Service management for multi-domain
Active Networks

Alvin K. K. Tan

Submitted to the Department of Electronic & Electrical Engineering in
partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Electronic & Electrical Engineering

University College London
2004
ACKNOWLEDGEMENTS

I would like to thank a whole spectrum of people who have helped contribute towards this exciting journey, which I started about four-and-a-half years ago.

My sincere gratitude goes to my supervisor, Chris Todd and Alex Galis, whose hard work in bringing valuable research funding to the group has enabled many of us to realise our ambitions of pursuing the Ph.D. I am also grateful to have George Pavlou from the University of Surrey and Abe Mamdani from Imperial College as my examiners, who have both helped to make this a stronger thesis.

Walter Eaves deserves my utmost thanks for being a wonderful teacher and friend. Without his interesting insights, many of the ideas presented in this thesis would not have materialised. Together with Chris, Dave Griffin taught me how much fun research could be when they supervised my final year undergraduate project.

Many interesting people have joined and left the Network Services Group during the time I have spent here. In particular, I would like to thank Nikos Vardalachos, Kun Yang, Kerry Jean, Lawrence Cheng, Ajibola Bakare, Jason Spencer, Jonas Griem, Yiannis Liabotis, George Eleftheriou, Wilson Lim, Richard Lewis, Jorge Castro and John Lam, for making rooms 621, 811 and 104 not only workplaces, but fun places.

I would also like to thank Epi Salamanca and Julio Vivero at Universitat Politècnica de Catalunya and Marcin Solarski at the Fraunhofer Institute FOKUS, with whom I have enjoyed interesting and valuable technical discussions regarding the FAIN project. As with those in the NSG, I also appreciate the friendship that we developed.

No doubt, tackling all the challenges inherent in the Ph.D experience would not have been possible without my loved ones. Words of love and encouragement during the weekly phone calls with my parents, thousands of miles away, have continuously rejuvenated me to persevere with my goal of attaining the Ph.D.

And finally, to my wonderful wife, Ming Ai: thank you for always believing in my abilities and aspirations. I owe a great debt of gratitude for your love, patience and support in helping me make a dream come true. You were and will always be my reason to make good things happen; and now, we can look forward to getting on with the rest of our lives.

Alvin Tan, Ph.D
15 April 2004
ABSTRACT

The Internet is an example of a multi-agent system. In our context, an agent is synonymous with network operators, Internet service providers (ISPs) and content providers. ISPs mutually interact for connectivity’s sake, but the fact remains that two peering agents are inevitably self-interested. Egoistic behaviour manifests itself in two ways. Firstly, the ISPs are able to act in an environment where different ISPs would have different spheres of influence, in the sense that they will have control and management responsibilities over different parts of the environment. On the other hand, contention occurs when an ISP intends to sell resources to another, which gives rise to at least two of its customers sharing (hence contending for) a common transport medium.

The multi-agent interaction was analysed by simulating a game theoretic approach and the alignment of dominant strategies adopted by agents with evolving traits were abstracted. In particular, the contention for network resources is arbitrated such that a self-policing environment may emerge from a congested bottleneck. Over the past 5 years, larger ISPs have simply peddled as fast as they could to meet the growing demand for bandwidth by throwing bandwidth at congestion problems. Today, the dire financial positions of Worldcom and Global Crossing illustrate, to a certain degree, the fallacies of over-provisioning network resources. The proposed framework in this thesis enables subscribers of an ISP to monitor and police each other’s traffic in order to establish a well-behaved norm in utilising limited resources. This framework can be expanded to other inter-domain bottlenecks within the Internet.

One of the main objectives of this thesis is also to investigate the impact on multi-domain service management in the future Internet, where active nodes could potentially be located amongst traditional passive routers. The advent of Active
Networking technology necessitates node-level computational resource allocations, in addition to prevailing resource reservation approaches for communication bandwidth. Our motivation is to ensure that a service negotiation protocol takes account of these resources so that the response to a specific service deployment request from the end-user is consistent and predictable.

To promote the acceleration of service deployment by means of Active Networking technology, a pricing model is also evaluated for computational resources (e.g., CPU time and memory). Previous work in these areas of research only concentrate on bandwidth (i.e., communication) – related resources. Our pricing approach takes account of both guaranteed and best-effort service by adapting the arbitrage theorem from financial theory. The central tenet for our approach is to synthesise insights from different disciplines to address problems in data networks.

The greater parts of research experience have been obtained during direct and indirect participation in the IST-10561 project known as FAIN (Future Active IP Networks) and ACTS-AC338 project called MIAMI (Mobile Intelligent Agent for Managing the Information Infrastructure). The Inter-domain Manager (IDM) component was integrated as an integral part of the FAIN policy-based network management systems (PBNM). Its monitoring component (developed during the MIAMI project) learns about routing changes that occur within a domain so that the management system and the managed nodes have the same topological view of the network. This enabled our reservation mechanism to reserve resources along the existing route set up by whichever underlying routing protocol is in place.
# TABLE OF CONTENTS

## CHAPTER 1 INTRODUCTION ......................................................................................12

1.1 PROBLEM SPACE, MOTIVATION AND SCOPE .................................................................12
1.2 CONTRIBUTIONS AND RELEVANCE ..............................................................................14
1.3 THESIS STRUCTURE AND ORGANISATION ..................................................................15

