number of successful cases) for the three users case.

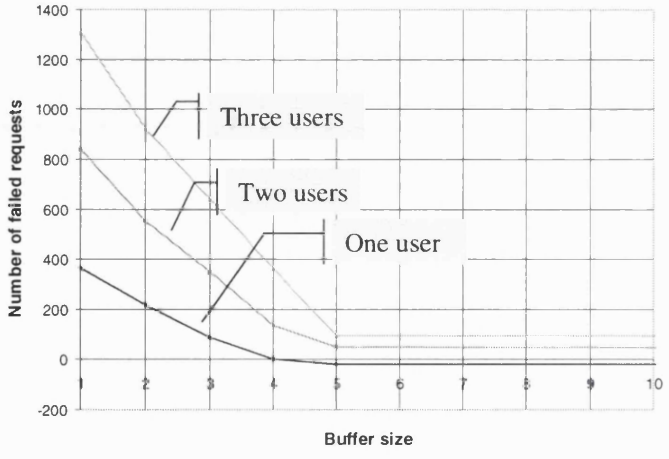


Figure 64. Failed requests in the simulation interval (estimated values)

The total number of **estimated** failed requests for the one, two and three users cases is represented in Figure 64 for different values of the buffer size. The negative values in the graph are due to these being estimated average values, calculated as described above. As expected, the more users in the system the larger the number of failed requests. This is mainly because, as stated earlier, the total number of requests – successful plus failed - is larger for larger number of users.

A more representative measure of failure is the percentage of failed requests, depicted in Figure 65. This graph shows that, as expected, the failure rate is larger for larger number of users. Again, negative values are due to the probabilistic calculation of total number of requests. Note that in order to improve the readability of the graph, the *x*-axis is only shown up to a value of 10. Therefore, only the results for the 1, 2, 3, 4, and 5 buffer sizes are shown in the plot. No results are available for buffer sizes between 6 and 10, but the percentage of failed requests decreases with higher values of buffer sizes, and the results showed that this percentage is 0 for a buffer size of 100.

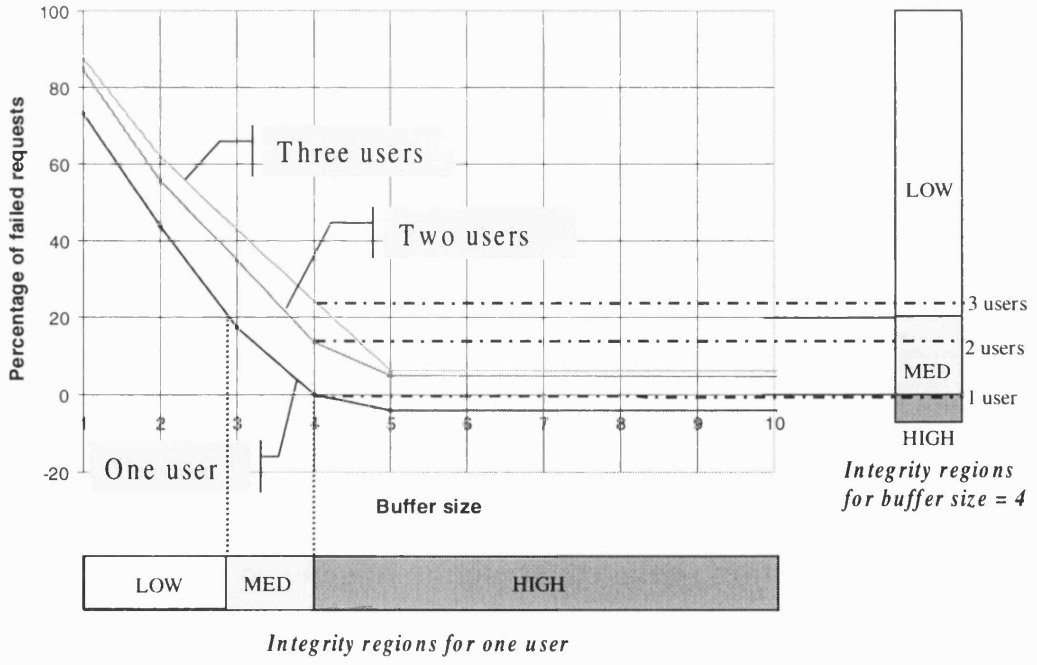


Figure 65. Estimated percentage of failed requests in the simulation time (estimated values)

As shown previously for other integrity measures, a collection of thresholds can be defined here for the percentage of failed request. The examples depicted in Figure 65 show three integrity regions defined as follows:

1. a high integrity region with 0% of failed requests
2. a medium integrity region with between 0 and 20% of failed requests
3. a low integrity region where the percentage of failed requests is greater than 20%

The following two examples are examined:

a) *Integrity regions for the one user case – impact of buffer size*
The results represented in the bottom graph of Figure 65 show that for the Magnet service to operate in the high integrity region with one user a minimum buffer size of 4 is required. With a buffer size of 3 the system would operate in the medium integrity region, whereas buffer sizes lower than 3 would cause the system to operate in the low integrity region.

b) *Integrity regions for a buffer size of 4 – impact of the number or users*
Assuming that the system is designed for a buffer size of 4, the results plotted on the right hand side of Figure 65 show that in order to operate within the high integrity region the Magnet service can only accept one user. If two users are allowed the service will operate in the medium integrity region, whereas three users would cause it to move to the low integrity region.

Again, these examples illustrate how the modelling and simulation results in conjunction with the proposed integrity framework can be used to understand the impact of the different design parameters in the integrity measures and to identify its working limitations, hence assisting in the design of the service and the network where it is provided.

7.7. Introduction to the construction of generic models

In section 3.4.1 the need for industrial organisations to develop formal, structured and generic modelling methodologies, based on re-usability of models, and applied consistently within the organisation was discussed. This requires identification of the requirements for such generic modelling methodologies for integrity modelling. The investigation of such requirements was one of the possible areas to pursue within this project. However, as discussed in previous

chapters, a different line of work was adopted, namely the application of modelling methodologies to a particular case study. As a side effect of this pursued line of work, some understanding of the needs and requirements for developing generic modelling frameworks using SDL were identified, and an introductory study in this subject was undertaken. The findings are discussed in this section.

Three basic types of components in models for integrity analysis have been identified, i.e. network descriptions, service descriptions and population characteristics. A modelling framework would need to include a collection of libraries of components of each of these three types. Note that the code written for the real services, i.e. the one that runs on the network, could be the basis for the construction of the service model libraries.

These library components should be able to be re-used by different people to model different scenarios. To achieve this, a complete separation of network, service and population descriptions is necessary.

Figure 66 shows an example of a possible modelling scenario in an SDL-like fashion. In the example, two networks are connected together. Each network is modelled as an instantiation of an SDL block type, i.e. Net1 of type NetworkType1 and Net2 of NetworkType2. These networks provide services, and this is modelling by connecting the network blocks to the service blocks. The services are modelled as instantiations of service types. In the example, Net1 provides two services, namely Service1_1 of ServiceType1 and Service2_1 of ServiceType2, and Net2 provides Service2_2, of ServiceType2. Each service will be used by a collection of customers, and this is modelled by connecting each service block to a population block. Again, each population block is an instance of a population block type. In Figure 66 Service1_1 in Net1 and Service2_2 in Net2 are coupled to a population model of type PopulationModelType1; Service2_1 in Net1 is coupled to a PopulationModelType2. This example shows how having independent library components for different types of networks, services and population models permits the combination of these components in any way to model specific scenarios.

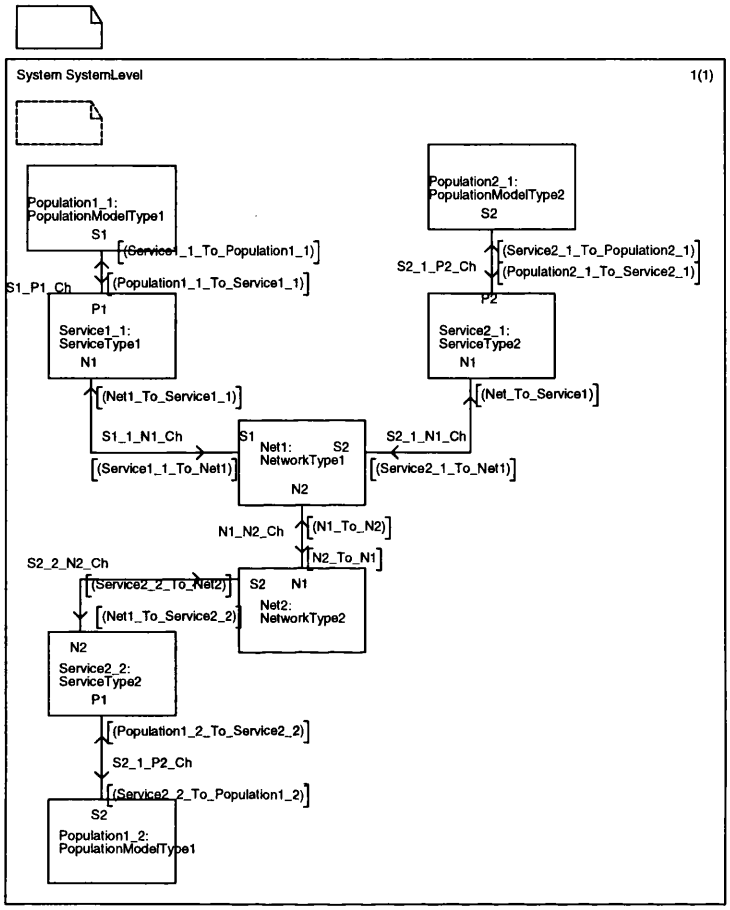


Figure 66. A possible modelling scenario using library model components

Ideally, the descriptions included in each of these libraries should be independent. This way, the user could build specific scenarios by combining elements from the different libraries. The need for independence between different components is justified as follows:

a) **Independence between population models and service descriptions:** the same service description will need to be studied for different population models, to understand the limitations and possible sources of errors in the service that are imposed by the characteristics of the population of users.

b) **Independence between service descriptions and network descriptions:** The behaviour of the services needs to be analysed for different network architectures. Any service description should be able to be coupled to any network that can provide the service.

c) **Independence between population models and network descriptions:** the characteristics of the population of users are different for each service, and are not determined by the network architecture. Each service will have a different service take up and their own population patterns.

Another essential aspect is the possibility to decide on the level of abstraction of the models, and to model different aspects with different levels of detail. For example, in section 7.3.1 it was discussed how a single-network model can be upgraded to a multi-network scenario by modelling relevant characteristics of the interconnect at a high level. In a generic case, there will be some factors external to the service under study that may influence the service performance. However, the details of these factors may not be known, or it may not be desirable to model them in detail. Hence, it must be possible to introduce statistical description of this type of factors – such as other services, network details or population models - in a given model, as some kind of noise in the system that effects the service in some arbitrary way.

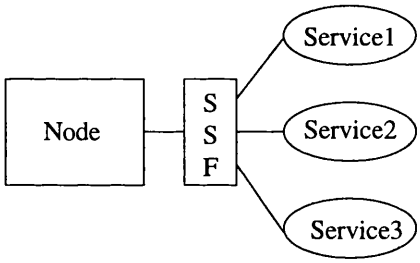


Figure 67. Service switching function (SSF) in a node

A major issue is how to achieve complete independence between components. Consider the case of network models and service descriptions. A service is provided by functionality in a network, so is it possible to build the service description without any information about the network? The reverse case also needs to be solved, i.e. how to design the components of a network without any information about the services that it is going to provide. These are complicated issues and will need to be carefully researched. For example, a service may require a database. This database can be part of the service description, since it contains information relevant to the service, e.g. user profiles. On the other hand, it could be argued that a database is a network component, and therefore needs to be included in the network model, where performance characteristics, such as average response time, would be incorporated. Therefore, the question is how to break down these elements in order to

guarantee independence between network and service elements. It seems clear that somewhere in the network model there will need to be a reference to the services that the network can provide. For example, a service switching function that selects the appropriate service object may be incorporated in the nodes. The signals received by a node will then need to carry a service identifier from which the service switching function will select the appropriate service object (Figure 67).

These service objects within a node are not the service descriptions, but the interface to the appropriate service description modules which, as previously stated, will reside outside the network module. All the service modules need to be directly connected to all the relevant nodes in the network. For this purpose, the network module will need to have a NetworkInterface block to allow for this type of connectivity (Figure 68).

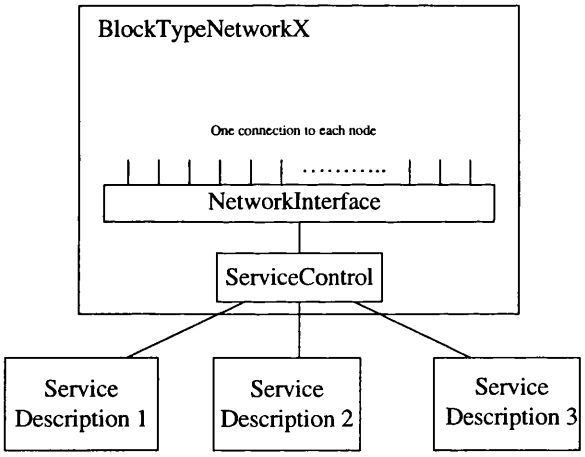


Figure 68. A network interface in a node facilitates the connection to the service blocks

The design of these NetworkInterface blocks must also be studied. Due to the restrictions imposed by SDL (refer to section 6.3), connecting these blocks to all the nodes could make the diagram unreadable. A possible solution is to break down the network interface into a collection of blocks that would de-multiplex the inputs to the network into a collection of channels in a hierarchical way as illustrated in Figure 69.

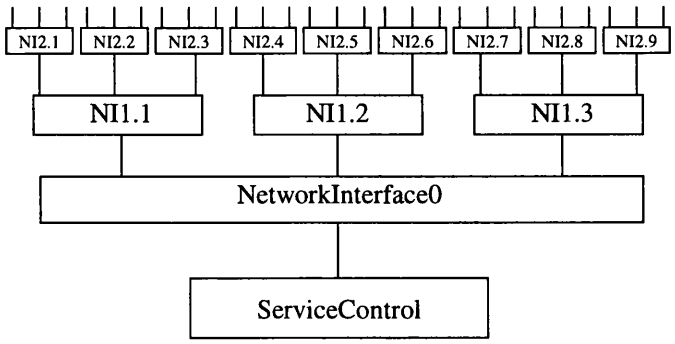


Figure 69. An example of multiplexing network interfaces

A network module simply provides a network topology (a specific interconnection of selected network components) and a routing capability. All the rest of the functionality would be provided by other modules, e.g. service descriptions. A network module is made of a number of network elements connected together. Libraries of network elements would also be provided, including components such as descriptions of different types of nodes, i.e. base nodes, database nodes, SCPs, SSPs, etc. An approach to do this in SDL is to develop a hierarchy of node types. For example, for the class BaseNode, the minimum element could be BaseNode_OneLink. Then, a type BaseNode_TwoLinks will be constructed that inherits BaseNode_OneLink and has an added link, and so forth up to a certain number of links (Figure 70). This is only one possible decomposition. Further research would be necessary to identify basic types of network components and how to build them.

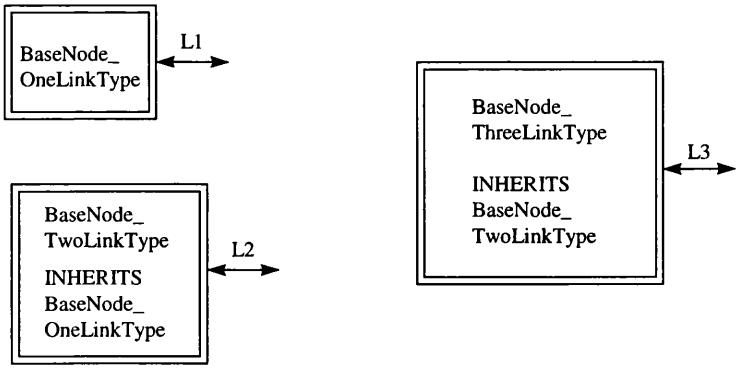


Figure 70. Hierarchy of base node components

Figure 71 shows an example of a network module composed of four nodes, the network interface and the service control blocks.

As previously stated, an essential issue is to guarantee compatibility between all the different model components. These components are likely to be developed separately by different people, and at different times. As discussed in section 3.4.1, a uniform and standardised modelling methodology must be followed by all the designers to guarantee compatibility. In order to ensure connectivity between different model elements, an appropriate design of their interfaces is essential. The library developer must account for possible extensions to the design, and incorporate flexibility. Another point is the need for a uniform naming convention and a coherent design style that can be understood by the different groups of people who build specific parts of the models, as well as by people who have not even been involved in the design process. These include potential users; or simply people to whom the work needs to be reported. Rules and guidelines will need to be developed to guarantee uniformity and compatibility. In addition, there is a need for tool development to support these modelling approaches.

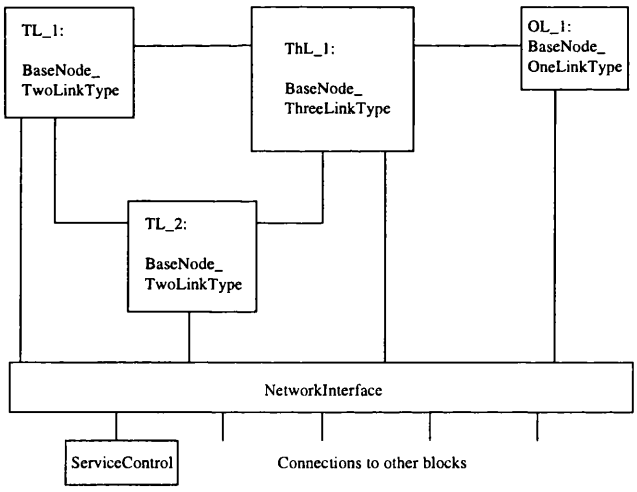


Figure 71. Arbitrary network created by connecting blocks from the nodes library

7.8. Conclusions

The focus of this chapter has been on the results obtained from the SDL modelling of the Magnet service, and more importantly on the analysis of the proposed modelling methodologies and integrity definition in the light of these results.

The chapter started with a discussion on how the simulation results from the SDL model can be applied to obtain integrity measures, to categorise the degree of integrity in regions and to understand the effect of control parameters in the integrity levels. Next, a general description of the results obtained for the Magnet service was presented. These results include performance measures such as the generated signalling load, and quality of service parameters such as delays in obtaining requested information. Several types of failure in the Magnet service are identified, e.g. the probability of losing data files and packets.

The modelling framework proposed in section 3.4 has been reviewed by describing how it applied for the Magnet service. The usability of most of the proposed ideas has been illustrated. There were a small number of ideas that could not be pursued, partly because of the limitations encountered in SDL and its associated tools. These refer to the identification of error states and calculation of probability trees.

The results from the SDL model of the Magnet service have also been used to formulate the connection to the network integrity definition proposed in section 3.2 aimed to avoid severe integrity degradation. Examples of how to define integrity regions and obtain the threshold values for control parameters have been presented. The study is limited to the analysis of one control parameter and one integrity measure at a time. Indication on possible further extensions has been included.

The Magnet service has been simulated for a fixed network size. As discussed in section 8.3, this was due to the SDL limitations to build large systems and to introduce structural changes in the models. Therefore, it has not been possible to test the service for larger networks. However, it must be emphasised that the model was designed to investigate concepts and test methodologies. The Magnet service was designed to embrace key aspects in current telecommunication services, but it was not the aim of the model to obtain the best implementation for it. It can be expected that increasing the size of the network would affect the performance of the Magnet service, but not the basic behaviour. Increasing the networks size would increase the average distance that files must travel, and hence the delays that users would experience in obtaining the data. However, the assumption is that the users residence times in a location are much longer that the file transfer times. Hence, the qualitative behaviour of the service should not be different to that in smaller networks. Performance parameters such as value of time-outs and buffer sizes would need to be tuned to the specific network conditions, and possibly different integrity regions and thresholds should be defined

for them, but it can be expected that the basic behaviour and type of errors identified in our model would remain the same.

The chapter ends with some introductory ideas about the construction of generic models, identified as a result of the modelling work. This is considered a very important area of research. Here it is believed that there is a need for commercial tools that permit modelling and simulation of services, clear and powerful visualisation capabilities to improve understanding of the systems modelled, and data processing features to analyse the results.

Chapter 8

Conclusions

The major motivation for this work was the need to develop structured and systematic approaches to maintain the integrity of the increasingly complex telecommunication networks and services. The first stage of the work was the analysis of the main characteristics of modern telecommunication networks and services, and the identification of areas related to network integrity. In each of these individual areas there is extensive expertise, but the contribution of this work has been to analyse them in an integrated approach, focussing on their connections to network integrity. The conclusions from this initial analysis were:

a) An increasing degree of intelligence is being introduced in telecommunication networks. This is giving rise to a dramatic growth in control complexity and fragility, exacerbated by increasingly sophisticated customer demands for quick deployment of services and the regulatory policies fostering competition and open interconnect. The work has focussed on integrity threats caused at these intelligent levels.

b) The problem domain is beyond the comprehension of single individuals and there is need for more formal and structured techniques and approaches for the design, development and maintenance of networks and services.

c) Pre-emptive approaches are required to avoid catastrophic failures cause by dramatic degradation of network integrity.

A multi-paradigm approach is needed including improvement of design methodologies as well as real time integrity quantification and restoration techniques.

A key contribution of this work has been to introduce a quantitative and measurable definition of network integrity that can be used in a pre-emptive way to reduce risk of integrity beach. The definition can be used to assess the risk of failure, to prevent failure situations and to assist rapid recovery in case failure occurs. The pre-emptive capabilities are a key feature of such definition, allowing rapid detection of variations in the integrity levels and hence permitting action to be taken before this integrity degradation exceeds some predefined limits.

The project started with an extremely open and large problem space, i.e. how can the integrity of telecommunication systems be improved. The need for decomposition became immediately apparent in order to break down the problem into manageable areas. This was done bearing in mind the need for a multi-paradigm approach mentioned above, and the result was the overall integrity framework embracing static and real-time activities. This framework is proposed as a guideline, highlighting a number of key interrelated activities that should be undertaken by telecommunication organisations to enhance the integrity of their systems. This has been defined at a very high level, and the problem domain at this stage was still well beyond the scope of this project. Hence the scope was narrowed down further by focussing on one of the proposed activities, namely the use of modelling and simulation techniques as a tool to assist in the improvement of network integrity.

In the area of modelling, an important part of the work has been the investigation of requirements that modelling approaches and methodologies should meet in order to be successful in commercial organisations. The commercial orientation has been a constant in this work. A key objective has been the investigation of approaches and techniques that would be useful in industrial organisations, as opposed to purely academic solutions. A main conclusion has been that there is a strong need for well structured and supported modelling frameworks where models are created by connecting existing library components. This requires defining and building generic re-usable modelling components. This is a very technically complex proposition and would require large amount of effort, time and investment. Some insight into this topic has been discussed in this thesis, but there is still some way to go before generic re-usable modelling approaches are in place.

The development of models requires an initial detailed analysis and planning phase. The main types of decisions and areas that need to be addressed in this planning phase have been discussed. This includes identifying the objectives of the models and the main inputs, deciding the languages to be used to build them, and developing a methodology for the construction of the models. A modelling case study, the so-called Magnet service, has been used in a number of ways. An important application has been to test the suitability of the languages employed. In particular, a main outcome has been the identification of limitations of SDL as a modelling language for large complex systems where performance information is crucial. In the latter area, a review of approaches to include performance into SDL specified systems has been introduced, followed by the author's proposed solution. The case study has also been used to test the proposed modelling techniques, and it has been argued that most of the ideas proved valid. It has also been discussed how part of the proposed methodology was impractical, mainly due to the intrinsic complexity of the problem, the limitations encountered in the modelling languages and the time constraints in this project. Another application of the case study has been to demonstrate how models can assist in the identification of design limitations, key integrity parameters in the system and the effect of external conditions e.g. population characteristics. Finally, it has been used to illustrate the use of the proposed integrity definition to categorise the integrity of a system into a collection of regions, identify the region in which the system operates and detect possible integrity violations.

The Magnet service is a simple example, and does not represent a specific existing service, but it was carefully designed to incorporate aspects of potential future services. In particular, the convergence of the Telco world and the information technology world and the proliferation of information services, which rises the problem of providing fast and easy customer access to large amounts of information. Thus, the management of information is a major feature of the Magnet service. Customers are mobile, and they demand ubiquity of services, as well as customisation of these services to meet their individual needs. This requires access to customer profiles wherever the customers are, and this is another essential feature of the Magnet service.

As previously stated, the commercial applicability of the work has been a mayor drive of this work. Hence, an essential requirement identified in the context of modelling is the use of languages and tools that are easy and intuitive to learn and use. This determined some of the decisions taken during the project, such as the selection of SDL for the modelling work. But SDL was not designed for this type of use, hence considerable part of the work has been the identification of limitations in SDL and, where possible, propose enhancements. These

limitations proved rather large, so too many enhancements and changes to SDL would be required to overcome them. The conclusion is that SDL is not really the best suited tool for this job, and other languages could have been more effective. The use of a high level programming language would have provided more flexibility and would have permitted to explore more aspects in the simulations.

The importance of using modelling techniques in commercial organisations such as operators, manufactures and service providers has been emphasised in this thesis. Modelling is proposed to aid understanding of new services and networks and to help eliminate potential problems before the service launch. But for models to be useful, an integral and consistent approach must be followed, and the right infrastructures need to be in place.

Building models from scratch for each specific case of study is inefficient and results in long delays. For models to be beneficial, the users of the models would need access to a well-defined and structured modelling environment, with a collection of libraries readily available for use. These modelling environments need to be user friendly, since the personnel likely to use them would be experts in the services and systems that they want to model, but not necessarily in programming and simulation techniques. This requires appropriate tools that combine the power requirements with user friendliness and short learning periods. Such types of tools are not available, and here it is believed that this is a very important area, and there is a great commercial need for them. Developing this entire modelling infrastructure requires a significant amount of investment in terms of time, money and resources. It is a long-term proposition with no immediate return. This makes it a difficult task within commercial organisations, specially in the competitive telecommunications environment, where time to market is crucial. However, it is believed here that the problems are becoming so critical and hard to solve that manual fixes will become impossible, and commercial organisations will need to evolve into this type of structured and automated methods.

Carrying out this work in academic institutions on their own would not be appropriate either, because it is very important that the techniques have all the aspects associated with commercial products in mind. Academic research is not limited by these constraints, and hence there is the risk of producing largely theoretical work with little application to commercial environments. Hence the proposed solution is collaborative work between industry and academia. Industry has the requirements, the money and the infrastructure. Academia has the flexibility, time and human resources.

Appendix A

Overview of IN

Intelligent Networks (IN) represent a major milestone in the recent evolution of telecommunications. The IN is gradually being adopted by major operators and it is believed that they will be the predominant technology across the world within the next few years. Hence, understanding service provision under the IN paradigm and emerging issues is of great relevance to identify integrity threats. This appendix contains a summary of the basics of IN.

The International Telecommunications Union -Telecommunications Technical Committee (ITU-T) has proposed an evolving sequence of standards that support IN. The starting point is a core structure of capabilities called Capabilities Set 1 (CS-1), which will be advanced by the addition of extra capabilities and services to form the next definition, CS-2. The process is proposed to continue to produce CS-x, each new set built on the last, with ever expanding capabilities.

The IN conceptual model defines four planes of decreasing abstraction - Service, Global Functional, Distributed Functional and Physical [Bale 95]. Each layer supports a one to many mapping; i.e. each entity at one layer is supported by many interacting entities from the layer below. The IN conceptual model is represented in Figure 72.

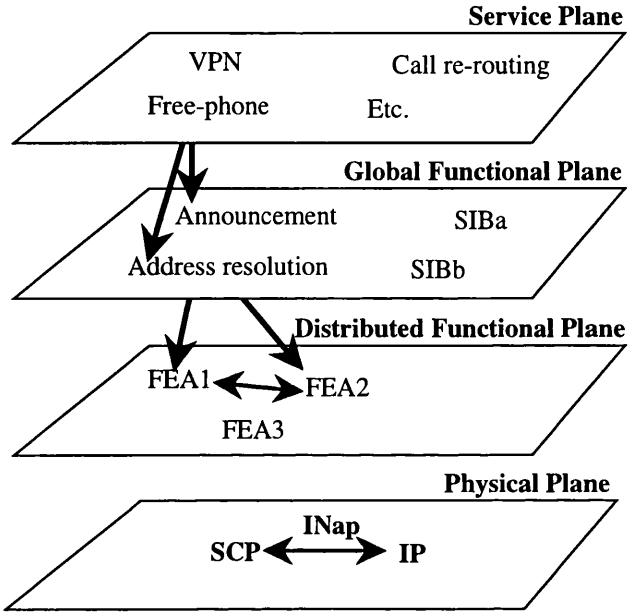


Figure 72. The four layers in the IN Conceptual Model

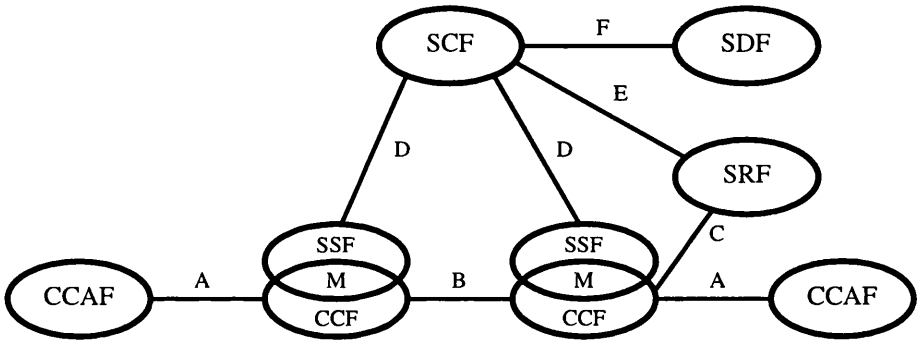
The **Service Plane** is the level where services are identified and defined. This level does not specify how services are designed and implemented. They are simply described and decomposed into service features in an implementation independent way. Each service is defined as a single program or Global Service Logic (GSL).

The service features are realised in the **Global Functional Plane** (GFP) by Global Service Logic. The GFP has an overall abstract view of the whole network and its capabilities. These capabilities are modelled as Service Independent Blocks (SIBs), which are a set of basic components used by the GFP to build services. The SIBs are still abstract functional capabilities that operate globally throughout the network. In other words, a SIB is a task that the network can perform and it always provides the result best suited to the location of the request, regardless of where it was initiated. The concept of SIBs permits isolation of the service from operational details. In principle, a service is a piece of logic that calls a set of SIBs to perform its task.

The SIB is the most powerful concept of the IN conceptual model. In IN capability set 1 (CS-1), which is the ITU-T specification of requirements for a defined set of IN based services, a number of SIBs are defined, covering areas such as charging, user interaction and call

queuing. A SIB worth a special mention is the Basic Call Process (BCP) which represents the basic call-processing capability of the network. This is basically the simple process of telephony, augmented by details of the interaction of the BCP and other SIBs.

The **Distributed Functional Plane** (DFP) is where all the functional entities that support the SIBs are identified. The functionality is distributed, because the network is made up of many distributed programs that provide a single unit action. Such functions or actions are referred to as entities; hence, they are called Functional Entity Actions (FEAs). Functional relationships between the functional entities are defined, i.e. the FEAs used to construct a SIB must communicate and information must flow between them. It is these functional relationships which determine the requirements of the signalling protocols used in the IN (Intelligent Networks Application Protocol (INAP) of the SS7 standard (dealt with in appendix B).



CCAF	Call control agent function	SRF	Specialised resource function
CCF	Call control function	SCF	Service control function
SSF	Service switching function	SDF	Service data function

Figure 73. Functional architecture of the DFP defined for IN CS-1

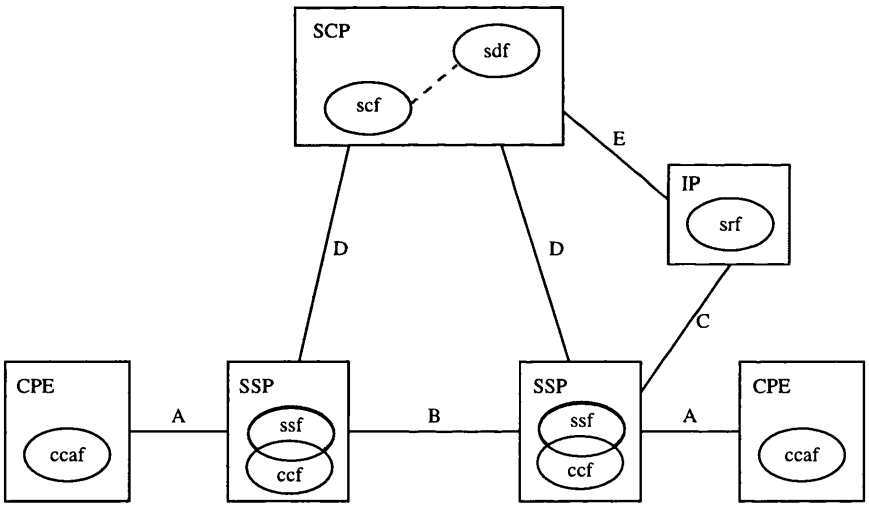
Figure 73 shows part of the functional architecture of the DFP as defined for IN CS-1. This architecture shows a number of functional entities and functional relationships. In the DFP each of the SIBs are broken down into actions to be performed by the FEs and information that flows between them. Each standard functional relationship for CS-1 requires the following capabilities:

◊ Carrier network connection control flows A, B and C.

◊ Standard call control functions for non-IN services are needed for A and B.

◊ IN service control is needed for D, E and F.

All the flow structure in the CS-1 model has been standardised, except for the flow between the service switching function and the call control function, M. This is because it is expected that this will be tightly coupled, held within one piece of hardware and be proprietary. The only times that calls can interact with IN processes are at special armed detection points in the call sequence. Various types of service logic can be invoked at the detection points depending upon the information that is gathered. At this stage the information flows are associated with a call, but once the service logic has been invoked, it can start to use the information flows D, E and F. The precise form of these information flows will be given by the operation of the SIB invoked and the method of implementation of that SIB.



CPE	Customers' premises equipment	ccf	Call control function
SSP	Service switching point	ssf	Service switching function
SCP	Service control point	srf	Specialised resource function
IP	Intelligent peripheral	scf	Service control function
ccaf	Call control agent function	sdf	Service data function

Figure 74. Example IN physical plane showing allocation of functional entities to physical entities

The **Physical Plane** is where each of the functional entities are mapped onto the physical network elements. The details of the physical network architecture specify the details of the

information flows and hence the implementation details. For some physical implementations, different functional entities may reside within the same physical entity, in which case the relationship between the functional entities is internal and not standardised. The basic proposed physical architecture is based around a Service Control Point (SCP), Service Switching Points (SSP), Intelligent Peripherals (IP) and a signalling network, as shown in Figure 74.

The planes of the IN Conceptual Model provide an interface between higher level functions and lower level implementations, as well as a route to construct IN services. The different planes are independent in the sense that changing the structure in one plane does not effect the higher level.

Services are the basis of the IN. CS-1 specifies the requirements for a defined set of, so called, Type A services. Type A services support "single ended" calls with a single point of control, i.e. one SCP. Branching and multi-threading is not supported. CS-1 specifies a set of benchmark Type A services, e.g. abbreviated dialling, call forwarding, call re-routing distribution, conference calling, premium rate, freephone, terminal call screening, virtual private network, etc.

Appendix B

Overview of SS7

The standard signalling system widely used in today's digital networks is the ITU-T Signalling System number 7 (SS7). This is a digital, message-based common channel signalling system. SS7 has a layered structure that consists of four levels. The SS7 protocol reference model is shown in Figure 75, with a comparison to the Open Systems Interconnection (OSI) 7 layered model. An important aspect of the SS7 model is that there is a clear separation of the application process (user part) from the underlying protocols (message transfer part - MTP -, which is common to all of the user parts).

Message Transfer Part (MTP)

The function of the MTP is to provide a reliable signalling connection between any two points in the network. It comprises three levels, namely:

1. Signalling data link (MTP level 1): provides a physical bi-directional transmission path between two directly connected networks elements
2. Signalling link (MTP level 2): provides a reliable logical signalling channel between two directly connected networks elements

3. Signalling network (MTP level 3): provides routing of signalling messages from any point in the SS7 signalling network to any other point. It also provides for traffic, route and link management of the signalling network; for example, automatically re-routing a signalling connection on detection of failure at level 2.

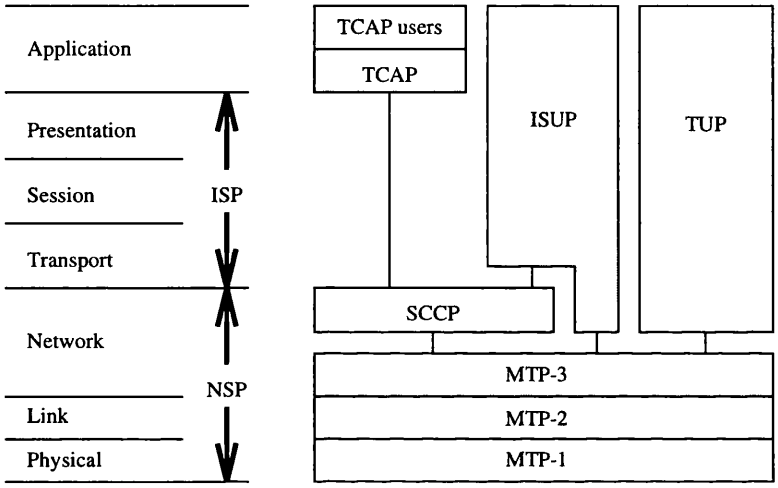


Figure 75. SS7 reference model. Comparison to the OSI 7 layered model

Signalling Connection Control Part (SCCP)

The SCCP was introduced by ITU-T to align SS7 with the OSI model. It provides an OSI network layer conformant interface to the MTP, enabling OSI conformant higher layers to be used with SS7. The combination of the MTP and the SCCP is referred to as the Network Service Part (NSP), which provides reliable transfer and global routing and addressing of signalling messages through the network.

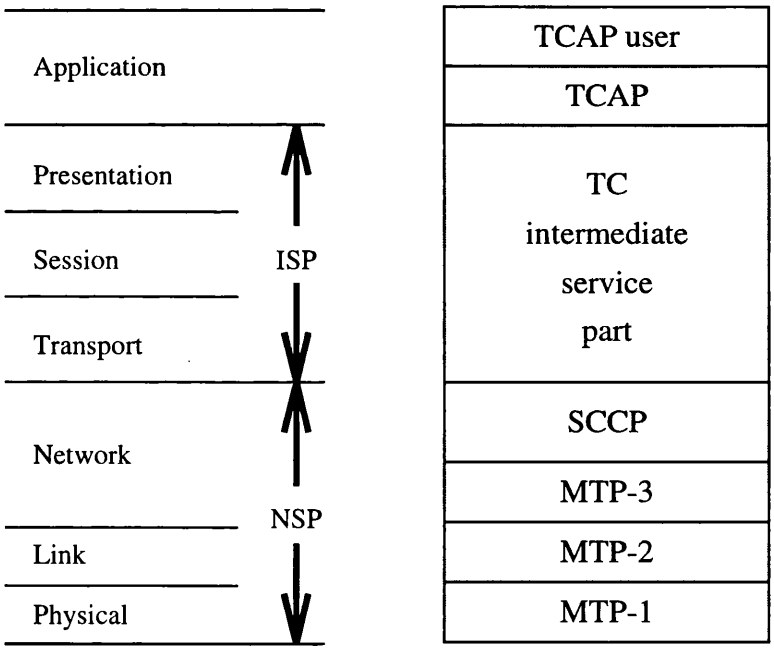
User Parts

The user parts are above the MTP, and tend to incorporate all the application functionality. A number of user parts have been defined by the ITU-T. These include the telephony user part (TUP), to support basic telephony, and the ISDN user part (ISUP), to support ISDN (Integrated Services Digital Network) call and connection control.

Transaction Capabilities

Intelligent services are going to make use of the network in varied ways, and are going to do this by capabilities given to them through the SCCP. The use of the capabilities, or Transaction Capabilities (TC), offered by this transmission protocol are all non-circuit related. Circuit related signalling is covered adequately by the other user parts.

There are a whole set of service structures defined to be supported by IN and these functions are to be built upon TC. The architecture of the TC is represented in Figure 76. Layers 1-3 of the OSI model are dealt with by the MTP and SCCP, and are referred to as Network Services Part (NSP). Layers 4-6 are dealt with by the Intermediate Service Part (ISP), leaving only the application layer to be specified. This is actually divided into two separate structures, the Transaction Capabilities Application Part (TCAP) and TCAP User.



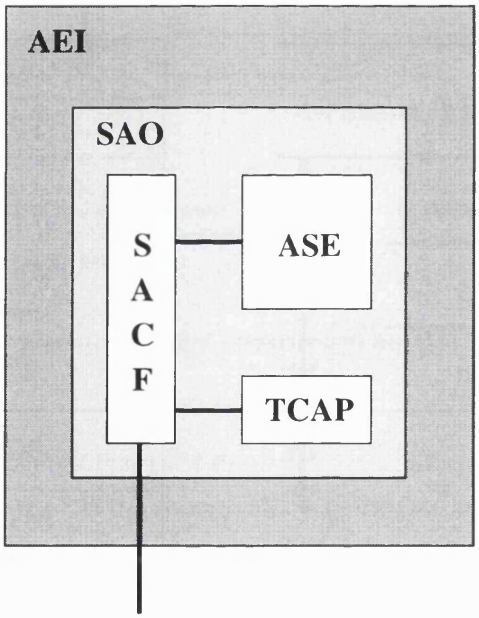
MTP	Message transfer part	SCCP	Signalling connection control part
TC	Transaction capabilities	TCAP	Transaction capabilities application part
ISP	Intermediate services part	NSP	Network services part

Figure 76. Transaction capabilities architecture

IN and Signalling

As previously stated, a feature of the intelligent network architecture is that it allows the platforms providing basic services (call processing) to be separated from the platforms providing supplementary services. The approach taken to define the signalling for the new interfaces defined in the IN was to have a clear separation between the application-specific protocol and the underlying protocols that support it. This enables the application-specific protocol to be designed for high functionality and flexibility, and the underlying protocols designed for efficiency and resilience.

The NSP of SS7 (MTP and SCCP) fulfils the requirements for the underlying protocols; the Transaction Capabilities Application Part (TCAP) and TCAP users fulfils those of the application specific protocol. The TCAP user for IN signalling is the Intelligent Network Application Protocol (INAP).



AEI	Application entity invocation	ASE	Application service elements
SAO	Single association object	TCAP	Transaction capabilities application part
SACF	Single association control function		

Figure 77. INAP protocol structure

The INAP is the protocol that supports the communication between functional entities in the IN. Each functional entity communicates with its peers through Application Entities (AE),

which are the protocols and procedures for communication. The AEs are part of the communication process, but they are also programmes in their own right. So the functional entity is the parent programme and the application entity is the communication procedure for a given type of peer. An AE that is running is referred to as Application Entity Invocation (AEI), which distinguishes the mechanism from a specific running instance. The structure of an AEI is shown in Figure 77 and described below.

An application entity must talk to, or be associated with, at least one peer AE. For each of these associations, a Single Association Object (SAO) is invoked. For a given SAO, there are a set of capabilities related to its communications ability, referred to as Application Service Elements (ASE). The service elements need to be controlled through a parent procedure, and this is called a Single Association Control Function (SACF).

Before all the above is set up, a connection has to be set up between the two AEs, and a number of AEs that are needed should be agreed upon. For this initiation procedure, TCAP is used as a universal language. In other words, TCAP performs the basic connection process.