## CHAPTER 2 SERVICE MANAGEMENT WITHIN THE INTERNET .......................18

2.1 CURRENT TRENDS AND PROBLEM SPACE ..................................................................19
2.2 STAKEHOLDERS ...........................................................................................................21
  2.2.1 From dial-up to backbone .......................................................................................22
  2.2.2 Active Network providers ......................................................................................23
  2.2.3 ISP relationships ....................................................................................................24
2.3 ROUTE CATEGORISATION ............................................................................................26
2.4 DOMAINS ......................................................................................................................28
  2.4.1 Domain types and nodes within ..............................................................................30
  2.4.2 Segregation within domains to enhance scalability ..............................................30
  2.4.3 Agency relationships .............................................................................................33
2.5 COMPARISON TO THE PSTN .....................................................................................36
2.6 SERVICE TYPES ..........................................................................................................39
2.7 QoS APPROACHES .......................................................................................................40
  2.7.1 Current design issues .............................................................................................40
  2.7.2 Integrated services ..................................................................................................42
  2.7.3 Differentiated service .............................................................................................43
  2.7.4 MPLS – layer 2.5 ....................................................................................................44
2.8 SUMMARY .....................................................................................................................44

## CHAPTER 3 THE IMPACT OF ACTIVE NETWORKS ...........................................46

3.1 ACTIVE NETWORKS .......................................................................................................46
  3.1.1 Objective ................................................................................................................46
  3.1.2 Implementation .......................................................................................................47
  3.1.3 Open interfaces in programmable networks .........................................................48
  3.1.4 Comparison ............................................................................................................50
3.2 NODE ARCHITECTURE .................................................................................................51
  3.2.1 Active node resource .............................................................................................53
  3.2.2 Active packets ........................................................................................................55
3.3 MANAGEMENT OF ACTIVE NODES ...........................................................................56
  3.3.1 State-of-the-art review ...........................................................................................58
  3.3.2 Analysis of Active Network management using a constitutional framework ..........61
3.4 EXPECTATIONS .............................................................................................................64
  3.4.1 Speeding service deployment and service customisation ......................................64
  3.4.2 Leveraged network and service management .......................................................65
  3.4.3 Diversification of services and business opportunities .........................................65
3.5 A NEW ROLE FOR STAKEHOLDERS ............................................................................66
3.6 ACCESS AND ALLOCATION STRATEGIES IN A MULTI-USER SYSTEM ............67
3.7 SERVICE EXAMPLES ....................................................................................................68
  3.7.1 Scenario for service provisioning using the active node ......................................69
  3.7.2 A secure framework for remote access .................................................................71
3.8 SUMMARY .....................................................................................................................73
CHAPTER 4  NORMS IN AGENT INTERACTION .................................................76
  4.1  UNDERSTANDING MULTI-DOMAIN COOPERATION AND COMPETITION ....77
  4.2  A BASIC GAME THEORETIC APPROACH ..................................................80
    4.2.1  Reasoning about strategies between two ISPs ........................................80
    4.2.2  Games of incomplete information ..........................................................83
    4.2.3  Formalising the problem space: tragedy of the commons .........................86
  4.3  STABILISING NORMS ..................................................................................89
    4.3.1  Current flow control strategies ..............................................................89
    4.3.2  The norms game in the context of access networks and bottlenecks ...........91
    4.3.3  Metanorms as a means to stabilise norms ................................................92
    4.3.4  Simulation ..................................................................................................94
    4.3.5  Practical implementation ..........................................................................99
    4.3.6  Reasoning about defections between ISPs ...............................................102
  4.4  SUMMARY .....................................................................................................104

CHAPTER 5  PRICING NODE RESOURCES ..........................................................108
  5.1  INTERNET ECONOMICS AND THE RELEVANCE OF FINANCIAL INSTRUMENTS ....109
  5.2  INCORPORATING THE PARIS METRO PRICING (PMP) IDEA .........................110
    5.2.1  Relevance to Chapter 4 ..........................................................................111
  5.3  MATHEMATICAL REPRESENTATION OF RESOURCE PRICE .........................111
  5.4  ‘ASSET CLASSES’ FOR ACTIVE NODES .........................................................113
    5.4.1  Resource pools .........................................................................................113
    5.4.2  Arbitrage theorem ....................................................................................115
    5.4.3  Adapting from financial theory .................................................................117
    5.4.4  Pricing the options ....................................................................................118
  5.5  A NUMERICAL EXAMPLE AND BASIC COMPUTATIONS .............................120
  5.6  SUMMARY .....................................................................................................121

CHAPTER 6  INTER-DOMAIN MANAGEMENT IN FAIN .........................................123
  6.1  STATE-OF-THE-ART REVIEWS ....................................................................124
    6.1.1  Policy-based network management ..............................................................124
    6.1.2  Inter-domain management .........................................................................135
  6.2  THE FAIN MANAGEMENT SYSTEM ................................................................136
    6.2.1  PDP manager ............................................................................................138
    6.2.2  Policy decision point (PDP) ......................................................................139
    6.2.3  Policy enforcement point (PEP) ..................................................................139
    6.2.4  Service manager .........................................................................................139
    6.2.5  Resource manager ......................................................................................140
  6.3  INTER-DOMAIN MANAGER ..........................................................................140
    6.3.1  Overview .....................................................................................................140
    6.3.2  Design and implementation ........................................................................141
    6.3.3  IDM enhancement .......................................................................................144
  6.4  INTER-DOMAIN RESOURCE-CENTRIC NEGOTIATION PROTOCOL (IRNP) ......145
    6.4.1  Motivation ...................................................................................................145
    6.4.2  Assumptions ...............................................................................................146
    6.4.3  The protocol ...............................................................................................147
    6.4.4  Discussion ...................................................................................................151
  6.5  CASE STUDY ..................................................................................................151
    6.5.1  Motivation and requirements ......................................................................152
    6.5.2  Use case .....................................................................................................155
    6.5.3  Management by delegation ..........................................................................156
TABLE OF FIGURES

Figure 1: Scope of research.......................................................................................................... 12
Figure 2: Transit and the flow of money across ISP tier hierarchy. CERFNet, UUNet and PSINet are the largest Tier 1 ISPs in the USA.............................................................. 21
Figure 3: A real world perspective of the Internet stakeholders................................................... 23
Figure 4: ISPs as Active Network providers (ANPs)................................................................... 24
Figure 5: Peering and transit arrangements .................................................................................. 24
Figure 6: Traffic flows in opposite direction from route advertisements. Peering achieves mutual reachability while transit achieves global reachability .................................... 27
Figure 7: Regions within a domain .............................................................................................. 31
Figure 8: Agency relationships.................................................................................................... 34
Figure 9: A good reason why service differentiation should be targeted at the OSI network layer – the pervasiveness of IP ................................................................. 41
Figure 10: Illustration of the data, control and management planes within an active node ......... 49
Figure 11: The three software layers running on an active router: NodeOS, EE, and the application. The VE is an abstraction that is used only for the purpose of partitioning the resources of the active node such that different communities of users can stay isolated from each other............................................................... 51
Figure 12: Mapping the constitution model towards the management of active nodes (cf. the IETF’s model in Figure 37) .................................................................................... 63
Figure 13: Reverse-path VPN provisioning using Injector and Interceptor Systems............... 69
Figure 14: Illustration of Active Networking technology used in securing remote access to a private network ................................................................. 71
Figure 15: Cooperation in providing end-to-end service ............................................................. 77
Figure 16: Alternate paths ............................................................................................................ 78
Figure 17: Competition in resource contention, coupled with the benefit of cooperation ......... 79
Figure 18: The assignment of points in this payoff matrix generates the classic conundrum of game theory, known as the Prisoner’s Dilemma. R, S, T, P definitions are relative to Player 1................................................................................ 81
Figure 19: Customer flows coming in at higher speed than ISP’s uplink, where the incoming traffic is over-subscribed ................................................................. 85
Figure 20: Graph of ISP utility versus the overall total number of packets on processor........... 87
Figure 21: Metanorms game ........................................................................................................ 91
Figure 22: Basic norms game. Lighter shaded plots indicate earlier generation. Red dot denotes final average after 500 generations................................. 95
Figure 23: Temporal adaptation of channel utilisation with no incentive to punish ‘defection’ .................................................................................................................. 96
Figure 24: Metanorms game. Boldness level drops and remains low.......................................... 97
Figure 25: Temporal adaptation of channel utilisation with incentive to punish ‘defection’ ........................................................................................................ 97
Figure 26: Group dynamics revert to high average vengefulness and low average boldness even after starting of the metanorms game biased with low vengefulness and high boldness. ................................................................. 98
Figure 27: An abstraction of the departmental research network; iptable rules are inserted at ‘george’ so that ‘jorg’ and ‘wilson’ can use SNMP to probe each others’ traffic through ‘george’.................................................... 100
Figure 28: Screen snapshot depicts an administrator setting the filtering rules on ‘george’ .... 101
Figure 29: Screen capture shows how 'jorg' may use the iptable MIB to inspect the chainByteCount OID on 'george' to find out the traffic size from 'wilson'...

Figure 30: Mapping the metanorms game into a network context

Figure 31: Actors in the system

Figure 32: Relationship between the Checker and the Defector

Figure 33: Relationship between the Accomplice and the Defector

Figure 34: Relationship between the Checker and the Accomplice

Figure 35: Grey arrow depicts flow of an end-user’s traffic into its own reserved EE (hence VE), each made up of predetermined memory and thread pools

Figure 36: Inverse relationship between utility and price for each unit of resource

Figure 37: The IETF PBNM reference model

Figure 38: Abstraction levels to aide policy translation

Figure 39: Introducing a feedback mechanism to policy enforcement

Figure 40: PxP basic architecture

Figure 41: FAIN two-tiered architecture within a single domain

Figure 42: The FAIN network-level management system (NMS) in relation to active service provisioning (ASP) and element management system (EMS)

Figure 43: RMI communication using stub and skeleton

Figure 44: UML sequence diagram depicts interaction between the subcomponents of the IDM with the other key components within the NMS, i.e., the resource manager and the service manager