If more than one AE needs to be dealt with, then several SAOs must be invoked. In this case, multiple SAOs may be called and then they have to be managed through a Multiple Association Control Function (MACF).

Appendix C

The analytical model

This appendix contains the analytical calculations corresponding to the first C++ version of the service in the case of random population relocation. The analysis is made for both the case of always moving files if they are requested and the threshold case.

C.1. No Threshold

We can obtain analytic results for the case of random motion of the model file. The basic parameter set is given by the fact that we are applying periodic boundary conditions to a square lattice network, the network being of size $X \times X$. The analysis is based on the case considered for the simulation where there are only two priorities for the file transfer probabilities, but there is no reason why this cannot be generalised.

Within this network, we are considering a single model file and a set of related files. In the simple case dealt with these files are related to one and only one model file. This means that there is no contention between the placement and requirement on any of the normal files. The effect of contention due to the sharing of files is examined in the second version of the service.

Due to the fact that the model file is always moved to a random location in the network, the distribution of files can always be regarded as uniform. When a model file is re-located, it initiates the movement of a random selection of files to this location. Because the current

location has no relationship to any previous location, the moved files can be at any location. This leads to the observation that the probability that a file is at a given location is X^{-2}. On subsequent movement of the model file, the previously moved files are at a location that has no relationship to the new one and the files were selected at random. Hence, their location relative to the new position is equally likely to be any site. The net effect is that the probability of file location remains constant in time. The uniform and temporally static distribution makes the calculation of average quantities such as bandwidth usage very straightforward. The only caveat is that the results are averages and must be compared to well averaged simulation results.

The file sizes have a distribution of values. This distribution can easily be changed within the simulation, but we chose to use one that led to a simple analytic expression average size. Provided that the total number of files, N, is relatively large, then our calculations are not susceptible to fluctuations. The files have randomly assigned lengths between S_{min} and S_{max}. Therefore the size of file i is

$$Si = S\min + (S\max - S_{min})ri, \qquad \textbf{1.1}$$

where the random number r_i is obtained from the defining distribution. The distribution chosen for our comparison was that of a uniform distribute, between 0 and 1.

$$P(r) = \begin{Bmatrix} 1, & \text{if } 0 \geq r < 1; \\ 0, & \text{otherwise} \end{Bmatrix} \qquad \textbf{1.2}$$

This can be used to calculate the average file size

$$\begin{aligned} \langle S \rangle &= S\min + \int_0^{\infty} (S\max - S\min) r P(r) dr \\ &= S\min + \frac{1}{N}(S\max - S\min) \int_0^1 r dr \qquad \textbf{1.3} \\ &= Smin + \frac{1}{2}(S\max - S\min) \end{aligned}$$

As we calculate bandwidth usage in a time step as the size of the file multiplied by the distance moved. Therefore, the next thing we need to know is the average distance that a file is moved.

This is also easy to calculate, due to the fact that the normal files are distributed uniformly and we are applying periodic boundary conditions to our square lattice of nodes. The average distance is simply the sum of probability of a file being at a node multiplied by the distance of that node from the current position. The number of sites at a given distance, i, form a diamond of 4*i sites, provided the diamond fits within the lattice, i.e. $i \leq X/2$. This contributes the first part of the average and is simply the sum

$$\langle X_{int} \rangle = \frac{1}{X^2} \sum_{i=1}^{X/2} 4i^2 \qquad \textbf{1.4}$$

If i exceeds X^2, then the number of sites decreases with increasing i and is given by $4(X-i)$, hence the remainder of the sum is

$$\langle X_{fin} \rangle = \frac{1}{X^2} \sum_{i=X/2+1}^{X} 4(X-i)i \qquad \textbf{1.5}$$

The average distance can be calculated by adding these two sums together.

$$\langle X \rangle = \langle X_{int} \rangle + \langle X_{fin} \rangle = \frac{X}{2} \qquad \textbf{1.6}$$

The final thing we need to consider is the number of files in the system and the likelihood that they are to be moved. We are dealing with the case where there are only two priorities, the general case being M. Each priority, j, which contains N_j files, is assigned a certain probability, p_j that the file will be accessed and hence moved each time the model file is moved. We consider the case that at every time step the model file is moved. In our simple example, extension of this simplification could be incorporated into a change in the values of p_j. The average number of files moved, K, at a given time is the calculated from the sum

$$K = \sum_{j=1}^{M} N_j p_j \qquad \textbf{1.7}$$

The case considered here is $M = 2$ and is given by

$$K = N\left(n_1 p_1 + (1-n_2)p_2\right) \qquad \textbf{1.8}$$

It is now a simple matter to calculate the average bandwidth, B, used during the time that the model file is at a given location. The first thing to do is to calculate the total amount of data moved, S_T

$$S_T = \sum_{j=1}^{M} \langle S_j \rangle N_j p_j \qquad \textbf{1.9}$$

where $\langle S_j \rangle$ is the average size of the file in priority j. In the case considered in the simulation both priorities have the same average size, hence

$$S_T = \langle S \rangle N\left(n_1 p_1 + (1-n_1)p_2\right) \qquad \textbf{1.10}$$

The total bandwidth used is then just this multiplied by the average distance a file travels

$$B = \langle X \rangle \langle S \rangle N\left(n_1 p_1 + (1-n_1)p_2\right) \qquad \textbf{1.11}$$

Which when it is noted that the minimum file size is chosen to be zero can be written in full as

$$B = \frac{1}{4} X S_{max} N\left(n_1 p_1 + (1-n_1)p_2\right) \qquad \textbf{1.12}$$

This is used to verify the simulation

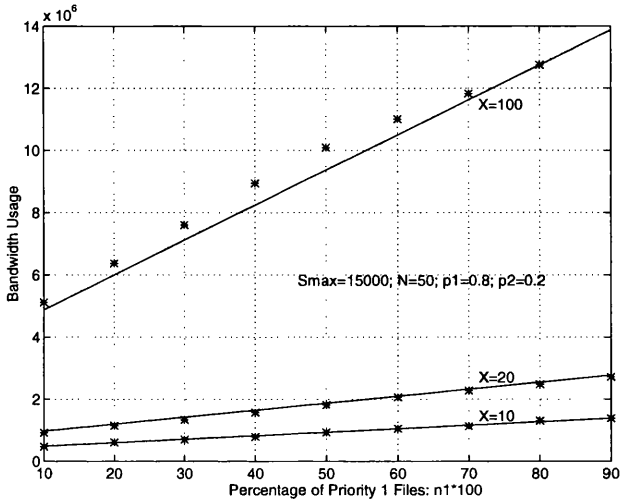


Figure 78. The bandwidth usage as a function of priority 1 files. The solid lines are the theoretical values and the stars are the results of the simulation

Figure 78 shows a comparison of Equation 1.12 with the simulation results using the appropriate parameters. The results are shown as a function of n_1 and shown for three separate values of X. The correspondence of course is not perfect. The minor deviation can be attributed to inadequate averaging and hence the results can be regarded as validation of the simulation.

C.2. Threshold

Lower Than Threshold Case

The details of the process for the case of moving files when the bandwidth is less than the threshold are considered below including the analysis of the special case of random relocation. Similar calculations can be performed for case **b**, but for brevity, they are omitted.

The introduction of a threshold complicates the analysis, but provided we consider the case of random relocation of the model, it is still possible. There are two distances that are important, the range within which all files would be below T, f_{min} and the range beyond which all files exceed T, f_{max}.

$$f_{min} = \frac{T}{S_{max}}, \quad f_{max} = \frac{T}{S_{min}} \qquad \textbf{2.1}$$

Due to the square nature of the lattice, the minimum distance defines a diamond region of sites. The number of sites in this region m_{min} is easily calculated, with the caveats that there are two case dictated by f being less or greater than $X/2$.

$$m_{min} = \begin{cases} 2f_{min}^2, & \text{if } f_{min} < X/2; \\ X^2 - 2f_{min}^2, & \text{if } X/2 \le f_{min} < X; \\ X^2, & \text{if } f_{min} \ge X. \end{cases} \qquad \textbf{2. 2}$$

The remainder of the calculation will be based on moving files if they are below the threshold. The average size of the files moved in this region is given by Equation 1.3. However, this is not true extended beyond this region.

If the range is extended by one site then a perimeter of sites is defined from which files could be moved, provided the file size is below a certain value. The maximum size of files that can be moved is given by

$$S_1 = \frac{T}{f_{min} + 1} \qquad \textbf{2. 3}$$

The average file size can be calculated simply by considering Equation 1.3, where S_{max} is replaced with S_1. The process can be repeated for the next set of sites and then indefinitely until either f_{max} or X is reached. Because the file sizes are assumed to be distributed uniformly, then the fraction of files below this value can be calculated as

$$s_f = \frac{S_i - S_{min}}{S_{max} - S_{min}} \qquad \textbf{2. 4}$$

For simplicity sake we fix $S_{min} = 0$, so this can be rewritten as

$$s_f = \frac{f_{min}}{(f_{min}+i)} \qquad \textbf{2.5}$$

The contribution to bandwidth usage for the ith perimeter is therefore given by

$$B_i = \frac{TN}{2X^2}\frac{f_{min}}{(f_{min}+i)^2} m_i\left(n_1p_1+(1-n_1)p_2\right) \qquad \textbf{2.6}$$

The total bandwidth usage can be written as

$$\begin{aligned} B \quad &= \frac{TN}{3X^2}(2f_{min}+1)(f_{min}+1)\left(n_1p_1+(1-n_1)p_2\right) \\ &+ \sum \frac{2TN}{X^2} f_{min}\left(n_1p_1+(1-n_1)p_2\right) \qquad \textbf{2.7} \\ &+ \sum_{j=X/2+1}^{X} \frac{2TN}{X^2}\frac{f_{min}}{(f_{min}+j)}(X-f_{min}-j)\left(n_1p_1+(1-n_1)p_2\right) \end{aligned}$$

It is a simple matter to rewrite this as

$$\begin{aligned} B \quad &= \frac{TN}{3X^2}(2f_{min}+1)(f_{min}+1)\left(n_1p_1+(1-n_1)p_2\right) \\ &+ \frac{2TN}{X^2} f_{min}(X-f_{min})\left(n_1p_1+(1-n_1)p_2\right) \qquad \textbf{2.8} \\ &+ \sum \frac{2TN}{X^2}\frac{jf_{min}}{(f_{min}+j)}\left(n_1p_1+(1-n_1)p_2\right) \end{aligned}$$

Note that the above equation assumes that $f_{min} < X/2$; it is straightforward to obtain the alternative case, or even the general solution.

Having derived Equation 2.8, a comparison to the simulation results is possible. We consider the usual test case where the parameters are set such that $N = 50$, $X = 20$ and:

$$s_1 = s_2 = 15000$$
$$p_1 = 0.8; \quad p_2 = 0.2$$

The value of n_1 is considered through the range 0.1-0.9,with each being a separate line on the graph. The threshold was set to be $T = 7500\sigma$, where σ ranges from 0 to X and is used to label the x-axis.

The simulation results are averaged over 100 separate runs for each set of parameters. This gives reasonable level of averaging for the quantities monitored. However, the sensitivity of a particular output variable can vary considerably.

The comparison in Figure 79 shows excellent agreement between theory and simulation. The surprising result is that although the process is governed by complex non-linearities, the growth of bandwidth usage with increasing threshold is predominantly linear. Only at very small thresholds and near saturation does it deviate from this, there does appear to be a slight disparity on the point at which the bandwidth usage saturates, the simulation saturating earlier than predicted theoretically. This is only slight and is hard to quantify as it is within the errors of the measured results. If the effect is real, it indicates that there are slightly more files moved, or they are on average slightly larger than predicted. This is perfectly possible as the exact way in which real numbers were truncated to integers in the simulation was treated in the analysis.

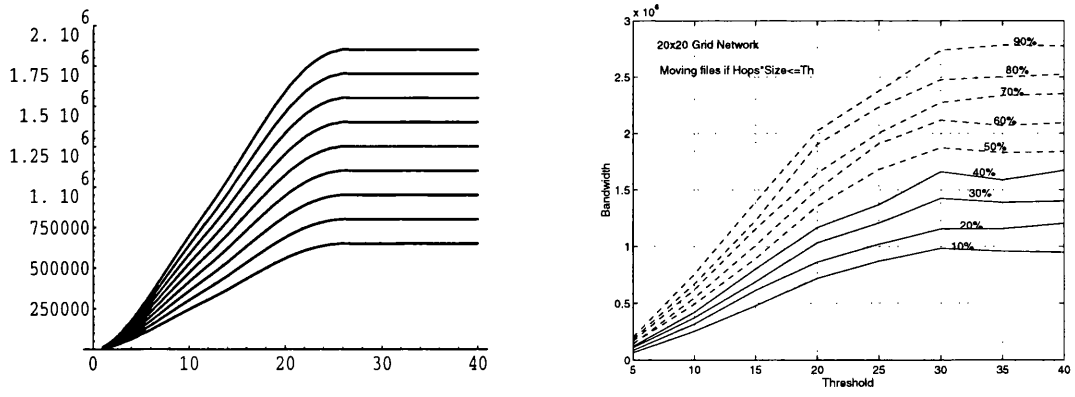


Figure 79. Comparison between theoretical and simulation results of bandwidth usage against σ, for the case of file relocation if the file size and distance product is less than the threshold

The linearity of bandwidth on threshold would have interesting consequences on the choice of control parameters, if it were to be applied to a more complex population of users. In the case considered, the range of activity for a user was the entire network. If, as is more realistic, users are expected to be confined to a particular region with high probability and only range beyond this rarely, the threshold should be chosen to suit that range.

It is worth noting here that we are examining only one parameter in isolation. To make a judgement as to which value is most applicable we need to consider the trade-offs and dependencies on other performance characteristics. In the context of our simple model their are a limited number, the most important being the QoS factor we are trying to obtain.

The performance can be demonstrated by considering the same system as above, but changing the value of S_{max}. In this case to look at values either side of that already chosen, i.e. $S_{max} = 10000$ and $S_{max} = 20000$.

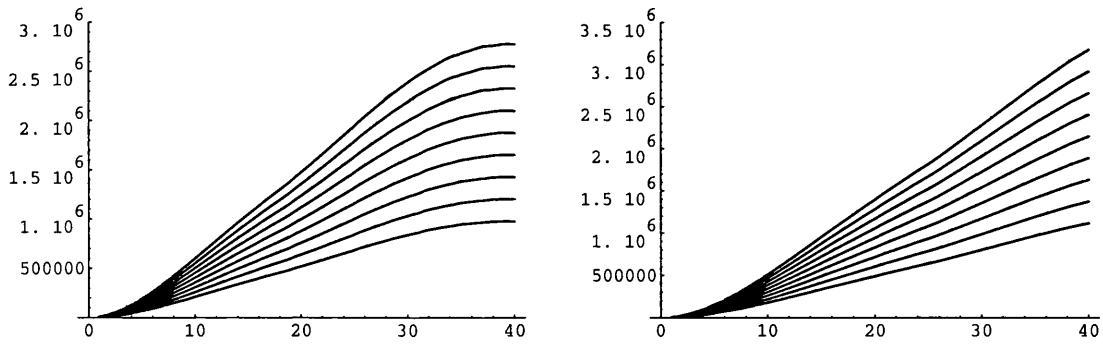


Figure 80. The data on the left shows the bandwidth usage if $S_{max} = 10000$ and that on the right shows the results for $S_{max} = 20000$. As previously, $T = 7500\sigma$

The results in Figure 80 show that if the threshold is chosen to be too small, i.e. the files are large relative to the threshold, saturation occurs and there are no gains to be made. On the other hand making it too large, it does not utilise its full range of flexibility. Careful choice of the control strategy as well as the parameters can be considered, which would be the long term aim of work in this domain, but at such preliminary stages only possible directions can be suggested.

Appendix D

Overview of SDL

D.1. Introduction

The specification methods employed by operators and other organisations such as the ITU have for many years relied on informal techniques such as English descriptions and message sequence diagrams. While these methods have been accepted in the past, the problems that can arise from such a free format are beginning to be understood. The difficulties of specifying OSI (Open Systems Interconnection) standards in natural language were recognised early, motivating CCITT and ISO to work on international standards for FDTs (Formal Description Techniques). This led to the development of the languages SDL, Stelle and LOTOS for producing formal specifications that are unambiguous, clear, complete, and consistent. FDTs are also intended to help in analysis, implementation and testing. This document focuses on SDL, starting with a general overview of the language and then presenting in more detail some of the language features and constructs, using an SDL model of a telecommunications service as an example.

D.2. The benefits of a formal specification language

It is probably widely accepted that the key for the success of a system is a thorough system specification and design. This requires, on the other hand, a suitable specification language, satisfying the following needs:

◊ a well defined set of concepts;

◊ unambiguous, clear, precise and concise specifications;

◊ a basis for verifying specifications with respect to completeness and correctness;

◊ a basis for determining whether or not an implementation conforms to the specifications;

◊ a basis for determining the consistency of specifications relative to each other;

◊ use of computer-based tools to create, maintain, verify, simulate and validate specifications;

◊ computer support for generating applications without the need of the traditional coding phase.

D.3. SDL history

SDL is the Specification and Description Language developed and standardised by CCITT (now ITU). The development of SDL began in 1972 after a period of investigations. The first version of the language was issued in 1976. The first CCITT Recommendation (Z.100 Recommendation) was in 1980, followed by updates in 1984, 1988, 1992 and 1996. During this period that language has experienced a considerable expansion. Object oriented features were included in the language in 1992 (SDL-92). In 1996 a few updates were made to the language in an addendum to the standard. Current SDL is now defined in the 1992 Z.100 standard with the 1996 addendum.

For systems engineering SDL is usually used in combination with other languages, such as OMT object model, MSC (Message Sequence Charts), ASN.1 and TTCN (Tree and Tabular Combined Notation). Some of these languages have also been studied within the same group within the ITU. The ITU Z.105 standard defines the use of SDL with ASN.1, and the Z.120 standard defines Message Sequence Charts. There has been work relating the SDL and TTCN semantic models, and the TTCN is used for testing standards and systems written using SDL. The use of OMT object model notation in combination with MSC and SDL is a powerful combination that covers most aspects of system engineering.

One of the advantages of SDL is the availability of commercial software tools for the specification and design of systems using SDL, and simulation of these systems, hence aiding in the validation process. These software tools have also experienced considerable advance within the past few years, incorporating the upgrades in the language and the integration with some of the other languages mentioned above.

Today, SDL is widely used in he telecommunications field, but it also being applied to a diverse number of other areas ranging over aircraft, train control, medical and packaging systems. In general, the suitability of the language is for real-time, stimulus-response systems.

D.4. SDL overview

D.4.1. Representation forms

SDL has two representation forms, a textual representation (SDL/PR) and a graphical representation (SDL/GR). The use of the graphical representation makes SDL a user friendly language, easy to learn and understand. In the graphical representation, graphical syntax is used to give overview, complemented by a textual syntax, since graphical symbols are missing for some concepts, e.g. abstract data type.

D.4.2. System structure

Since systems that are subject to specification are usually quite complex, any specification language should have some structuring concepts. There are three basis concepts in an SDL structure, namely system, block and process.

1. **System:** contains one or more blocks, interconnected with each other and with the boundary of the system by *channels*

2. **Blocks** : can be subsequently partitioned into sub-blocks; there are two types of blocks
   - ◊ partitioned blocks do not contain any processes; they only contain sub-blocks that communicate via *channels.*
   - ◊ leaf blocks are not partitioned, and contain only processes; within a leaf signals are conveyed on *signal routes* between processes, and between processes and the boundary of the block.
3. **Processes:** must always be contained in a block; they are the entities in the SDL description that describe *behaviour.*

A system description is thus structured into block (and sub-blocks) descriptions and process descriptions, resulting in a tree structure with the system as the root. Figure 81 shows an example.

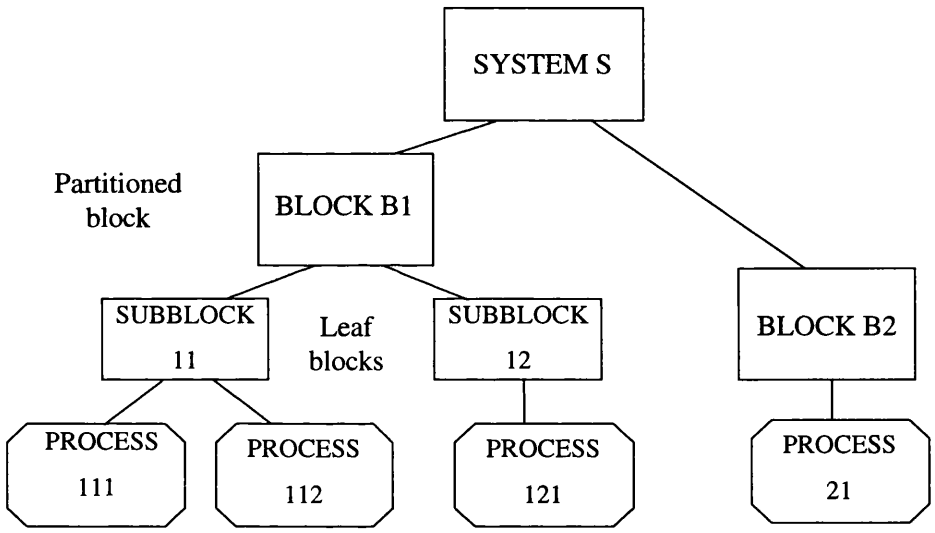


Figure 81. Structuring of a system into block, sub-blocks and processes

D.4.3. System behaviour

The behaviour of a system is constituted by the combined behaviour of a number of processes in the system. A process in SDL is an extended finite state machine, i.e. a finite state machine that can use and manipulate data stored in variables local to the machine. A process in SDL communicates autonomously and concurrently with other processes. The co-operation between processes is performed asynchronously by discrete messages, called signals. A process can also send signals to an receive signals from the environment to the system. It is assumed that the environment acts in as SDL-like fashion, and must obey the constraints given by the system descriptions.

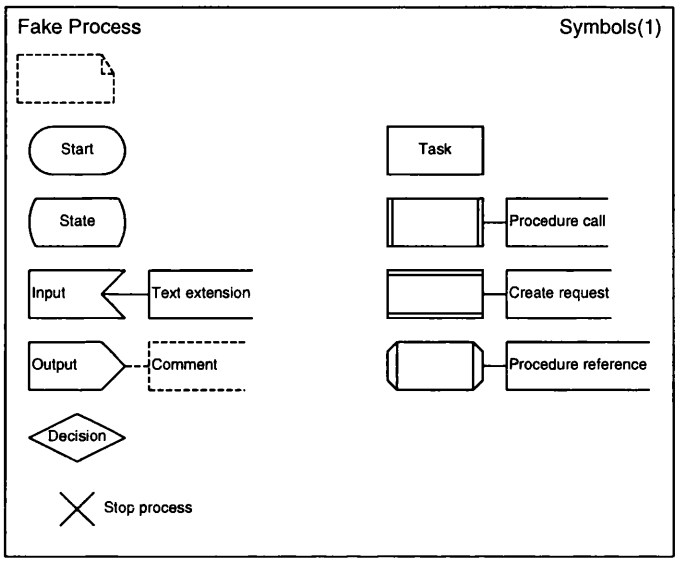


Figure 82. SDL symbols in the body of a process

Each process has an input FIFO (first in first out) queue, where the received signals are stored until they are consumed. The length of the queue is unlimited. A process is either in a state (waiting) or performs a transition between two states. A transition is initiated by the first signal in the input queue. When a signal has initiated a transition, it is removed from the input queue (and is said to be consumed). In a transition, variables can be manipulated, decisions can be made, new processes can be created, signals can be send (to other processes or the process itself), etc. The description of the finite state machine of the process constitutes the process graph or body of the process, composed of a set of symbols. Some of these symbols and their meaning are depicted in Figure 82. An example of a simple finite state machine defined by a direct graph is shown in Figure 83, with the equivalent SDL description in Figure 84.

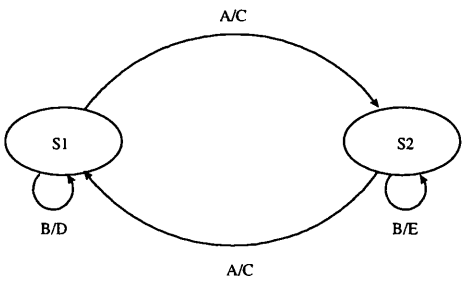


Figure 83. Behaviour of a finite-state machine (this is not SDL syntax)

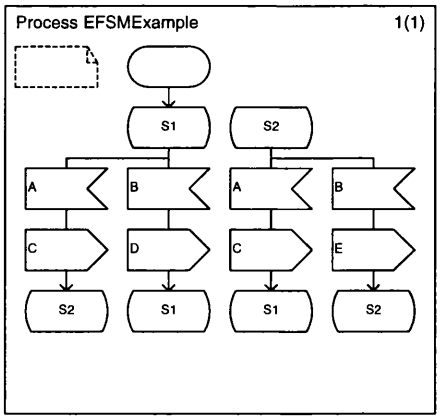


Figure 84. SDL description of the finite-state machine in Figure 83 by a process diagram

D.4.4. Data

In SDL, the *abstract data type* approach has been chosen. This means that all data types, predefined as well as user-defined, are defined in an implementation-independent way only in terms of their properties. Data types are called sorts in SDL. The specification of a *sort* has the following components:

◊ set of values,

◊ set of literals (value names),

◊ set of operators on these values,

◊ set of equations defining the operators.

Some predefined sorts are included in SDL, e.g. Integer, Boolean, Character, PId (process identifier), String, Charstring, Natural, Real, Array, Powerset, Time, and Duration. Added to these, the users can specify their user defined data types. The syntax for the specification of user defined data types is illustrated in the following example:

```
NEWTYPE UserList
array(UserIndex,Charstring)
ADDING
LITERALS
NewUserList;
OPERATORS
Search:Charstring,UserList->SearchResType;
ListFull:UserList->Boolean;
Add:Charstring,UserList->UserList;
Remove:Charstring, UserList->UserList;
ENDNEWTYPE;
```

The example above specifies a new data type called UserList, defined as an array of Charstrings, in which the index of the array is of type UserIndex. This data type has one literal, called NewUserList. Four operators are defined, namely Search, ListFull, Add and Remove. The types of the input parameters to the operators and the results must be specified. For example, the operator Search takes two parameters, one of type Charstring and one of type UserList, and returns a result of type SearchResType. Note that in the example above, the specification of the data type UserList uses other data types. Some are predefined sorts, such as Charstring and Boolean, and others, such as User Index and SearchResType, are on their own user defined data types which must also be specified somewhere else.

D.4.5. Types and instances

A definition in SDL is either a type definition or an instance definition. In SDL-92 the concepts of system type, block type, process type (amongst others) were introduced to enable reuse of the same definition in different parts o a system.
Types can be used to derive new (sub)types through specialisation, or to generate instances. For example, a block type is a template which can be *instantiated* as block instances or *specialised* a new block types, which are subtypes of the original one. Types and instances will be discussed in more detail in section D.6.2.

D.5. Service modelling using SDL

One of the areas of use of SDL is the modelling of telecommunication services. In particular, within BT SDL models are being used for the analysis of services interactions, mostly in the context of IN services. The advantages of using SDL to model services/features are varied. Some of the applications of such models are as follows:

a) To produce SDL specifications of the service, based on existing informal specifications. This exercise usually helps to identify issues related to the clarity and completeness of the specifications.

b) The commercial tools permit to simulate the behaviour of the service from the SDL description. This simulations can be used to validate the service against original requirements, allowing a rapid and cheap redevelopment of the service if necessary.

c) SDL models of different services can be used to simulate the combined behaviour of these services, aiding in the detection of services interactions.

d) The SDL models can also be used as a basis for specification, implementation and testing of the services.

D.6. SDL in more detail

This sections provides a deeper insight of SDL, using a specific SDL system that models the behaviour of an example service. The service as such is not relevant in this context. It is only used to illustrate some of the SDL concepts and features. Hence, the reader must avoid to get lost in the details of the service description, and focus on understanding the SDL concepts. Nevertheless, a brief description of the example service is given, to provide a general overview of the system that is modelled. Next, some diagrams from the SDL specification of the service are used to introduce the SDL concepts. This is done in a top down fashion, i.e. starting at the top level of the hierarchical structure (the system), and continuing with blocks and finally processes.

D.6.1. Description of the example service

The objective of the modelling work is a generic representation of management of location of resources within a network. The customers in the system have full mobility capability, and different mobility patterns may be considered. The location of resources available for each user is described in a user profile file, which is always moved to the user's local node. The resources in this case are data files whose location is not fixed. Several file relocation strategies can also be considered.

The service allows users to register at any location in the network, and change their location according to certain population patterns. When a user changes location, his/her customer profile is moved to the new location. Data files, containing services logic and data, are assumed to be distributed around the network and can be relocated. The current location of the different data files is stored in a central database.

When a user requests a particular service, his/her customer profile is checked, and the addresses of the necessary data files are requested to the database. When the location of the data files is known, two alternative actions are possible, namely the information can be extracted from the data files in their current location, or the whole data file can be moved to the user's current locations. The choice will depend on the selected relocation strategy.
At the service level, the model service described above can be viewed as composed of three main features:

a) an addressing function, that deals with the searching of the requested data files within the network;

b) database consultation, or file access function, to access the requested information in the data files, and

c) a file transfer function, that performs the relocation of the data files according to the selected relocation criteria.

D.6.2. System level

The system description constitutes the top level of detail, and it is what the SDL description specifies. The rest of the world is called the environment, and is separate from the system by the system boundary. The system communicates with its environment via channels. In SDL/GR the system description is called system diagram. Figure 85 shows an example.

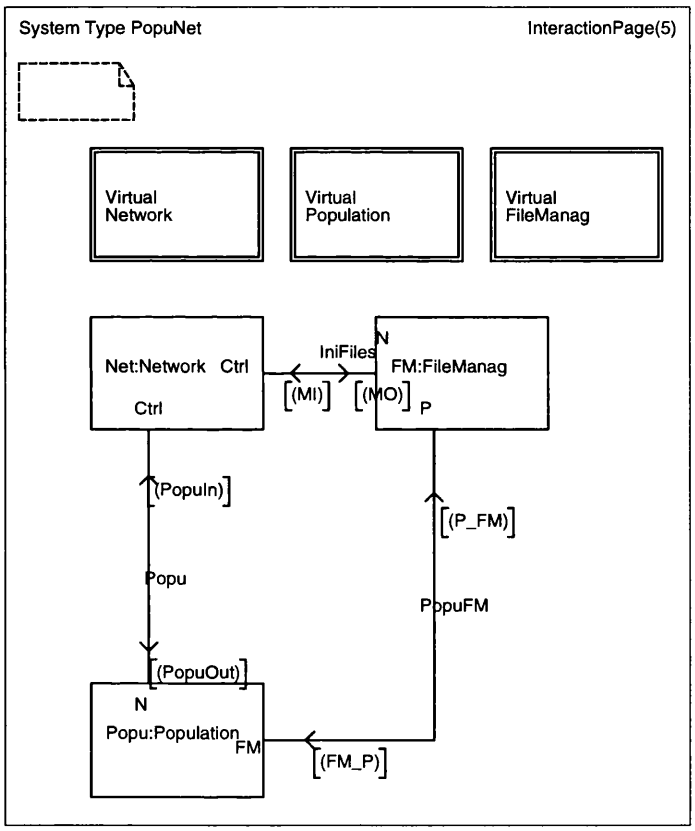


Figure 85. A system diagram

Figure 85 represents a system type where three block types are referenced, namely *Network, Population and FileManag*. Here, only a reference to these block types is included. The specification of these block types must be included elsewhere. The block types in Figure 85 have been declared as virtual. This is related to the concept of specialisation, introduced for the first time in SDL-92. Specialisation allows one type to be based on the properties of another type. Conceptually, the types in a specification are grouped in a type hierarchy (i.e. a tree) where the nearest parent node of a type in the hierarchy denotes the supertype from which the (sub)type has been specialised. All parent nodes of a subtype are supertypes of the type. Likewise, all child nodes of a type are subtypes of the type. Specialisation can be strict, i.e. addition of properties, or it can allow redefinition of virtual properties. In the latter case,

the keyword *Virtual* in the supertype indicates parts of the supertype that can be redefined by subtypes.

The system in Figure 85 is composed of the interaction between three block instances, namely *Net*, of type *Network*, *FM*, of type *FileManag* and *Popu*, of type *Population*. These block instances are connected via channels, that can convey specific types of signals. The block *Net* contains the description of the network and service described in section D.6.1, whereas the blocks *FM* and service *Popu* deal with the management of files and users in the system. The diagram in Figure 85 is not the complete definition of the system level. In SDL it is possible to split system descriptions into more than one page. This is indicated in the top right corner of the diagram and the syntax is *PageName(Total number of pages)*. The page shown in Figure 85 has been named *InteractionPage*, and it is one of a total of 5 pages that compose the system description in this example.

A system diagram usually contains the following elements:

◊ System name (Figure 85 represents in fact a system type, named *PopuNet*);

◊ Channels descriptions, for the channel connecting the blocks of the system or between the blocks and the environment (in Figure 85 *IniFiles, Popu* and *PopuFM*)

◊ Signals and signals lists descriptions, for the types of signals interchanged between the blocks o the system of between the blocks and the environment; signals can be grouped in signal lists, which are simply a set of signals; signal lists are included in brackets, to differentiate them from single signals (in Figure 85 only signal lists are used, e.g. *(MI), (MO), (PopuIn),* etc.); the actual description of the signals and signal lists is included in other pages (see section D.6.2.2)

◊ Data types descriptions, for the user defined data types, visible in the whole system and its environment.

◊ Blocks (block types) descriptions, for the blocks into which the system is partitioned (in Figure 85 *Net, FM* and *Popu*)

◊ Other elements, such as macros definitions (not covered in this document)

D.6.2.1. Channels

A channel description contains a channel name, a list of signal names for signals that can be transported by the channel and the identification of the end points of the channel (block or environment). For example, in Figure 85 the channel *Popu* may transport the signals included in the signal lists *(PopuIn)* from the block *Popu* to the block *Net*, and *(PopuOut)* from *Net* to *Popu*.

A channel may be bi-directional (all the channels in Figure 85) or uni-directional, in which case the channel symbol has only one arrowhead. Also the channels can be delaying (as is the case of all the channels in Figure 85) or non-delaying, in which case the arrowheads are placed at the en of the channel line. Delaying channels introduce an indeterminate non-constant delay in the signals that they convey.

D.6.2.2. Signals

A signal description contains a signal name and the type of values conveyed by the signal.

Figure 86 shows the page containing the declaration of signals and signal lists corresponding to the system level of the case example.

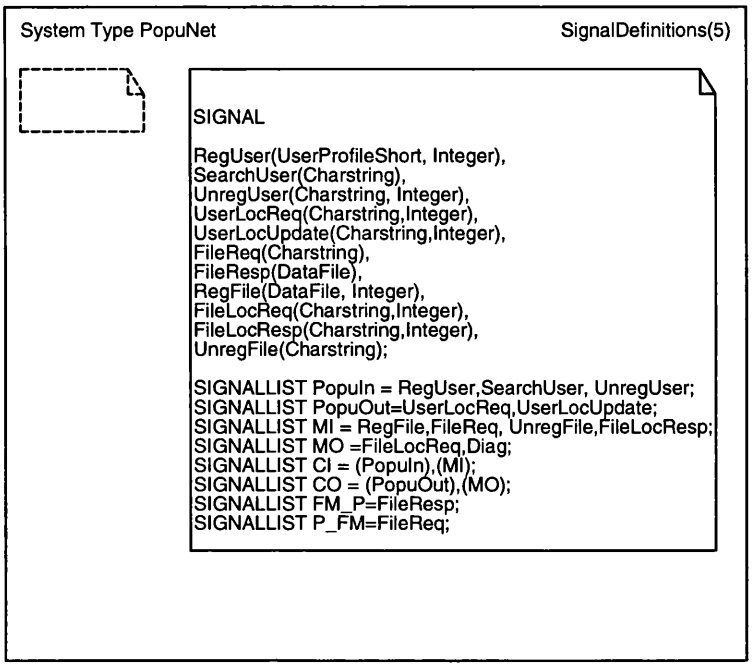


Figure 86. Declaration of signals and signal lists

D.6.3. Block level

A block is a part of the system that can be treated in various respects (development, description, understanding, etc.)as a self contained object. In SDL/GR a block is described by a block diagram.

Figure 87 shows an example of a block type, called *BaseNode*, representing a node in the network of the case example. A node is modelled in a structured fashion, and consists of four layers, namely Data Link, Network, Application and User layers. The functionality of each of these blocks is as follows:

◊ *DataLinkLayer* provides communication between nodes by sending packets, represented by the signal *P;*

◊ *NetworkLayer* deals with routing functions;

◊ *ApplicationLayer* deals with the management of users and files in the node;

◊ *UserLayer* is the interface of the node with the users of the system; it is the only block that can exchange signals with the environment of the system;

Communication between layers is done by means of service primitives, represented in the form of signals called 'SPxx' (these signals are defined in a different page, not shown here). Each layer is defined in an SDL block type, only referenced here, but specified elsewhere. A *BaseNode* type consists of one block instance of each of the four layer types connected together as shown in Figure 87. Note that this is an example of a partitioned block type, since the block is further subdivided into more sub-blocks.

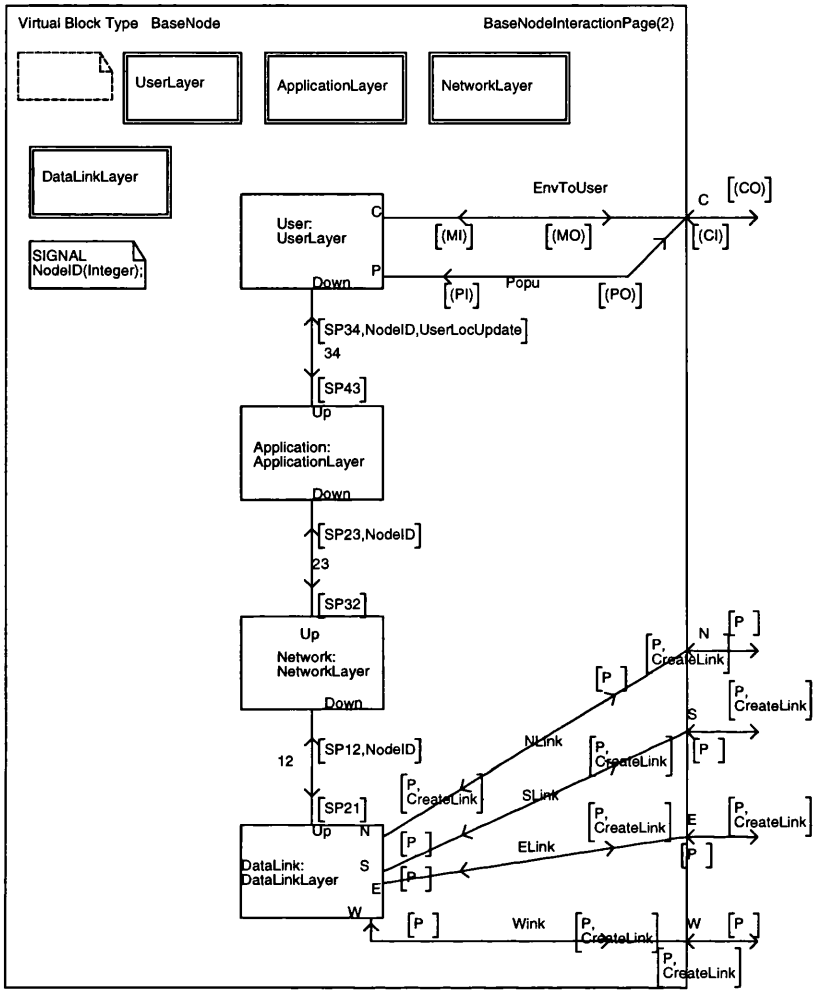


Figure 87. A partitioned block diagram

A partitioned block diagram usually contains the following elements:

◊ Block Name (Figure 87 represents in fact a block type named *BaseNode*);

◊ Signal (and signal lists) descriptions, for the signals local to the block, i.e. not visible outside the block (in Figure 87 the signal *NodeID* is defined);

◊ Channel definitions, for the connections between the different sub-blocks (*34, 23,*and *12*), and the block with the channels external to the block (*EnvToUser, Popu, NLink, SLink, ELink* and *WLink*);

◊ Gates definitions, i.e. connection points of the channels (for example in Figure 87 the block *User*, of type *UserLayer* has three gates, namely *C, P,* and *Down*);

◊ Sub-blocks definitions (in Figure 87 *DataLinkLayer, NetworkLayer, ApplicationLayer* and *UserLayer*);

◊ Other elements, such a s data types definitions, macro definitions, etc. local to the block.

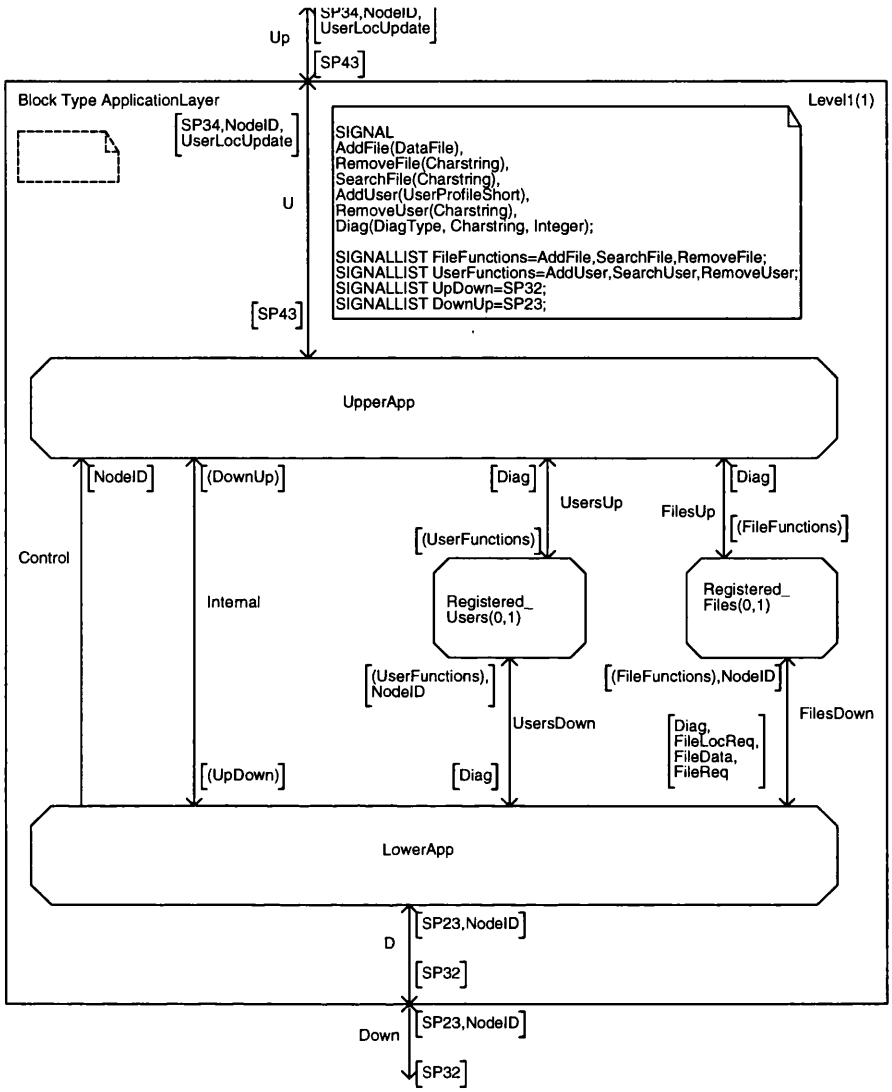


Figure 88. A leaf block

As an example of a leaf block, i.e. a block containing only processes, Figure 88 shows the definition of one of the sub-blocks that compose the specification of the *BaseNode*, namely the block type *ApplicationLayer*.

The block type *ApplicationLayer* consists of 4 processes, namely:

◊ *UpperApp*: is the interface with the *UserLayer* via gate *Up*. The function of *UpperApp* is to:

◊ interpret the service primitives received from *UserLayer* and send the appropriate signals to the processes *RegisteredFiles* and *RegisterdUsers*,

◊ construct service primitives and pass them to the *UserLayer* upon receipt of signals and parameters from the processes *RegisteredUsers*, *RegisteredFiles* and *LowerApp*;

◊ *LowerApp*: is the interface with the *NetworkLayer*, in a similar fashion to *UpperApp*;

◊ *RegisteredUsers*: provides the functionality for the registration of users in the node;

◊ *RegisteredFiles*: provides the functionality for the management of files within the node

The numbers in brackets in the procedures *RegisteredUsers* and *RegisteredFiles* mean that the initial number of processes instances of these types is 0, and the maximum number is 1 (see section D.6.4.2 Process creation).

Processes communicate between them via signal routes (the equivalent of channels at the block interconnection level). For example, in Figure 88 '*UsersUp*' is a bi-directional signal route between the processes *UpperApp* and *RegisteredUsers*, and can convey the signal *Diag* in one direction and the signals specified in the signal list *UserFunctions* in the other direction.

D.6.4. Process level

In SDL/GR, a process description is called a process diagram. An example containing the description of the process *UpperApp*, which as previously explained is part of the block type *ApplicationLayer* , is shown in Figure 89. A process diagram usually contains the following elements:

◊ Process name (in Figure 89 *UpperApp*)

◊ Formal parameters (not in this example)

◊ Variable descriptions (in the example in Figure 89 the variables used in the process, e.g. *TempSP4, SP4Name, i*, etc. have been declared in a different page)

◊ Procedure definitions (in the process *UpperApp* 7 procedures are referenced, namely *SendDiagToUser, AddUserReq, RemUserReq, AddFileReq, RemFileReq, MakeSP3* and *RemoveRemoteUser*)

◊ Other elements, such as timer descriptions, signal list descriptions, macro descriptions and data type descriptions, not included in this example

◊ Process graph or body of the process, for the description of the finite state machine

◊ The body of the process is a sequence of states and transitions, represented in SDL/GR by a set of symbols interconnected. The most commonly used symbols in a process body are shown in Figure 82. A process starts with a **start symbol**, followed by **optional initial actions** and the **initial state**. Figure 89 only shows one of the two pages that specify the process, and this is the reason why the **start symbol** is not present. The part of the process graph shown here starts with the state called *ReadInput*. The rest of the body is a set of transitions. Each transition consists of

  i) An initial state

  ii) The input that triggers the transition

  iii) One or more actions, e.g. assignment of a value to a variable, decisions according to the values of variables, variables export, send output signals, process creation, procedure calls.

  iv) A final state (or termination symbol)

In the example of Figure 89 only one state is included, and for all the transitions represented there the final state is the same as the initial state, *ReadInput*.

The behaviour represented in the part of the process *UpperApp* shown in Figure 89 is as follows. When the process is in the state *ReadInput* it can receive two signals, namely a service primitive *SP43* and *Diag*.

◊ When a service primitive is received (*SP43*), the parameter it carries is stored in the variable *TempSP4*. Its contents are then inspected (by means of the task symbol and user defined operators), and a decision is made depending on the type of message. For example, if the message is *RegUser*, the procedure *AddUserReq* is called, and after the procedure return the process goes back to state *ReadInput*. Similar actions are taken for the other cases

◊ When a 'Diag' signal is received, the three parameters that it carries are stored in the variables *DiagMess, DPar1* and *DPar2*. Then the variable *DPar2* is checked. If its . value is not greater than 0, the transition finishes and the process goes back to state *ReadInput*. If *DPar2* is greater than 0, a new decision is made based on the contents of the variable *DiagMess*. If this is anything other than *'UReg'*, the process goes back to state *ReadInput*. If, on the contrary, it is 'UReg', the procedure *RemoveRemoteUser* is called. After the procedure return the signal *UserLocUpdate* is sent out, and the process goes back to state *ReadInput*.

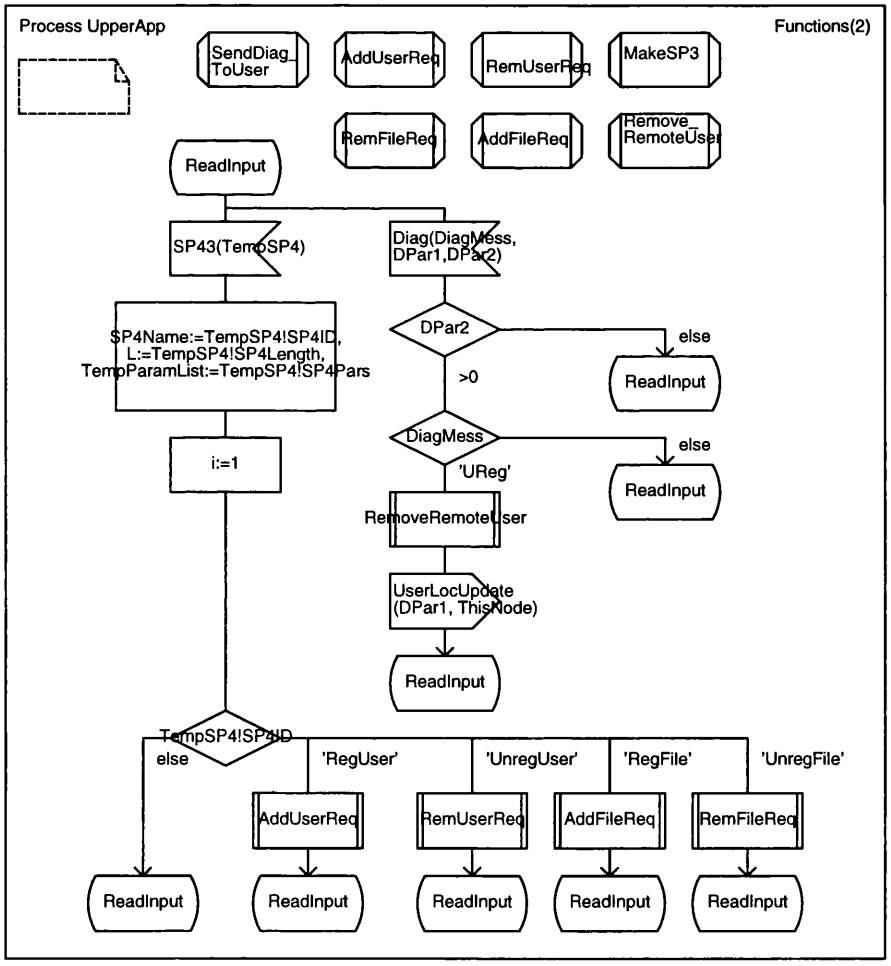


Figure 89. A process description

D.6.4.1. Procedures

A procedure is a finite state machine within a process. It is created when a procedure call is interpreted (see symbol in Figure 82), and it dies when it terminates, i.e. when a return construct is interpreted (see Figure 90). A procedure has the same complete valid input signal set as the enclosing process, but a separate set of states. When a procedure is running, the calling process or procedure is suspended in the transition containing the procedure call. This transitions is continued when the procedure terminates. Procedures are often used to provide more clarity, avoiding long sequences in the same page.

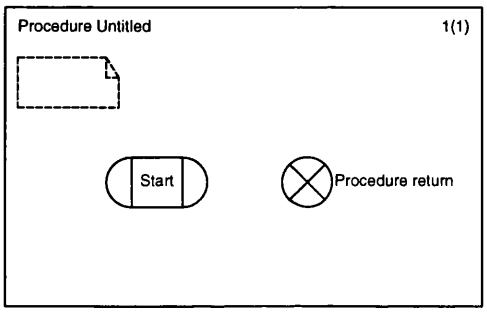


Figure 90. Proceudre start and procedure return symbols

D.6.4.2. Process creation

A process can be created during the system initiation, or dynamically, i.e. at interpretation time, as part of the execution of a transition of another process in the same block. For the symbol used in SDL/GR for creation of a process refer to Figure 82. In the creation of a process parameters that distinguish between different instances can be passed.

In SDL, each process has a unique identification which is a value of the type *PId*. The following predefined *PId* expressions exist:

◊ SELF - this process itself

◊ OFFSPRING - the most recent process instance created by this process

◊ PARENT - the creating process, if this process was dynamically created

◊ SENDER - the sender of the signal most recently consumed by this process

The creating and created processes can get information on their addresses by the predefined expressions described above.

When a process type is referenced in as SDL description, the initial number of instances of that type, i.e. the number of instances that are created during system initiation, and the total number of instances that may exist can be specified. For example, in Figure 88 for the

processes *RegisteredUsers* and *RegisteredFiles* there are initially 0 instances, and the maximum number of instances that can be created is 1. If an attempt is made to create more processes, then no process will be created and the expression OFFSPRING of the creating process gives the value Null.

D.6.5. Management of time

In SDL there is not a strict control of time. The exact time at which signals are consumed, transitions are executed, etc. can not be specified. Moreover, since delaying channels introduce an arbitrary delay in he signals, implicit non determinism occurs due to race conditions.

Even though a strict control of time is not possible in SDL, it is possible to introduce certain time constraints. This is done with the ***timer*** construct. A timer is an object, owned by a process, that is able to generate a timer signal and put it in the input queue of the process. During a transition a timer can be activated with the ***set*** construct. The set construct has two arguments. The first one is the absolute time for the expiration of the timer, and the other one is the name of the timer. For the specification of the expiration time, the expression *NOW*, of the predefined type *Time*, can be used. *NOW* always gives the current time during the interpretation of the system description. For example, if a timer T is set as follows

SET (NOW+13, T)

after time has advanced 13 time units, a timer signal with name T is put into the input queue of the process.
Timers are deactivated with a RESET construct. After resetting the timer, the process will behave in a way as if the timer has never been activated.

D.6.6. Non determinism

A real system may behave unreliably. The unreliable behaviour can have many causes, such as malfunction of congestion due to imperfect implementation. Since a formal specification is not concerned with implementation issues, there is a need to express unreliable behaviour of a real world system without mentioning the cause. Non-determinism is a means to achieve this. In SDL, there spontaneous transition and the any expression constructs were introduced to express non-determinism explicitly. Also, implicit non-determinism occurs due to the behaviour of the environment and to race conditions.

D.6.6.1. Implicit non-determinism

◊ An SDL system interacts with its environment by means of signals. No assumptions are made about the behaviour of the environment, except that it behaves in a SDL-like manner. This undefined behaviour of the environment is an implicit non-determinism in SDL systems.

◊ The other source of implicit nondeterminism are race conditions due to delaying channels. The indeterminate, non-constant channel delays are an implicit nondeterminism which may result in alternative behaviours.

D.6.6.2. Explicit non-determinism

1. *Any* expression:

A predefined ***any*** expression contains the keyword ***any*** and the identifier of a sort, and denotes an unspecified value of the given sort. For example, the expression

value:=ANY (integer)

denotes an unspecified integer.

2. Decision with ANY

In a non-deterministic decision the question contains only the keyword ***any*** and the answers are omitted (Figure 91). When a non-deterministic decision is evaluated, an unspecified transition branch is chosen.

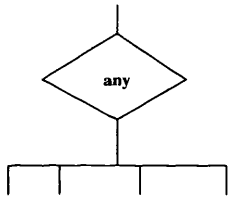


Figure 91. Nondeterministic decision

3. Spontaneous transitions:

A spontaneous transition allows a state transition to be triggered non-deterministically irrespective of whether there are any signals in the input buffer. This is expressed by writing ***none*** in an input symbol

D.7. Other issues

SDL was originally conceived to describe protocols behaviour and functionality in a time-independent fashion (as previously mentioned in section D.6.5 in SDL there is no strict time control). Since a formal specification has to be as implementation-independent as possible, specification languages do not cover performance aspects. However, system developers often require performance measures, e.g. throughput and response time, to decide on implementation design alternatives. To obtain performance measurements in addition to the functional behaviour the quantitative properties have to be specified. The use of a formal description language as a paradigm for performance modelling requires the extensions of the language with temporal and probabilistic specifications. The temporal specifications are necessary to describe the time lapse between consecutive events, and the probabilistic specifications are necessary to describe the selection among different possible behaviours. The integration of performance evaluation with SDL descriptions is not straightforward issue, and considerable amounts of research is being done in this area.

The application area of SDL are real time, reactive, discrete event driven systems. SDL uses a particular message passing paradigm, based on asynchronous communication. It is not well suited for data intensive systems or synchronous systems such as digital hardware. SDL has not been intended to be an implementation language, although existing tools permit a more or less automatic translation of SDL specification to programming languages. In this area SDL has to compete with other methods, such as visual object orientated methods.

Appendix E

Complete simulation results for the SDL model

E.1. Introduction

This appendix contains the complete set of results obtained from the SDL model of the Magnet service. The discussion of the results is included in chapter 9.

The appendix consists of three main section, each containing the results for the single user, two users and three users case respectively. The plots are all labelled and the captions underneath are self-explanatory. A list of the principal parameters in the model is reproduced here for convenience:

- ◊ Size of the network: 9 nodes (3 rows by 3 columns grid)
- ◊ Total number of data files in the system: 10
- ◊ Number of data files associated to each user: 5
- ◊ Probability of a data file being requested for a user: 0.5
- ◊ Speed of movement of users: users change location every 5000 time units
- ◊ Packet transmission delay: 100 time units
- ◊ Database access delay: 100 time units
- ◊ Time out to obtain a file address: FLOC_TIMEOUT = 400 time units
- ◊ Time out to obtain a file after a *FileRequest* has been sent: F_TIMEOUT= 400 time units
- ◊ Initial simulation time: the first movement of users occurs at t_i=1000 time units
- ◊ Final simulation time: t_f= 1000000
- ◊ Frequency at which measures are taken: every 5000 time units (Δ=5000)
- ◊ Offset of the measures with respect to the movement of users: measures are taken 500 time units after the movement of users

For each value of the number of users and size of input buffers a total of 10 simulations have been done, and the results have been averaged for the 10 different simulations.

Note: The simulation time in the axis *x* of the plots is scaled by the measurements interval, $5x10^3$. Hence, the total simulation time is 10^6.

E.2. One user

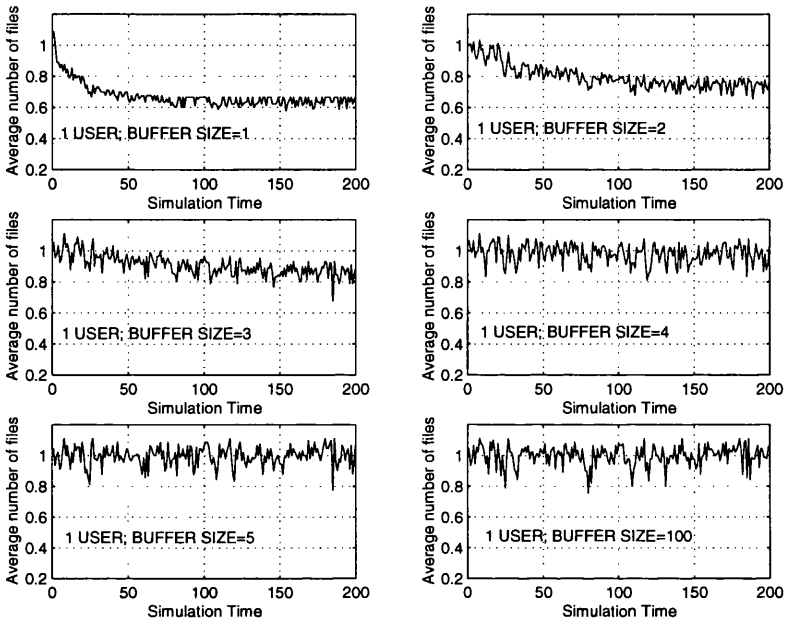


Figure 92. Average number of data files per node for different sizes of the input buffer, and 1 user in the system

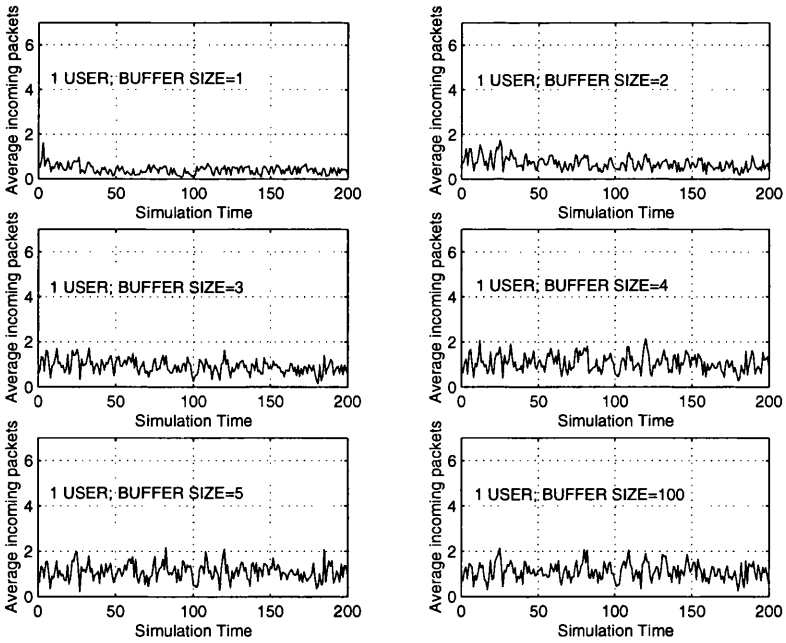


Figure 93. Average number of incoming packets in each node every *D* = 5000 time units for the one user case

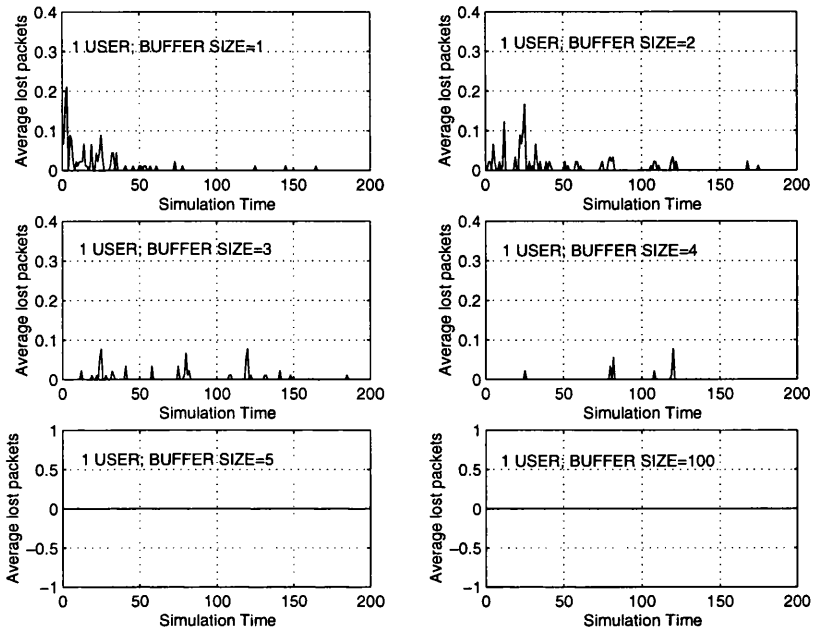


Figure 94. Average number of packets discarded in each node every $D = 5000$ time units for the 1 user case

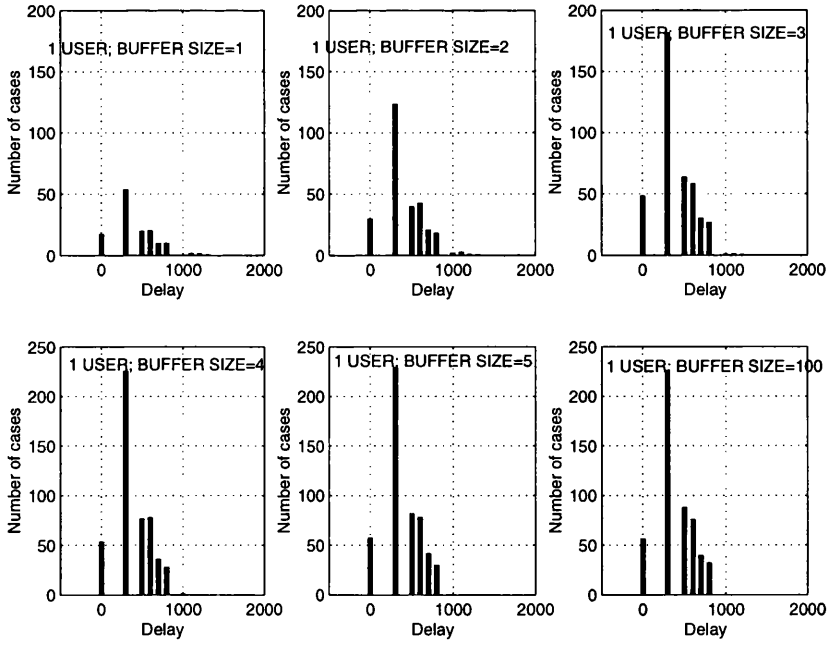


Figure 95. Histogram of the delays in obtaining a file for the 1 user case

E.3. Two users

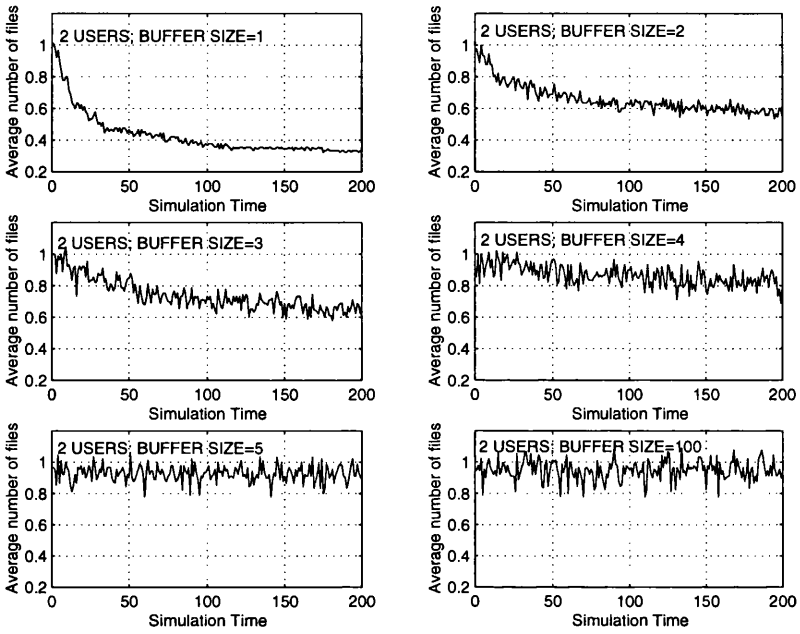


Figure 96. Average number of data files per node for different sizes of the input buffer, and 2 users in the system

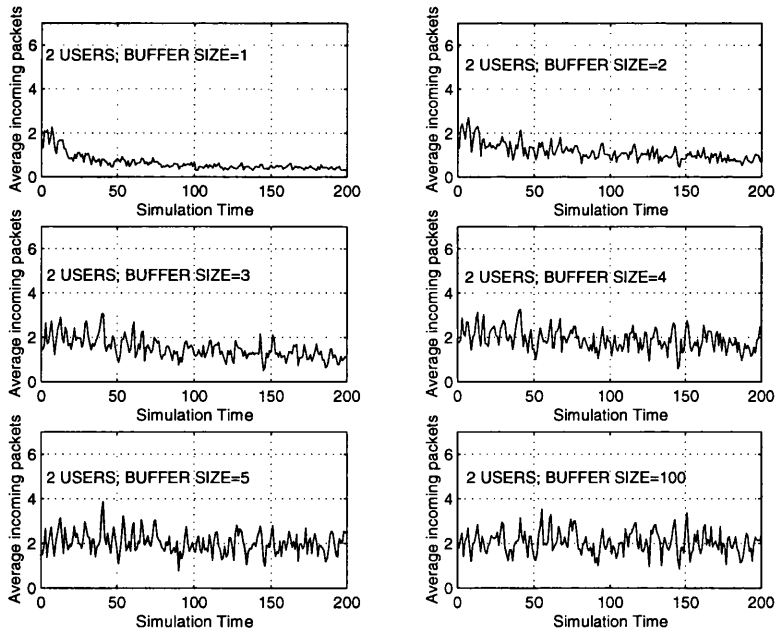


Figure 97. Average number of incoming packets in each node every *D* = 5000 time units for the two users case

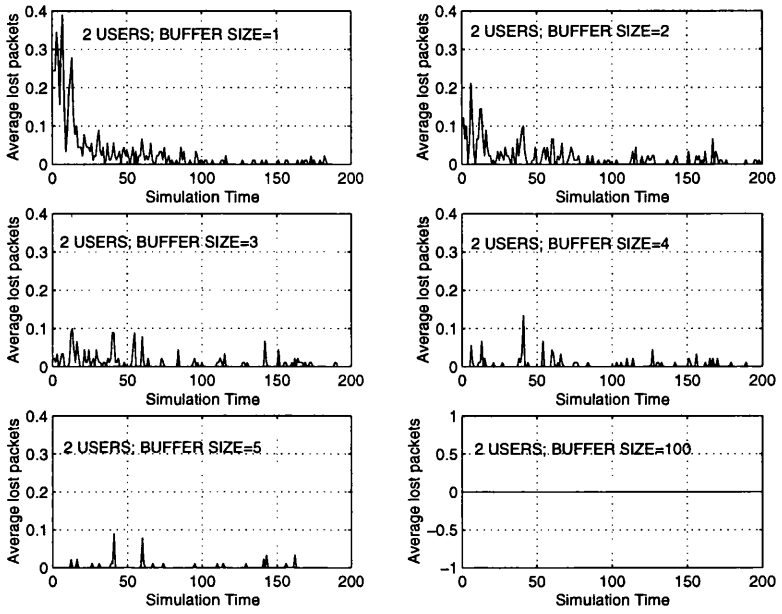


Figure 98. Average number of packets discarded in each node every $D = 5000$ time units for the 2 users case

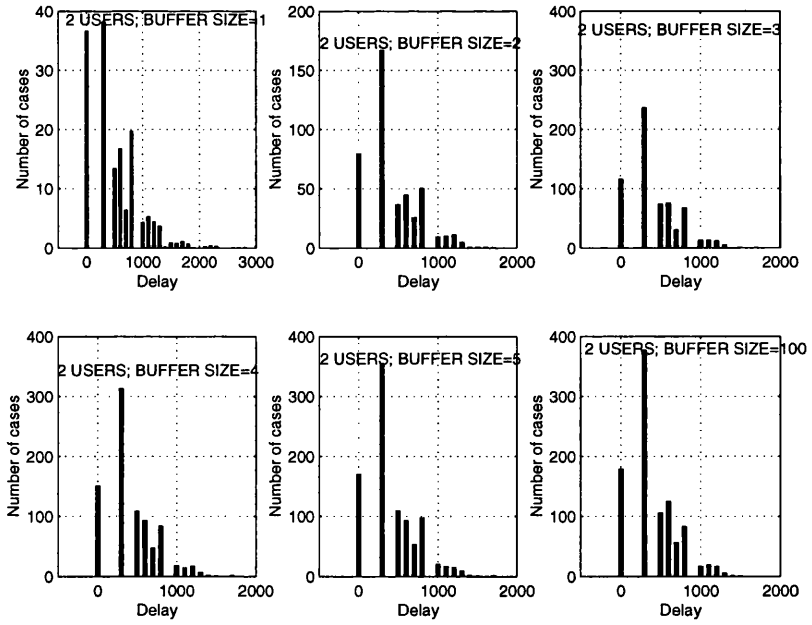


Figure 99. Histogram of the delays in obtaining a file for the 2 users case

E.4. Three users

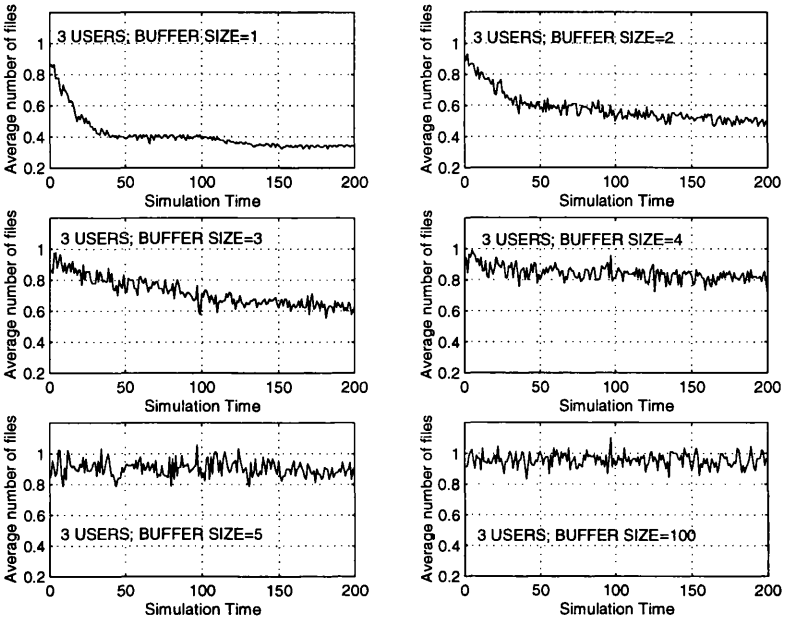


Figure 100. Average number of data files per node for different sizes of the input buffer, and 3 users in the system

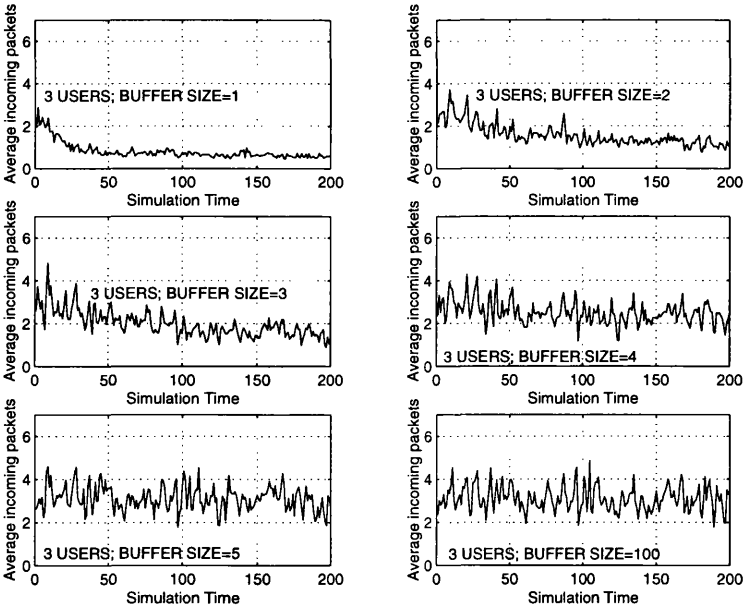


Figure 101. Average number of incoming packets in each node every D = 5000 time units for the three users case

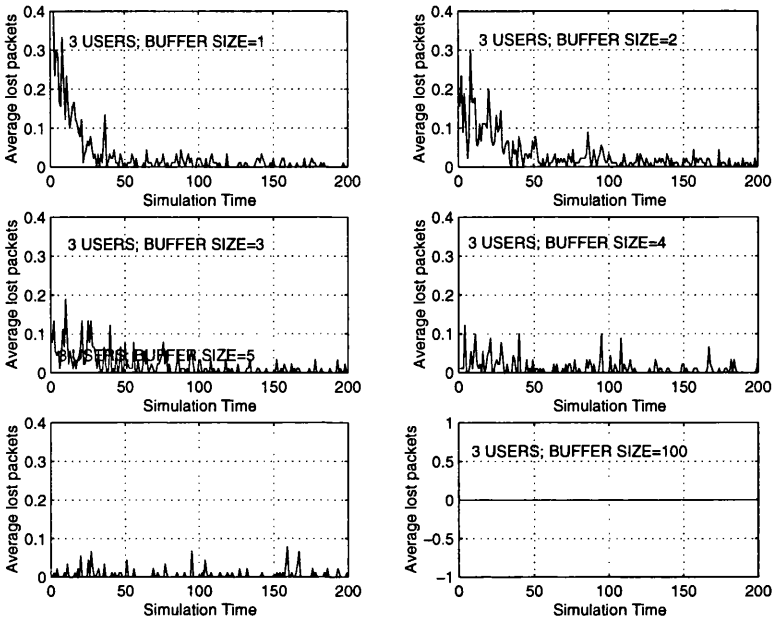


Figure 102. Average number of packets discarded in each node every D = 5000 time units for the 3 users case

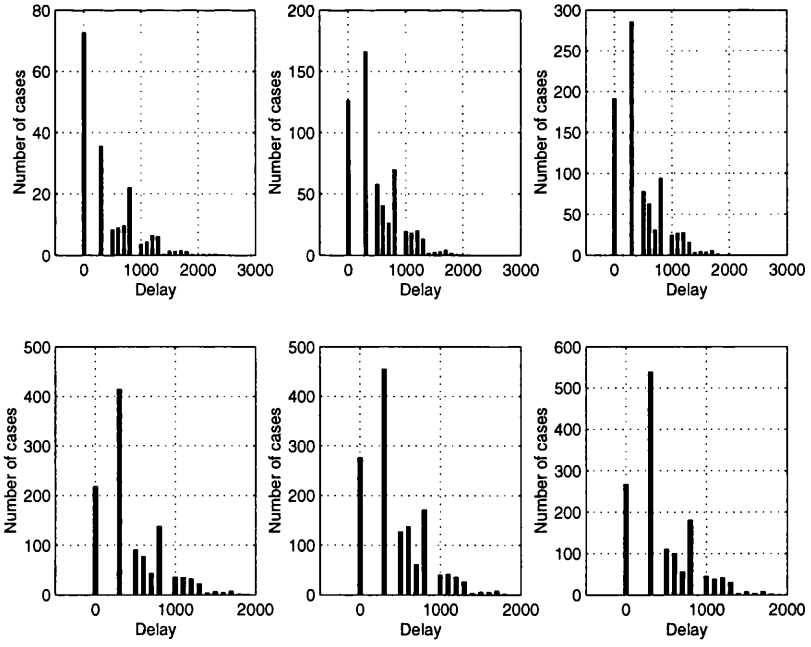
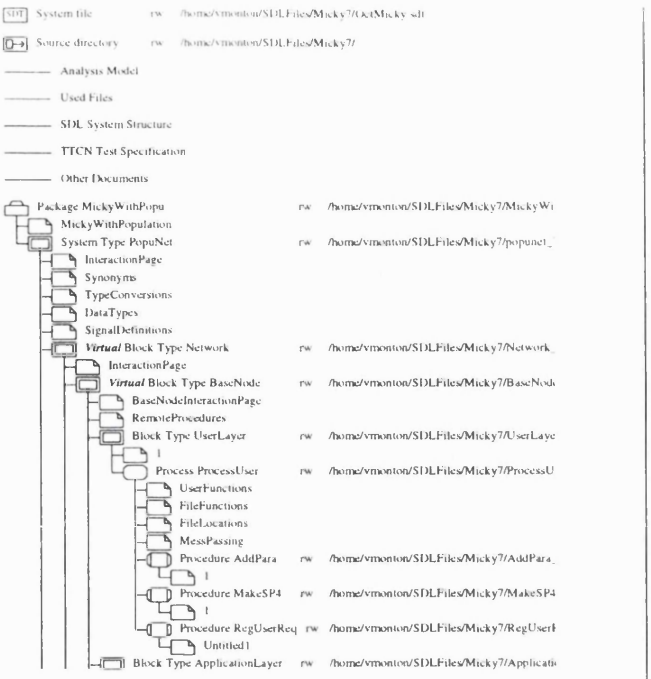
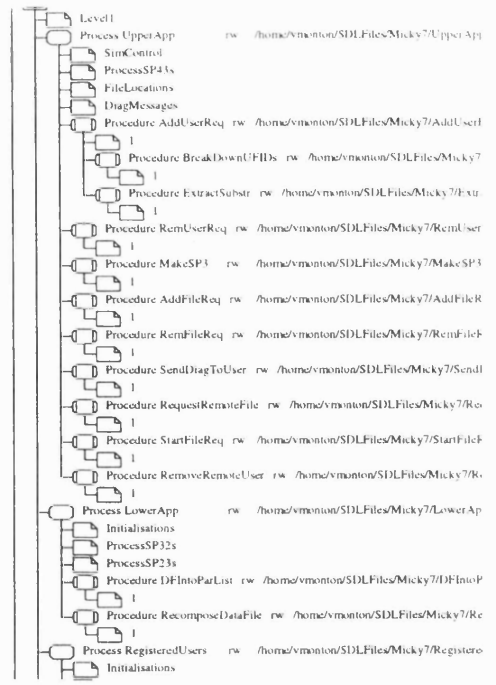


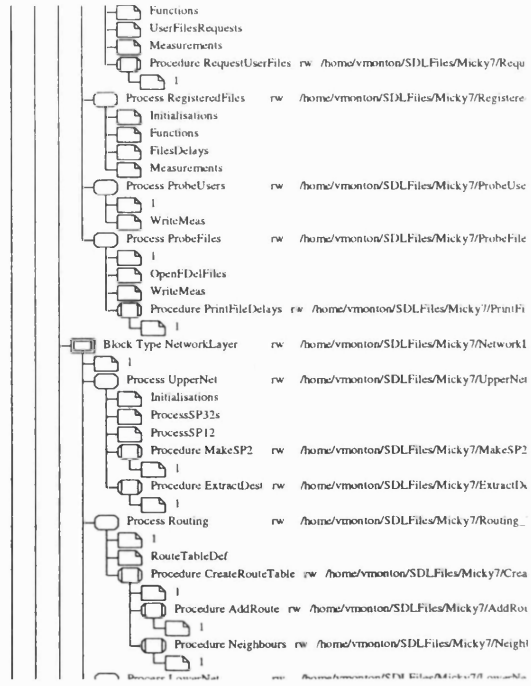
Figure 103. Histogram of the delays in obtaining a file for the 3 users case

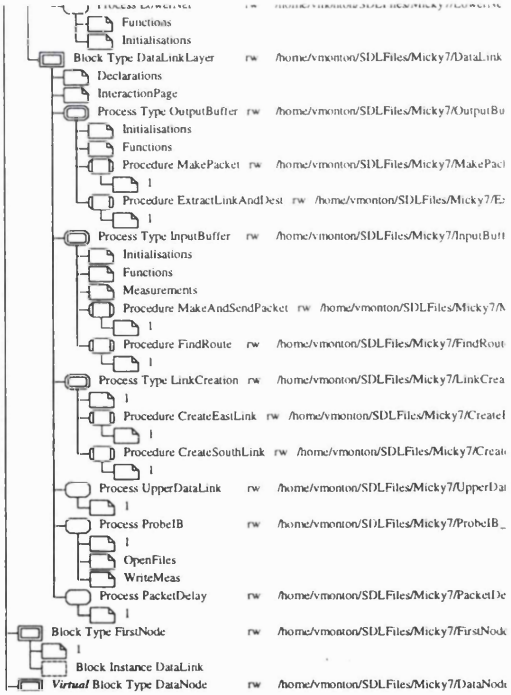
Appendix F

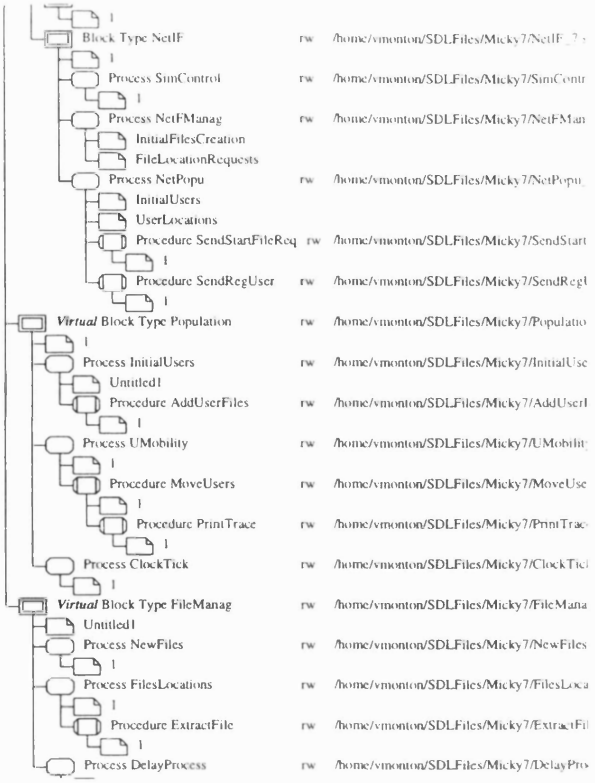
The complete SDL model

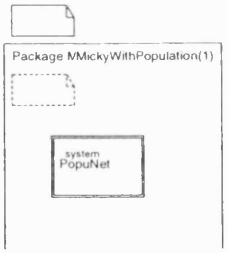

System file rw /home/vmonton/SDLFiles/Micky7/OldMicky.sdt
Source directory rw /home/vmonton/SDLFiles/Micky7/
Analysis Model
Used Files
SDL System Structure
TTCN Test Specification
Other Documents
Package MickyWithPopu rw /home/vmonton/SDLFiles/Micky7/MickyWi
MickyWithPopulation
System Type PopuNet rw /home/vmonton/SDLFiles/Micky7/popunet_
InteractionPage
Synonyms
TypeConversions
DataTypes
SignalDefinitions
Virtual Block Type Network rw /home/vmonton/SDLFiles/Micky7/Network_
InteractionPage
Virtual Block Type BaseNode rw /home/vmonton/SDLFiles/Micky7/BaseNod
BaseNodeInteractionPage
RemoteProcedures
Block Type UserLayer rw /home/vmonton/SDLFiles/Micky7/UserLaye
1
Process ProcessUser rw /home/vmonton/SDLFiles/Micky7/ProcessU
UserFunctions
FileFunctions
FileLocations
MessPassing
Procedure AddPara rw /home/vmonton/SDLFiles/Micky7/AddPara_
1
Procedure MakeSP4 rw /home/vmonton/SDLFiles/Micky7/MakeSP4
1
Procedure RegUserReq rw /home/vmonton/SDLFiles/Micky7/RegUserI
Untitled1
Block Type ApplicationLayer rw /home/vmonton/SDLFiles/Micky7/Applicati

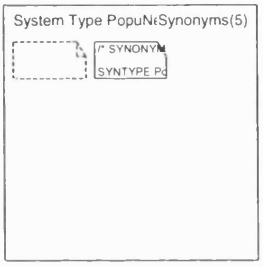

Level1
Process UpperApp rw /home/vmonton/SDLFiles/Micky7/UpperApp
SimControl
ProcessSP43s
FileLocations
DiagMessages
Procedure AddUserReq rw /home/vmonton/SDLFiles/Micky7/AddUserI
1
Procedure BreakDownUFIDs rw /home/vmonton/SDLFiles/Micky7
1
Procedure ExtractSubstr rw /home/vmonton/SDLFiles/Micky7/Extr
1
Procedure RemUserReq rw /home/vmonton/SDLFiles/Micky7/RemUser
1
Procedure MakeSP3 rw /home/vmonton/SDLFiles/Micky7/MakeSP3
1
Procedure AddFileReq rw /home/vmonton/SDLFiles/Micky7/AddFileR
1
Procedure RemFileReq rw /home/vmonton/SDLFiles/Micky7/RemFileF
1
Procedure SendDiagToUser rw /home/vmonton/SDLFiles/Micky7/SendI
1
Procedure RequestRemoteFile rw /home/vmonton/SDLFiles/Micky7/Re
1
Procedure StartFileReq rw /home/vmonton/SDLFiles/Micky7/StartFileF
1
Procedure RemoveRemoteUser rw /home/vmonton/SDLFiles/Micky7/R
1
Process LowerApp rw /home/vmonton/SDLFiles/Micky7/LowerAp
Initialisations
ProcessSP32s
ProcessSP23s
Procedure DFIntoParList rw /home/vmonton/SDLFiles/Micky7/DFIntoP
1
Procedure RecomposeDataFile rw /home/vmonton/SDLFiles/Micky7/Re
1
Process RegisteredUsers rw /home/vmonton/SDLFiles/Micky7/Registere
Initialisations