Figure 45: Manager-agent relationship in SNMP

Figure 46: Sequence of signalling messages to conduct negotiation as dictated by the inter-domain resource-centric negotiation protocol (IRNP)

Figure 47: Sequence of signalling messages to conduct negotiation when reservation is required across two (or more) other domains

Figure 48: The FAIN pan-European testbed logically divided to represent separate administrative domains

Figure 49: Video-on-demand delivery

Figure 50: Management instances illustrated on simplified FAIN management systems

Figure 51: Use case depicts the functions of the ASP mechanism. In FAIN the Active Network Provider is referred to as ANSP
# ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>Active Application</td>
</tr>
<tr>
<td>ANP</td>
<td>Active Network Provider</td>
</tr>
<tr>
<td>ANEP</td>
<td>Active Network Encapsulation Protocol</td>
</tr>
<tr>
<td>ANSP</td>
<td>Active Network Service Provider</td>
</tr>
<tr>
<td>ANTS</td>
<td>Active Network Transport System</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>BB</td>
<td>Bandwidth Broker</td>
</tr>
<tr>
<td>CAML</td>
<td>Categorical Abstract Machine Language</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>CNF</td>
<td>Conjunctive Normal Form</td>
</tr>
<tr>
<td>COPS</td>
<td>Common Open Policy Service</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defence Advanced Research Projects Agency</td>
</tr>
<tr>
<td>DEN</td>
<td>Directory-Enabled Networking</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DMTF</td>
<td>Distributed Management Task Force</td>
</tr>
<tr>
<td>DNF</td>
<td>Disjunctive Normal Form</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DSLAM</td>
<td>Digital Subscriber Line Access Multiplier</td>
</tr>
<tr>
<td>EE</td>
<td>Execution Environment</td>
</tr>
<tr>
<td>EMS</td>
<td>Element Management System</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>FAIN</td>
<td>Future Active IP Networks</td>
</tr>
<tr>
<td>FDDI</td>
<td>Fibre Distributed Data Interface</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>HDLC</td>
<td>High-level Data Link Control</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IDL</td>
<td>Interface Definition Language</td>
</tr>
<tr>
<td>IDM</td>
<td>Inter-Domain Manager</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IOS</td>
<td>Internetwork Operating System</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IRNP</td>
<td>Inter-domain Resource-centric Negotiation Protocol</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organisation</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
</tbody>
</table>
MAE  Metropolitan Area Exchange
MPLS  Multiprotocol Label Switching
MIAMI Mobile Intelligent Agent for Managing the Information Infrastructure
MIB  Management Information Base
NAP  Network Access Point
NMS  Network Management System
NSP  Network Service Provider
ODP  Open Distributed Processing
OID  Object Identifier
OSI  Open Systems Interconnect
PBNM Policy-Based Network Management
PDP  Policy Decision Point
PEP  Policy Enforcement Point
PHB  Per Hop Behaviour
PLAN Programmable Language for Active Networks
PMP  Paris Metro Pricing
PNO  Public Network Operator
POP  Point of Presence
POTS Plain Old Telephone Service
PSTN Public Switched Telephone Network
QoS  Quality of Service
RIP  Routing Information Protocol
RISC Reduced Instruction Set Computer
RMI  Remote Method Invocation
RSVP Resource Reservation Protocol
SCP  Service Control Point
SDH  Synchronous Digital Hierarchy
SLA Service Level Agreement
SMTP Simple Mail Transfer Protocol
SNMP Simple Network Management Protocol
SONET  Synchronous Optical NETwork
SS7  Signalling System 7
SSP  Service Switching Point
TINA Telecommunications Information Networking Architecture
TMN  Telecommunications Management Network
VAN  Virtual Active Network
VPN Virtual Private Network
XML Extensible Mark-up Language
Chapter 1

Introduction

1.1 Problem space, motivation and scope

The Internet is an example of a multi-agent system, where we represent the controlling entity (for access control and resource allocation) for each administrative domain as an agent. In our context, an agent is synonymous with Internet service providers (ISPs); and it is well known that ISPs mutually interact for connectivity’s sake. However, the fact remains that two peering agents are inevitably self-interested when we consider their respective commercial goals, particularly as we explore service management issues.

![Figure 1: Scope of research](image-url)
Our research agenda in this thesis addresses two main concerns (see Figure 1) in service provisioning: communication (e.g., bandwidth) and computational (e.g., node processing time) resource contention, as well as resource reservation. Contention occurs at network bottlenecks. The multi-agent interaction is analysed in Chapter 4 by simulating a game theoretic approach and the alignment of dominant strategies adopted by agents with evolving traits were abstracted. Since the advent of Active Networking technology elaborated in Chapter 3 necessitates node-level computational resource allocations, the service negotiation protocol discussed in Chapter 6 takes account of these resources so that the response to a specific service deployment request from the end-user is consistent and predictable.

As of today, no complete integrated solution exists for a provider that wants to offer and manage enhanced IP services. Difficulties occur if the customer is able to order a service online (service automation) and the service requires the collaboration of several providers. In other words, inter-administrative domain management research initiatives have all but provided an automated and seamless collaborative framework for managing multi-provider services. Thus, it is a research goal to derive a protocol that can introduce the notion of negotiation within a resource reservation sequence.