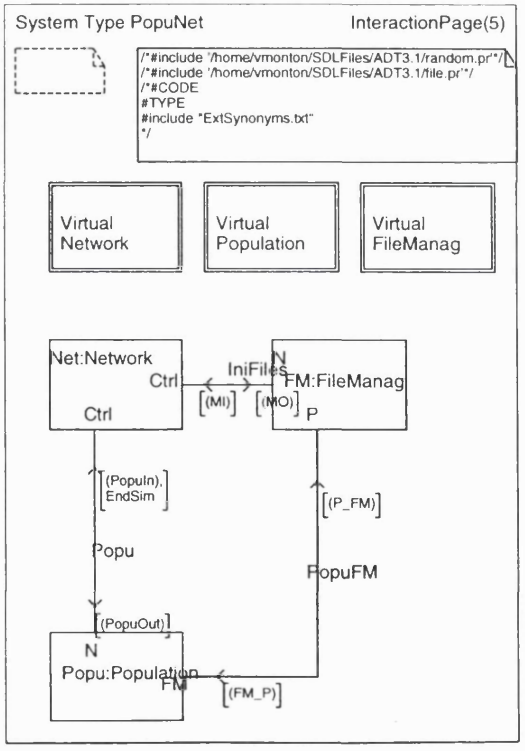

Functions
UserFilesRequests
Measurements
Procedure RequestUserFiles rw /home/vmonton/SDLFiles/Micky7/Requ
1
Process RegisteredFiles rw /home/vmonton/SDLFiles/Micky7/Registere
Initialisations
Functions
FilesIDelays
Measurements
Process ProbeUsers rw /home/vmonton/SDLFiles/Micky7/ProbeUse
1
WriteMeas
Process ProbeFiles rw /home/vmonton/SDLFiles/Micky7/ProbeFile
1
OpenFDelFiles
WriteMeas
Procedure PrintFileDelays rw /home/vmonton/SDLFiles/Micky7/PrintFi
1
Block Type NetworkLayer rw /home/vmonton/SDLFiles/Micky7/Network1
1
Process UpperNet rw /home/vmonton/SDLFiles/Micky7/UpperNet
Initialisations
ProcessSP32s
ProcessSP12
Procedure MakeSP2 rw /home/vmonton/SDLFiles/Micky7/MakeSP2
1
Procedure ExtractDest rw /home/vmonton/SDLFiles/Micky7/ExtractDe
1
Process Routing rw /home/vmonton/SDLFiles/Micky7/Routing_
1
RouteTableDef
Procedure CreateRouteTable rw /home/vmonton/SDLFiles/Micky7/Crea
1
Procedure AddRoute rw /home/vmonton/SDLFiles/Micky7/AddRou
1
Procedure Neighbours rw /home/vmonton/SDLFiles/Micky7/Neighl
1


Functions
Initialisations
Block Type DataLinkLayer rw /home/vmonton/SDLFiles/Micky7/DataLink
Declarations
InteractionPage
Process Type OutputBuffer rw /home/vmonton/SDLFiles/Micky7/OutputBu
Initialisations
Functions
Procedure MakePacket rw /home/vmonton/SDLFiles/Micky7/MakePacl
1
Procedure ExtractLinkAndDest rw /home/vmonton/SDLFiles/Micky7/E
1
Process Type InputBuffer rw /home/vmonton/SDLFiles/Micky7/InputBuff
Initialisations
Functions
Measurements
Procedure MakeAndSendPacket rw /home/vmonton/SDLFiles/Micky7/M
1
Procedure FindRoute rw /home/vmonton/SDLFiles/Micky7/FindRout
1
Process Type LinkCreation rw /home/vmonton/SDLFiles/Micky7/LinkCrea
1
Procedure CreateEastLink rw /home/vmonton/SDLFiles/Micky7/CreateI
1
Procedure CreateSouthLink rw /home/vmonton/SDLFiles/Micky7/Create
1
Process UpperDataLink rw /home/vmonton/SDLFiles/Micky7/UpperDat
1
Process ProbeIB rw /home/vmonton/SDLFiles/Micky7/ProbeIB_
1
OpenFiles
WriteMeas
Process PacketDelay rw /home/vmonton/SDLFiles/Micky7/PacketDe
1
Block Type FirstNode rw /home/vmonton/SDLFiles/Micky7/FirstNode
1
Block Instance DataLink
Virtual Block Type DataNode rw /home/vmonton/SDLFiles/Micky7/DataNode



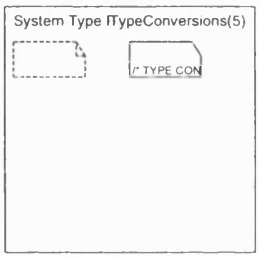






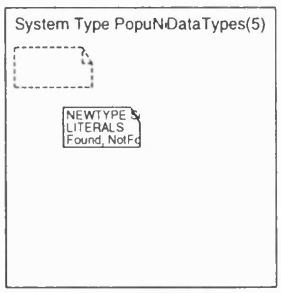

```
/* SYNONYM TYPES */

SYNTYPE PositiveInteger=Integer
 CONSTANTS 0:1000000
ENDSYNTYPE;

SYNTYPE NodeIndex=Integer
 CONSTANTS 1:MAXNODE
ENDSYNTYPE;

SYNTYPE UserIndex=Integer
 CONSTANTS 1:MAXREGUSERS
ENDSYNTYPE;

SYNTYPE FileIndex=Integer
 CONSTANTS 1:MAXREGFILES
ENDSYNTYPE;

SYNTYPE UFileIndex=Integer
 CONSTANTS 1:FILES_PER_USER
ENDSYNTYPE;

SYNTYPE ExpectedFLocIndex=Integer
 CONSTANTS 1:MAX_EXP_FLOC
ENDSYNTYPE;

SYNTYPE ExpectedFileIndex=Integer
 CONSTANTS 1:MAX_EXP_FILES
ENDSYNTYPE;

SYNTYPE FDelIndex=Integer
 CONSTANTS 0:MAX_F_DEL_ARRAY
ENDSYNTYPE;

SYNTYPE FlagVal=Integer
 CONSTANTS 1,-1
ENDSYNTYPE;

SYNONYM MAX_F_DEL_ARRAY Integer=100; /*MaxSize of the array containing file delays*/

/* SIMULATION PARAMETERS */

/* External synonyms are included as macro */
/* definitions, in the file "ExtSynonyms.txt" */

SYNONYM IBUF_SIZE Integer = external; /*Maximum number of delay processes in the input buffers*/
SYNONYM MAX_SEED Integer = 32767; /*Maximum seed for the SDT random number generator*/
SYNONYM INIUSERS Integer = external; /*Initial number of users in the UserLocationsDatabase*/
SYNONYM TOTALUSERS Integer=5; /*Max number of users in the system*/
SYNONYM MAXREGUSERS Integer=5; /*Max users per node */
SYNONYM MAXREGFILES Integer=10; /*Max files in each node */
SYNONYM FILES_PER_USER Integer=5; /*Number of data files in a user profile */
SYNONYM TOTALFILES Integer=100; /*Max number of different files that can be generated*/
SYNONYM MAXSIZE Integer=external; /*Maximum size of files*/
SYNONYM MINSIZE Integer=external; /*Minimum size of files */

SYNONYM INIFILES Integer = 10; /*Initial number of files in the network*/
SYNONYM PRIONEFILES=INIFILES/2;/*Initial number of Priority 1 files*/
SYNONYM PRITWOFILES=INIFILES-PRIONEFILES;/*Initial number of prio. 2 files*/

SYNONYM FM_START Duration=100;
SYNONYM POPU_START Duration=200;
SYNONYM TICK_START Duration=1000;
SYNONYM TOTAL_SIM_TIME Time=1000000;
SYNONYM USER_CADENCE Duration=5000;/*Frequency with which users are moved*/
```

```
SYNONYM MEAS_FREQ Duration = 5000;/*Frequency with which measurements are taken*/
SYNONYM PACK_MEAS_FREQ Duration=5000;
SYNONYM MEAS_OFFSET Time=500;/*Time offset (with respect to TICK_START to take measurements*/

SYNONYM TOTALROW Integer=3;
SYNONYM TOTALCOL Integer=3;
SYNONYM MAXNODE Integer=TOTALROW*TOTALCOL;

SYNONYM MAX_EXP_FLOC Integer=10;/*Max size of the ExpectedFLocList*/
SYNONYM MAX_EXP_FILES Integer=10;/*Max size of the ExpectedFileList*/
SYNONYM FLOC_TIMEOUT Duration=400;
SYNONYM FILE_TIMEOUT Duration=400;
SYNONYM F_LOC_DEL Duration=100;
SYNONYM PACKET_DELAY Duration=100;

SYNONYM MAXITEM Integer=5;
SYNONYM MAXPARA Integer=5; /*Max parameters in a service primative*/
```




```
/* TYPE CONVERSIONS */

/******************* IntegerString conversions *************/

NEWTYPE IntegerStringOperators
 OPERATORS
/*Transforms positive integers <=9999999 into an array of chars*/
  IntToString : Integer->Charstring;
/*Transforms strings into integers*/
  StringToInt : Charstring->Integer;

/*#ADT(B)
#BODY

SDL_Charstring #(IntToString)(I)
SDL_Integer I;

{
  SDL_Charstring result=NULL;
  char tmp[8];
  tmp[0]='V';
  sprintf(&(tmp[1]),"%-7d",I);
  xAss_SDL_Charstring(&result,tmp,XASS);
  result[0]='T';

  return result;
}


SDL_Integer #(StringToInt)(C)
SDL_Charstring C;
{
  SDL_Integer I;
  I=atoi(++C);
  return I;
}

*/
ENDNEWTYPE;

/******************* TimeString conversions *************/

NEWTYPE TimeStringOperators
 OPERATORS
/*Transforms SDL <=999999999 into an array of chars*/
  TimeToString : Time->Charstring;
/*Transforms strings into integers*/
  StringToTime : Charstring->Time;

/*#ADT(B)
#BODY

SDL_Charstring #(TimeToString)(T)
SDL_Time T;

{
  SDL_Charstring result=NULL;
  char tmp[10];
  tmp[0]='V';
  sprintf(&(tmp[1]),"%-9d",T.s);
  xAss_SDL_Charstring(&result,tmp,XASS);
```

```
  result[0]='T';

  return result;
}


SDL_Time #(StringToTime)(C)
SDL_Charstring C;
{
  SDL_Time T;
  T.s=atoi(++C);
  return T;
}

*/
ENDNEWTYPE;



/***************User Profiles Conversion *****************/

NEWTYPE UserProfilesConversions
 OPERATORS
/*Transforms UserProfile int UserProfileShort*/
  UPFToUPFShort:UserProfile,UserProfileShort->UserProfileShort;

/*#ADT(B)
#BODY


#(UserProfileShort) #(UPFToUPFShort)(Param1, Param2)
#(UserProfile) Param1;
#(UserProfileShort) Param2;
{
#(UFileIndex) cnt;

  Param2.UID=Param1.UID;
  Param2.UPrevLoc=Param1.UPrevLoc;
  Param2.UCurrLoc=Param1.UCurrLoc;
  Param2.TotalUFiles=Param1.TotalUFiles;
  for(cnt=0;cnt<#(FILES_PER_USER);cnt++){
    Param2.UFiles.A[cnt]=Param1.UFiles.A[cnt].FileID;
  }
  return Param2;
}

*/
ENDNEWTYPE UserProfilesConversions;
```



```
NEWTYPE SearchResType
LITERALS
Found, NotFound;
ENDNEWTYPE;

/********** BEGIN Data File *************/
/****************************************/

NEWTYPE DataFile
   STRUCT
      FileID Charstring;
      Prio Integer;
      FSize Integer;
ADDING
LITERALS
NewDataFile;
OPERATORS
Clear:DataFile->DataFile;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(DataFile) #(NewDataFile) (void)
#else
extern #(DataFile) #(NewDataFile) ()
#endif
{
return yMake_#(DataFile)("V\0",-1,-1);
}

#ifndef XNOPROTO
extern #(DataFile) (Clear) (#(DataFile) Param1)
#else
extern #(DataFile) #(Clear) (Param1)
#(DataFile) Param1;
#endif
{
  Param1.FileID="V\0";
  Param1.Prio=-1;
  Param1.FSize=-1;
  return Param1;
}
*/
ENDNEWTYPE;

/********** END DATAFILE *****************/

/************** Files IDS LIST **********/
/****************************************/

NEWTYPE FileIDsList
array(UFileIndex,Charstring)
ADDING
LITERALS
NewFileIDsList;
OPERATORS
Search:Charstring,FileIDsList->SearchResType;
ListFull:FileIDsList->Boolean;
Add:Charstring,FileIDsList->FileIDsList;
Remove:Charstring, FileIDsList->FileIDsList;
GetFID:Integer, FileIDsList->Charstring;
```

```
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(FileIDsList) #(NewFileIDsList) (void)
#else
extern #(FileIDsList) #(NewFileIDsList) ()
#endif
{
return yMake_#(FileIDsList)("V\0");
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(FileIDsList) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(FileIDsList) Param2;
#endif
{
  #(UFileIndex) cnt;
  for(cnt=0;cnt<#(FILES_PER_USER);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt])) return #(Found);
  }
  return #(NotFound);
}

#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(FileIDsList) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(FileIDsList) Param1;
#endif
{
  #(UFileIndex) cnt;
  for(cnt=0;cnt<#(FILES_PER_USER);cnt++)
    if(xEq_SDL_Charstring(Param1.A[cnt],"V\0")){
      return SDL_False;
    }
  return SDL_True;
}

#ifndef XNOPROTO
extern #(FileIDsList) #(Add) (SDL_Charstring Param1,#(FileIDsList) Param2)
#else
extern #(FileIDsList) #(Add)(Param1,Param2)
SDL_Charstring Param1;
#(FileIDsList) Param2;
#endif
{
  #(UFileIndex) cnt;
  for(cnt=0;cnt<#(FILES_PER_USER);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt],"V\0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
extern #(FileIDsList) #(Remove) (SDL_Charstring Param1,#(FileIDsList) Param2)
#else
extern #(FileIDsList) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
```

```
#(FileIDsList) Param2;
#endif
{
  #(UFileIndex) cnt;
  for(cnt=0;cnt<#(FILES_PER_USER);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt])) {
      Param2.A[cnt]= "V\0";
      return Param2;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(Charstring) #(GetFID) (SDL_Integer Param1,#(FileIDsList) Param2)
#else
extern #(Charstring) #(GetFID)(Param1,Param2)
SDL_Integer Param1;
#(FileIDsList) Param2;
#endif
{
 return Param2.A[Param1];
}

*/
ENDNEWTYPE;

/************ END FILES IDS LIST *******/

/********** BEGIN FileList ************/
/****************************************/

NEWTYPE FileList
array(FileIndex,DataFile)
ADDING
LITERALS
NewFileList;
OPERATORS
Search:Charstring,FileList->SearchResType;
ListFull:FileList->Boolean;
Count:FileList->Integer;
Add:DataFile,FileList->FileList;
Remove:Charstring, FileList->FileList;
GetFile:Integer,FileList->DataFile;
GetFileByName:Charstring,FileList->DataFile;
Clear:FileList->FileList;


/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(FileList) #(NewFileList) (void)
#else
extern #(FileList) #(NewFileList) ()
#endif
{
return yMake_#(FileList)(yMake_#(DataFile)("V\0",-1,-1));
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(FileList) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
```

```
#(FileList) Param2;
#endif
{
  #(FileIndex) cnt;
  for(cnt=0;cnt<#(MAXREGFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FileID)) return #(Found);
  }
  return #(NotFound);
}

#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(FileList) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(FileList) Param1;
#endif
{

 #(FileIndex) cnt;
  for(cnt=0;cnt<#(MAXREGFILES);cnt++)
    if(xEq_SDL_Charstring(Param1.A[cnt].FileID,"V\0")){
      return SDL_False;
    }
  return SDL_True;
}


#ifndef XNOPROTO
extern SDL_Integer #(Count) (#(FileList) Param1)
#else
extern SDL_Integer #(Count) (Param1)
#(FileList) Param1;
#endif
{

 #(FileIndex) cnt;
 #(FileIndex) n=0;
  for(cnt=0;cnt<#(MAXREGFILES);cnt++)
    if(yNEqF_SDL_Charstring(Param1.A[cnt].FileID,"V\0")){
      n=n+1;
    }
  return n;
}


#ifndef XNOPROTO
extern #(FileList) #(Add) (#(DataFile) Param1,#(FileList) Param2)
#else
extern #(FileList) #(Add)(Param1,Param2)
#(DataFile) Param1;
#(FileList) Param2;
#endif
{
  #(FileIndex) cnt;
  for(cnt=0;cnt<#(MAXREGFILES);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt].FileID,"V\0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
```

```
extern #(FileList) #(Remove) (SDL_Charstring Param1,#(FileList) Param2);
#else
extern #(FileList) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(FileList) Param2;
#endif
{
  #(FileIndex) cnt;
  for(cnt=0;cnt<#(MAXREGFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FileID)) {
      Param2.A[cnt].FileID= "V\0";
      return Param2;
    }
  }
  return Param2;
}


#ifndef XNOPROTO
extern #(DataFile) #(GetFile) (SDL_Integer Param1,#(FileList) Param2)
#else
extern #(DataFile) #(GetFile)(Param1,Param2)
SDL_Integer Param1;
#(FileList) Param2;
#endif
{
 return Param2.A[Param1];
}


#ifndef XNOPROTO
extern #(DataFile) #(GetFileByName) (SDL_Charstring Param1,#(FileList) Param2)
#else
extern #(DataFile) #(GetFileByName)(Param1,Param2)
SDL_Charstring Param1;
#(FileList) Param2;
#endif
{
  #(FileIndex) cnt;
  for(cnt=0;cnt<#(MAXREGFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FileID)) {
      return Param2.A[cnt];
    }
  }
}


#ifndef XNOPROTO
extern #(FileList) #(Clear) (#(FileList) Param1)
#else
extern #(FileList) #(Clear)(Param1)
#(FileList) Param1;
#endif
{
  #(FileIndex) cnt;
  for(cnt=0;cnt<#(MAXREGFILES);cnt++){
    Param1.A[cnt].FileID= "V\0";
  }

  return Param1;
}

*/
ENDNEWTYPE;
```

```
/************ END FileList **************/

/******BEGIN FILE LOCATIONS DATABASE *******/

SYNTYPE FLocDatabIndex=Integer
 CONSTANTS 0:TOTALFILES
ENDSYNTYPE;


NEWTYPE FLocEle
STRUCT
  FID Charstring;
  FPrio Integer;
  Node Integer;
  Flag Integer;
ADDING
LITERALS
NewFLocEle;
OPERATORS
Clear:FLocEle->FLocEle;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(FLocEle) #(NEWFLocEle) (void)
#else
extern #(FLocEle) #(NewFLocEle) ()
#endif
{
 return yMake_#(FLocEle)("V\0", -1, -1, -1);
}

#ifndef XNOPROTO
extern #(FLocEle) #(Clear) (#(FLocEle) Param1)
#else
extern #(FLocEle) #(Clear) (Param1)
#(FLocEle) Param1;
#endif
{
  Param1.FID="V\0";
  Param1.Node=-1;
  Param1.FPrio=-1;
  Param1.Flag=-1;
  return Param1;
}
*/
ENDNEWTYPE;


NEWTYPE FLocDatabase
array(FLocDatabIndex,FLocEle)
ADDING
LITERALS
NewFLocDatabase;
OPERATORS
Search(Charstring,FLocDatabase->SearchResType;
IsListFull:FLocDatabase->Boolean;
Add:FLocEle,FLocDatabase->FLocDatabase;
Remove:Charstring, FLocDatabase->FLocDatabase;
GetFPrio(Charstring,FLocDatabase->Integer;
GetFLocation:Charstring,FLocDatabase->Integer;
PutFLocation:Charstring,FLocDatabase,Integer->FLocDatabase;
PutFlag:Charstring,FLocDatabase,FlagVal->FLocDatabase;
GetFlag:Charstring,FLocDatabase->FlagVal;
Clear:FLocDatabase->FLocDatabase;
```

```
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(FLocDatabase) #(NewFLocDatabase) (void)
#else
extern #(FLocDatabase) #(NewFLocDatabase) ()
#endif
{
return yMake_#(FLocDatabase)(yMake_#(FLocEle)("V\0", -1, -1, -1));
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(FLocDatabase) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(FLocDatabase) Param2;
#endif
{
  #(FLocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)) return #(Found);
  }
  return #(NotFound);
}


#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(FLocDatabase) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(FLocDatabase) Param1;
#endif
{
  #(FLocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALFILES);cnt++){
  if(xEq_SDL_Charstring(Param1.A[cnt].FID,"V\0"))
    return SDL_False;
  }
  return SDL_True;
}


#ifndef XNOPROTO
extern #(FLocDatabase) #(Add) (#(FLocEle) Param1,#(FLocDatabase) Param2)
#else
extern #(FLocDatabase) #(Add)(Param1,Param2)
#(FLocEle) Param1;
#(FLocDatabase) Param2;
#endif
{
  #(FLocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALFILES);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt].FID,"V\0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
extern #(FLocDatabase) #(Remove) (SDL_Charstring Param1,#(FLocDatabase) Param2)
#else
extern #(FLocDatabase) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(FLocDatabase) Param2;
#endif
{
  #(FLocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)) {
      Param2.A[cnt].FID = "V\0";
      return Param2;
    }
  }
  return Param2;
}


#ifndef XNOPROTO
extern #(Integer) #(GetFPrio) (SDL_Charstring Param1,#(FLocDatabase) Param2)
#else
extern #(Integer) #(GetFPrio)(Param1,Param2)
SDL_Charstring Param1;
#(FLocDatabase) Param2;
#endif
{
  #(FLocDatabIndex) cnt;
  for (cnt=0;cnt<#(TOTALFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)){
      return Param2.A[cnt].FPrio;
    }
  }
  return -1;
}


#ifndef XNOPROTO
extern #(Integer) #(GetFLocation) (SDL_Charstring Param1,#(FLocDatabase) Param2)
#else
extern #(Integer) #(GetFLocation)(Param1,Param2)
SDL_Charstring Param1;
#(FLocDatabase) Param2;
#endif
{
  #(FLocDatabIndex) cnt;
  for (cnt=0;cnt<#(TOTALFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)){
      return Param2.A[cnt].Node;
    }
  }
  return -1;
}

#ifndef XNOPROTO
extern #(FLocDatabase) #(PutFLocation) (SDL_Charstring Param1,#(FLocDatabase) Param2, SDL_Integer Param3)
#else
extern #(FLocDatabase) #(PutFLocation)(Param1,Param2,Param3)
SDL_Charstring Param1;
#(FLocDatabase) Param2;
SDL_Integer Param3;
#endif
{
  #(FLocDatabIndex) cnt;
  for (cnt=0;cnt<#(TOTALFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)){
      Param2.A[cnt].Node=Param3;
      return Param2;
    }
```

```
  }
  return Param2;
}


#ifndef XNOPROTO
extern #(FLocDatabase) #(PutFlag) (SDL_Charstring Param1,#(FLocDatabase) Param2, #(FlagVal) Param3)
#else
extern #(FLocDatabase) #(PutFlag) (Param1,Param2,Param3)
SDL_Charstring Param1;
#(FLocDatabase) Param2;
#(FlagVal) Param3;
#endif
{
  #(FLocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)){
      Param2.A[cnt].Flag=Param3;
      return Param2;
    }
  }
  return Param2;
}


#ifndef XNOPROTO
extern #(FlagVal) #(GetFlag)(SDL_Charstring Param1,#(FLocDatabase) Param2)
#else
extern #(FlagVal) #(GetFlag)(Param1,Param2)
SDL_Charstring Param1;
#(FLocDatabase) Param2;
#endif
{
  #(FLocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALFILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)){
      return Param2.A[cnt].Flag;
    }
  }
  return -1;
}


#ifndef XNOPROTO
extern #(FLocDatabase) (Clear) (#(FLocDatabase) Param1)
#else
extern #(FLocDatabase) #(Clear) (Param1)
#(FLocDatabase) Param1;
#endif
{
  #(FLocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALFILES);cnt++){
    Param1.A[cnt].FID="V0";
    Param1.A[cnt].FPrio=-1;
    Param1.A[cnt].Node=-1;
    Param1.A[cnt].Flag=-1;
  }
  return Param1;
}
*/

ENDNEWTYPE;

/***************END FILE LOCATIONS DATABASE *******/
```

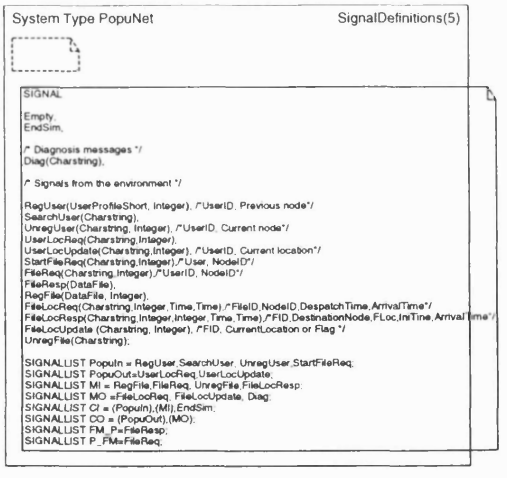
```
/********** BEGIN User Profile SHORT ************/
/************************************************/

NEWTYPE UserProfileShort
  STRUCT
    UID Charstring;
    UPrevLoc Integer;
    UCurrLoc Integer;
    TotalUFiles Integer;
    UFiles FileIDsList;
ADDING
LITERALS
NewUserProfileShort;
OPERATORS
Clear:UserProfileShort->UserProfileShort;
GetUFiles:UserProfileShort->FileIDsList;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(UserProfileShort) #(NewUserProfileShort) (void)
#else
extern #(UserProfileShort) #(NewUserProfileShort) ()
#endif
{
return yMake_#(UserProfileShort)("V0",-1,-1,-1,yMake_#(FileIDsList)("V0"));
}

#ifndef XNOPROTO
extern #(UserProfileShort) #(Clear) (#(UserProfileShort) Param1)
#else
extern #(UserProfileShort) #(Clear) (Param1)
#(UserProfileShort) Param1;
#endif
{
  Param1.UID="V0";
}


#ifndef XNOPROTO
extern #(FileIDsList) #(GetUFiles) (#(UserProfileShort) Param1)
#else
extern #(FileIDsList) #(GetUFiles) (Param1)
#(UserProfileShort) Param1;
#endif
{
  return Param1.UFiles;
}

*/
ENDNEWTYPE;

/********** END USERPROFILE SHORT*****************/

/************ BEGIN User Profile *************/
/*********************************************/

NEWTYPE UserProfile
  STRUCT
    UID Charstring;
```

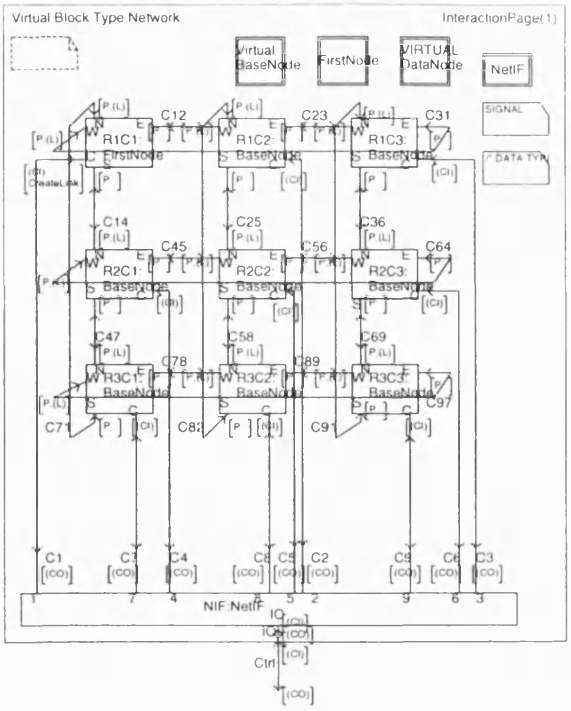
```
    UPrevLoc Integer;
    UCurrLoc Integer;
    TotalUFiles Integer;
    UFiles FileList;
ADDING
LITERALS
NewUserProfile;
OPERATORS
Clear:UserProfile->UserProfile;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(UserProfile) #(NewUserProfile) (void)
#else
extern #(UserProfile) #(NewUserProfile) ()
#endif
{
return yMake_#(UserProfile)("V0",-1,-1,-1,yMake_#(FileList)(yMake_#(DataFile)("V0",-1,-1)));
}

#ifndef XNOPROTO
extern #(UserProfile) (Clear) (#(UserProfile) Param1)
#else
extern #(UserProfile) #(Clear) (Param1)
#(UserProfile) Param1;
#endif
{
  Param1.UID="V0";
}
*/
ENDNEWTYPE;

/********** END USERPROFILE *****************/

SYNTYPE ULocDatabIndex=Integer
 CONSTANTS 0 TOTALUSERS
ENDSYNTYPE;

/******** BEGIN ULocEle *****************/
/****************************************/


NEWTYPE ULocEle
STRUCT
  UserEle UserProfile;
  UPrevLoc Integer;
  UCurrLoc Integer;
  Flag Integer;/* Flag=1 when user location is undetermined */
ADDING
LITERALS
NewULocEle;
OPERATORS
Clear:ULocEle->ULocEle;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(ULocEle) #(NewULocEle) (void)
#else
extern #(ULocEle) #(NewULocEle) ()
#endif
{
return yMake_#(ULocEle)(yMake_#(UserProfile)("V0",-1,-1,-1,yMake_#(FileList)(yMake_#(DataFile)("V0",-1,-1)),-1,-1,-1);
```

```
}

#ifndef XNOPROTO
extern #(ULocEle) (Clear) (#(ULocEle) Param1)
#else
extern #(ULocEle) #(Clear) (Param1)
#(ULocEle) Param1;
#endif
{
  #(FileIndex) cnt;
  Param1.UserEle.UID="V0";
  for(cnt=0;cnt<#(MAXREGFILES);cnt++){
    Param1.UserEle.UFdes.A[cnt].FileID="V0";
  }
  Param1.UPrevLoc=-1;
  Param1.UCurrLoc=-1;
  Param1.Flag=-1;
  return Param1;
}
*/
ENDNEWTYPE;
/*************END ULocEle ******************/

/******BEGIN USER LOCATIONS DATABASE *******/
/*****************************************/

NEWTYPE ULocDatabase
array(ULocDatabIndex,ULocEle)
ADDING
LITERALS
NewULocDatabase;
OPERATORS
Search:Charstring,ULocDatabase->SearchResType;
ListFull:ULocDatabase->Boolean;
Add:ULocEle,ULocDatabase->ULocDatabase;
Remove:Charstring, ULocDatabase->ULocDatabase;
Clear:ULocDatabase->ULocDatabase;
PutUPrevLoc:Charstring,ULocDatabase,Integer->ULocDatabase;
PutUCurrLoc:Charstring,ULocDatabase,Integer->ULocDatabase;
PutFlag:Charstring,ULocDatabase,FlagVal->ULocDatabase;
GetUProfile:Charstring,ULocDatabase->UserProfile;


/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(ULocDatabase) #(NewULocDatabase) (void)
#else
extern #(ULocDatabase) #(NewULocDatabase) ()
#endif
{
return yMake_#(ULocDatabase)(yMake_#(ULocEle)(yMake_#(UserProfile)("V0",yMake_#(FileList)(yMalle_#(DataFile)("V0",-1,-
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(ULocDatabase) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(ULocDatabase) Param2;
#endif
{
  #(ULocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UserEle.UID)) return #(Found);
```

```
  }
  return #(NotFound);
}

#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(ULocDatabase) Param1);
#else
extern SDL_Boolean #(ListFull) (Param1)
#(ULocDatabase) Param1;
#endif
{
  #(ULocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALUSERS);cnt++){
  if(xEq_SDL_Charstring(Param1.A[cnt].UserEle.UID,"VV0"))
    return SDL_False;
  }
  return SDL_True;
}


#ifndef XNOPROTO
extern #(ULocDatabase) #(Add) (#(ULocEle) Param1,#(ULocDatabase) Param2)
#else
extern #(ULocDatabase) #(Add)(Param1,Param2)
#(ULocEle) Param1;
#(ULocDatabase) Param2;
#endif
{
  #(ULocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALUSERS);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt].UserEle.UID,"VV0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
extern #(ULocDatabase) #(Remove) (SDL_Charstring Param1,#(ULocDatabase) Param2)
#else
extern #(ULocDatabase) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(ULocDatabase) Param2;
#endif
{
  #(ULocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UserEle.UID)) {
      Param2.A[cnt].UserEle.UID = "VV0";
      return Param2;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(ULocDatabase) (Clear) (#(ULocDatabase) Param1)
#else
extern #(ULocDatabase) #(Clear) (Param1)
#(ULocDatabase) Param1;
#endif
{
  #(ULocDatabIndex) cnt;
  for(cnt=0;cnt<#(TOTALUSERS);cnt++){
    Param1.A[cnt].UserEle.UID= "VV0";
    Param1.A[cnt].UPrevLoc=-1;
    Param1.A[cnt].UCurrLoc=-1;
    Param1.A[cnt].Flag=-1;
  }
  return Param1;
}

#ifndef XNOPROTO
extern #(ULocDatabase) #(PutUPrevLoc) (SDL_Charstring Param1,#(ULocDatabase) Param2, SDL_Integer Param3)
#else
extern #(ULocDatabase) #(PutUPrevLoc)(Param1,Param2,Param3)
SDL_Charstring Param1;
#(ULocDatabase) Param2;
SDL_Integer Param3;
#endif
{
  #(ULocDatabIndex) cnt;
  for (cnt=0;cnt<#(TOTALUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UserEle.UID)){
      Param2.A[cnt].UserEle.UPrevLoc=Param3;
      Param2.A[cnt].UPrevLoc=Param3;
      return Param2;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(ULocDatabase) #(PutUCurrLoc) (SDL_Charstring Param1,#(ULocDatabase) Param2, SDL_Integer Param3)
#else
extern #(ULocDatabase) #(PutUCurrLoc)(Param1,Param2,Param3)
SDL_Charstring Param1;
#(ULocDatabase) Param2;
SDL_Integer Param3;
#endif
{
  #(ULocDatabIndex) cnt;
  for (cnt=0;cnt<#(TOTALUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UserEle.UID)){
      Param2.A[cnt].UserEle.UCurrLoc=Param3;
      Param2.A[cnt].UCurrLoc=Param3;
      return Param2;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(ULocDatabase) #(PutFlag) (SDL_Charstring Param1,#(ULocDatabase) Param2, SDL_Integer Param3)
#else
extern #(ULocDatabase) #(PutFlag)(Param1,Param2,Param3)
SDL_Charstring Param1;
#(ULocDatabase) Param2;
SDL_Integer Param3;
#endif
{
  #(ULocDatabIndex) cnt;
  for (cnt=0;cnt<#(TOTALUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UserEle.UID)){
      Param2.A[cnt].Flag=Param3;
      return Param2;
    }
  }
```

```
  return Param2;
}

#ifndef XNOPROTO
extern #(UserProfile) #(GetUProfile) (SDL_Charstring Param1,#(ULocDatabase) Param2)
#else
extern #(UserProfile) #(GetUProfile)(Param1,Param2)
SDL_Charstring Param1;
#(ULocDatabase) Param2;
#endif
{
  #(ULocDatabIndex) cnt;
  for (cnt=0;cnt<#(TOTALUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UserEle.UID)){
      return Param2.A[cnt].UserEle;
    }
  }
}

*/

ENDNEWTYPE;

/***************END USER LOCATIONS DATABASE *******/
```






```
SIGNAL

/*Network setup */
CreateLink(integer),

/*Data Link Layer*/
P(Packet),


/* Service primitives) */

SP12(SP2), /* DataLink to Network layers*/
SP21(SP2), /* Network to Data Link */
SP23(SP3), /* Network to Application */
SP32(SP3), /* Application to Network */
SP34(SP4), /* Application to User */
SP43(SP4); /* User to Application */

/*******************************/
/*      SIGNAL LISTS           */
/*******************************/

/* Network */
SIGNALLIST L=CreateLink;
SIGNALLIST PO=(PopuOut);
SIGNALLIST PI=(PopuIn), EndSim;

/****** BEGIN BaseNode SIGNALLISTS ******/

/* Between the different layers*/

/*SIGNALLIST NetworkToDataLink = */
/*SIGNALLIST DataLinkToNetwork = */
/*SIGNALLIST ApplToNetwork = */
/*SIGNALLIST NetworkToAppl = */
/*SIGNALLIST NetworkToAppl = */
/*SIGNALLIST UserToAppl = */

/****** END BaseNode SIGNALLISTS ******/
```

```
/* DATA TYPES */

SYNTYPE ParamIndex = Integer
CONSTANTS 0:MAXPARA
ENDSYNTYPE;

SYNTYPE ItemIndex=Integer
CONSTANTS 1:MAXITEM
ENDSYNTYPE;

SYNTYPE LinkIDType=Charstring
CONSTANTS 'N','S','E','W','V0','U' /* 'U' for Undefined */
ENDSYNTYPE LinkIDType;


SYNTYPE RealLinkType=LinkIDType
CONSTANTS 'N','S','E','W','U'
ENDSYNTYPE RealLinkType;


NEWTYPE AddResType
LITERALS
Added, NotAdded;
ENDNEWTYPE;

/********** BEGIN UserDatabaseShort ************/
/*Where the user profiles contain only the user files IDs*/
/***********************************************/

NEWTYPE UserDatabaseShort
array(UserIndex,UserProfileShort)
ADDING
LITERALS
NewUserDatabaseShort;
OPERATORS
Search:Charstring, UserDatabaseShort->SearchResType;
GetUser:Charstring, UserDatabaseShort->UserProfileShort;
ListFull:UserDatabaseShort->Boolean;
Count:UserDatabaseShort->Integer;
Add:UserProfileShort,UserDatabaseShort->UserDatabaseShort;/*In the first free position*/
ModifyUser:UserProfileShort,UserDatabaseShort->UserDatabaseShort;

Remove:Charstring, UserDatabaseShort->UserDatabaseShort;
Clear:UserDatabaseShort->UserDatabaseShort;
/*#ADT(B)
#BODY

#ifndef XNOPROTO
extern #(UserDatabaseShort) #(NewUserDatabaseShort) (void)
#else
extern #(UserDatabaseShort) #(NewUserDatabaseShort) ()
#endif
{
return yMake_#(UserDatabaseShort)(yMake_#(UserProfileShort)("V\0",-1,-1,-1,yMake_#(FileIDsList)("V\0")));
}


#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(UserDatabaseShort) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(UserDatabaseShort) Param2;
```

```
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++){
    if(tEq_SDL_Charstring(Param1.Param2.A[cnt].UID)) return #(Found);
  }
  return #(NotFound);
}


#ifndef XNOPROTO
extern #(UserProfileShort) #(GetUser) (SDL_Charstring Param1,#(UserDatabaseShort) Param2)
#else
extern #(UserProfileShort) #(GetUser) (Param1,Param2)
SDL_Charstring Param1;
#(UserDatabaseShort) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++){
    if(tEq_SDL_Charstring(Param1.Param2.A[cnt].UID)) return (Param2.A[cnt]);
  }
}



#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(UserDatabaseShort) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(UserDatabaseShort) Param1;
#endif
{

 #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++)
    if(tEq_SDL_Charstring(Param1.A[cnt].UID,"V\0")){
      return SDL_False;
    }
  return SDL_True;
}



#ifndef XNOPROTO
extern SDL_Integer #(Count) (#(UserDatabaseShort) Param1)
#else
extern SDL_Integer #(Count) (Param1)
#(UserDatabaseShort) Param1;
#endif
{

 #(UserIndex) cnt;
 #(UserIndex) n=0;

  for(cnt=0;cnt<#(MAXREGUSERS);cnt++)
    if(yNEqF_SDL_Charstring(Param1.A[cnt].UID,"V\0")){
      n=n+1;
    }
  return n;
}

#ifndef XNOPROTO
extern #(UserDatabaseShort) #(Add) (#(UserProfileShort) Param1,#(UserDatabaseShort) Param2)
```

```
#else
extern #(UserDatabaseShort) #(Add)(Param1,Param2)
#(UserProfileShort) Param1;
#(UserDatabaseShort) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt].UID,"V\0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
extern #(UserDatabaseShort) #(ModifyUser) (#(UserProfileShort) Param1,#(UserDatabaseShort) Param2)
#else
extern #(UserDatabaseShort) #(ModifyUser) (Param1,Param2)
#(UserProfileShort) Param1;
#(UserDatabaseShort) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++)
    if(xEq_SDL_Charstring(Param1.UID,Param2.A[cnt].UID)){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}


#ifndef XNOPROTO
extern #(UserDatabaseShort) #(Remove) (SDL_Charstring Param1,#(UserDatabaseShort) Param2)
#else
extern #(UserDatabaseShort) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(UserDatabaseShort) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UID)) {
      Param2.A[cnt].UID= "V\0";
      return Param2;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(UserDatabaseShort) #(Clear) (#(UserDatabaseShort) Param1)
#else
extern #(UserDatabaseShort) #(Clear)(Param1)
#(UserDatabaseShort) Param1;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++){
    Param1.A[cnt].UID= "V\0";
  }
  return Param1;
}

}

*/
ENDNEWTYPE;

/*************** END UserDatabaseShort ***************/


/************ BEGIN UserDatabase ************/
/*Where the user profiles contain the list files*/
/********************************************/

NEWTYPE UserDatabase
array(UserIndex,UserProfile)
ADDING
LITERALS
NewUserDatabase;
OPERATORS
Search(Charstring,UserDatabase->SearchResType;
ListFull(UserDatabase->Boolean;
Add(UserProfile,UserDatabase->UserDatabase;
Remove(Charstring, UserDatabase->UserDatabase;
Clear(UserDatabase->UserDatabase;

/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(UserDatabase) #(NewUserDatabase) (void)
#else
extern #(UserDatabase) #(NewUserDatabase) ()
#endif
{
return yMake_#(UserDatabase)(yMake_#(UserProfile)("V\0",-1,-1,yMake_#(DataFile)("V\0",-1,-1)));
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(UserDatabase) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(UserDatabase) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UID)) return #(Found);
  }
  return #(NotFound);
}

#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(UserDatabase) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(UserDatabase) Param1;
#endif
{

 #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++)
    if(xEq_SDL_Charstring(Param1.A[cnt].UID,"V\0")){
      return SDL_False;
    }
  return SDL_True;
}
```

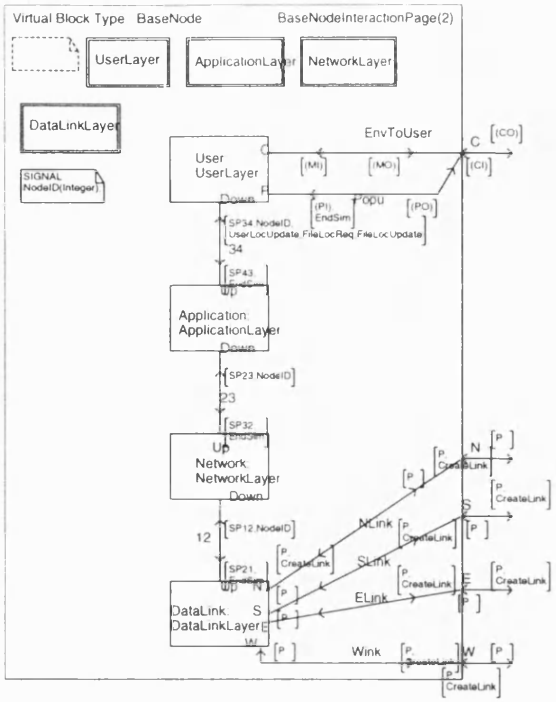
```
#ifndef XNOPROTO
extern #(UserDatabase) #(Add) (#(UserProfile) Param1,#(UserDatabase) Param2)
#else
extern #(UserDatabase) #(Add)(Param1,Param2)
#(UserProfile) Param1;
#(UserDatabase) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt].UID,"V\0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
extern #(UserDatabase) #(Remove) (SDL_Charstring Param1,#(UserDatabase) Param2)
#else
extern #(UserDatabase) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(UserDatabase) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].UID)) {
      Param2.A[cnt].UID= "V\0";
      return Param2;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(UserDatabase) #(Clear) (#(UserDatabase) Param1)
#else
extern #(UserDatabase) #(Clear)(Param1)
#(UserDatabase) Param1;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXREGUSERS);cnt++){
    Param1.A[cnt].UID= "V\0";
  }
  return Param1;
}

*/
ENDNEWTYPE;

/*************** END UserDatabase ***************/


/************ BEGIN UserList ************/
/****************************************/
NEWTYPE UserList
array(UserIndex,Charstring)
ADDING
LITERALS
NewUserList;
OPERATORS
Search(Charstring,UserList->SearchResType;
ListFull(UserList->Boolean;
Add(Charstring,UserList->UserList;
Remove(Charstring, UserList->UserList;

/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(UserList) #(NewUserList) (void)
#else
extern #(UserList) #(NewUserList) ()
#endif
{
return yMake_#(UserList)("V\0");
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(UserList) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(UserList) Param2;
#endif
{
  #(ItemIndex) cnt;
  for(cnt=0;cnt<#(MAXITEM);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt])) return #(Found);
  }
  return #(NotFound);
}

#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(UserList) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(UserList) Param1;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXITEM);cnt++)
    if(xEq_SDL_Charstring(Param1.A[cnt],"V\0")){
      return SDL_False;
    }
  return SDL_True;
}


#ifndef XNOPROTO
extern #(UserList) #(Add) (SDL_Charstring Param1,#(UserList) Param2)
#else
extern #(UserList) #(Add)(Param1,Param2)
SDL_Charstring Param1;
#(UserList) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXITEM);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt],"V\0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
```

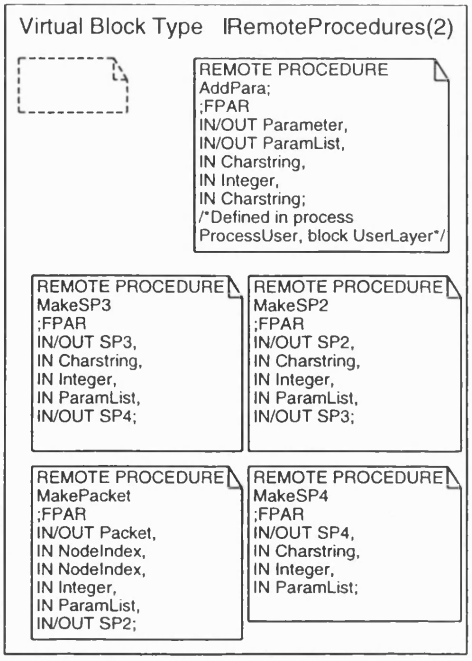
```
extern #(UserList) #(Remove) (SDL_Charstring Param1,#(UserList) Param2)
#else
extern #(UserList) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(UserList) Param2;
#endif
{
  #(UserIndex) cnt;
  for(cnt=0;cnt<#(MAXITEM);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt])) {
      Param2.A[cnt]= "V\0";
      return Param2;
    }
  }
  return Param2;
}

*/
ENDNEWTYPE;
/********** END UserList **********/


/*********************/
/*   PDU's           */
/*********************/

SYNTYPE APDUType=Charstring
CONSTANTS
 'RegUser', 'UnregUser', 'RegFile', 'UnregFile', 'Diag','StartFileReq',
 'FileReq', 'RemoteFileReq'
ENDSYNTYPE APDUType;


/*********************/
/* SERVICE DATA UNITS */
/*********************/

NEWTYPE Parameter
 STRUCT
  ParaName Charstring;
  Length   Integer;
  Data     Charstring;
ADDING
LITERALS
NewParameter;
OPERATORS
Clear(Parameter->Parameter;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(Parameter) #(NewParameter) (void)
#else
extern #(Parameter) #(NewParameter) ()
#endif
{
return yMake_#(Parameter)("V\0",-1, "V\0");
}

#ifndef XNOPROTO
extern #(Parameter) (Clear) (#(Parameter) Param1)
#else
extern #(Parameter) #(Clear) (Param1)
#(Parameter) Param1;
#endif
{
  Param1.ParaName:= "V\0";
  Param1.Length:=-1;
  Param1.Data:= "V\0";
  return Param1;
}
*/
ENDNEWTYPE;


/********** BEGIN ParamList **********
*************************************/
NEWTYPE ParamList
array(ParamIndex,Parameter)
ADDING
LITERALS
NewParamList
OPERATORS
Search(Charstring,ParamList->SearchResType;
ListFull(ParamList->Boolean
Add(Parameter,ParamList->ParamList
Remove(Charstring, ParamList->ParamList
GetParam(Integer,ParamList->Parameter
GetParamByName(Charstring,ParamList->Parameter;
Clear(ParamList->ParamList

/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(ParamList) #(NewParamList) (void)
#else
extern #(ParamList) #(NewParamList) ()
#endif
{
return (yMake_#(ParamList)(yMake_#(Parameter)("V\0",-1, "V\0")));
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(ParamList) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(ParamList) Param2;
#endif
{
  #(ParamIndex) cnt;
  for(cnt=0;cnt<#(MAXPARA);cnt++){
    if(xEq_SDL_Charstring(Param1, Param2.A[cnt].ParaName)) return #(Found);
  }
  return #(NotFound);
}


#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(ParamList) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(ParamList) Param1;
#endif
{
  if(xEq_SDL_Charstring(Param1.A[#(MAXPARA)-1].ParaName,"V\0")) return SDL_False;
  else return SDL_True;
}


#ifndef XNOPROTO
```

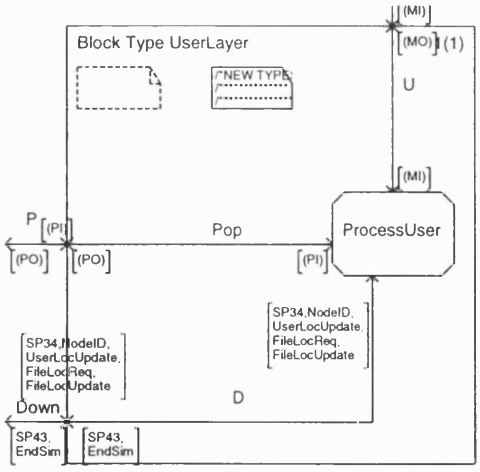
```
extern #(ParamList) #(Add) (#(Parameter) Param1,#(ParamList) Param2)
#else
extern #(ParamList) #(Add)(Param1,Param2)
#(Parameter) Param1;
#(ParamList) Param2;
#endif
{
  #(ParamIndex) cnt;
  for(cnt=0;cnt<#(MAXPARA);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt].ParaName,"V\0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
extern #(ParamList) #(Remove) (SDL_Charstring Param1,#(ParamList) Param2)
#else
extern #(ParamList) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(ParamList) Param2;
#endif
{
  #(ParamIndex) cnt;
  for(cnt=0;cnt<#(MAXPARA);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].ParaName)) {
      Param2.A[cnt].ParaName = "V\0";
      return Param2;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(Parameter) #(GetParam) (SDL_Integer Param1,#(ParamList) Param2)
#else
extern #(Parameter) #(GetParam)(Param1,Param2)
SDL_Integer Param1;
#(ParamList) Param2;
#endif
{
return Param2.A[Param1];
}


#ifndef XNOPROTO
extern #(Parameter) #(GetParamByName) (SDL_Charstring Param1,#(ParamList) Param2)
#else
extern #(Parameter) #(GetParamByName)(Param1,Param2)
SDL_Charstring Param1;
#(ParamList) Param2;
#endif
{

  #(ParamIndex) cnt;
  for(cnt=0;cnt<#(MAXPARA);cnt++){
    if(xEq_SDL_Charstring(Param1, Param2.A[cnt].ParaName))
      return Param2.A[cnt];
  }
}


#ifndef XNOPROTO
```

```
extern #(ParamList) (Clear) (#(ParamList) Param1)
#else
extern #(ParamList) #(Clear) (Param1)
#(ParamList) Param1;
#endif
{
  #(ParamIndex) cnt;
  for(cnt=0;cnt<#(MAXPARA);cnt++){
    Param1.A[cnt].ParaName:="V\0";
    Param1.A[cnt].Length:=-1;
    Param1.A[cnt].Data:="V\0";
  }
  return Param1;
}

*/
ENDNEWTYPE;
/********** END ParamList **********/


/********** BEGIN SP4 **********/
/*******************************/
NEWTYPE SP4
 STRUCT
  SP4ID APDUType;
  SP4Length Integer;
  SP4Parc ParamList;
ADDING
LITERALS
NewSP4;

/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(SP4) #(NewSP4) (void)
#else
extern #(SP4) #(NewSP4) ()
#endif
{
return yMake_#(SP4) ("V\0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V\0",-1, "V\0")));
}
*/
ENDNEWTYPE;
/********** END SP4 **********/

/********** BEGIN SP3 **********/
/*******************************/
NEWTYPE SP3
 STRUCT
  SP3ID    Charstring;
  SP3Length Integer;
  SP3Parc  ParamList;
  SP4Part  SP4;
ADDING
LITERALS
NewSP3;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(SP3) #(NewSP3) (void)
#else
extern #(SP3) #(NewSP3) ()
#endif
{
return yMake_#(SP3) ("V\0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V\0",-1, "V\0")),
```

```
yMake_#(SP4) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1,"V0"))));
}
*/
ENDNEWTYPE;
/*************** END SP1 ******************/

/*************** BEGIN SP2 ****************/
/*****************************************/
NEWTYPE SP2
 STRUCT
  SP2ID    Charstring;
  SP2Length Integer;
  SP2Para   ParamList;
  SP1Part   SP3;
ADDING
LITERALS
 NewSP2;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(SP2) #(NewSP2) (void)
#else
extern #(SP2) #(NewSP2) ()
#endif
{
return yMake_#(SP2) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1,"V0")),
yMake_#(SP3) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1,"V0")),
yMake_#(SP4) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1,"V0")))));

}
*/
ENDNEWTYPE;
/*************** END SP2 ******************/

/*************** BEGIN Packet ****************/
/*****************************************/
NEWTYPE Packet
 STRUCT
  Origin    NodeIndex;
  Dest      NodeIndex;
  PackLength Integer;
  PackPara  ParamList;
  SP2Part   SP2;
ADDING
LITERALS
 NewPacket;
OPERATORS
Clear(Packet->Packet;
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(Packet) #(NewPacket) (void)
#else
extern #(Packet) #(NewPacket) ()
#endif
{
return yMake_#(Packet) (-1,-1,-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1,"V0")),
yMake_#(SP2) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1,"V0")),
yMake_#(SP3) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1,"V0")),
yMake_#(SP4) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1,"V0")))));

}

#ifndef XNOPROTO
```



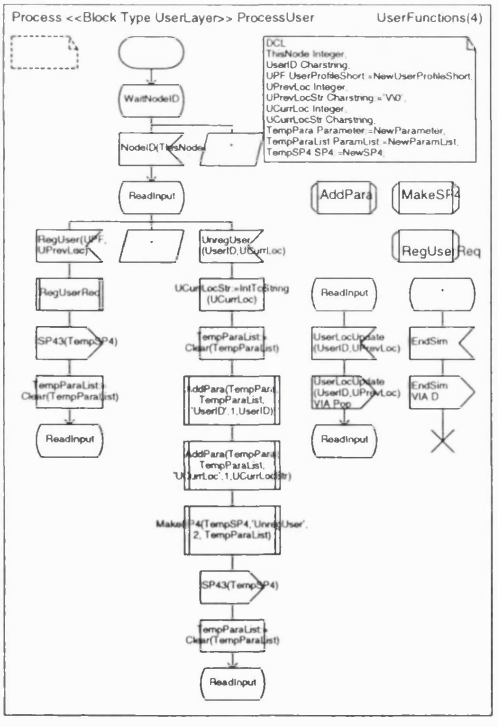


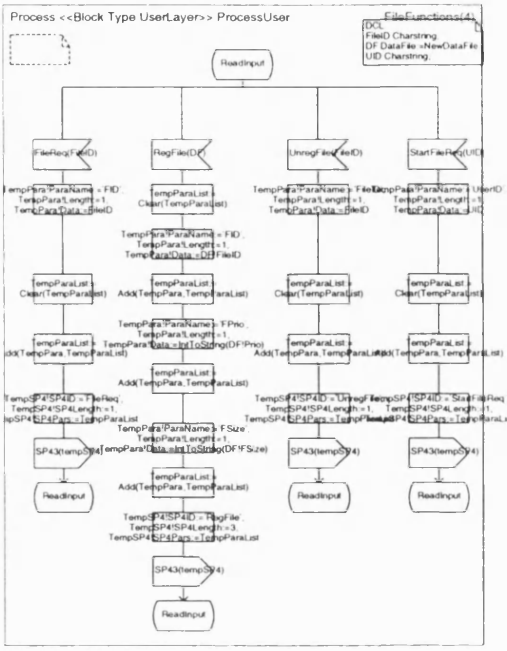


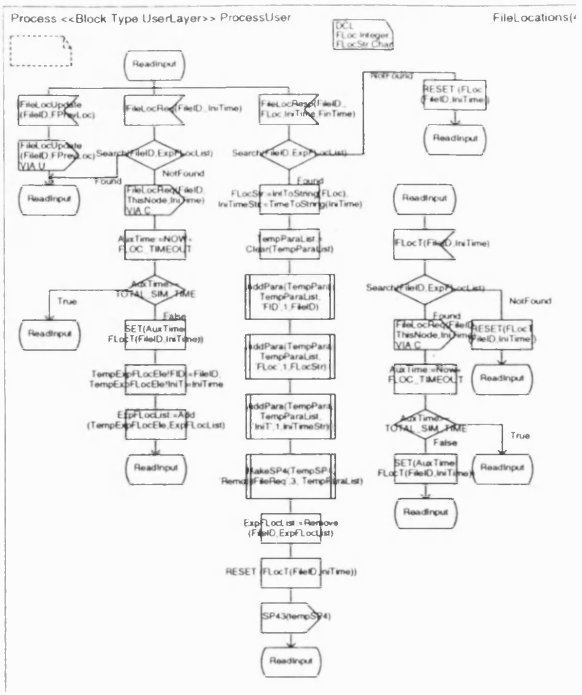
```
/*NEW TYPES DECLARATIONS*/
/************** EXPECTED_FLOC_ELE **********/
/*****************************************/
SYNONYM Zero_time Time=0;

NEWTYPE ExpectedFLocEle
  STRUCT
    FID Charstring;
    InuT Time;
ADDING
LITERALS
NewExpectedFLocEle;
OPERATORS
Clear:ExpectedFLocEle->ExpectedFLocEle;

/*ADT(B)
#BODY
#ifndef XNOPROTO
extern #(ExpectedFLocEle) #(NewExpectedFLocEle) (void)
#else
extern #(ExpectedFLocEle) #(NewExpectedFLocEle) ()
#endif
{
return yMake_#(ExpectedFLocEle)("V\0",#(Zero_Time));
}

#ifndef XNOPROTO
extern #(ExpectedFLocEle) (Clear) (#(ExpectedFLocEle) Param1)
#else
extern #(ExpectedFLocEle) #(Clear) (Param1)
#(ExpectedFLocEle) Param1;
#endif
{
  Param1.FID="V\0";
  Param1.InuT=#(Zero_Time);
  return Param1;
}
*/
ENDNEWTYPE;

/********** END EXPECTED_FLOC_ELE **********/


/************** EXPECTED_FLOC LIST **********/
/*****************************************/

NEWTYPE ExpectedFLocList
array(ExpectedFLocIndex,ExpectedFLocEle)
 ADDING
LITERALS
NewExpectedFLocList;
OPERATORS
Search:Charstring,ExpectedFLocList->SearchResType;
ListFull:ExpectedFLocList->Boolean;
Count:ExpectedFLocList->integer;
Add:ExpectedFLocEle,ExpectedFLocList->ExpectedFLocList;
Remove:Charstring, ExpectedFLocList->ExpectedFLocList;

/*ADT(B)
#BODY
#ifndef XNOPROTO
extern #(ExpectedFLocList) #(NewExpectedFLocList) (void)
#else
extern #(ExpectedFLocList) #(NewExpectedFLocList) ()
```

```
#endif
{
return yMake_#(ExpectedFLocList)(yMake_#(ExpectedFLocEle)("V\0",#(Zero_Time)));
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(ExpectedFLocList) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(ExpectedFLocList) Param2;
#endif
{
  #(ExpectedFLocIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FLOC);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)) return #(Found);
  }
  return #(NotFound);
}

#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(ExpectedFLocList) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(ExpectedFLocList) Param1;
#endif
{

 #(ExpectedFLocIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FLOC);cnt++)
    if(xEq_SDL_Charstring(Param1.A[cnt].FID,"V\0")){
      return SDL_False;
    }
  return SDL_True;
}


#ifndef XNOPROTO
extern SDL_Integer #(Count) (#(ExpectedFLocList) Param1)
#else
extern SDL_Integer #(Count) (Param1)
#(ExpectedFLocList) Param1;
#endif
{

 #(ExpectedFLocIndex) cnt;
 #(ExpectedFLocIndex) n=0;
  for(cnt=0;cnt<#(MAX_EXP_FLOC);cnt++)
    if(yNEqF_SDL_Charstring(Param1.A[cnt].FID,"V\0")){
      n=n+1;
    }
  return n;
}


#ifndef XNOPROTO
extern #(ExpectedFLocList) #(Add) (#(ExpectedFLocEle) Param1,#(ExpectedFLocList) Param2)
#else
extern #(ExpectedFLocList) #(Add)(Param1,Param2)
#(ExpectedFLocEle) Param1;
#(ExpectedFLocList) Param2;
#endif
{
```

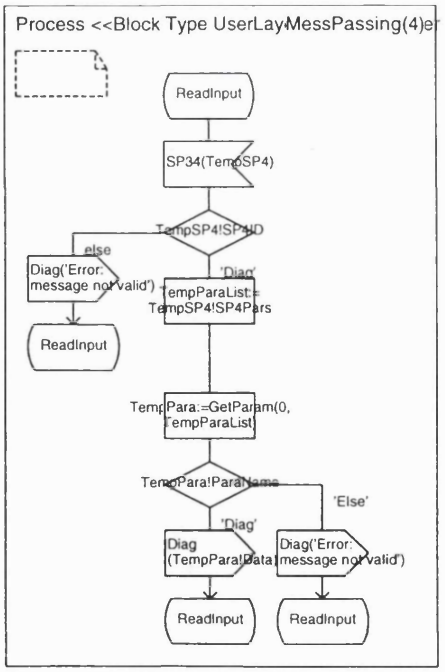
```
  #(ExpectedFLocIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FLOC);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt].FID,"V\0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
extern #(ExpectedFLocList) #(Remove) (SDL_Charstring Param1,#(ExpectedFLocList) Param2)
#else
extern #(ExpectedFLocList) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(ExpectedFLocList) Param2;
#endif
{
  #(ExpectedFLocIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FLOC);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)) {
      Param2.A[cnt].FID= "V\0";
      return Param2;
    }
  }
  return Param2;
}

*/
ENDNEWTYPE;

/************** END EXPECTED_FLOC LIST **********/
```

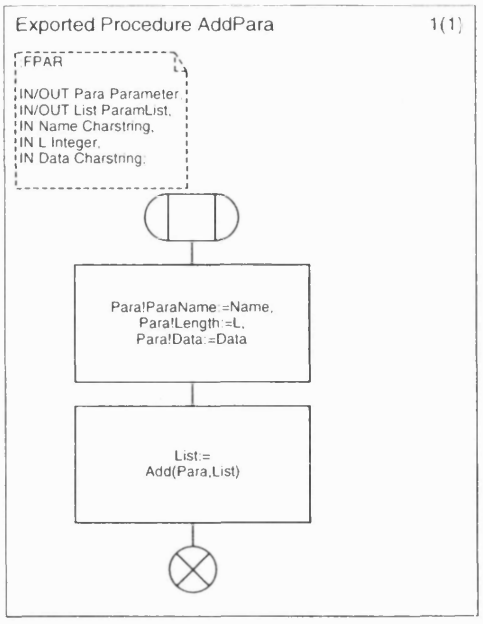


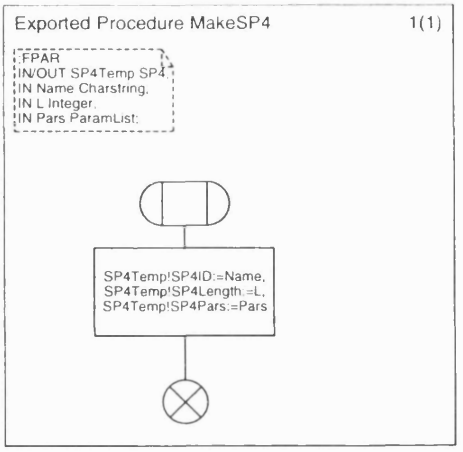


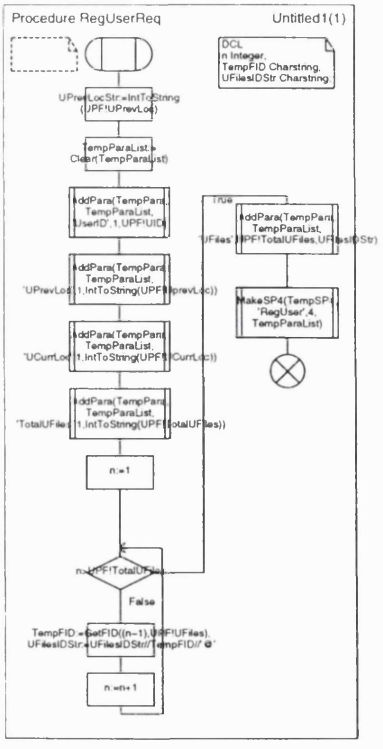


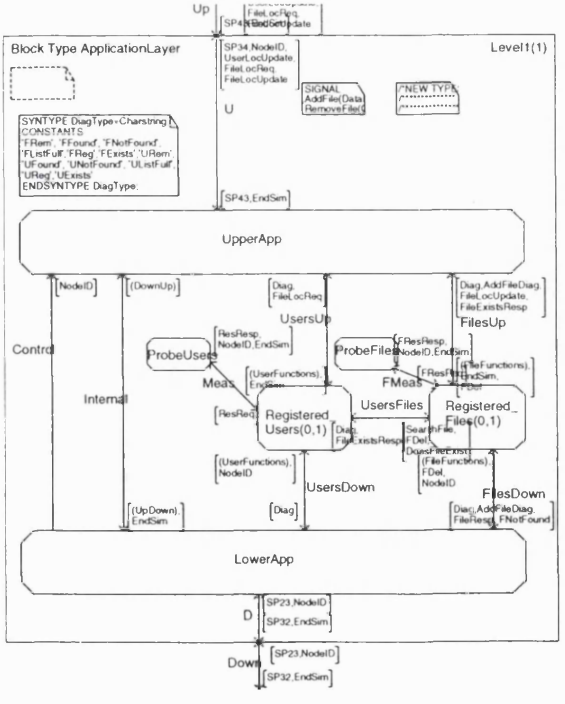

```
DCL
FLoc Integer,
FLocStr Charstring,
FPrevLoc Integer,
IniTime Time,
IniTimeStr charstring,
FinTime Time,
AuxTime Time,
TempExpFLocEle ExpectedFLocEle:=NewExpectedFLocEle,
ExpFLocList ExpectedFLocList:=NewExpectedFLocList;

TIMER
FLocT(Charstring,Time);/*FID,IniTime*/
```




Exported Procedure AddPara
1(1)
FPAR
IN/OUT Para Parameter,
IN/OUT List ParamList,
IN Name Charstring,
IN L Integer,
IN Data Charstring;
Para!ParaName:=Name,
Para!Length:=L,
Para!Data:=Data
List:=
Add(Para,List)


Exported Procedure MakeSP4
1(1)
FPAR
IN/OUT SP4Temp SP4,
IN Name Charstring,
IN L Integer,
IN Pars ParamList;
SP4Temp!SP4ID:=Name,
SP4Temp!SP4Length:=L,
SP4Temp!SP4Pars:=Pars


Procedure RegUserReq
Untitled1(1)
DCL
n Integer,
TempFID Charstring,
UFilesIDStr Charstring;
UPrevLocStr:=IntToString
(UPF!UPrevLoc)
TempParaList:=
Clear(TempParaList)
AddPara(TempPara
TempParaList,
'UserID',1,UPF!UID)
AddPara(TempPara
TempParaList,
'UPrevLoc',1,IntToString(UPF
AddPara(TempPara
TempParaList,
'UCurrLoc',1,IntToString(UPF
AddPara(TempPara
TempParaList,
'TotalUFiles',1,IntToString(UPF
n:=1
n>UPF!TotalUFiles
True
AddPara(TempPara
TempParaList,
'UFiles',UPF!TotalUFiles,UFi
MakeSP4(TempSP4
'RegUser',4,
TempParaList)
False
TempFID:=GetFID((n-1),UPF!UFiles),
UFilesIDStr:=UFilesIDStr//TempFID//';'
n:=n+1


Up
Block Type ApplicationLayer
Level1(1)
SP34,NodeID,
UserLocUpdate,
FileLocReq,
FileLocUpdate
SIGNAL
AddFile(Data
RemoveFile(
NEW TYPE
U
SYNTYPE DiagType=Charstring
CONSTANTS
'FRem', 'FFound', 'FNotFound',
'FListFull','FReq','FExists','URem',
'UFound', 'UNotFound', 'UListFull',
'UReq','UExists'
ENDSYNTYPE DiagType;
SP43,EndSim
UpperApp
NodeID
(DownUp)
Diag,
FileLocReq
Diag,AddFileDiag,
FileLocUpdate,
FileExistsResp
UsersUp
FilesUp
Control
ProbeUsers
ResResp,
NodeID,EndSim
ProbeFiles
FResResp,
NodeID,EndSim
(FileFunctions),
EndSim,
Meas
(UserFunctions),
EndSim
FMeas
Internal
UsersFiles
ResReq
Registered_
Users(0,1)
Registered_
Files(0,1)
Diag,
FileExistsResp
SearchFile,
FDel,
DoesFileExist
(FileFunctions),
FDel,
NodeID
(UserFunctions),
NodeID
UsersDown
FilesDown
(UpDown),
EndSim
Diag
Diag,AddFileDiag,
FileResp,FNotFound
LowerApp
D
SP23,NodeID,
SP32,EndSim
Down
SP23,NodeID
SP32,EndSim

```
SIGNAL
AddFile(DataFile),
RemoveFile(Charstring),
SearchFile(Charstring),
AddUser(UserProfileShort),
RemoveUser(Charstring),
Diag(DiagType, Charstring, Integer),
AddFileDiag(DiagType, Charstring, Integer),
DoesFileExist(Charstring),
FNotFound(Charstring),
FileExistsResp(SearchResType),
FDel(Time, Duration), /*NOW, Delay in obtaining the file*/
StartFileReq(Charstring,Integer), /*UserID,nothing*/
ResReq,ResResp(UserDatabaseShort),/*Measures' Result request, Result response*/
FResReq,FResResp(FileList,FDelArray)/*Measures' Files Result request, Files Result response*/;

SIGNALLIST FileFunctions= AddFile,SearchFile,DoesFileExist,FileExistsResp,RemoveFile,FileReq,FileResp;
SIGNALLIST UserFunctions= AddUser,SearchUser,RemoveUser,StartFileReq;
SIGNALLIST UpDown=SP32;
SIGNALLIST DownUp=SP23,FileLocUpdate, FileLocReq;
```

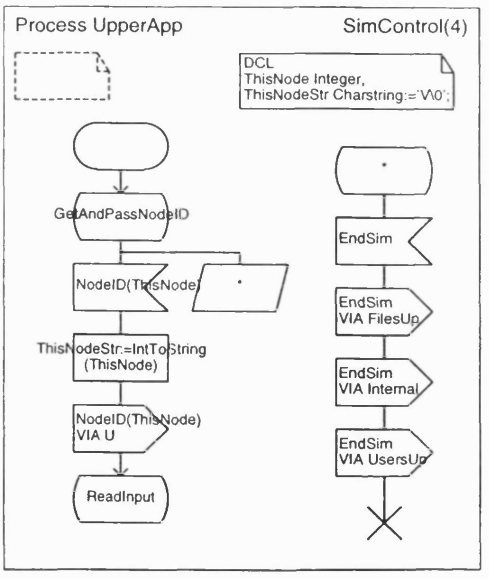
```
/*NEW TYPES DECLARATIONS*/
/************** EXPECTED_FLOC_ELE **********/
/*******************************************/
SYNONYM Zero_time Time=0;
SYNONYM Zero_duration Duration=0;

NEWTYPE ExpectedFileEle
  STRUCT
    FID Charstring;
    FileSP3 SP3;
    InuT Time;
ADDING
LITERALS
NewExpectedFileEle;
OPERATORS
Clear:ExpectedFileEle->ExpectedFileEle;

/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(ExpectedFileEle) #(NewExpectedFileEle) (void)
#else
extern #(ExpectedFileEle) #(NewExpectedFileEle) ()
#endif
{
return yMake_#(ExpectedFileEle)("V0",
(yMake_#(SP3) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1, "V0")),
              yMake_#(SP4) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1, "V0"))))),
#(Zero_Time));
}

#ifndef XNOPROTO
extern #(ExpectedFileEle) #(Clear) (#(ExpectedFileEle) Param1)
#else
extern #(ExpectedFileEle) #(Clear) (Param1)
#(ExpectedFileEle) Param1
#endif
{
  Param1.FID="V0";
  Param1.InuT=#(Zero_Time);
  return Param1;
}
*/
ENDNEWTYPE;

/********** END EXPECTED_FLOC_ELE ******************/


/************** EXPECTED_FLOC LIST **********/
/*******************************************/

NEWTYPE ExpectedFileList
array(ExpectedFileIndex,ExpectedFileEle)
ADDING
LITERALS
NewExpectedFileList;
OPERATORS
Search:Charstring,ExpectedFileList->SearchResType;
ListFull:ExpectedFileList->Boolean;
Count:ExpectedFileList->Integer;
Add:ExpectedFileEle,ExpectedFileList->ExpectedFileList;
Remove:Charstring, ExpectedFileList->ExpectedFileList;
Modify:ExpectedFileEle,ExpectedFileList->ExpectedFileList;
GetFileEle:Charstring, ExpectedFileList->ExpectedFileEle;
```

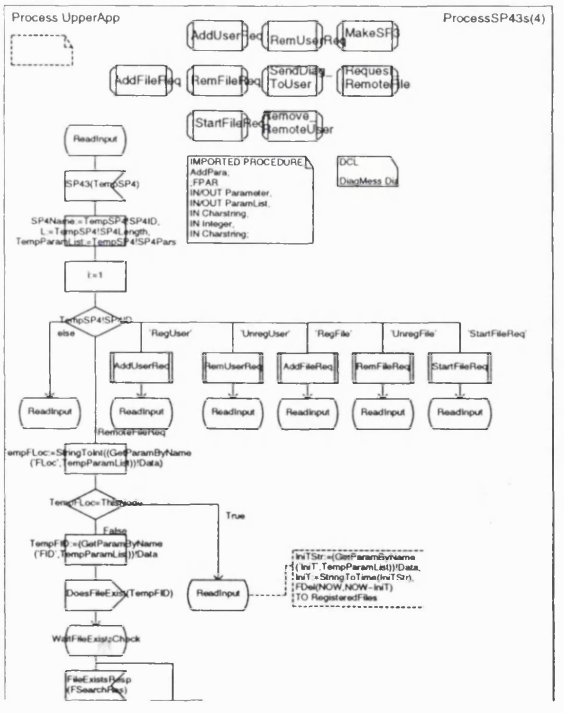
```
/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(ExpectedFileList) #(NewExpectedFileList) (void)
#else
extern #(ExpectedFileList) #(NewExpectedFileList) ()
#endif
{
return yMake_#(ExpectedFileList)(yMake_#(ExpectedFileEle)("V0",
(yMake_#(SP3) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1, "V0")),
              yMake_#(SP4) ("V0",-1, yMake_#(ParamList)(yMake_#(Parameter)("V0",-1, "V0"))))),
#(Zero_Time)));
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (SDL_Charstring Param1,#(ExpectedFileList) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
SDL_Charstring Param1;
#(ExpectedFileList) Param2;
#endif
{
  #(ExpectedFileIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)) return #(Found);
  }
  return #(NotFound);
}

#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(ExpectedFileList) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(ExpectedFileList) Param1;
#endif
{

 #(ExpectedFileIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FILES);cnt++)
    if(xEq_SDL_Charstring(Param1.A[cnt].FID,"V0")){
      return SDL_False;
    }
  return SDL_True;
}


#ifndef XNOPROTO
extern SDL_Integer #(Count) (#(ExpectedFileList) Param1)
#else
extern SDL_Integer #(Count) (Param1)
#(ExpectedFileList) Param1;
#endif
{

 #(ExpectedFileIndex) cnt;
 #(ExpectedFileIndex) n=0;
  for(cnt=0;cnt<#(MAX_EXP_FILES);cnt++)
    if(yNEqF_SDL_Charstring(Param1.A[cnt].FID,"V0")){
      n=n+1;
    }
  return n;
}
```

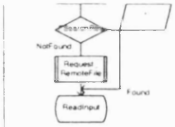
```
#ifndef XNOPROTO
extern #(ExpectedFileList) #(Add) (#(ExpectedFileEle) Param1,#(ExpectedFileList) Param2)
#else
extern #(ExpectedFileList) #(Add)(Param1,Param2)
#(ExpectedFileEle) Param1;
#(ExpectedFileList) Param2;
#endif
{
  #(ExpectedFileIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FILES);cnt++)
    if(xEq_SDL_Charstring(Param2.A[cnt].FID, "V0")){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}

#ifndef XNOPROTO
extern #(ExpectedFileList) #(Remove) (SDL_Charstring Param1,#(ExpectedFileList) Param2)
#else
extern #(ExpectedFileList) #(Remove)(Param1,Param2)
SDL_Charstring Param1;
#(ExpectedFileList) Param2;
#endif
{
  #(ExpectedFileIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FILES);cnt++){
    if(xEq_SDL_Charstring(Param1,Param2.A[cnt].FID)) {
      Param2.A[cnt].FID= "V0";
      return Param2;
    }
  }
  return Param2;
}


#ifndef XNOPROTO
extern #(ExpectedFileList) #(Modify) (#(ExpectedFileEle) Param1,#(ExpectedFileList) Param2)
#else
extern #(ExpectedFileList) #(Modify)(Param1,Param2)
#(ExpectedFileEle) Param1;
#(ExpectedFileList) Param2;
#endif
{
  #(ExpectedFileIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FILES);cnt++){
    if(xEq_SDL_Charstring(Param1.FID,Param2.A[cnt].FID)) {
      Param2.A[cnt]= Param1;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(ExpectedFileEle) #(GetFileEle) (SDL_Charstring Param1,#(ExpectedFileList) Param2)
#else
extern #(ExpectedFileEle) #(GetFileEle)(Param1,Param2)
SDL_Charstring Param1;
#(ExpectedFileList) Param2;
#endif
{
  #(ExpectedFileIndex) cnt;
  for(cnt=0;cnt<#(MAX_EXP_FILES);cnt++){
```

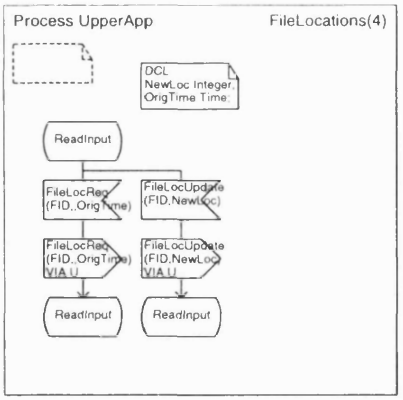
```
    if(yEq_SDL_Charstring(Param1.Param2.A[cnt].FID)) {
      return Param2.A[cnt];
    }
  }
}

*/
ENDNEWTYPE;

/************** END EXPECTED_FILE LIST **********/

/************** BEGIN F_DEL_ELE **************/
NEWTYPE FDelEle
  STRUCT
    ArrivalTime Time;
    FDel Duration;
ADDING
LITERALS
NewFDelEle;
OPERATORS
Clear(FDelEle->FDelEle;

/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(FDelEle) #(NewFDelEle) (void)
#else
extern #(FDelEle) #(NewFDelEle) ()
#endif
{
return yMake_#(FDelEle)(#(Zero_time),#(Zero_duration));
}

#ifndef XNOPROTO
extern #(FDelEle) #(Clear) (#(FDelEle) Param1)
#else
extern #(FDelEle) #(Clear)(Param1)
#(FDelEle) Param1;
#endif
{
  Param1.ArrivalTime=#(Zero_time);
  return Param1;
}
*/
ENDNEWTYPE;

/********** END F_DEL_ELE **********/

/************** F_DEL_ARRAY **********/
/*****************************************/

NEWTYPE FDelArray
array(FDelIndex,FDelEle)
ADDING
LITERALS
NewFDelArray;
OPERATORS

Add(FDelEle,FDelArray->FDelArray;
Clear(FDelArray->FDelArray;

/*#ADT(B)
```

```
#BODY
#ifndef XNOPROTO
extern #(FDelArray) #(NewFDelArray) (void)
#else
extern #(FDelArray) #(NewFDelArray) ()
#endif
{
return yMake_#(FDelArray)(yMake_#(FDelEle)(#(Zero_time),#(Zero_duration)));
}


#ifndef XNOPROTO
extern #(FDelArray) #(Add) (#(FDelEle) Param1,#(FDelArray) Param2)
#else
extern #(FDelArray) #(Add)(Param1,Param2)
#(FDelEle) Param1;
#(FDelArray) Param2;
#endif
{
  #(FDelIndex) cnt;
  for(cnt=0;cnt<#(MAX_F_DEL_ARRAY);cnt++)
    if(yEq(F_SDL_Time(Param2.A[cnt].ArrivalTime,#(Zero_time))){
      Param2.A[cnt]=Param1;
      return Param2;
    }
  return Param2;
}


#ifndef XNOPROTO
extern #(FDelArray) #(Clear) (#(FDelArray) Param1)
#else
extern #(FDelArray) #(Clear)(Param1)
#(FDelArray) Param1;
#endif
{
  #(FDelIndex) cnt;
  for(cnt=0;cnt<#(MAX_F_DEL_ARRAY);cnt++){
      Param1.A[cnt].ArrivalTime=#(Zero_time);
    }
  return Param1;
}

*/
ENDNEWTYPE;

/****************** END F_DEL_ARRAY **********/
```



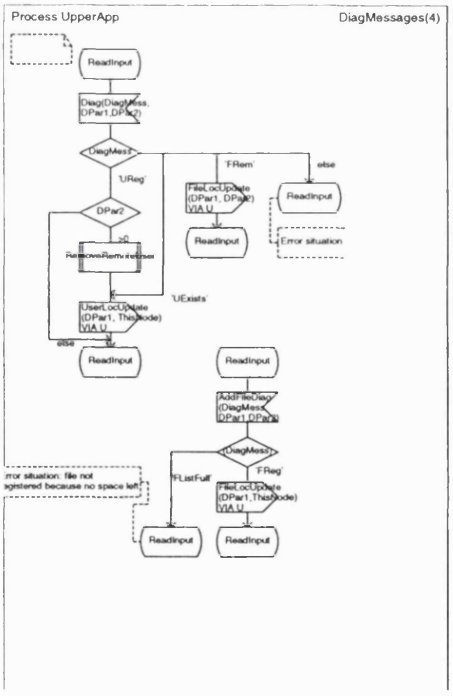


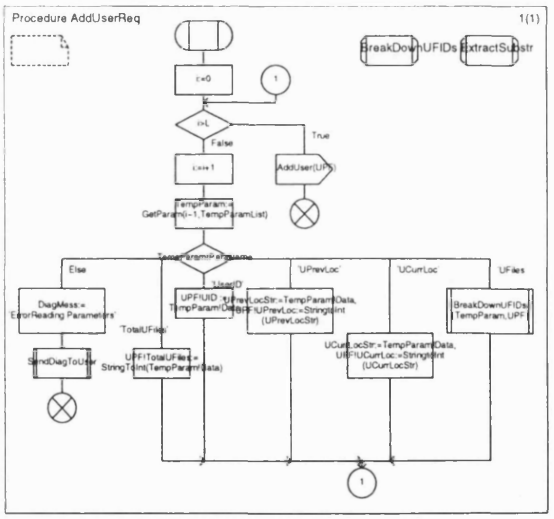


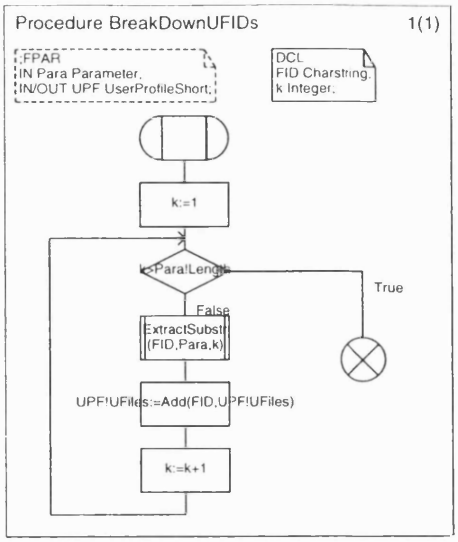


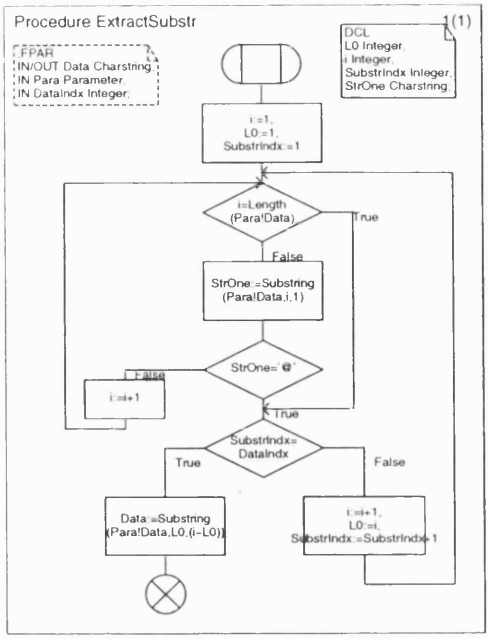
DCL

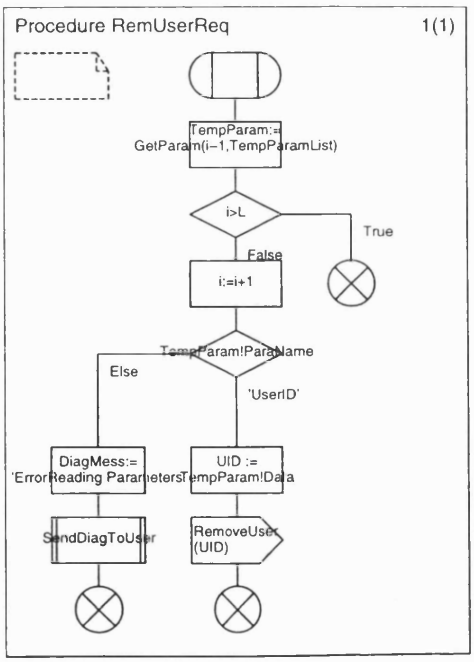
DiagMess DiagType,
DPar1 Charstring,
DPar2 Integer,
TempSP4 SP4:=NewSP4,
TempSP3 SP3:=NewSP3,
TempParam Parameter:=NewParameter,
TempParamList ParamList:=NewParamList,
SP4Name Charstring,
L Integer,
i Integer,
UPF UserProfileShort:=NewUserProfileShort,
UID Charstring,
UPrevLocStr Charstring,
UCurrLocStr Charstring,
FID Charstring,
TempFID Charstring,
TempFLoc Integer,
DF DataFile,
FSearchRes SearchResType,
IniT Time,
IniTStr Charstring;