In our view, discussions on inter-domain service management would be incomplete without a mention of pricing. Thus, a new pricing approach that takes account of both guaranteed and best effort service was introduced. The arbitrage theorem was adapted from financial theory and this work echoed Semret’s [130] proposal for establishing derivative assets from underlying network resources.

One of our motivations is to synthesise insights from different disciplines and also to encourage practitioners to design strategies and implement software that can be tested in our proposed environment.
1.2 Contributions and relevance

1. Formalising admission and allocation strategies of Internet players using first-order propositional logic;

2. Metanorms simulation in network context to extract lessons in achieving stability in inter-provider cooperation for end-to-end service provisioning;

3. MIB for discovering metanorm parameters;

4. Inter-domain signalling protocol to enable negotiation, including the implementation of the inter-domain manager; and

5. Adapting financial derivatives for pricing communicational and computational network resources.

Using our approach, we hope to convince ISPs that we can help establish a more autonomous framework that will increase their efficiency in managing customer traffic. Secondly, with a fair pricing mechanism in place as well as a multi-domain service provisioning protocol (both of which take account of unique computational resources for Active Networking), ISPs will thus be motivated to introduce node programmability for its customers as this will increase the product range it can offer to them (mainly for content providers). Router vendors should also be interested to enhance their product MIB to enable new actions to be performed in network management. In our context, this implies the discovery of metanorms parameters.

In terms of the immediate research circle, previous researchers who have worked on the relevant fields would be interested to see how their initial (seminal in some cases) works have been enhanced. For instance, Axelrod’s metanorms game would have been fully implemented in a networking context, while Semret would see that his initial notion of offering bandwidth derivatives has been expanded and unified to include a pricing mechanism that takes account of best-effort and guaranteed service
on top of a derivative product. Finally, we also believe that a further number of Ph.Ds would be derived from this work, and in addition, there would be opportunities for M.Sc and M.Eng projects at UCL.

1.3 Thesis structure and organisation

In Chapter 2, we describe what the Internet can do today in contrast to what end-users expect it to be, hence the existence of a plethora of research initiatives that is geared towards realising a service-provisioning platform. In further sub-sections, we also explain the Internet constituents and their relationships, before proceeding to define our understanding of domains in this thesis so that our perception of inter-domain interaction is well comprehended. We understand that service management is a topic that has been extensively researched in the telecommunications field and revisited the use of intelligent network (IN) for service provisioning and compare its motivations with regards to what we hope to achieve with data networks (in particular, those of inter-domain resource reservation as well as active and programmable networks). We also explain what we mean by a service by differentiating between basic and high-level services in the Internet and we then go on to review related work in high-level service management.

In Chapter 3, we move on to describe the impact of active and programmable networks on service management within the network model we described earlier. In particular, we highlight the currency of computational resources brought forth by Active Networking technology. Further to discussions with project partners in FAIN, we point out the expectations of incumbent local exchange carriers (ILECs) with regards to the advent of Active Networking technologies. As with Chapter 2, this chapter comprises an extensive literature survey, as well as our views and insights on the current state of the corresponding research issues. We used a mathematical approach to generalise access
control policies for computational resources and these generic definitions are equally applicable as strategies that may be implemented by ISPs to promote their respective goals. In trying to understand the impact brought forth by the Active Networking technology, two use cases that may be taken up by future research initiatives in the area of secure service provisioning were also conceptualised.

In Chapter 4, we explore the fundamental issue of allocating limited network communication and computational resources. Our goal is to exploit the economic situation of the stakeholders so that they can apply a range of simple local policies that should give rise to more stable global situations. We model the inter-domain environment such that Axelrod's notion of norms in social setting can be mapped on to our problem space. Using a Java-based simulation, we showed how Axelrod's idea of metanorms may be adopted by network architects in establishing stability and order at congestion points; and presented ideas for practical implementation.

We recognise that accurate pricing of bandwidth has long been a challenging goal for both commercial parties and researchers alike. We further propose the use of the Paris Metro Pricing approach as we foresee the need for an explicit yet simple charging mechanism to prevent the 'tragedy of the commons' explained in Chapter 4. In Chapter 5, we complement this initiative by presenting, in our knowledge, the first framework for pricing computational resources to be reserved for setting up virtual environments (VEs) for Active Networks. Here, we first explore current work in Internet Economics before proceeding to explain our approach based on financial theory, i.e., arbitraging.

We discuss the inter-domain manager (IDM) in Chapter 6 and introduce a protocol where two IDMs may negotiate and arrive at a consensus with regards to resource allocation across administrative domains. The need for this sub-component arose within the FAIN project when we had to reserve computational and
communicational resources for service deployment in our WebTV case study, particularly as requests occur between different administrative entities. For completeness, we also briefly introduce the other components within the FAIN network-level management system.

We conclude this thesis in Chapter 7 by reasserting our contributions to the inter-domain service management research community. We also discuss open issues that are left unanswered and present a list of questions that may shape future research agenda. The publications achieved during the Ph.D research period is listed in §A.12, which includes key IEEE conferences like IWAN, OpenArch, ICN and ICT.
Chapter 2

Service management within the Internet

I am far from thinking that nomenclature is a remedy for every defect in art or science: still I cannot but feel that confusion of terms generally springs from, and always leads to, confusion of ideas.