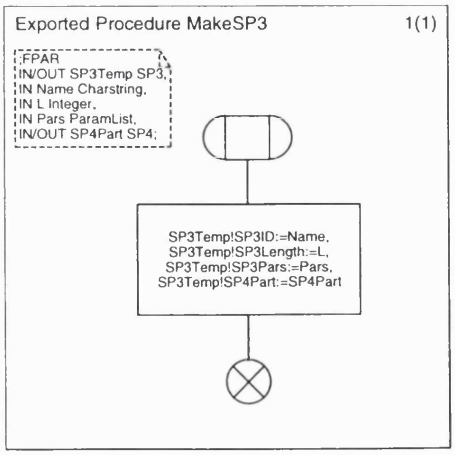


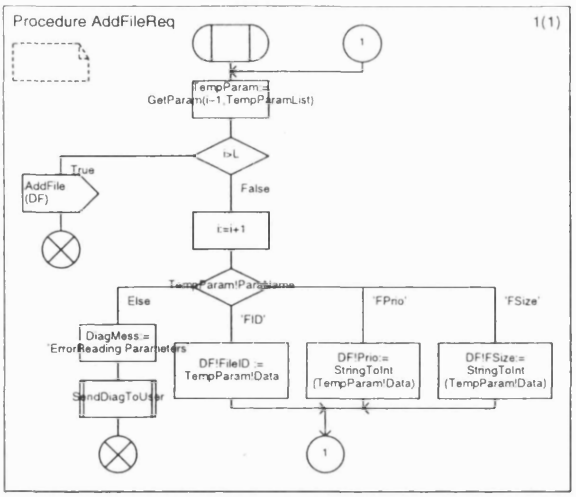


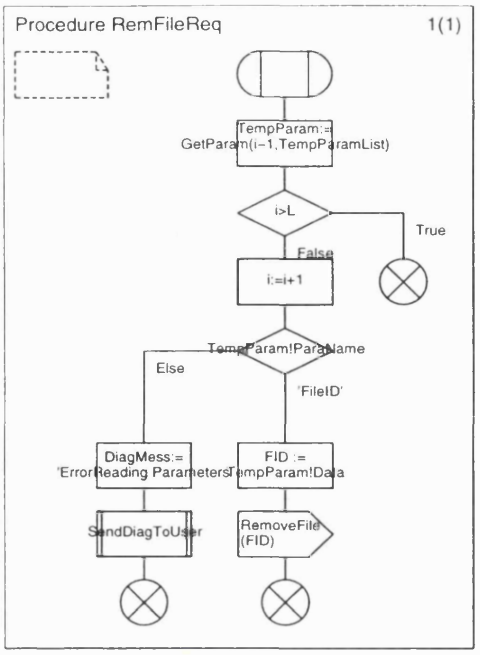

Procedure BreakDownUFIDs
1(1)
FPAR
IN Para Parameter,
IN/OUT UPF UserProfileShort;
DCL
FID Charstring,
k Integer;
k:=1
k>Para!Length
True
False
ExtractSubstr
(FID,Para,k)
UPF!UFiles:=Add(FID,UPF!UFiles)
k:=k+1

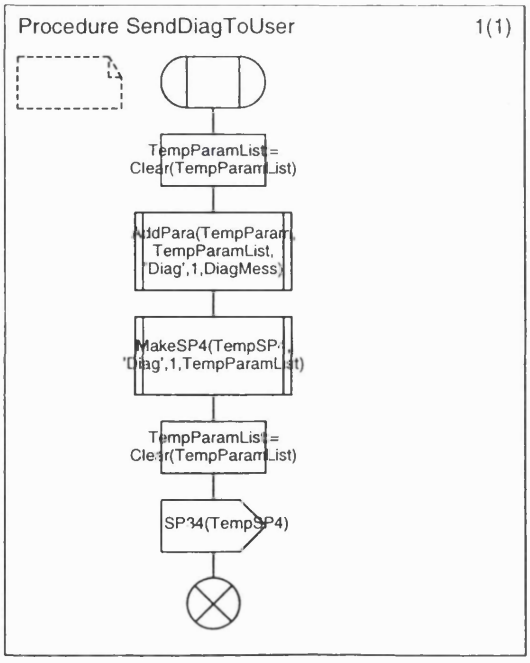

Procedure ExtractSubstr
1(1)
FPAR
IN/OUT Data Charstring,
IN Para Parameter,
IN DataIndx Integer;
DCL
L0 Integer,
i Integer,
SubstrIndx Integer,
StrOne Charstring;
i:=1,
L0:=1,
SubstrIndx:=1
i=Length
(Para!Data)
True
False
StrOne:=Substring
(Para!Data,i,1)
StrOne='@'
False
i:=i+1
True
SubstrIndx=
DataIndx
True
False
Data:=Substring
(Para!Data,L0,(i-L0))
i:=i+1,
L0:=i,
SubstrIndx:=SubstrIndx+1

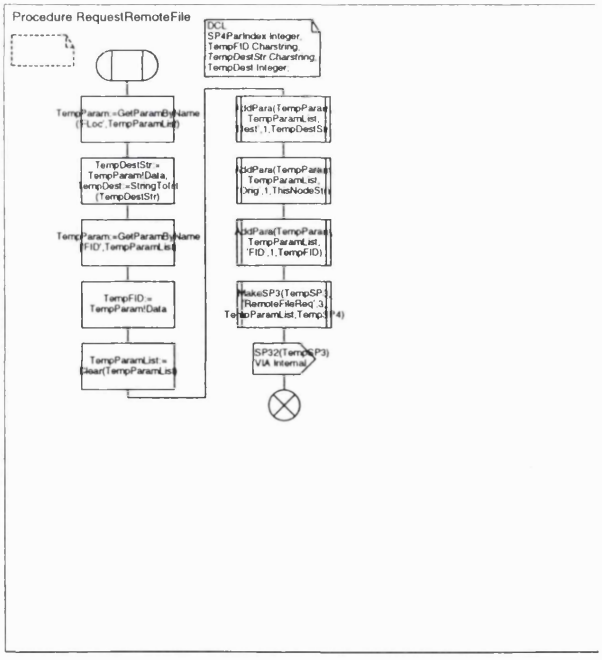

Procedure RemUserReq
1(1)
TempParam:=
GetParam(i-1,TempParamList)
i>L
True
False
i:=i+1
TempParam!ParaName
Else
'UserID'
DiagMess:=
'Error Reading Parameters'
UID :=
TempParam!Data
SendDiagToUser
RemoveUser
(UID)

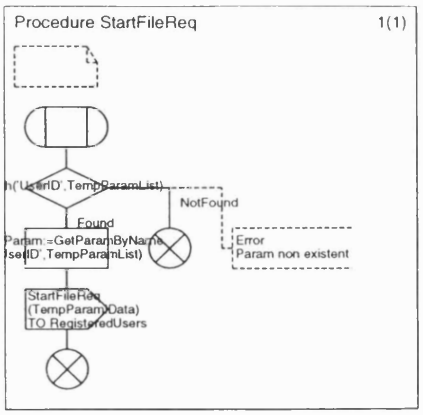

Exported Procedure MakeSP3
1(1)
FPAR
IN/OUT SP3Temp SP3,
IN Name Charstring,
IN L Integer,
IN Pars ParamList,
IN/OUT SP4Part SP4;
SP3Temp!SP3ID:=Name,
SP3Temp!SP3Length:=L,
SP3Temp!SP3Pars:=Pars,
SP3Temp!SP4Part:=SP4Part

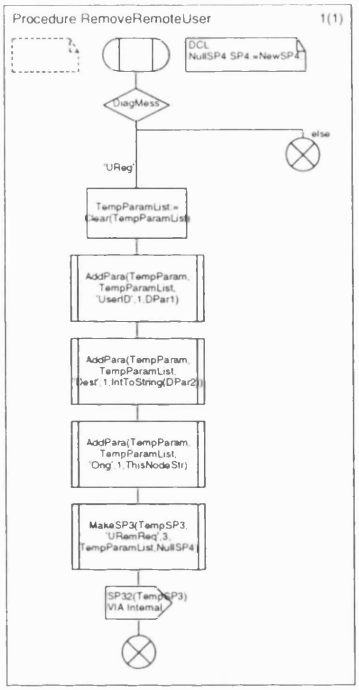

Procedure AddFileReq
1(1)
TempParam:=
GetParam(i-1,TempParamList)
i>L
True
False
AddFile
(DF)
i:=i+1
TempParam!ParaName
Else
'FID'
'FPrio'
'FSize'
DiagMess:=
'ErrorReading Parameters'
DF!FileID :=
TempParam!Data
DF!Prio:=
StringToInt
(TempParam!Data)
DF!FSize:=
StringToInt
(TempParam!Data)
SendDiagToUser

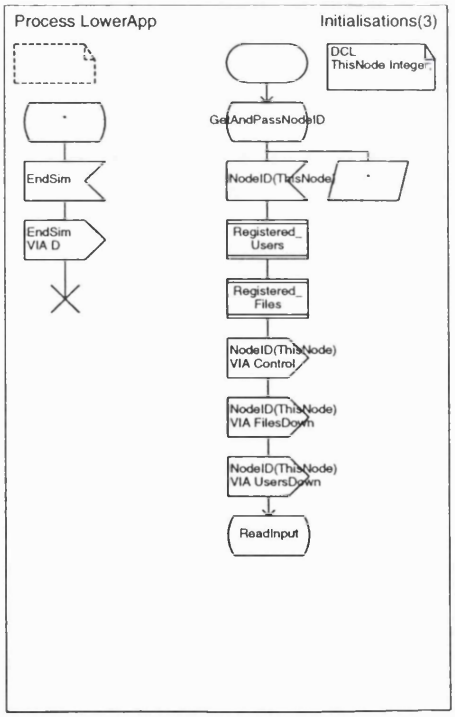

Procedure RemFileReq
1(1)
TempParam:=
GetParam(i-1,TempParamList)
i>L
True
False
i:=i+1
TempParam!ParaName
Else
'FileID'
DiagMess:=
'ErrorReading Parameters'
FID :=
TempParam!Data
SendDiagToUser
RemoveFile
(FID)

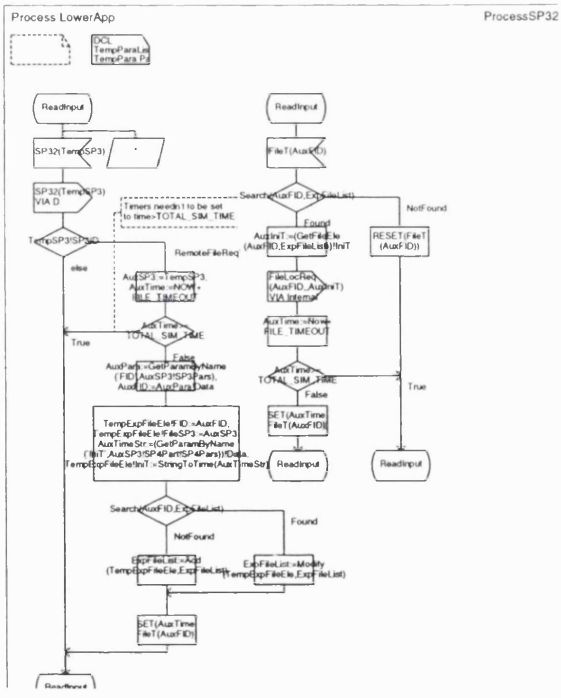

Procedure SendDiagToUser
1(1)
TempParamList:=
Clear(TempParamList)
AddPara(TempParam,
TempParamList,
'Diag',1,DiagMess)
MakeSP4(TempSP4,
'Diag',1,TempParamList)
TempParamList:=
Clear(TempParamList)
SP34(TempSP4)


Procedure RequestRemoteFile
DCL
SP4ParIndex Integer,
TempFID Charstring,
TempDestStr Charstring,
TempDest Integer;
TempParam:=GetParamByName
(FLoc',TempParamList)
TempDestStr:=
TempParam!Data,
TempDest:=StringToInt
(TempDestStr)
TempParam:=GetParamByName
(FID',TempParamList)
TempFID:=
TempParam!Data
TempParamList:=
Clear(TempParamList)
AddPara(TempParam,
TempParamList,
'Diag',1,TempDestStr)
AddPara(TempParam,
TempParamList,
'FID',1,TempFID)
MakeSP3(TempSP3,
'RemoteFileReq',3,
TempParamList,TempSP4)
SP32(TempSP3)
VIA Internal

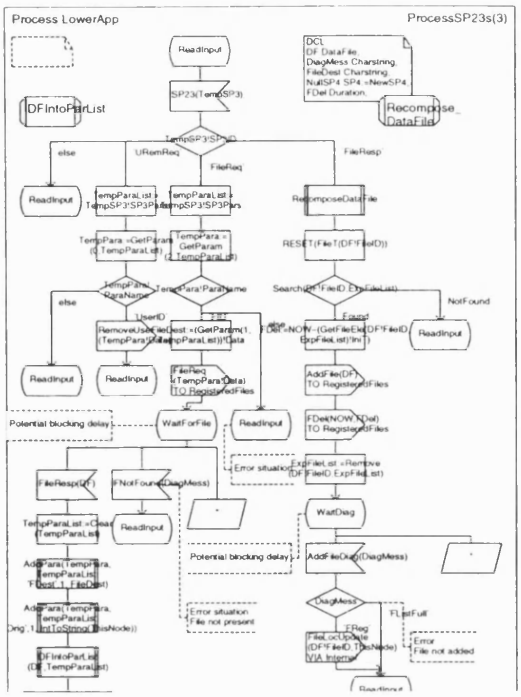

Procedure StartFileReq
1(1)
Found
NotFound
Error
Param non existent
StartFileReq
(TempParamData)
TO RegisteredUsers

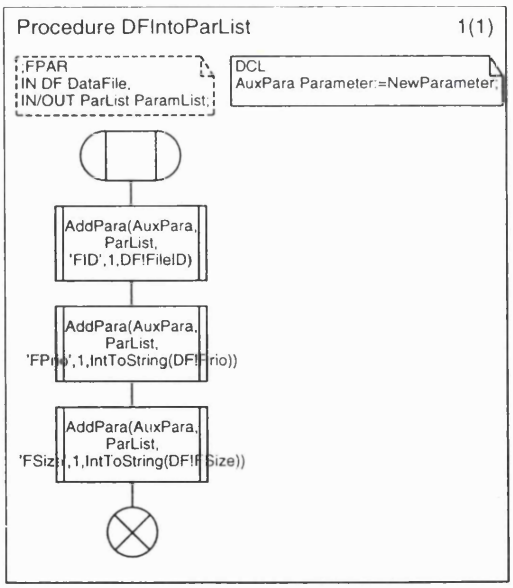

Procedure RemoveRemoteUser
1(1)
DCL
NullSP4 SP4 :=NewSP4
DiagMess
else
'UReg'
TempParamList :=
Clear(TempParamList)
AddPara(TempParam,
TempParamList,
'UserID',1,DPar1)
AddPara(TempParam,
TempParamList,
'Dest',1,IntToString(DPar2))
AddPara(TempParam,
TempParamList,
'Orig',1,ThisNodeStr)
MakeSP3(TempSP3,
'URemReq',3,
TempParamList,NullSP4)
SP32(TempSP3)
VIA Internal

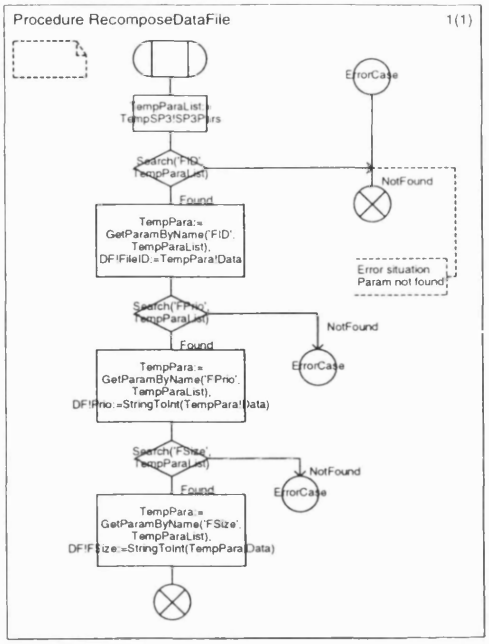

Process LowerApp
Initialisations(3)
DCL
ThisNode Integer
GetAndPassNodeID
NodeID(ThisNode)
Registered_
Users
Registered_
Files
NodeID(ThisNode)
VIA Control
NodeID(ThisNode)
VIA FilesDown
NodeID(ThisNode)
VIA UsersDown
ReadInput
EndSim
EndSim
VIA D

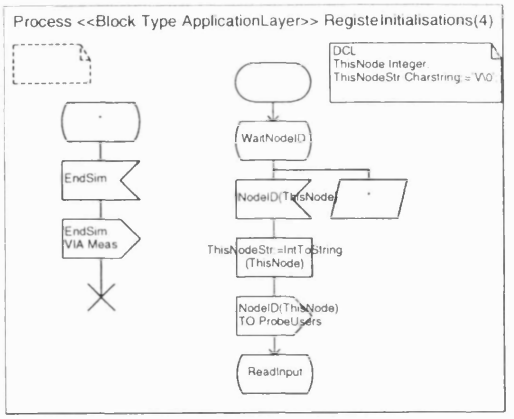

Process LowerApp
ProcessSP32
ReadInput
SP32(TempSP3)
SP32(TempSP3)
VIA D
Timers need(1) to be set
to time=TOTAL_SIM_TIME
RemoteFileReq
else
True
False
NotFound
Found
RESET(FileT
(AuxFID))
FileLocReq
(AuxFID,AuxInvT)
VIA Internal
SET(AuxTime
FileT(AuxFID))
ReadInput

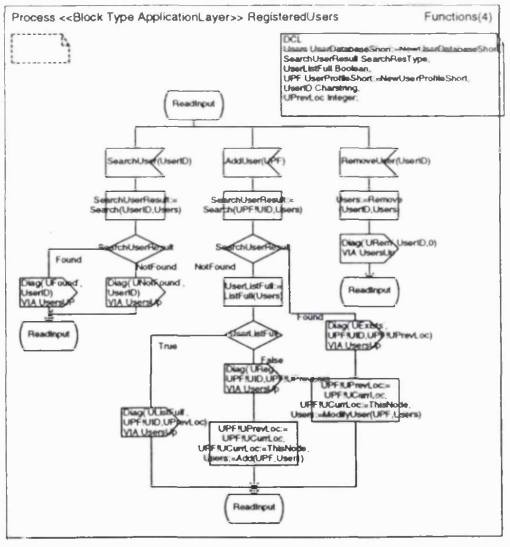
```
DCL
TempParaList ParamList:=NewParamList,
TempPara Parameter:=NewParameter,
AuxPara Parameter:=NewParameter,
AuxFID Charstring,
TempSP3 SP3:=NewSP3,
AuxSP3 SP3:=NewSP3,
IniTime Time,
FinTime Time,
AuxTime Time,
AuxTimeStr Charstring,
AuxIniT Time,
TempExpFileEle ExpectedFileEle:=NewExpectedFileEle,
ExpFileList ExpectedFileList:=NewExpectedFileList;

TIMER
FileT(Charstring);/*FID*/
```

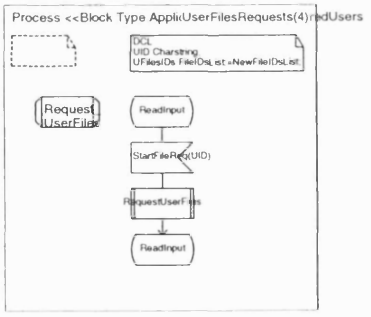


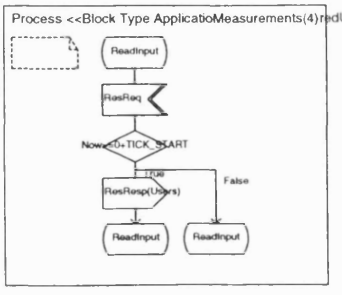


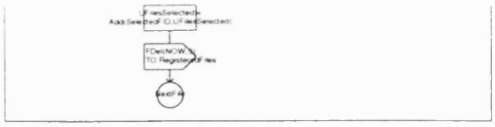

Procedure RecomposeDataFile
1(1)
ErrorCase
TempParaList:=
TempSP3!SP3Pars
Search('FID',
TempParaList)
NotFound
Found
TempPara:=
GetParamByName('FID',
TempParaList),
DF!FileID:=TempPara!Data
Error situation
Param not found
Search('FPrio',
TempParaList)
NotFound
Found
ErrorCase
TempPara:=
GetParamByName('FPrio',
TempParaList),
DF!Prio:=StringToInt(TempPara!Data)
Search('FSize',
TempParaList)
NotFound
Found
ErrorCase
TempPara:=
GetParamByName('FSize',
TempParaList),
DF!FSize:=StringToInt(TempPara!Data)

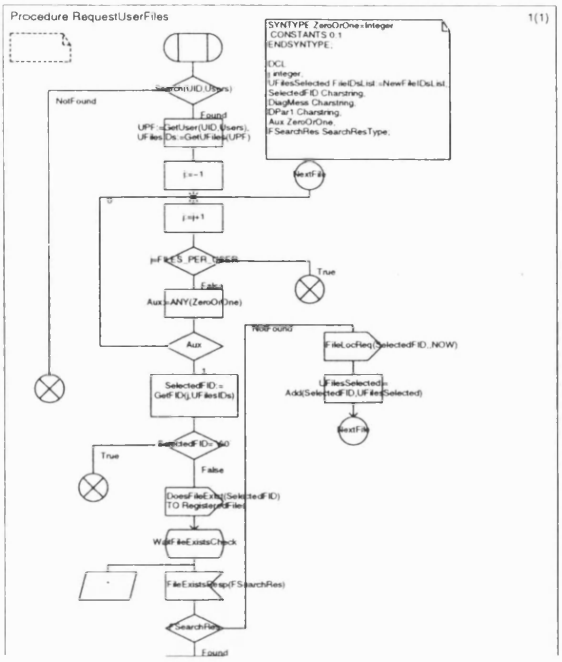

Process <<Block Type ApplicationLayer>> RegisteInitialisations(4)
DCL
ThisNode Integer,
ThisNodeStr Charstring:='V\0'
WaitNodeID
EndSim
NodeID(ThisNode)
EndSim
VIA Meas
ThisNodeStr:=IntToString
(ThisNode)
NodeID(ThisNode)
TO ProbeUsers
ReadInput

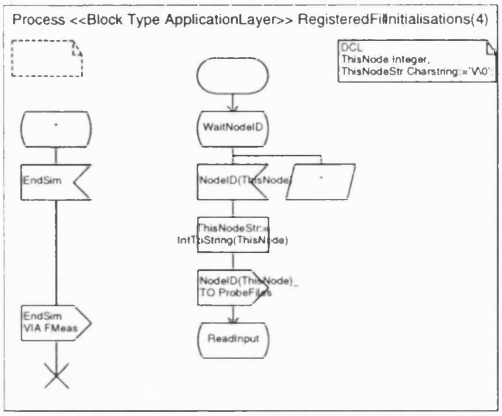

Process <<Block Type ApplicationLayer>> RegisteredUsers
Functions(4)
DCL
SearchUserResult SearchResType,
UserListFull Boolean,
UPF UserProfileShort :=NewUserProfileShort,
UserID Charstring,
!UPrevLoc Integer;
ReadInput
SearchUser(UserID)
AddUser(UPF)
RemoveUser(UserID)
Found
NotFound
NotFound
Found
NotFound
True
False
ReadInput

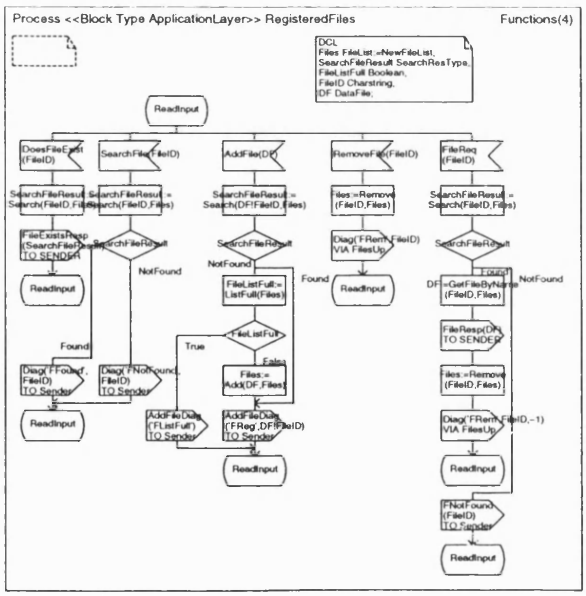

Process <<Block Type ApplicUserFilesRequests(4)redUsers
DCL
UID Charstring,
UFilesIDs FileIDsList :=NewFileIDsList;
Request
!UserFiles
ReadInput
StartFileReq(UID)
RequestUserFiles
ReadInput

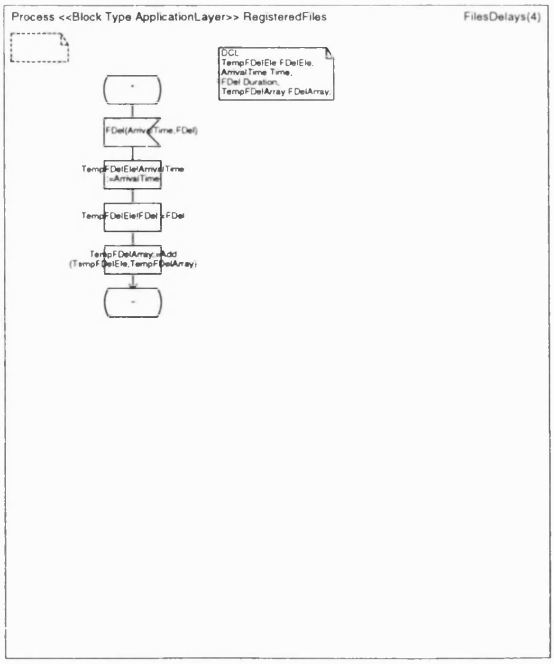

Process <<Block Type ApplicatioMeasurements(4)redUsers
ReadInput
ResReq
TICK_START
True
False
ReadInput
ReadInput

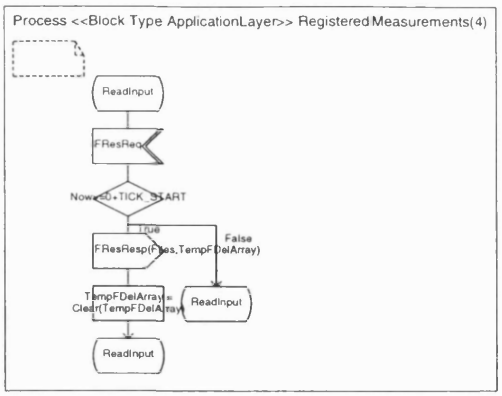


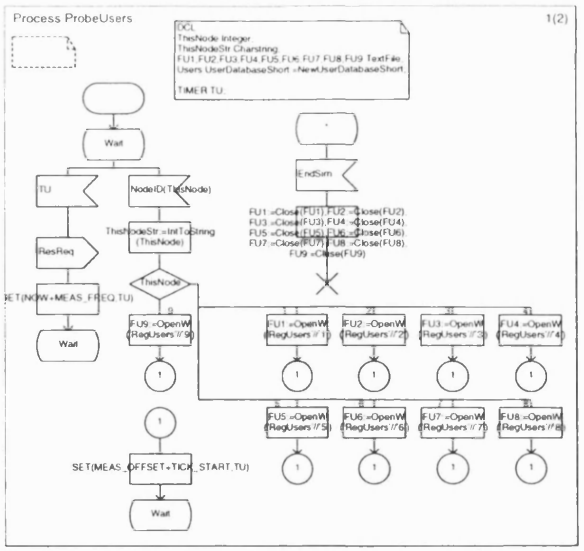

Procedure RequestUserFiles
1(1)
SYNTYPE ZeroOrOne=Integer
CONSTANTS 0,1
ENDSYNTYPE;
DCL
i integer,
UFilesSelected FileIDsList :=NewFileIDsList,
SelectedFID Charstring,
DiagMess Charstring,
DPart Charstring,
Aux ZeroOrOne,
FSearchRes SearchResType;
NextFile
NotFound
Found
i:=-1
i:=i+1
i=FILES_PER_USER
True
False
Aux:=ANY(ZeroOrOne)
Aux
NotFound
FileLocReq(SelectedFID,_NOW)
SelectedFID:=
Get(FID,UFilesIDs)
UFilesSelected:=
Add(SelectedFID,UFilesSelected)
NextFile
SelectedFID=0
True
False
DoesFileExist(SelectedFID)
TO RegisteredFiles
WaitFileExistsCheck
FileExistsResp(FSearchRes)
SearchFiles
Found

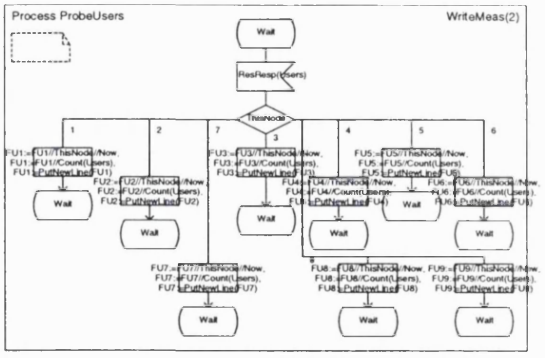

Process <<Block Type ApplicationLayer>> RegisteredFilInitialisations(4)
DCL
ThisNode Integer,
ThisNodeStr Charstring:='V\0';
WaitNodeID
EndSim
NodeID(ThisNode)
ThisNodeStr:=
IntToString(ThisNode)
NodeID(ThisNode)_
TO ProbeFiles
EndSim
VIA FMeas
ReadInput

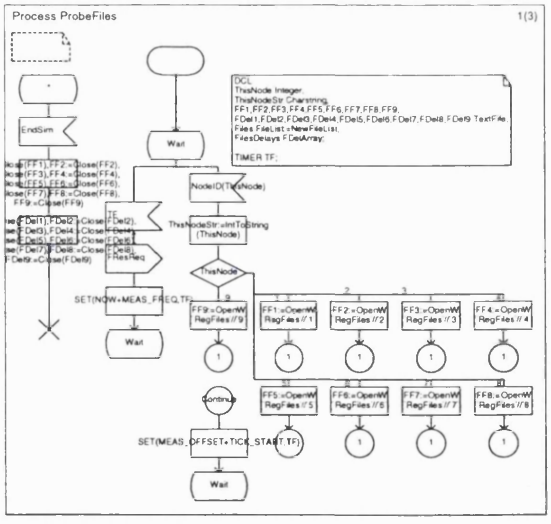

Process <<Block Type ApplicationLayer>> RegisteredFiles
Functions(4)
DCL
Files FileList :=NewFileList,
SearchFileResult SearchResType,
FileListFull Boolean,
FileID Charstring,
DF DataFile;
ReadInput
DoesFileExist
(FileID)
SearchFile(FileID)
AddFile(DF)
RemoveFile(FileID)
FileReq
(FileID)
SearchFileResult :=
Search(FileID,Files)
FileExistsResp
(SearchFileResult)
TO SENDER
NotFound
Found
Diag('FFound',
FileID)
TO Sender
Diag('FNotFound',
FileID)
TO Sender
FileListFull:=
ListFull(Files)
True
False
Files:=
Add(DF,Files)
AddFileDone
(FListFull)
TO Sender
AddFileDone
(FFReq,DF,FileID)
TO Sender
Diag('FRem',FileID)
VIA FilesUp
DF:=GetFileByName
(FileID,Files)
FileResp(DF)
TO SENDER
Files:=Remove
(FileID,Files)
Diag('FRem',FileID,-1)
VIA FilesUp
FNotFound
(FileID)
TO Sender

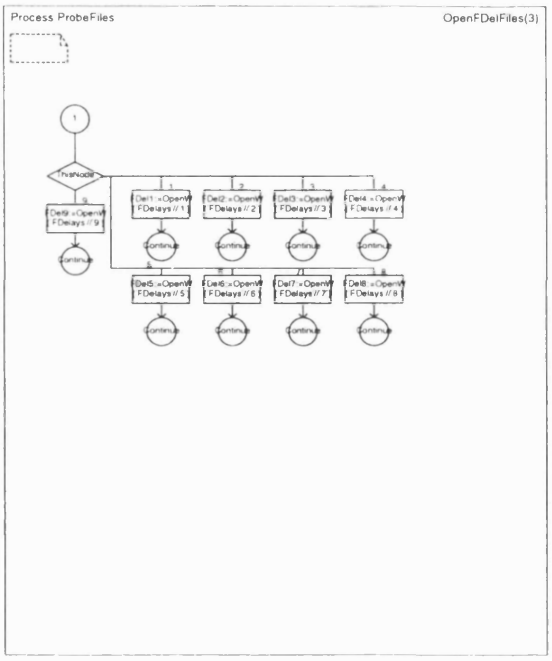

Process <<Block Type ApplicationLayer>> RegisteredFiles
FilesDelays(4)
DCL
TempFDelEle FDelEle,
ArrivalTime Time,
FDel Duration,
TempFDelArray FDelArray;
FDel(ArrivalTime,FDel)
TempFDelEle:=ArrivalTime
TempFDelEle!FDel:=FDel
TempFDelArray:=Add
(TempFDelEle,TempFDelArray);

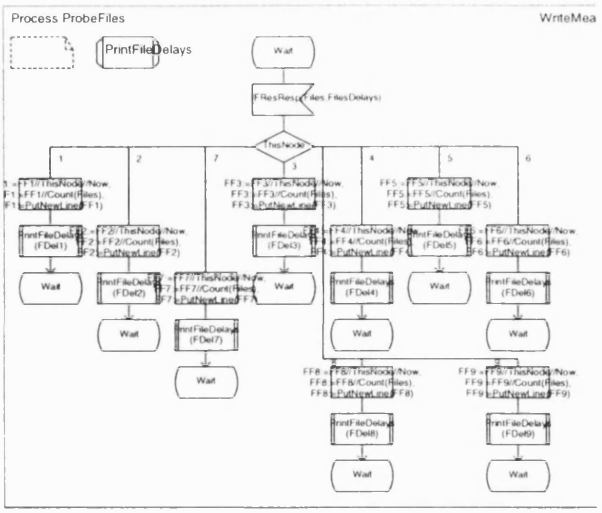

Process <<Block Type ApplicationLayer>> RegisteredMeasurements(4)
ReadInput
FResReq
Now<SD+TICK_START
TRUE
False
FResResp(FFiles,TempFDelArray)
TempFDelArray:=
ClearI(TempFDelArray)
ReadInput
ReadInput

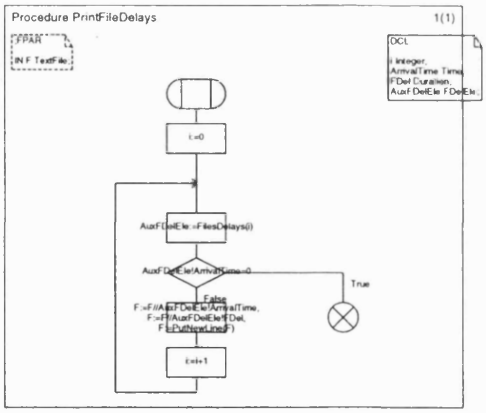

Process ProbeUsers
1(2)
DCL
ThisNode Integer,
ThisNodeStr Charstring,
FU1,FU2,FU3,FU4,FU5,FU6,FU7,FU8,FU9 TextFile,
Users UserDatabaseShort :=NewUserDatabaseShort,
TIMER TU;
Wait
TU
NodeID(ThisNode)
ThisNodeStr:=IntToString
(ThisNode)
ResReq
ThisNode
SET(NOW+MEAS_FREQ,TU)
Wait
FU9:=OpenW
(RegUsers // 9)
FU1:=OpenW
(RegUsers // 1)
FU2:=OpenW
(RegUsers // 2)
FU3:=OpenW
(RegUsers // 3)
FU4:=OpenW
(RegUsers // 4)
FU5:=OpenW
(RegUsers // 5)
FU6:=OpenW
(RegUsers // 6)
FU7:=OpenW
(RegUsers // 7)
FU8:=OpenW
(RegUsers // 8)
SET(MEAS_OFFSET+TICK_START,TU)
Wait
EndSim
FU1:=Close(FU1);FU2:=Close(FU2);
FU3:=Close(FU3);FU4:=Close(FU4);
FU5:=Close(FU5);FU6:=Close(FU6);
FU7:=Close(FU7);FU8:=Close(FU8);
FU9:=Close(FU9)

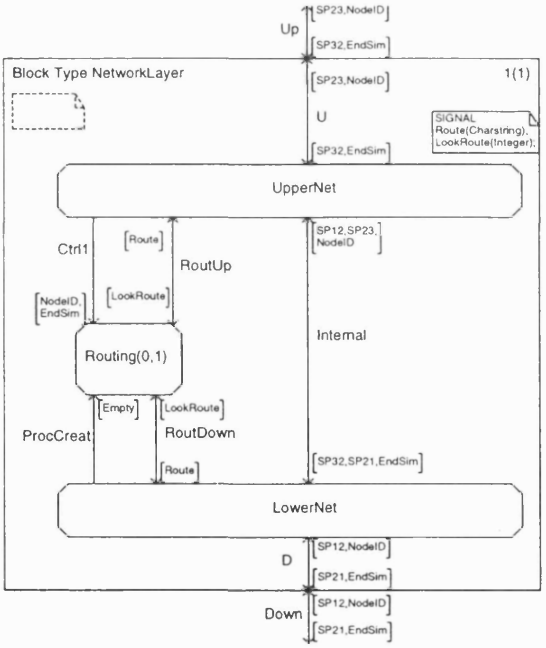

Process ProbeUsers
WriteMeas(2)
Wait
ResResp(Users)
ThisNode
Wait
Wait
Wait
Wait
Wait
Wait
Wait
Wait

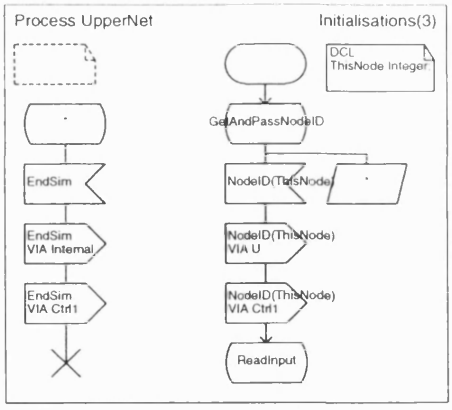

Process ProbeFiles
1(3)
DCL
ThisNode Integer,
ThisNodeStr Charstring,
FF1,FF2,FF3,FF4,FF5,FF6,FF7,FF8,FF9,
FDel1,FDel2,FDel3,FDel4,FDel5,FDel6,FDel7,FDel8,FDel9 TextFile,
Files FileList:=NewFileList,
FilesDelays FDelArray;
TIMER TF;
EndSim
Wait
NodeID(ThisNode)
ThisNodeStr:=IntToString
(ThisNode)
ThisNode
SET(NOW+MEAS_FREQ,TF)
Wait
FF9:=OpenW
(RegFiles // 9)
FF1:=OpenW
(RegFiles // 1)
FF2:=OpenW
(RegFiles // 2)
FF3:=OpenW
(RegFiles // 3)
FF4:=OpenW
(RegFiles // 4)
FF5:=OpenW
(RegFiles // 5)
FF6:=OpenW
(RegFiles // 6)
FF7:=OpenW
(RegFiles // 7)
FF8:=OpenW
(RegFiles // 8)
SET(MEAS_OFFSET+TICK_START,TF)
Wait

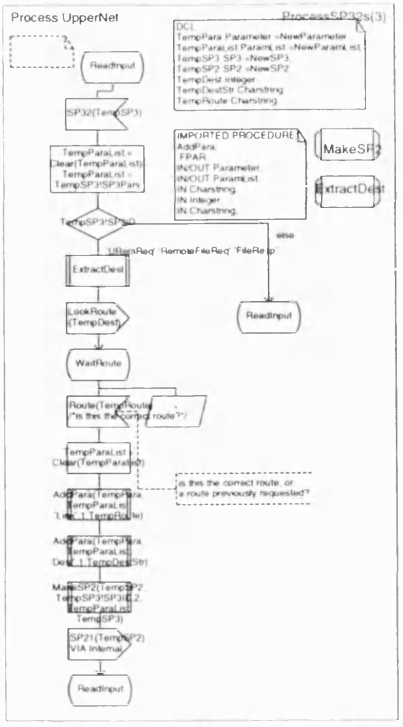

Process ProbeFiles
OpenFDelFiles(3)

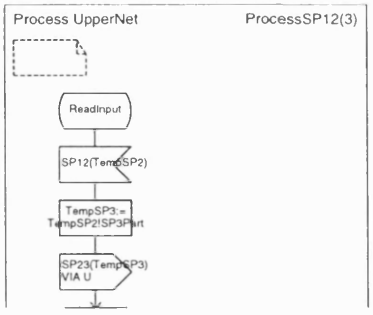

Process ProbeFiles
WrteMea
PrintFileDelays
Wait

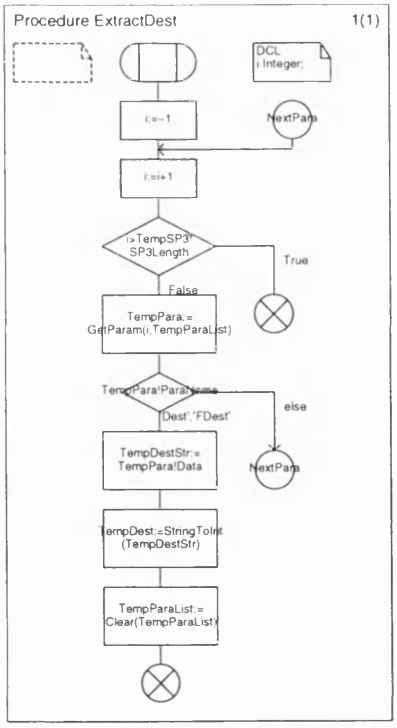

Procedure PrintFileDelays
1(1)
FPAR
INF TextFile
OCL
I=0
True

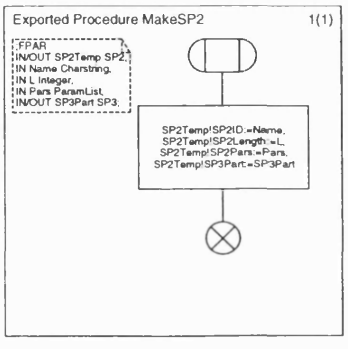

Block Type NetworkLayer
1(1)
Up
U
D
Down
SIGNAL
Route(Charstring),
LookRoute(Integer);
UpperNet
LowerNet
Routing(0,1)
Ctrl1
RoutUp
RoutDown
ProcCreat
Internal
[SP23,NodeID]
[SP32,EndSim]
[Route]
[LookRoute]
[Empty]
[NodeID, EndSim]
[SP12,SP23, NodeID]
[SP32,SP21,EndSim]
[SP12,NodeID]
[SP21,EndSim]

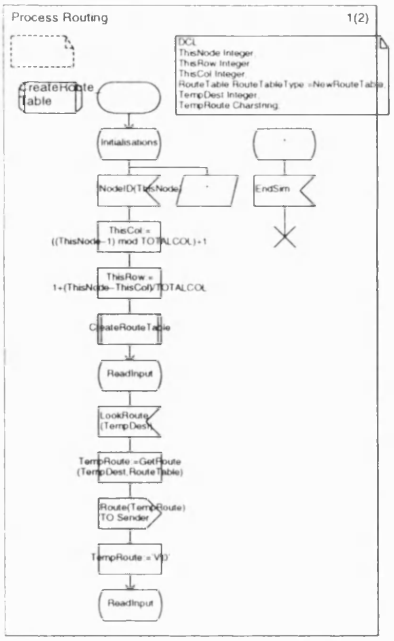

Process UpperNet
Initialisations(3)
DCL
ThisNode Integer;
GetAndPassNodeID
NodeID(ThisNode)
EndSim
EndSim
VIA Internal
NodeID(ThisNode)
VIA U
EndSim
VIA Ctrl1
NodeID(ThisNode)
VIA Ctrl1
ReadInput

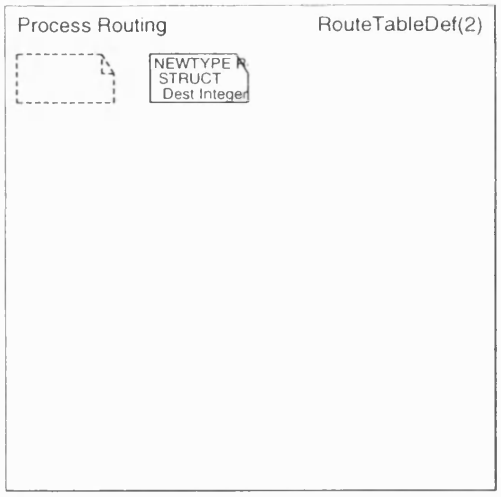

Process UpperNet
ProcessSP32s(3)
DCL
TempPara Parameter =NewParameter
TempParaList ParamList =NewParamList
TempSP3 SP3 =NewSP3
TempSP2 SP2 =NewSP2
TempDest Integer
TempDestStr Charstring
TempRoute Charstring
ReadInput
SP32(TempSP3)
IMPORTED PROCEDURE
AddPara
FPAR
IN/OUT Parameter
IN/OUT ParamList
IN Charstring
IN Integer
IN Charstring
MakeSP2
ExtractDest
else
ExtractDest
ReadInput
LookRoute
(TempDest)
WaitRoute
SP21(TempSP2)
VIA Internal
ReadInput


Process UpperNet
ProcessSP12(3)
ReadInput
SP12(TempSP2)
TempSP3:=
TempSP2!SP3Part
SP23(TempSP3)
VIA U


Procedure ExtractDest
1(1)
DCL
i Integer;
i:=-1
NextPara
i:=i+1
(i>TempSP3!
SP3Length
True
False
TempPara:=
GetParam(i,TempParaList)
else
Dest','FDest'
TempDestStr:=
TempPara!Data
NextPara
TempDest:=StringToInt
(TempDestStr)
TempParaList:=
Clear(TempParaList)

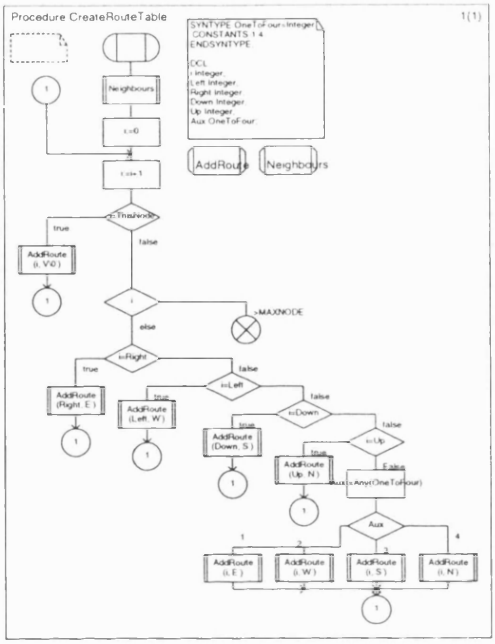

Exported Procedure MakeSP2
1(1)
FPAR
IN/OUT SP2Temp SP2,
IN Name Charstring,
IN L Integer,
IN Pars ParamList,
IN/OUT SP3Part SP3;
SP2Temp!SP2ID:=Name,
SP2Temp!SP2Length:=L,
SP2Temp!SP2Pars:=Pars,
SP2Temp!SP3Part:=SP3Part

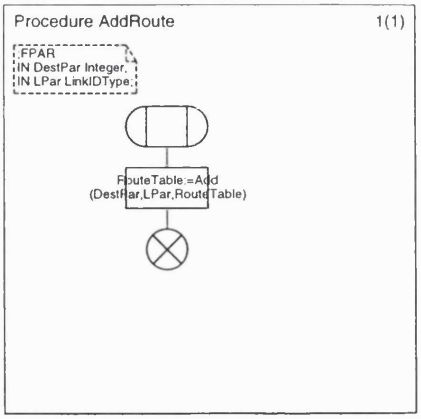


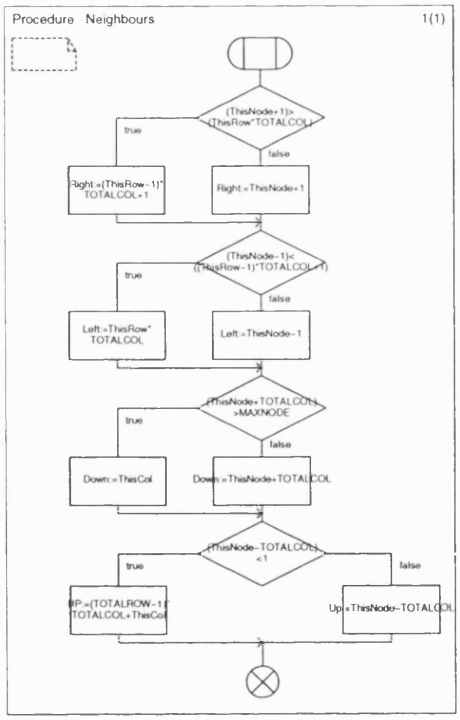


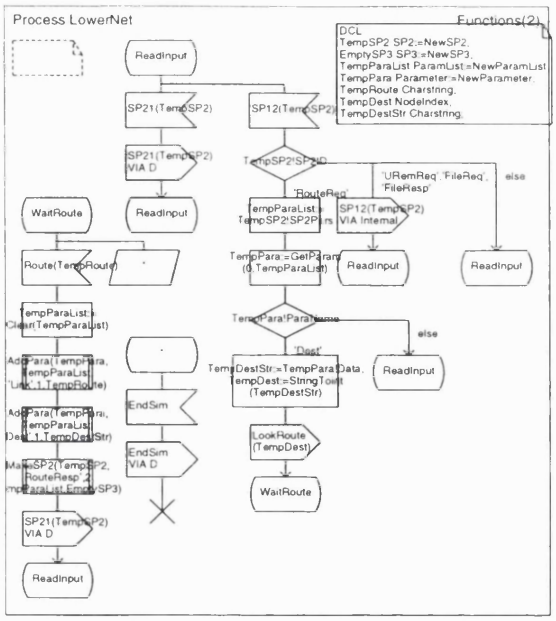

```
NEWTYPE RouteType
 STRUCT
  Dest Integer;
  Link LinkIDType;
ADDING
LITERALS
NewRoute;
/*ADT(B)
#BODY
#ifndef XNOPROTO
extern #(RouteType) #(NewRoute) (void)
#else
extern #(RouteType) #(NewRoute) ()
#endif
{
return yMake_#(RouteType)(-1, "V0");
}
*/
ENDNEWTYPE RouteType;


NEWTYPE RouteTableType
array(NodeIndex,RouteType)
ADDING
LITERALS
NewRouteTable;
OPERATORS

Search:NodeIndex,RouteTableType->SearchResType;
ListFull:RouteTableType->Boolean;
Add:Integer,LinkIDType,RouteTableType->RouteTableType;
Remove:RouteTableType,NodeIndex->RouteTableType;
GetRoute:NodeIndex,RouteTableType->LinkIDType;
Clear:RouteTableType->RouteTableType;
ChangeRoute: NodeIndex,LinkIDType,RouteTableType->RouteTableType;

/*#ADT(B)
#BODY
#ifndef XNOPROTO
extern #(RouteTableType) #(NewRouteTable) (void)
#else
extern #(RouteTableType) #(NewRouteTable) ()
#endif
{
return (yMake_#(RouteTableType)(yMake_#(RouteType)(-1, "V0")));
}

#ifndef XNOPROTO
extern #(SearchResType) #(Search) (#(NodeIndex)Param1,#(RouteTableType) Param2)
#else
extern #(SearchResType) #(Search) (Param1,Param2)
#(NodeIndex) Param1;
#(RouteTableType) Param2;
#endif
{
  #(NodeIndex) cnt;
  for(cnt=0;cnt<#(MAXNODE);cnt++){
    if(Param1==Param2.A[cnt].Dest) return #(Found);
  }
  return #(NotFound);
}

#ifndef XNOPROTO
extern SDL_Boolean #(ListFull) (#(RouteTableType) Param1)
#else
extern SDL_Boolean #(ListFull) (Param1)
#(RouteTableType) Param1;
#endif
{
  if(Param1.A[#(MAXNODE)-1].Dest==-1) return SDL_False;
  else return SDL_True;
}


#ifndef XNOPROTO
extern #(RouteTableType) #(Add) (SDL_Integer Param1,#(LinkIDType)Param2,
#(RouteTableType) Param3)
#else
extern #(RouteTableType) #(Add)(Param1,Param2,Param3)
SDL_Integer Param1;
#(LinkIDType) Param2;
#(RouteTableType) Param3;
#endif
{
  #(NodeIndex) cnt;
  for(cnt=0;cnt<#(MAXNODE);cnt++)
    if(Param3.A[cnt].Dest==-1){
      Param3.A[cnt].Dest=Param1;
      Param3.A[cnt].Link=Param2;
      return Param3;
    }
  return Param3;
}

#ifndef XNOPROTO
extern #(RouteTableType) #(Remove) (#(NodeIndex) Param1,#(RouteTableType) Param2)
#else
extern #(RouteTableType) #(Remove)(Param1,Param2)
#(NodeIndex) Param1;
#(RouteTableType) Param2;
#endif
{
  #(NodeIndex) cnt;
  for(cnt=0;cnt<#(MAXNODE);cnt++){
    if(Param1==Param2.A[cnt].Dest) {
      Param2.A[cnt].Dest = -1;
      return Param2;
    }
  }
  return Param2;
}

#ifndef XNOPROTO
extern #(LinkIDType) #(GetRoute) (#(NodeIndex) Param1,#(RouteTableType) Param2)
#else
extern #(LinkIDType) #(GetRoute)(Param1,Param2)
#(NodeIndex) Param1;
#(RouteTableType) Param2;
#endif
{
  #(NodeIndex) cnt;
  for(cnt=0;cnt<#(MAXNODE);cnt++){
  if(Param2.A[cnt].Dest==Param1)
    return Param2.A[cnt].Link;
  }
}

#ifndef XNOPROTO
extern #(RouteTableType) (Clear) (#(RouteTableType) Param1)
#else
```

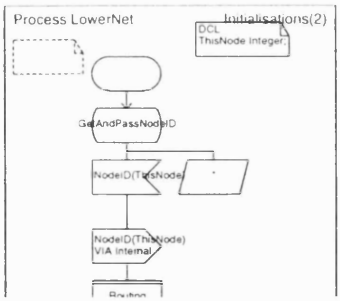
```
extern #(RouteTableType) #(Clear) (Param1)
#(RouteTableType) Param1;
#endif
{
  #(NodeIndex) cnt;
  for(cnt=0;cnt<#(MAXNODE);cnt++){
    Param1.A[cnt].Dest=-1;
    Param1.A[cnt].Link="V\0";
  }
  return Param1;
}

#ifndef XNOPROTO
extern #(RouteTableType) #(ChangeRoute) (#(NodeIndex) Param1,
#(LinkIDType) Param2, #(RouteTableType) Param3)
#else
extern #(RouteTableType) #(ChangeRoute)(Param1,Param2,Param3)
#(NodeIndex) Param1;
#(LinkIDType) Param2;
#(RouteTableType) Param3;
#endif
{
  #(NodeIndex) cnt;
  for(cnt=0;cnt<#(MAXNODE);cnt++){
    if(Param1==Param3.A[cnt].Dest) {
      Param3.A[cnt].Link = Param2;
      return Param3;
    }
  }
  return Param3;
}


*/
ENDNEWTYPE RouteTableType;
```

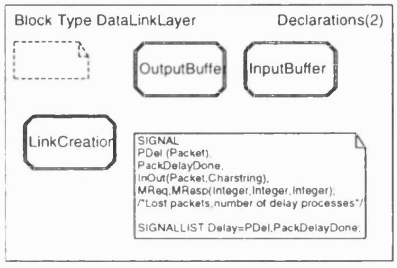


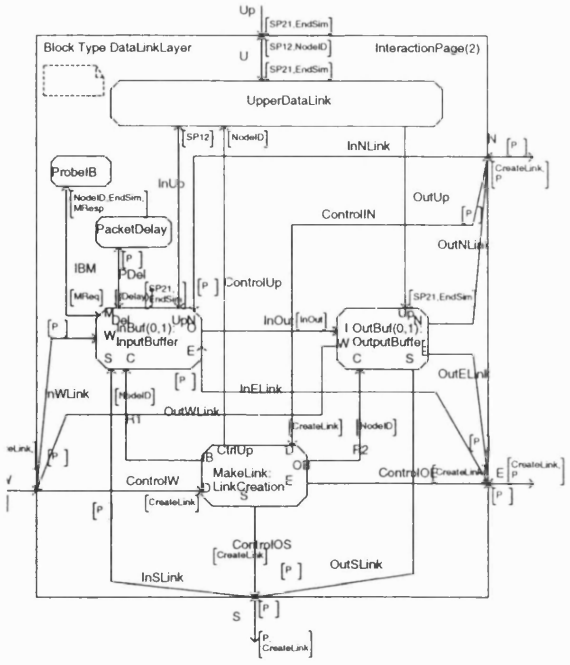


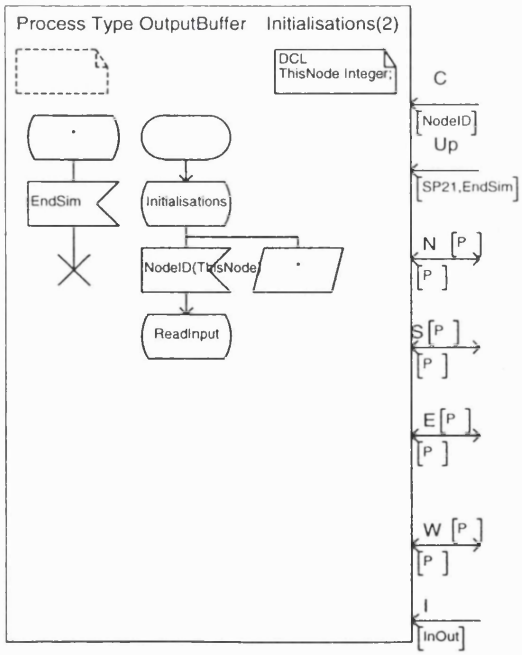


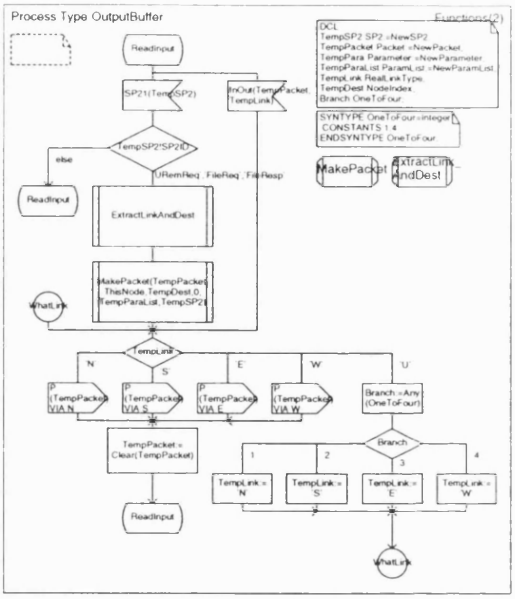

Process LowerNet
Functions(2)
ReadInput
DCL
TempSP2 SP2:=NewSP2,
EmptySP3 SP3:=NewSP3,
TempParaList ParamList:=NewParamList
TempPara Parameter:=NewParameter,
TempRoute Charstring,
TempDest NodeIndex,
TempDestStr Charstring;
SP21(TempSP2)
SP12(TempSP2)
SP21(TempSP2)
VIA D
TempSP2!SP2ID
'URemReq','FileReq',
'FileResp'
else
'RouteReq'
WaitRoute
ReadInput
TempParaList:=
TempSP2!SP2Paras
SP12(TempSP2)
VIA Internal
Route(TempRoute)
TempPara:=GetParam
(0,TempParaList)
ReadInput
ReadInput
TempPara!ParaName
'Dest'
else
TempDestStr:=TempPara!Data,
TempDest:=StringToInt
(TempDestStr)
ReadInput
EndSim
EndSim
VIA D
LookRoute
(TempDest)
WaitRoute
SP21(TempSP2)
VIA D
ReadInput

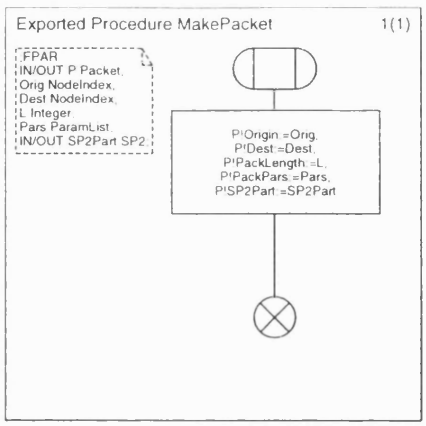

Process LowerNet
Initialisations(2)
DCL
ThisNode Integer;
GetAndPassNodeID
NodeID(ThisNode)
NodeID(ThisNode)
VIA Internal

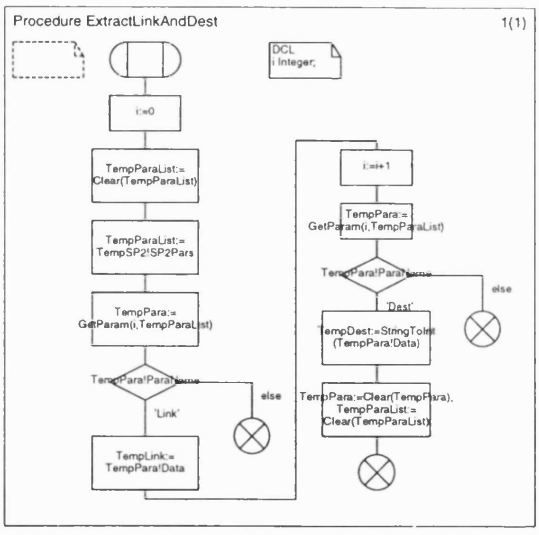

Block Type DataLinkLayer
Declarations(2)
OutputBuffer
InputBuffer
LinkCreation
SIGNAL
PDel (Packet),
PackDelayDone,
InOut(Packet,Charstring),
MReq,MResp(Integer,Integer,Integer);
/*Lost packets,number of delay processes*/
SIGNALLIST Delay=PDel,PackDelayDone;

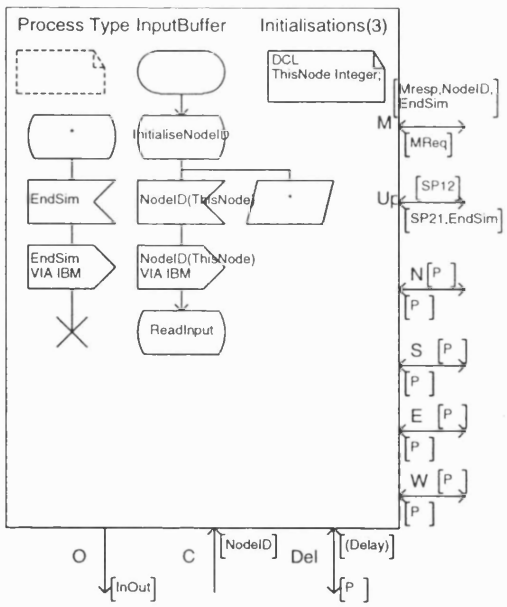

Up
[SP21,EndSim]
Block Type DataLinkLayer
[SP12,NodeID]
InteractionPage(2)
U
[SP21,EndSim]
UpperDataLink
[SP12]
[NodeID]
InNLink
N
[P]
ProbeIB
InUp
[CreateLink,P]
OutUp
[NodeID,EndSim,MResp]
ControlIN
PacketDelay
OutNLink
IBM
PDel
[MReq]
[P]
ControlUp
[SP21,EndSim]
InOut
[InOut]
Up
InBuf(0,1)
InputBuffer
OutBuf(0,1)
OutputBuffer
OutELink
InELink
InWLink
[NodeID]
OutWLink
[CreateLink]
[NodeID]
CtrlUp
MakeLink:
LinkCreation
ControlW
ControlOS
[CreateLink]
InSLink
OutSLink
[P]

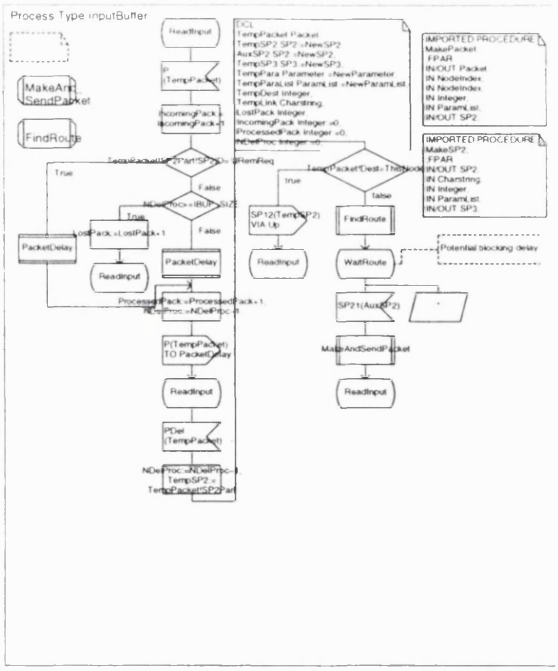

Process Type OutputBuffer
Initialisations(2)
DCL
ThisNode Integer;
EndSim
Initialisations
NodeID(ThisNode)
ReadInput
C
[NodeID]
Up
[SP21,EndSim]
N
[P]
[P]
S
[P]
[P]
E
[P]
[P]
W
[P]
[P]
I
[InOut]

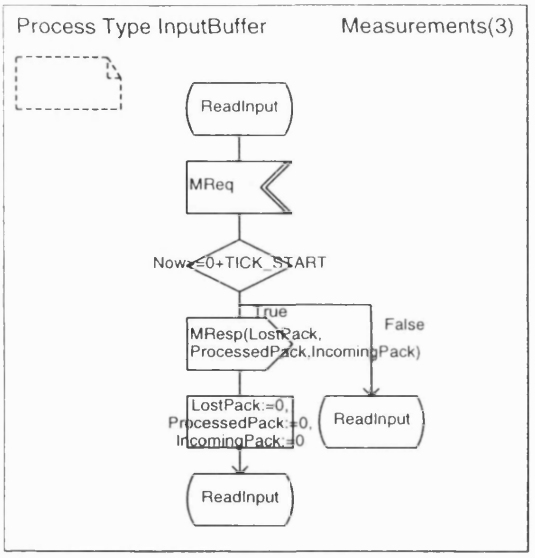

Process Type OutputBuffer
Functions(2)
ReadInput
SP21(TempSP2)
InOut(TempPacket, TempLink)
TempSP2!SP2ID
else
URemReq, FileReq, FileResp
ReadInput
ExtractLinkAndDest
MakePacket(TempPacket, ThisNode, TempDest.0, TempParaList, TempSP2)
WhatLink
TempLink
N
S
E
W
U
P (TempPacket VIA N)
P (TempPacket VIA S)
P (TempPacket VIA E)
P (TempPacket VIA W)
Branch:=Any (OneToFour)
TempPacket:= Clear(TempPacket)
Branch
1
2
3
4
TempLink:= N
TempLink:= S
TempLink:= E
TempLink:= W
ReadInput
WhatLink
DCL
TempSP2 SP2 :=NewSP2,
TempPacket Packet :=NewPacket,
TempPara Parameter :=NewParameter,
TempParaList ParamList :=NewParamList,
TempLink RealLinkType,
TempDest NodeIndex,
Branch OneToFour;
SYNTYPE OneToFour=Integer
CONSTANTS 1:4
ENDSYNTYPE OneToFour
MakePacket
ExtractLinkAndDest

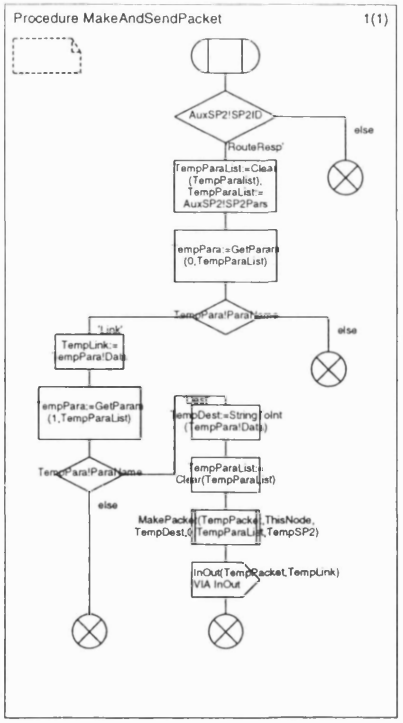

Exported Procedure MakePacket
1(1)
FPAR
IN/OUT P Packet,
Orig NodeIndex,
Dest NodeIndex,
L Integer,
Pars ParamList,
IN/OUT SP2Part SP2,
P!Origin :=Orig,
P!Dest :=Dest,
P!PackLength :=L,
P!PackPars :=Pars,
P!SP2Part :=SP2Part

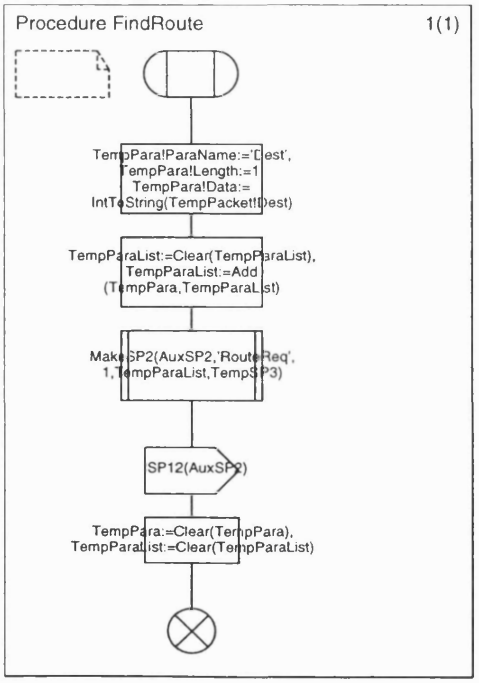

Procedure ExtractLinkAndDest
1(1)
DCL
i Integer;
i:=0
TempParaList:= Clear(TempParaList)
TempParaList:= TempSP2!SP2Pars
TempPara:= GetParam(i,TempParaList)
TempPara!ParaName
'Link'
else
TempLink:= TempPara!Data
i:=i+1
TempPara:= GetParam(i,TempParaList)
TempPara!ParaName
'Dest'
else
TempDest:=StringToInt (TempPara!Data)
TempPara:=Clear(TempPara), TempParaList:= Clear(TempParaList)

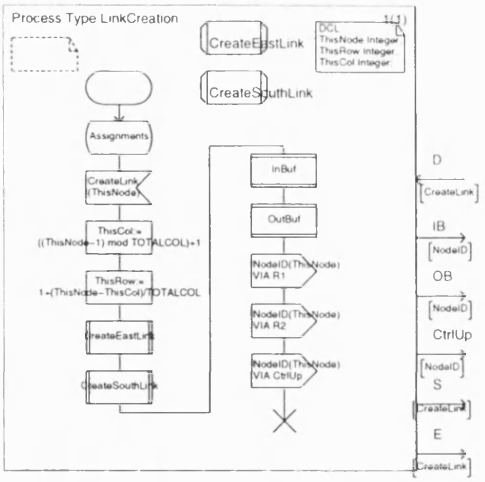

Process Type InputBuffer
Initialisations(3)
DCL
ThisNode Integer;
InitialiseNodeID
EndSim
NodeID(ThisNode)
EndSim VIA IBM
NodeID(ThisNode) VIA IBM
ReadInput
Mresp,NodeID, EndSim
M
MReq
U
SP12
SP21,EndSim
N [P]
[P]
S [P]
[P]
E [P]
[P]
W [P]
[P]
O
[InOut]
C
[NodeID]
Del
(Delay)
[P]

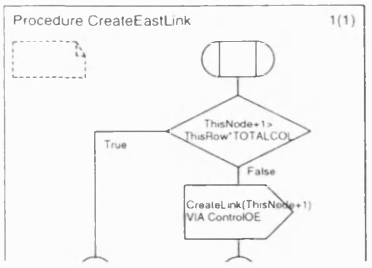

Process Type InputBuffer
ReadInput
MakeAnd SendPacket
FindRoute
PacketDelay
ReadInput
PacketDelay
ReadInput
ReadInput
P(TempPacket) TO PacketDelay
ReadInput
PDel (TempPacket)
SP12(TempSP2) VIA Up
ReadInput
FindRoute
WaitRoute
SP21(AuxSP2)
MakeAndSendPacket
ReadInput
Potential blocking delay

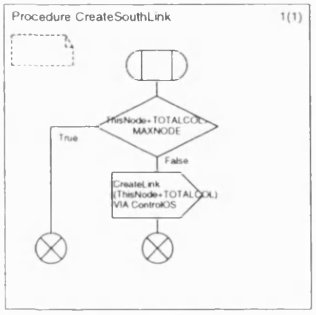

Process Type InputBuffer
Measurements(3)
ReadInput
MReq
Now:=0+TICK_START
True
False
MResp(LostPack, ProcessedPack,IncomingPack)
LostPack:=0, ProcessedPack:=0, IncomingPack:=0
ReadInput
ReadInput

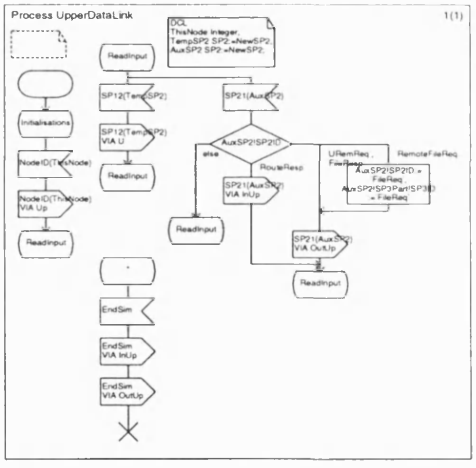

Procedure MakeAndSendPacket
1(1)
AuxSP2!SP2ID
else
RouteResp'
TempParaList:=Clear (TempParaList), TempParaList:= AuxSP2!SP2Para
TempPara:=GetPara (0,TempParaList)
'Link'
else
TempLink:= TempPara!Data
TempPara:=GetPara (1,TempParaList)
'Dest'
TempDest:=String(Point (TempPara!Data))
TempPara!ParaName
TempParaList:= Clear(TempParaList)
else
MakePacket(TempPacket,ThisNode, TempDest,0,TempParaList,TempSP2)
InOut(TempPacket,TempLink) VIA InOut

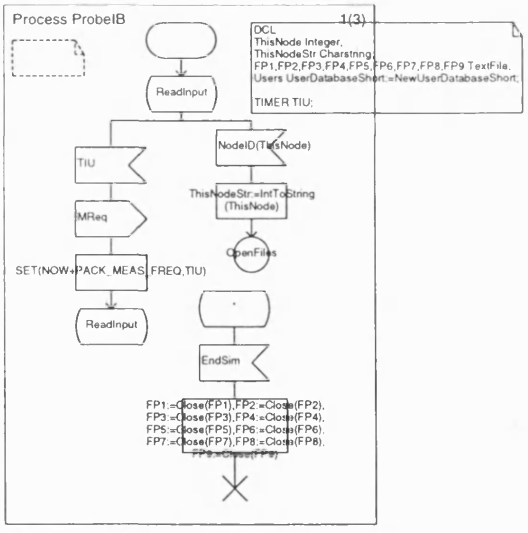

Procedure FindRoute
1(1)
TempPara!ParaName:='Dest', TempPara!Length:=1 TempPara!Data:= IntToString(TempPacket!Dest)
TempParaList:=Clear(TempParaList), TempParaList:=Add (TempPara,TempParaList)
MakeSP2(AuxSP2,'RouteReq', 1,TempParaList,TempSP3)
SP12(AuxSP2)
TempPara:=Clear(TempPara), TempParaList:=Clear(TempParaList)

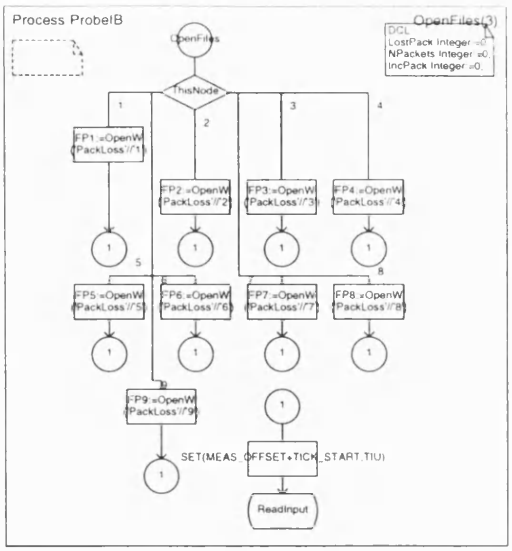

Process Type LinkCreation
1(1)
CreateEastLink
CreateSouthLink
DCL
ThisNode Integer,
ThisRow Integer,
ThisCol Integer;
Assignments
CreateLink
(ThisNode)
ThisCol:=
((ThisNode-1) mod TOTALCOL)+1
ThisRow:=
1+(ThisNode-ThisCol)/TOTALCOL
CreateEastLink
CreateSouthLink
InBuf
OutBuf
NodeID(ThisNode)
VIA R1
NodeID(ThisNode)
VIA R2
NodeID(ThisNode)
VIA CtrlUp
D
CreateLink
IB
NodeID
OB
NodeID
CtrlUp
NodeID
S
CreateLink
E
CreateLink

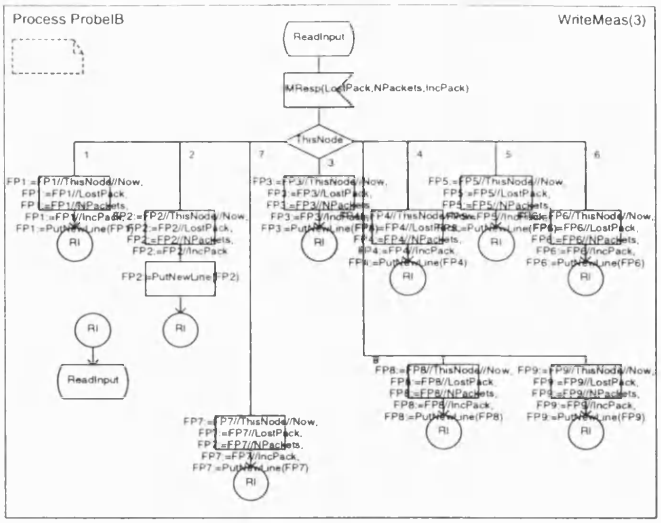

Procedure CreateEastLink
1(1)
ThisNode+1>
ThisRow*TOTALCOL
True
False
CreateLink(ThisNode+1)
VIA ControlOE

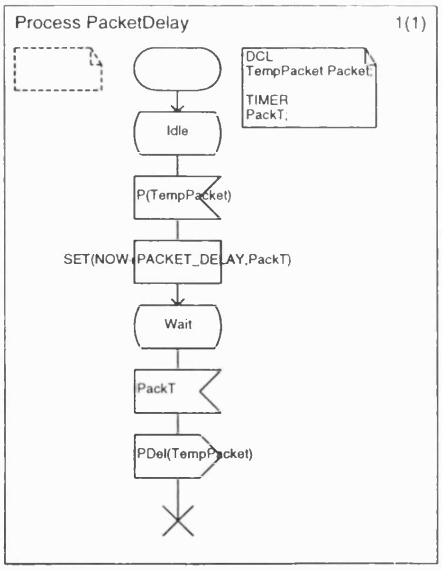

Procedure CreateSouthLink
1(1)
ThisNode+TOTALCOL>
MAXNODE
True
False
CreateLink
(ThisNode+TOTALCOL)
VIA ControlOS

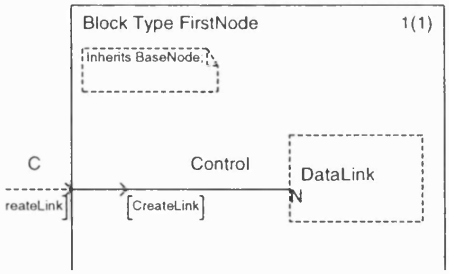

Process UpperDataLink
1(1)
DCL
ThisNode Integer,
TempSP2,SP2:=NewSP2,
AuxSP2:SP2:=NewSP2;
ReadInput
Initialisations
NodeID(ThisNode)
NodeID(ThisNode)
VIA Up
ReadInput
SP12(TempSP2)
SP12(TempSP2)
VIA U
ReadInput
SP21(AuxSP2)
AuxSP2!SP2ID
else
RouteResp
UItemReq
FileReq
RemoteFileReq
AuxSP2!SP2ID:=
FileReq;
AuxSP2!SP2Part!SP3ID
:= FileReq
SP21(AuxSP2)
VIA InUp
ReadInput
SP21(AuxSP2)
VIA OutUp
ReadInput
EndSim
EndSim
VIA InUp
EndSim
VIA OutUp

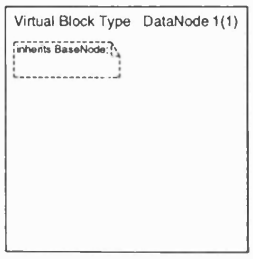

Process ProbeIB
1(3)
DCL
ThisNode Integer,
ThisNodeStr Charstring,
FP1,FP2,FP3,FP4,FP5,FP6,FP7,FP8,FP9 TextFile,
Users UserDatabaseShort:=NewUserDatabaseShort;
TIMER TIU;
ReadInput
TIU
MReq
SET(NOW+PACK_MEAS_FREQ,TIU)
ReadInput
NodeID(ThisNode)
ThisNodeStr:=IntToString
(ThisNode)
OpenFiles
EndSim
FP1:=Close(FP1),FP2:=Close(FP2),
FP3:=Close(FP3),FP4:=Close(FP4),
FP5:=Close(FP5),FP6:=Close(FP6),
FP7:=Close(FP7),FP8:=Close(FP8),
FP9:=Close(FP9)

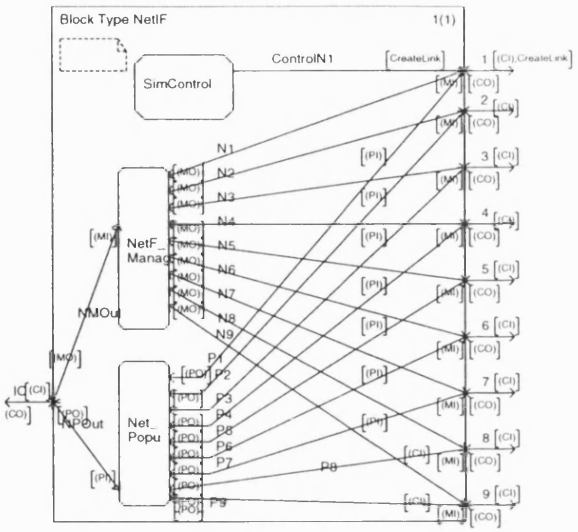

Process ProbeIB
OpenFiles(3)
OpenFiles
ThisNode
SET(MEAS_OFFSET+TICK_START,T1U)
ReadInput

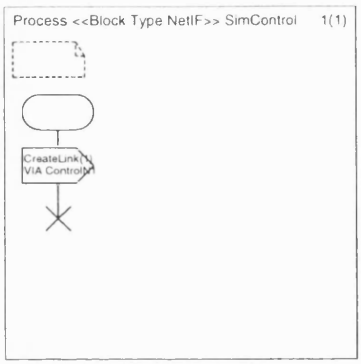

Process ProbeIB
WriteMeas(3)
ReadInput
MResp(LostPack,NPackets,IncPack)
ThisNode
ReadInput

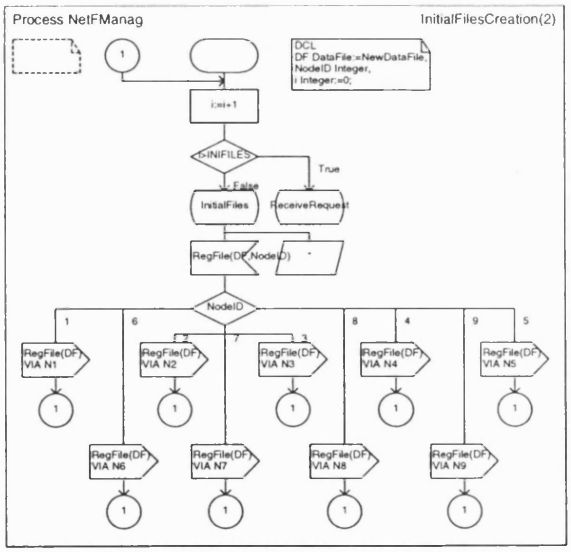

Process PacketDelay
1(1)
DCL
TempPacket Packet;
TIMER
PackT;
Idle
P(TempPacket)
SET(NOW+PACKET_DELAY,PackT)
Wait
PackT
PDel(TempPacket)

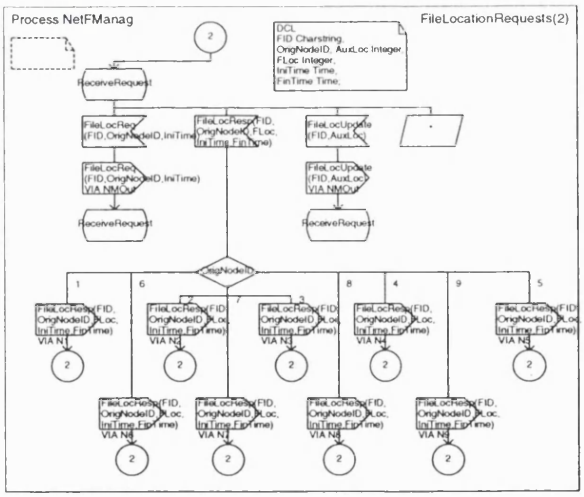

Block Type FirstNode
1(1)
Inherits BaseNode;
C
Control
DataLink
reateLink
CreateLink