John Louis Petit, Architectural Studies in France, 1854

We discuss the Internet landscape in which Active Networking technologies may be deployed in the future. If the motivation for Active Networking is one that would potentially accelerate service deployment, then a thorough understanding of the relationships among current stakeholders of this ‘network of networks’ is an essential precursor to the thesis. We also define our perception of domains to demarcate different ownership regions that make up the Internet so that our discussions within a multi-domain context naturally extend from it.

This chapter also looks at the issues that call for innovation in implementation approaches in order to provide a better service management framework as well as guaranteed levels of expected quality. These are comparable motivations that gave rise to the creation of the active and programmable networking paradigm, which aims to enable the set-up of provider-specific services ‘on-the-fly’. According to Barillaud et al [26], the lack of a management framework for Internet service has become the major drawback to the development of further Internet services, which meant that earlier development of Internet management had focused on network management; and recent management initiatives for the Internet have been directed towards service-specific context (see, for instance, [98] and [117]).
2.1 Current trends and problem space

The Internet has undoubtedly become a distribution channel for a wide array of end-user services with varied requirements, ranging from home entertainment to business applications. The underlying Internet infrastructure, as it stands today, is far from that of a service-delivery platform. The current service model is adequate for the applications that were prevalent when the Internet was first designed: remote access [123], file transfer [139], and electronic mail [92]. The design, motivated by simplicity, is based on best-effort delivery. End-users now demand end-to-end flexibility; service differentiation; isolation and privacy; as well as manageability. In riposte, the research community attempts to keep pace by means of active and programmable networks [146]; quality of service (QoS) implementations [131]; virtual private network (VPN) developments [65]; and policy-based and directory technologies [27].

The VPN approach enables private communication to be conducted across a network infrastructure shared by more than a single organisation. Interoperability issues arise due to administrative heterogeneity, i.e., the Internet has many owners. Using IP security (IPSec) protocol, the de facto industry standard for an IP-based VPN infrastructure, end-to-end privacy may be attained. However, achieving the required QoS remains to be guaranteed unless the various providers agree to support a common tunnelling mechanism and as a consequence unambiguous VPN identity can be assigned across multiple domains. This implies some form of cooperation between different administrative entities.

Asynchronous transfer mode (ATM) networks have long provided link-layer traffic engineering, first in voice networks, then in data networks. However, when an end-to-end path does not consist of a single, pervasive data-link layer, a packet will inevitably traverse links that cannot provide any type of differentiated service at the link layer, rendering an effort to provide QoS solely at this layer an inadequate solution.
Thanks to the ubiquity and the pervasiveness of IP as the standard Layer 3 protocol, we now have the option to utilise the unused bits within the IP header for end-to-end channel aggregation.

QoS-enabling technologies such as differentiated services [34] and integrated services [39] have been thoroughly researched, developed and refined over the past decade, and have been ready for deployment since the late 90s. So why are they not extensively rolled out yet? Chief among the reasons is that an inter-domain basis for the end-user to witness premium end-to-end quality has yet to be fully realised. This could be due to hardware heterogeneity and the ISP’s apprehension towards a higher level of cooperation that is required beyond basic physical connectivity.

To address the drawback in scalability\(^1\) in a multi-domain context, the Inter-domain Bandwidth Broker Signalling (SIBBS) mechanism [113] was created. From 1999 to 2001, the Qbone community conducted substantial research work to test and deploy scalable QoS mechanisms in the Internet\(^2\) environment [75]; however due to intractable deployment problems, the Internet2 QBone Premium Service initiative has been suspended indefinitely. To our knowledge, there has been no follow-up work to QBone’s SIBBS initiative, \textit{i.e.}, a simple protocol that consists of a request-response communication between the bandwidth broker peers, which carries the essential information for requesting a service in general.

The advent of Active Networking technology further introduces an additional parameter that could benefit from a reservation mechanism, \textit{i.e.}, computational resources on network nodes (\textit{e.g.}, per-flow processor time allocation and node memory space). Ergo, this thesis addresses some pertinent issues in end-to-end service

\(^1\) Also see §2.4.2 where scalability is addressed by demarcating the various regions of network elements within a domain into core, distribution and access.

\(^2\) The Internet2 (http://www.internet2.edu/) is a consortium being led by 205 universities working in partnership with industry and government to develop and deploy advanced network applications and technologies, in order to accelerate the creation of the next generation Internet.
management, particularly those that involve negotiation of computational and communication resources between Internet service providers (ISPs). Inter-provider relationships and strategies are examined in order to establish a stabilised norm for basic contention management at inter-domain bottlenecks. This holistic scheme is expected to encourage the notion of self-management among peers.

2.2 Stakeholders

To better understand the issues in this thesis, the ownership of the various segments within the Internet are described, where key assumptions regarding the relevant stakeholders within today's Internet are also stated. The Internet 'players' are categorised into network service providers (NSPs) who provide transit, long-distance, raw capacity, backbone connectivity; Internet service providers (ISPs)\(^3\) who provide points of presence for the geographic area served, \textit{i.e.}, sell access to the Internet; and end-users. The ISPs themselves connect to the Internet via transit service (see §2.2.3) from the upstream NSP. End-users range from corporate networks to retail, dial-up home users.