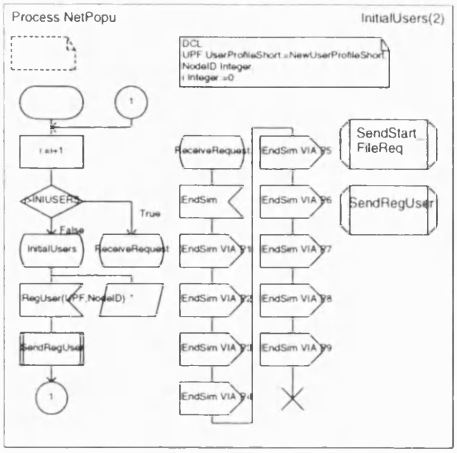

Virtual Block Type DataNode 1(1)
Inherits BaseNode;

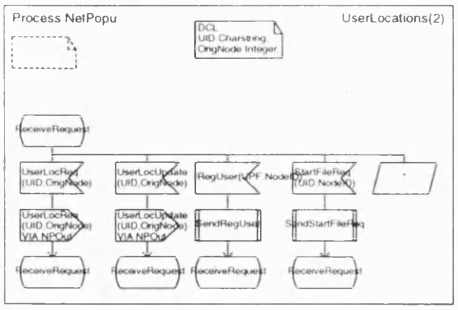

Block Type NetIF
1(1)
SimControl
ControlN1
CreateLink
NetF_
Manag
Net_
Popu
NMOut
NPOut
N1
N2
N3
N4
N5
N6
N7
N8
N9
P1
P2
P3
P4
P5
P6
P7
P8
P9

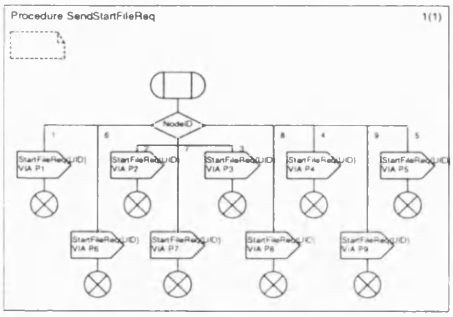

Process <<Block Type NetIF>> SimControl
1(1)
CreateLink(1)
VIA ControlN1

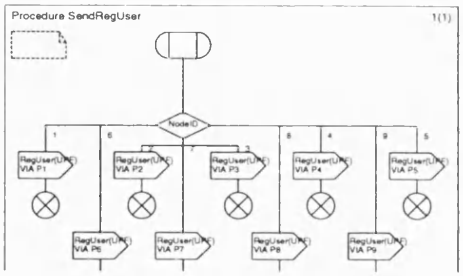

Process NetFManag
InitialFilesCreation(2)
DCL
DF DataFile:=NewDataFile,
NodeID Integer,
I Integer:=0;
I:=I+1
I<NIFILES
True
False
InitialFiles
ReceiveRequest
RegFile(DF,NodeID)
NodeID
RegFile(DF)
VIA N1
RegFile(DF)
VIA N2
RegFile(DF)
VIA N3
RegFile(DF)
VIA N4
RegFile(DF)
VIA N5
RegFile(DF)
VIA N6
RegFile(DF)
VIA N7
RegFile(DF)
VIA N8
RegFile(DF)
VIA N9

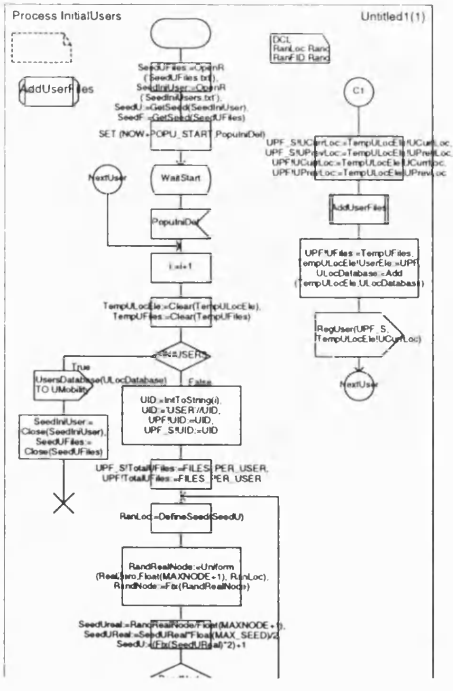

Process NetFManag
FileLocationRequests(2)
DCL
FID Charstring,
OrigNodeID, AuxLoc Integer,
FLoc Integer,
IniTime Time,
FinTime Time;
ReceiveRequest
FileLocReq
(FID,OrigNodeID,IniTime)
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
FileLocUpdate
(FID,AuxLoc)
FileLocReq
(FID,OrigNodeID,IniTime)
VIA NMOut
FileLocUpdate
(FID,AuxLoc)
VIA NMOut
ReceiveRequest
ReceiveRequest
OrigNodeID
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N1
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N2
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N3
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N4
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N5
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N6
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N7
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N8
FileLocResp(FID,
OrigNodeID,FLoc,
IniTime,FinTime)
VIA N9

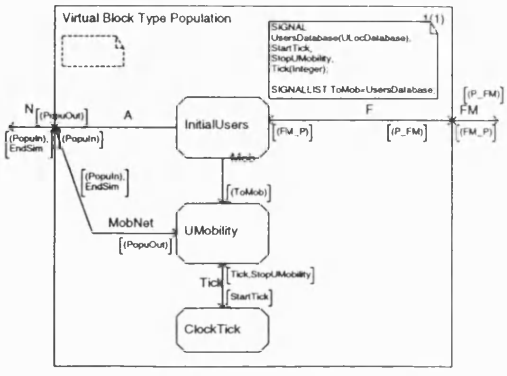

Process NetPopu
InitialUsers(2)
DCL
UPF_UserProfileShort :=NewUserProfileShort,
NodeID Integer,
i Integer :=0;
i:=i+1
<<NIUSERS>>
True
False
InitialUsers
ReceiveRequest
RegUser(UPF_NodeID) *
SendRegUser
EndSim VIA P5
EndSim VIA P6
EndSim VIA P7
EndSim VIA P8
EndSim VIA P9
SendStart_
FileReq
SendRegUser

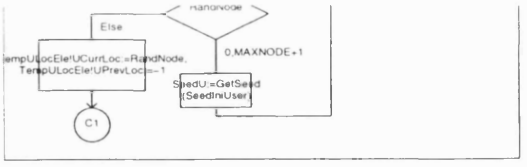

Process NetPopu
UserLocations(2)
DCL
UID Charstring,
OrigNode Integer
ReceiveRequest
UserLocReq
(UID,OrigNode)
UserLocUpdate
(UID,OrigNode)
RegUser(UPF_NodeID)
StartFileReq
(UID,NodeID)
UserLocReq
(UID,OrigNode)
VIA NPOut
UserLocUpdate
(UID,OrigNode)
VIA NPOut
SendRegUser
SendStartFileReq
ReceiveRequest

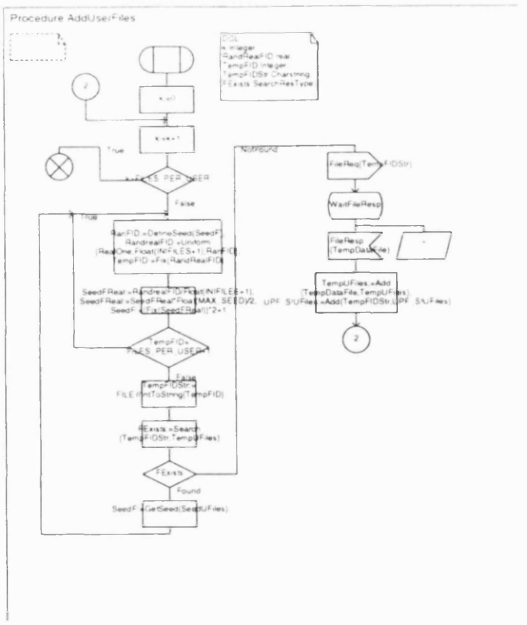

Procedure SendStartFileReq
1(1)
NodeID
StartFileReq(UID)
VIA P1
StartFileReq(UID)
VIA P2
StartFileReq(UID)
VIA P3
StartFileReq(UID)
VIA P4
StartFileReq(UID)
VIA P5
StartFileReq(UID)
VIA P6
StartFileReq(UID)
VIA P7
StartFileReq(UID)
VIA P8
StartFileReq(UID)
VIA P9


Procedure SendRegUser
1(1)
NodeID
RegUser(UID)
VIA P1
RegUser(UID)
VIA P2
RegUser(UID)
VIA P3
RegUser(UID)
VIA P4
RegUser(UID)
VIA P5
RegUser(UID)
VIA P6
RegUser(UID)
VIA P7
RegUser(UID)
VIA P8
RegUser(UID)
VIA P9

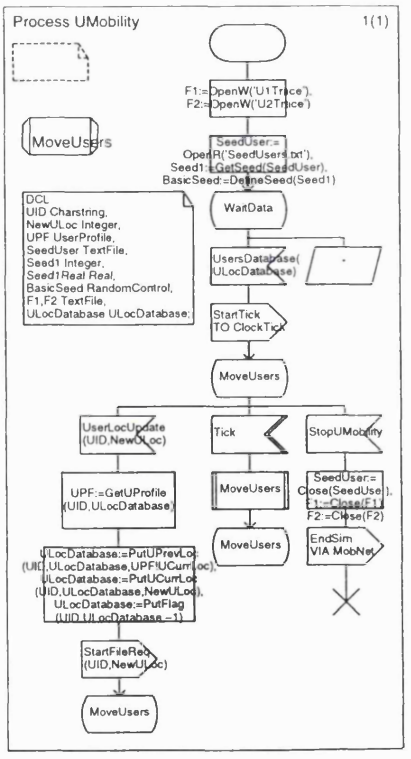

Process InitialUsers
Untitled1(1)
DCL
RanLoc Rand,
RanFID Rand;
AddUserFiles
SeedUFiles :=OpenR
(SeedUFiles.txt),
SeedInUser :=OpenR
(SeedInUsers.txt),
SeedU :=GetSeed(SeedInUser),
SeedF :=GetSeed(SeedUFiles)
SET (NOW+POPU_START, PopulnDel)
NextUser
WaitStart
PopulnDel
i:=i+1
TempULocEle :=Clear(TempULocEle),
TempUFiles :=Clear(TempUFiles)
<<NIUSERS>>
True
False
UsersDatabase(ULocDatabase)
TO UMobility
UID :=IntToString(i),
UPF_S!UID :=UID,
UPF!UID :=UID,
UPF_S!UID:=UID
SeedInUser :=
Close(SeedInUser),
SeedUFiles :=
Close(SeedUFiles)
UPF_S!TotalUFiles :=FILES_PER_USER,
UPF!TotalUFiles :=FILES_PER_USER
RanLoc :=DefineSeed(SeedU)
RandRealNode :=Uniform
(RealZero,Float(MAXNODE+1), RanLoc),
RandNode :=Fix(RandRealNode)
C1
AddUserFiles
NextUser

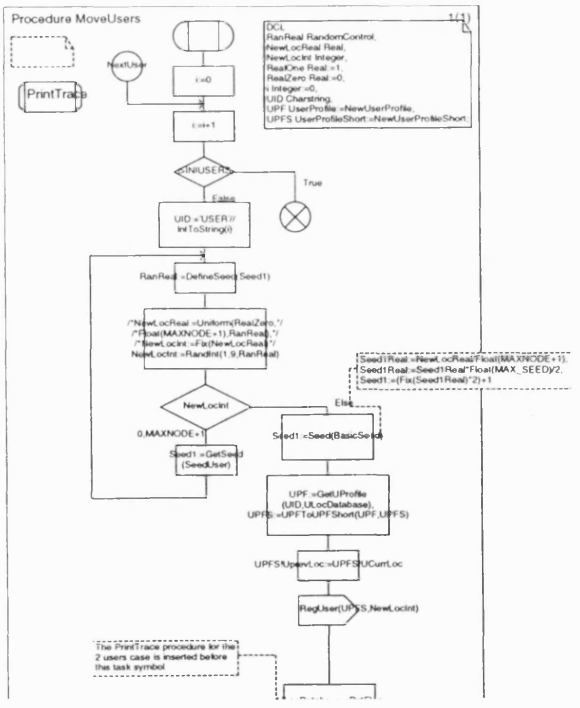

Virtual Block Type Population
1(1)
SIGNAL
UsersDatabase(ULocDatabase),
StartTick,
StopUMobility,
Tick(Integer);
SIGNALLIST ToMob=UsersDatabase
N
(PopuOut)
A
InitialUsers
F
(FM_P)
(P_FM)
P_FM
(PopuIn),
EndSim
(PopuIn)
ToMob
(PopuIn),
EndSim
(ToMob)
MobNet
UMobility
(PopuOut)
Tick
Tick,StopUMobility
StartTick
ClockTick

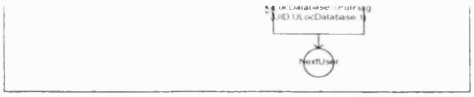


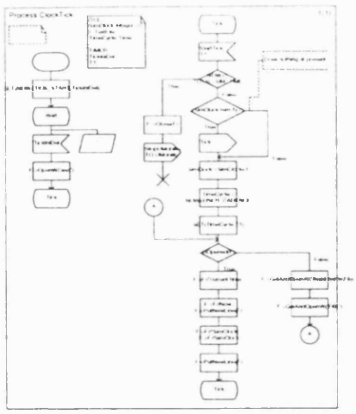


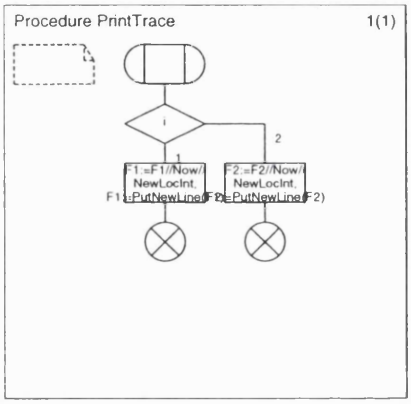

```
DCL
RanLoc RandomControl,
RanFID RandomControl,
SeedU Integer,
SeedUReal Real,
SeedF Integer,
SeedFReal Real,
SeedUFiles TextFile,
SeedIniUser TextFile,

i integer:=0,
RandRealNode Real,
RandNode Integer,
RealOne Real:=1,
RealZero Real:=0,
TempULocEle ULocEle:=NewULocEle,
ULocDatabase ULocDatabase,
UPF_S UserProfileShort:=NewUserProfileShort,
UPF UserProfile:=NewUserProfile,
TempUFiles FileList:=NewFileList,
TempDataFile DataFile:=NewDataFile,
UID Charstring;

TIMER PopuIniDel;
```

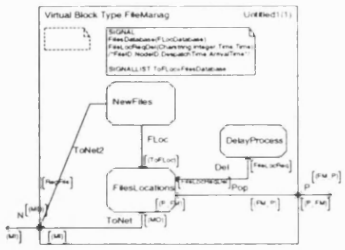


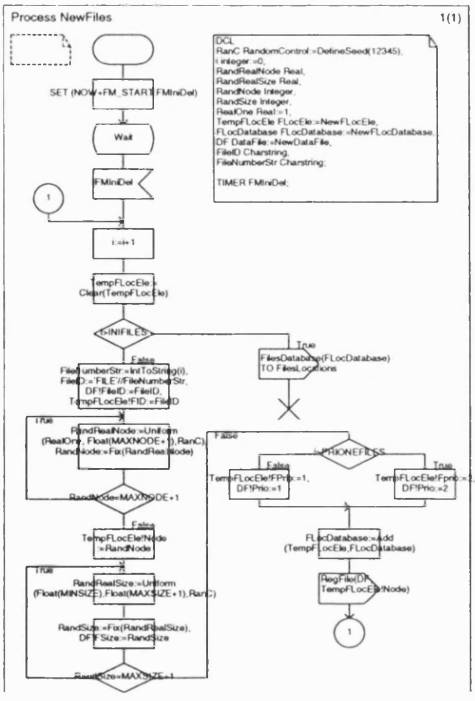


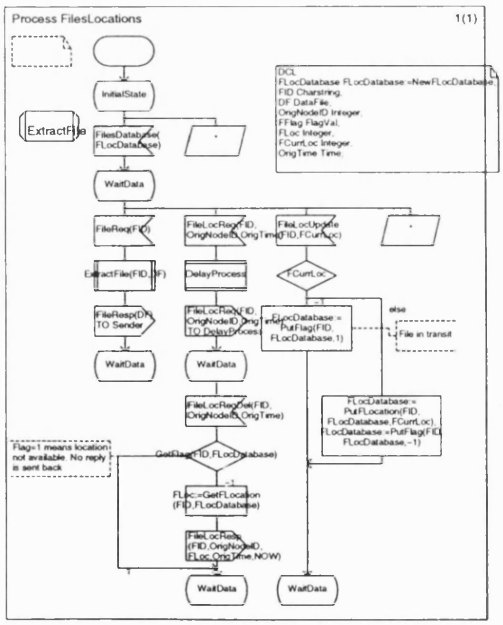

NextUser

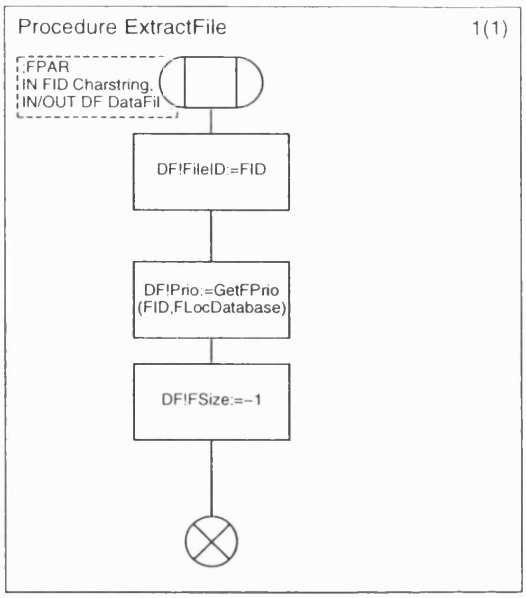


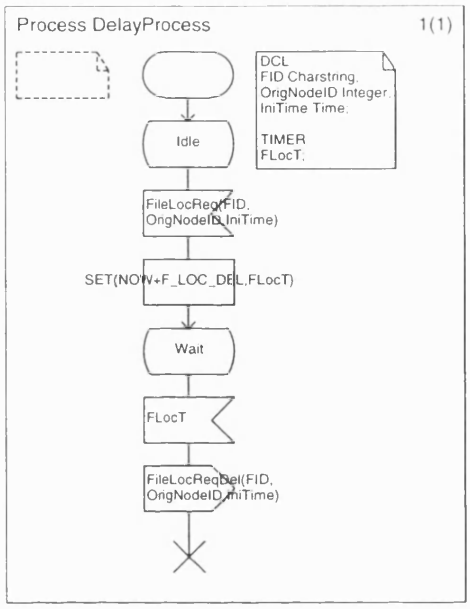

Procedure PrintTrace
1(1)
1
2
F1:=F1//Now//
NewLocInt
F1:=PutNewLine(F2)
F2:=F2//Now//
NewLocInt
F2:=PutNewLine(F2)


Virtual Block Type FileManag
Untitled1(1)
NewFiles
FLoc
ToNet2
DelayProcess
Del
FilesLocations
Pop
ToNet


Process NewFiles
1(1)
SET (NOW+FM_START,FMtnDel)
Wait
FMtnDel
TempFLocEle:=
Clear(TempFLoc(le))
FILENUMBERS
FilesDatabase(FLocDatabase)
TO FilesLocations
RandRealNode:=Uniform
RandNode:=Fix(RandRealNode)
TempFLocEle!FNode
:=RandNode
RandRealSize:=Uniform
RandSize:=Fix(RandRealSize),
DF!FSize:=RandSize
PRIONEFILES
TempFLocEle!FPrio:=1,
DF!Prio:=1
TempFLocEle!Fprio:=2,
DF!Prio:=2
FLocDatabase:=Add
(TempFLocEle,FLocDatabase)
RegFile(DF,
TempFLocEle!Node)


Process FilesLocations
1(1)
InitialState
ExtractFile
FilesDatabase(
FLocDatabase)
WaitData
FileReq(FID)
ExtractFile(FID,DF)
DelayProcess
FCurrLoc
FileResp(DF)
TO Sender
else
File in transit
FLocDatabase:=
PutFlag(FID,
FLocDatabase,1)
FLocDatabase:=
PutFLocation(FID,
FLocDatabase,FCurrLoc),
FLocDatabase:=PutFlag(FID,
FLocDatabase,-1)
Flag=1 means location
not available. No reply
is sent back
FLoc:=GetFLocation
(FID,FLocDatabase)


Procedure ExtractFile
1(1)
FPAR
IN FID Charstring,
IN/OUT DF DataFil
DF!FileID:=FID
DF!Prio:=GetFPrio
(FID,FLocDatabase)
DF!FSize:=-1


Process DelayProcess
1(1)
DCL
FID Charstring,
OrigNodeID Integer,
IniTime Time;
TIMER
FLocT;
Idle
FileLocReq(FID,
OrigNodeID,IniTime)
SET(NOW+F_LOC_DEL,FLocT)
Wait
FLocT
FileLocReqDel(FID,
OrigNodeID,IniTime)

References

[ACTS AC080 96] MISA: Management of Integrated SDH and ATM Networks, Deliverable 3, ACTS Project AC080, October, 1996.

[Avendaño and Hamelberg 93] M. Avendaño, P. C. Hamelberg, "A new co-operative framework for European public network operators", British Telecommunication Engineering, Vol. 12, July 1993, pp 141-145.

[Bale 95] M. C. Bale, "Signalling in the Intelligent Network", BT Technology Journal, Vol. 13, No 2, April 1995, pp 30-42.

[Bause and Buchholz 90] F. Bause, P. Buchholz, "Protocol analysis using a timed version of SDL", Proceeding 3rd International Conference on Formal Description Techniques (FORTE'90), Madrid(Spain), Nov 1990 (North Holland 1991).

[Beilner et al. 88] H. Beilner, J. Mäter, N. Weirenberg, "Towards a performance modelling environment. News on HIT", 4th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, 1988, pp. 57-75.

[Belina and Hogrefe 89] F. Belina, D. Hogrefe, "The CCITT-Specification and Description Language SDL", Computer Networks and ISDN systems, Vol. 16 (1988/89), pp. 311-341

[Belina et al 91] F. Belina, D. Hogrefe, A. Sarma, "SDL with applications from protocol specification", Prentice Hall 1991

[Black 86] W. J. Black, "Intelligent Knowledge Based Systems. An introduction", Van Nostrand Reinhold (UK), 1986, ISBN: 0-442-31772-7.

[Bochman and Vaucher 88] G. V .Bochman, J. Vaucher, "Adding performance aspects to specification languages", Proceedings of the 8th International Conference on Protocol Specification, Testing and Verification, New York, Elsevier Science, 1988.

[Boulter 88] R. Boulter, "Integrated services digital network", Chapter 12 in Local Telecommunications 2 – into the digital era, Revised edition, 1988. Edited by J.M.Griffiths, IEE Telecommunications Series 17, Peter Peregrinus Ltd., ISBN 0 86341 080 4.

[Bræk 95] R. Bræk, "SDL Basics", SDL'95 with MSC in CASE. Tutorials 95.09.25, pp. 11-29.

[Brosemer and Enright 92] J. J. Brosemer, D. J. Enright, "Virtual networks: past, present and future", IEEE Communications Magazine, March 1992, pp. 80-85.

[Brothers et al. 93] L. R. Brothers, E. J. Cameron, Y. J. Lin, M. E. Wilson and E. Silverstein, "Feature interaction detection", *IEEE International Conference in Communications,* vol. 3, pp. 1553-1557, 1993.

[Burroughs 93] A. Burroughs, "Making virtual private networking a reality", Telecommunications (International Edition), March 1993, Vol.27, No.3, pp 63-64.

[Cameron and Velthuijsen 93] E. J. Cameron and H. Velthuijsen, "Feature interactions in telecommunications systems", *IEEE Communications Magazine,* pp. 18-23, August 1993.

[Cameron et al. 93] E. J. Cameron, N. Griffeth, Y. J. Lin, M. E. Nilson, W. K. Schnure and H. Velthuijsen, "A feature interaction benchmark for IN and beyond", *IEEE Communications Magazine,* vol. 31, no. 3, pp. 64-69, 1993.

[CCITT Red Book 85] CCITT, "Terms and definitions related to quality of service, availability and reliability", Red Book, vol. III, Geneva, 1985.

[Chung 94] F. R. K. Chung, "Reliable software and communication I: an overview", IEEE Journal on Selected Areas in Communications, vol. 12, No 1, January 1994, pp. 23-32

[Crowther 93] M. J. Crowther, "Modelling and specifying C7 using SDL", BT Technology Journal, Vol. 11, No. 4, October 1993

[Deck et al. 91] E. Deck, D. Hogrefe, B. Müller-Clostermann, "Hierarchical performance evaluation based on formally specified communication protocols", IEEE Transactions on Computers, Vol. 40, pp. 500-513, April 1991.

[Dieferenbruch et al. 95] M. Dieferenbruch, E. Heck, J. Hintelmann, B. Müller-Clostermann, "Performance evaluation of SDL systems adjunct by queuing models", SDL'95 with MSC in CASE, Proceedings of the Seventh SDL Forum, Oslo, Norway, September 1995, Elsevier Science, pp 231-242.

[DongGill Lee et al. 97] DongGill Lee, JooKyung Lee, Wan Choi, Byung Sun Lee, Chimoon Han, "A new integrated software development environment based on SDL, MSC and CHILL for large-scale switching systems", ETRI Journal, vol. 18, No. 4, January 1997, pp 265-285.

[Dou 95] C. Dou, "A timed SDL for performance modelling of communication protocols", IEEE Global Telecommunications Conference, 1995, Vol.3, pp 1585-1589.

[EURESCOM 307] EURESCOM Project 307 Reliability Engineering. Deliverable No. 1 "Methods for fault handling and Prevention in Signalling System No. 7.

[Fægemand and Ølsen 92] O. Fægemand, A. Ølsen, FORTE-92 Tutorial on new features in SDL-92.

[Fujioka et al. 91] M. Fujioka, H. Yagi, Y. Ikeda, "Universal service creation and provission environment for IN", IEEE Communications Magazine, January 1991, pp 44-51.

[Fuyoka and Wakahara 94] M. Fuyoka, Y. Wakahara, "Consideration on common channel signalling evolution for global intelligent networking", IEEE Journal on Selected Areas in Communications, Vol.12, No.3, April 1994, pp 510-516.

[González and Danke 93] A. J. González, D. D. Dankel, "The engineering of knowledge-based systems. Theory and practise", Prentice Hall International Editions, 1993, ISBN 0-13-334293-x

[Hall and Magedauz 93] J. Hall, T. Magedauz, "Uniform modelling of management and telecommunications services in future telecommunication environments based on the ROSA approach", IFIP Transactions in C: Communications Systems 93, NC-1912, pp 521-532

[Harris-Jones 95] C. Harris-Jones, "Knowledge based systems methods. A practitioner's guide", Prentice Hall, 1995, ISBN 0-13-185315-5

[Haugen 95] Ø. Haugen, "Using MSC-92 effectively", SDL'95 and MSC in CASE, R. Bræk and A. Sarma editors, pp. 37-49, North Holland, 1995, ISBN 0 444 82269 0

[Hlady et al. 95] M. Hlady, R. Kovacevic, J. J. Li, B. R. Pekilis, D. Prairie, T. Savor, R. E. Seviora, D. Simser, A.Vorobiev, "An approach to automatic detection of software failures", Proceedings of the International Symposium on Software Reliability Engineering, 1995, pp 314-323.

[Hoang et al. 93] B. Hoang, D. J. Bastien, B. Lewin, "Common channel signalling network integrity experiences from the U.S." IEEE International Conference on Communications, ICC'93, Geneva, vol. 2, pp 644-649, 1993

[Hung et al. 94] N. L. Hung, A. R. Jacob, S. E. Makris, "Alternatives to achieve software diversity in common channel signalling networks", IEEE Journal on Selected Areas in Communications, Vol. 12, No. 3, April 1994.

[ISO/IEC 7498-1:1994] ISO ISO/IEC 7498-1:1994 Open Systems Interconnection – Basic reference model: the basic model.

[ISO/IS 9646 91] ISO Conformance testing methodology and framework, IS 9646 Part 3: The Tree and Tabular Combined Notation (TTCN), September 1991.

[ITU-T M.3010] ITU-T M.3010, Principles for a Telecommunications Management Network, May 96.

[ITU-T Q.1211] ITU Recommendation Q.1211: *Introduction to intelligent network capability set 1,* March 1993.

[ITU-T Q.1221] ITU Draft Recommendation Q.1221: *Introduction to intelligent network capability set 2,* Sept 1997.

[ITU-T Z.100] ITU-T Recommendation Z.100. "CCITT Specification and Description Language", June 1994

[ITU-T Z.120 94] ITU-T Recommendation Z.120. "Message Sequence Chart (MSC)", September 1994

[Jackson 90] P. Jackson, "Introduction to Expert Systems", 2nd Edition, Wokingham, England: Addison-Wesley, 1990.

[Jain 87] H. K. Jain, "A comprehensive model for the design of distributed systems", IEEE Transactions on Software Engineering, Vol. SE-13, No. 10, Oct 1987.

[Kelly et al. 94] B. Kelly, M. Crowther and J. King, "Feature interaction detection using SDL models", *IEEE Global Telecommunications Conference,* vol. 3, pp. 1857-1861, 1994.

[Kühn et al. 94] P. J. Kühn, C. D. Pack, R. A. Skoog, "Common Channel Signalling Networks: past, present and future", IEEE Journal on Selected Areas in Communications, Vol. 12, No. 3, April 1994, pp 383-393.

[Kurose and Simha 89] J. F. Kurose, R. Simha, "A microeconomic approach to optimal resource allocation in distributed computer systems", IEEE Transactions on Computers, Vol. 38, May 1989.

[Lai and Rauscher 93] M. Y. Lai, K. F. Rauscher, "Total reliability management for telecommunications software", IEEE Global Telecommunications Conference, Globecom 93, vol. 1, pp 505-509, 1993.

[Leakey 94] D. M. Leakey, "Network interconnection in a liberalised environment", No 014 in 20259, pp 1-11, IEE Colloquium on What's new in Telecommunications, January 1994.

[LeBlanc 93] R. R. LeBlanc, "Intelligent Network basic call model rapid prototype", IEEE International Conference on Communications 1993, pp 1543-1547

[Lehrenfeld et al. 95] G. Lehrenfeld, W. Mueller, C. Tahedl, "Transforming SDL diagrams into a complete visual representation", Proceedings of the IEEE Sympusium on visual languages 1995, pp 148-155.

[Lewin and Branflick 94] B. Lewin, D. Branflick, "Future directions for national interoperability testing for network integrity", IEEE Global Telecommunications Conference Globecom 94, Communications: the Global Bridge, vol. 1, pp 222-226, 1994.

[Magedanz and Popescu 96] T. Magedanz, R. Popescu-Zeletin, "Intelligent Networks: basic technology, standards and evolution", 1996, ISBN 1-85032-293-7.

[Mahmoud and Riordon 76] S. Mahmoud, J. S. Riordon, "Optimal allocation of resources in distributed information networks", ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, pp 66-78.

[Maillot et al. 95] D. Maillot, J. Ølnes, P. Spilling, "The new European regulatory environment for telecommunications – implications for service management and its security", IEEE International Conference Communications, 1995, Vol. 1, pp 110-116.

[Majeti and Prasad 93] V. C. Majeti, R. A. Prasad, "Advanced Intelligent Network Directions", IEEE Global Telecommunications Conference, 1993, Vol. 3, pp 1938-1942

[Manterfield 91] R. J. Manterfield, *Common-channel signalling,* London, United Kingdom: Peter Peregrinus Ltd., 1991.

[Mauw and Reniers 94] S. Mauw, M. A. Reniers, "An algebraic semantics of basic Message Sequence Charts", The Computer Journal, Vol.37, No.4, 1994.

[McConnell et al. 93] V. K. McConnell, M. E. Nilson and E. E. Silverstein, "Feature interaction analysis in the Advanced Intelligent Network: A telephone company perspective", *IEEE International Conference in Communications,* Vol. 3, pp 1548-1552, 1993.

[McDonald 94] J.C. McDonald, "Public network integrity – avoiding a crisis in trust", IEEE Journal on Selected Areas in Communicaions, vol. 12, No. 1, Jan. 1994, pp 5-12.

[Mierop et al. 93] J. Mierop, S. Tax and R. Janmaat, "Service interaction in an object-oriented environment", *IEEE Communications Magazine,* pp 46-51, August 1993.

[Montón 96] V. Montón, "An approach to tackling network integrity", *Third Communication Networks Symposium,* the Manchester Metropolitan University, July 1996.

[Montón et al. 97] V. Montón, K. Ward, M. Wilby, R. Masson, "Risk assessment methodology for network integrity", BT Technology Journal, Vol. 15, No. 1, January 1997, pp 223-234.

[Montón et al. 97 b] V. Montón, K. Ward, M. wilby, "Maintaining integrity in the context of intelligent services and networks", Intelligent in Services and Networks: Technology for Cooperative Competition, Proc. 4th International Conference on Intelligence in Services and Networks, IS&N'97, May 1997, pp 427-436, Springer, ISBN 3-540-63135-6

[Nordvik 94] J. P. Nordvik, N. Mitchison, M. Wilikens, "The role of the goal tree-success model in the real-time supervision of hazardous plants", Reliability Engineering and System Safety 1994, Elsevier Science Limited, pp 345-360

[Oliveira and Carrapatoso 96] P. C. Oliveira, E. M. Carrapatoso, "Fast creation of distributed services: SDL to ANSA translation", Proceedings of the IEEE Conference of Electrical and Electronic Engineers, Israel 1996, pp 17-20.

[Olsen et al. 94] A. Olsen, O. Færgemand, B. Møller-Pedersen, R. Reed, J. R. W. Smith, "Systems engineering using SDL-92", North Holland, 1994, ISBN 0 444 89872 7.

[ONP-UCL 94] "Final report of a study entitled Network Integrity in an Open Network Provision (ONP) Environment for the Commission of the European Union", University College London, November 1994.

[Papadimitriou et al. 94] C. H. Papadimitriou, S. Ramanathan, P. V. Rangan, "Information caching for delivery of personalised video programs on home entertainment channels", Proceedings of the International Conference on Multimedia Computing and Systems, 1994, pp 214-223.

[Probert and Monkewich 92] R. L. Probert, O. Monkewich, "TTCN: The international notation for specifying tests of communication systems", *Computer Networks and ISDN Systems,* vol. 23, pp 417-438, Feb. 1992.

[RACE 2116 95] TOMQAT: Total Management of Service Quality for Multimedia Applications on IBC Infrastructure. RACE Project 2116, 1995.

[Ramanathan and Viu 94] S. Ramanathan, H. M. Viu, " Towards personalised multimedia dial-up services", Computer Networks and ISDN Systems, July 1994, No. 10, pp 1305-1322

[Ramaswani 95] V. Ramaswani, "The essential role of traffic performance analysis in Intelligent Networks", IEEE Global Telecommunications Conference, Globecom 95. Communications for Global Harmony, vol. 2, pp. 1254-1259, 1995.

[Redmill and Valdar 94] F J. Redmill, A. R. Valdar, "SPC digital telephone exchanges" Rev. Ed., Peter Peregrinus, 1994, ISBN 086341298x

[Rumbaugh et al. 91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, "Object oriented modelling and design", Prentice Hall, International Editions, 1991.

[Sarikaya and Wiles 92] B. Sarikaya, A. Wiles, "Standard conformance test specification language TTCN", Computer Standards and Interfaces 1992, Vol.14, No.2, pp 117-144.

[Scott and Stansell 92] J. Scott, D. Stansell, "Private virtual networks: key considerations", IEE Conference Publication, 1992, No. 357, pp 37-43.

[Sexton and Reid 97] M. Sexton, A. Reid, "Broadband networking. ATM, SDH, and SONET", Boston Artech House 1997, ISBN 0890065780

[Smith 93] D. J. Smith, "Reliability, Maintainability and Risk. Practical methods for engineers", Fourth edition, Butterworth-Heinemann Ltd., 1993, ISBN 0 7506 0854 4.

[Stroustrup 91] B. Stroustrup, "The C++ Programming Language", Second Edition, Addison Wesley, 1991, ISBN 0-201-53992-6

[T1A1.2/93-001R3] Committee T1-Telecommunications, Document T1A1.2/93-001R3, Technical report No 24, "A technical report on network survivability performance", November 1993.

[T1A1.2/95-001R4] Committee T1-Telecommunications, Document T1A1.2/95-001R4, Technical report No 42, "A technical report on enhanced analysis of FCC-reportable service outage data", August 1995.

[T1A1.2/97-001R3] Committee T1-Telecommunications, Document T1A1.2/97-001R3, Technical report No 24A, "A technical report on network survivability performance (supplement to technical report No 24)", August 1997.

[Thomé et al 94] B. Thomé, B. Glas, R. Nahm, "Validation of real-time systems: from 'soft' to 'hard' ", Proc. 1994 Tutorial Workshop Syst Eng Comput Based Syst, 1994, pp 152-158.

[TINA-C 95] TINA-C Deliverable, "Overall concepts and principles of TINA", v.1, February 1995.

[Töro and Ziegler 94] M. Töro, G. Ziegler, "Validation of abstract test suites with use of SDL", Microprocessing and Microprogramming, December 1994, Vol.40, No.10-12, pp 711-714.

[Trought 91] M. Trought, "Private networking: a roadmap for the future", Telecommunications, 1991, Vol.25, No.9, pp 39-46.

[Tsang et al. 95] S. Tsang, E. H. Magill, B. Kelly, "The feature interaction problem in networked multimedia services – present and future", BT Technology Journal, Vol.5, No. 1, January 1995, pp 235-246.

[Turner 93] K. J. Turner, editor, "Using formal description techniques an introduction to ESTELLE, LOTOS and SDL", Chichester Wiley 1993, ISBN 0471934550.

[Turner 95] G. D. Turner, "Service Creation", BT Technology Journal, Vol. 13, No. 2. April 1995, pp 80-86.

[Unger et al. 94] B. Unger, D. J. Goetz, S. W. Maryka, "Simulation of SS7 common channel signalling", EEE Communications Magazine, March 1994, pp 52-62

[Ural and Williams 94] H. Ural, A. Williams, "Test generation by exposing control and data dependencies within system specifications in SDL", IFIP Transactions C: Communication Systems, 1994, No. C-22, pp 335-350.

[Valdar 89] A. R. Valdar, "Provision of telecommunication service in a competitive and deregulated environment", British Telecommunications Engineering, Vol. 8, Oct. 1989

[Valdar et al. 92] A. Valdar, D. Newman, R. Wood, D. Greenop, "A vision of the future network", British Telecommunications Engineering, Vol. 11, Oct. 1992

[Ward 95] K. Ward, "Impact of network interconnection on network integrity", *British Telecommunications Engineering,* vol. 13, pp 296-303, January 1995.

[Willis and Dufour 95] P. Willis, I. G. Dufour, "Towards knowledge-based networks", BT Technology Journal, Vol. 13, No. 2, April 1995, pp 87-93.

[Wilson and McDermid 95] S. P. Wilson, J. A. McDermid, "Integrated analysis of complex safety critical systems", The Computer Journal, Vol. 38, N0. 10, 1995

[Wirth 95] P. E. Wirth, "Teletraffic implications of database architectures in mobile personal communications", IEEE Communications Magazine, June 1995

[Wohlin 92] C. Wohlin, "Performance and functional prototyping from software descriptions", Proceedings of the International Conference on Software Engineering for Telecommunications Systems and Services, 1992, pp 102-106.

[Woollard 93] K. Woollard, "Verification and Testing", *BT Technology Journal,* vol. 11, January 1993.

[Woollard 95] K. Woollard, "What's IN a model? Modelling IN services using formal methods", *BT Technology Journal,* Vol. 13, No. 2, April 1995, pp 43-50.

[Yuang et al. 92] M. C. Yuang, S. J. Hsu, K. Y. Lee, C. Huang, "An integrated System for modelling, analysis and simulation for real-time networks", Proceedings of the 17th Conference on Local Computer Networks, 1992, pp 231-240.

[Zolfaghari and Kaudel 94] A. Zolfaghari, F. J. Kaudel, "Framework for network survivability performance", IEEE Journal on Selected Areas in Communications, vol. 12, No 1, January 1994, pp 46-51.

Glossary of terms

ACTS	Advanced Communications Technologies and Services - an EC programme, successor to RACE
AE	Application entity
AEI	Application entity invocation
AIN	Advanced Intelligent Network (Bellcore IN)
ANSI	American National Standards Institute
ASE	Application service elements
ASN	Abstract Syntax Notation
ATIS	Alliance for Telecommunications Industry Solutions (USA)
CAS	Channel associated signalling
CCAF	Call control agent function
CCF	Call control function
CCITT	International Consultative Committee on Telephony and Telegraphy - a Standards making committee of the ITU (now ITU-T)
CCS	Common Channel Signalling
CPE	Customers' Premises Equipment
CS1	Capability Set 1 (CCITT/ITU-T specification series)
DFP	Distributed Functional Plane
DPE	Distributed Processing Environment

DUP	Data User Part
EC	Commission of the European Union
ETSI	European Telecommunications Standards Institute
EU	European Union
FCC	Federal Communications Commission
FDT	Formal Description Technique
FE	Feature Entity
FEA	Feature Entity Agent
GFP	Global Functional Plane
GSL	Global Service Logic
IN	Intelligent Network
INA	Intelligent Network Architecture
INAP	Intelligent Network Application Protocol
INAP	Intelligent Network Access Point
IP	Intelligent peripheral
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO	International Standards Organisation
ISUP	ISDN User Part
ISP	Intermediate Services Part
ITU	International Telecommunications Union
ITU-T	ITU Telecommunications standards (was CCITT to 1993)
KBS	Knowledge Based System
LATA	Local Access and Transport Area
LE	Local Exchange
MSC	Message Sequence Charts
MTP	Message transfer part
NRIC	Network Reliability and Interoperability Council (USA)
NRC	Network Reliability Council (USA)
NSP	Network Services Part
ODP	Open Distributed Processing
OFTEL	The government office responsible for the telecommunications regulation in the UK

OLO	Other licensed operator
OMT	Object Modelling Technique
ONA	Open Network Architecture
ONP	Open Network Provision
OSI	Open Systems Interconnection
POI	Point of Interconnect
POTS	Plain Old Telephony Service
PTO	Public Telecommunications Operator
QoS	Quality of Service
RACE	Research and development on Advanced Communications in Europe, an EC programme
SACF	Single Association Control Function
SAO	Single Association Object
SCCP	Signalling Connection Control Part
SCE	Service Creation Environment
SCF	Service Control Function
SCP	Service Control Point
SDF	Service Data Function
SDL	Specification and Description Language
SIB	Service Independent Building Block
SLEE	Service Logic Execution Environment
SP	Service provider
SP	Signalling Point
SRF	Specialised Resource Function
SS7	Signalling System Number 7
SSF	Service Switching Function
SSP	Service Switching Point
STP	Signal Transfer Point
TC	Transaction Capabilities
TCAP	Transaction Capabilities Application Part
TINA	Telecommunications Information Networking Architecture
TMN	Telecommunications Management Network
TO	Telecommunications Organisation

TTCN	Tree and Tabular Combined Notation
TUP	Telephone User Part
UDP	User Datagram Protocol
ULE	User Lost Erlang
UMTS	Universal Mobile Telecommunication Service
UPT	Universal Personal Telecommunications
VPN	Virtual Private Network