![Diagram showing ISP tier hierarchy]

Figure 2: Transit and the flow of money across ISP tier hierarchy. CERFNet, UUNet and PSINet are the largest Tier 1 ISPs in the USA.

\(^3\) The size of an ISP is determined by its geographical reach (\textit{e.g.}, regional or local); as such, one can imagine further hierarchies within this category in practice.
The relative magnitude\(^4\) of ISPs can be translated into tiers\(^5\) (see Figure 2). Local ISPs provide retail-based Internet connectivity to end-users and connect to Regional ISPs on a wholesale basis. Regional ISPs, in turn, connect to the NSPs, who collectively provide the backbone connectivity services for the Internet. Many of the NSPs and regional ISPs today are national in scope and several have global reach. They are often affiliated with major carriers because access to fibre optic systems is critical when building large high-speed networks. These backbone providers exchange traffic and routing information at specific exchange points (illustrated as network access points or NAPs and metropolitan access exchanges (MAEs) in Figure 2). So how do all these stakeholders fit into the general landscape of the Internet infrastructure, as we know of today, as well as the specific case where active nodes exist together with passive ones? We look at these questions in the following sub-sections.

### 2.2.1 From dial-up to backbone

Within the UK, call minutes for dial-up IP traffic constituted 25% of all local call minutes on BT's network by the end of 1999 [35]. For most ISPs, the provision of dial-up connectivity at the local level relies on interaction with a number of other stakeholders of the network. In general, dial-up Internet access is supported by ISPs where end-users are connected to the public Internet via the public switched telephone network (PSTN). Because the majority of the lower tiered ISPs possess no local loop infrastructure, and only limited point of presence (POP) infrastructure, they have been forced to rely heavily upon the networks of incumbent local exchange carriers (ILECs) and other licensed operators like new-entrant telecommunication operators for the delivery of their services.

\(^4\) We consider end-user base and financial strength as a measure of relative magnitude.

\(^5\) The diagram presents an idealised abstraction of the Internet structure [105]. In reality, the delineation between wholesale and retail is unclear due to the diversity of end-user connectivity and its ability to undertake multiple upstream services.
2.2.2 Active Network providers

Active Networking technology provides ISPs with a new revenue stream. By incorporating active routers (hence establishing an Active Network within its WAN), ISPs offer content providers the ability to customise the network 'on the fly' for the delivery of a service (e.g., video on demand), without delays due to lengthy standardisation procedures [19].
Thus, a more specific role played by ISPs will be the role of the Active Network provider (ANP). The role of a content provider is introduced as a precursor to Chapter 6. The WebTV (cf. §6.4.4 on the FAIN demonstration) providers play the role of content providers, i.e., they are effectively providers who ‘add value’ to the network as opposed to just basic IP connectivity. Our description of service types in §2.6 provides a clearer distinction.

2.2.3 ISP relationships

Transit service provides access to the entire Internet routing table for a fee, while peering is typically a no-cost arrangement, where each ISP reciprocally provides access only to each other’s customers. A peering relationship between two (or more) ISPs involve physical interconnection, either using a private peering arrangement via a leased line to join both points of presence (POPs), or to hook up with a third-party public peering fabric.
Peering is not a transitive relationship; e.g., although ISP D (see Figure 5) establishes a private peering link with ISP C, the latter will only advertise reachability information of its own end-users to the former. ISP C will not provide ISP D with 'free lunch' such that reachability information to ISPs A and B (as well as those of the Regional ISP) are passed on to ISP D. In other words, a peering relationship means that only the reachability information of the end-users of the peers is advertised. However the mutual benefits are magnified when two NSPs peer with each other since their respective 'end-users' are actually regional ISPs.

If ISP D intends to extend its coverage, then it would either need to connect to the public peering exchange to gain access to ISPs A and B, or to purchase transit service from the regional ISP, or both. Apart from the Tier 1 providers, most ISPs do not have the geographical presence to sustain global connectivity via peering, thus buying transit service is essential. This inevitably introduces a resource allocation and contention problem at uplink bottlenecks.

It is thus concluded that underneath the veneer of the ISP's competitive retail and wholesale environment, every ISP must cooperate\textsuperscript{7} with neighbouring networks in order to provide comprehensive connectivity, as well as to reduce transit cost when a flow destination can be reached by cost-free peering\textsuperscript{8}. This paradoxical relationship between ISPs arises from their physical connectivity as well as commercial interests. As we can see from the following example, a simple traceroute experiment conducted between two ISPs in Malaysia returned pretty surprising results. The path taken by the traceroute packets literally went half way across the world before arriving at the destination, which is physically less than a few kilometres away. This demonstrates that

\textsuperscript{7} In our context (i.e., one that takes account of Active Networking technology), to cooperate implies allocation of basic connectivity, as well as node computational resources for active packet processing en-route.

\textsuperscript{8} Any traffic that can pass between two ISPs should take that shortest path and not be forced to traverse the upstream provider's network.
without a comprehensive peering relationship, end-to-end connectivity becomes inefficient.

This thesis focuses on novel approaches that can provide a better multi-domain congestion control framework and to introduce a protocol that can be used to ensure guaranteed levels of expected end-to-end quality within this fabric of providers. These relationships are further explored in §4.1. The mathematics that illustrates how Hardin’s [71] much-cited tragedy of the commons aptly models this resource contention scenario will be presented. A program running on an active node may consume all node resources, therefore blocking other programs. Therefore, Axelrod’s [22] notion of metanorms is adopted to promote a self-stabilising system within the ‘commons’.

2.3 Route categorisation

The potential traffic and investment asymmetry between ISPs means that careful consider has to be taken when exchanging routing information since accepting traffic from another provider implies additional bandwidth and node processor usage in their own networks. Unless an ISP has excess resources, excessive peering with other
ISPs would incur additional 'cost' in terms of congestion and lower quality of service for the ISP’s own network because there is a resource drain every time a smaller ISP, say, transits with a Tier 1 entity.

Since our inter-domain negotiation (see §6.4) only reserves resources along the existing route set up by whichever underlying routing protocol in place, we need to consider routing issues in slightly more depth – suffice to understand a potential counter-party in any negotiation. Reachability is the definition for the existence of a link between two domains, with the currency of interconnection being routing entries.

![Traffic flows in opposite direction from route advertisements. Peering achieves mutual reachability while transit achieves global reachability](image)

An ISP forwards traffic based on routing information that it obtains by means of routing advertisements. Route advertisements and traffic flow move in opposite directions. We categorise the types of routes that an ISP has in its routing table by the manner in which this routes are acquired.

The ISP’s internal machines advertise service routes used for providing basic end-user services such as HTTP caches, SMTP servers, and DNS. These are solely for the access of the ISP’s end-users. End-user routes correspond to the ISP’s own customers. These routes could either be dynamically assigned (for dial-up users) or...
statically configured (for corporate customers who have leased line contracts). Transit routes are obtained from upstream providers. Indeed, these routes are arguably the most important category since it enables an ISP to connect its end-users to the rest\(^9\) of the Internet. Peer routes are the routes that an ISP obtains from its peers via a cost-free arrangement.

Associated with transit and peer routes is a list of identifiers for domains traversed by the route, known as autonomous system (AS) path [125]. Service and end-user routes obviously do not have this list of identifiers because they are located within the same AS. ISPs could put policies in place to handle the traffic flows and route advertisements. A domain realises these policies by independently selecting, and selectively propagating the four different types of routes described above.

### 2.4 Domains

A domain is defined in the context of this thesis so that it intuitively leads to our perception of what an inter-domain relationship entails. Part 19 of ISO/IEC 10164 [119], which describes the international standard on policy and domains, defines a management domain as a set of entities grouped by means of a specific functionality or service, while management policies are a set of zero, or more, rules that when enforced yield a desired result. On the other hand, domains can also be segregated by virtue of their ownership by a variety of different organisations. We define the former as functional domains while the latter, which is of most interest, will be aptly called administrative domains, which are an autonomously administered portion of the Internet’s routing fabric, \textit{i.e.,} each domain signifies an ISP’s ‘territory’ that varies in size and geographic extent.

\(^9\) As extensive as the upstream provider can provide
Our view of an administrative domain is derived from that of the Telecommunications Information Networking Architecture Consortium (TINA-C) definition [137], where an administrative domain is viewed as a collection of resources that are under the control of a single network administration, hence providing a clear delineation in terms of resources ownership. This notion is fundamentally important since the Internet may have parts under the control of different administrations, and these administrations have to cooperate with each other (e.g., exchange routing information and negotiate guaranteed resources) to provide a comprehensive service to each of their end-users. To cope with the (technical and commercial) complexities of larger scale networks, a network administration should have a detailed view of resources in its domain, and only an abstracted (or summarised) view of resources in other administrative domains.

The complexity of accessing resources in other administrative domains impedes provisioning of end-to-end services [72]. This is especially true when active packets need to be processed on nodes other than that of the sender’s ISP. While security issues [100] probably present the most challenging research agenda in order for AN to be properly rolled out over the Internet, we focus on the resource allocation issues at the node level. The basic objective here is to study how cooperation between peers, if appropriate, can regulate resource contention.

Nichols et al [113] observed that multilateral agreements rarely work. Instead, end-to-end services need to be constructed out of purely bilateral agreements. It is hoped that inter-domain hop-by-hop relationship is achieved by allowing any two or more domain managers (akin to bandwidth brokers) to perform automated negotiation prior to service provisioning.
2.4.1 Domain types and nodes within

**Stub domains** carry only traffic originating from or destined for an end-user within the domain and typically have a local area network (LAN) or wide area network (WAN) that collocates an **edge router** with an access concentrator within an ISP's point-of-presence (POP). On the other hand, **transit domains** (i.e., that of regional ISPs and NSPs) may carry traffic that neither originates nor terminate within the domain. The edge routers are termed **egress** or **ingress** nodes depending on the direction of flow that passes through it. Stub domains are usually comprised of high-functionality systems that terminate customer-access links from the high-capacity systems that drive the high-speed forwarding in the transit domains. This helps the former provide service differentiation while keeping the transit domain mechanisms simple and aggregated.

Ideally, the management system for a single domain should passively participate in inter-domain (e.g., Border Gateway Protocol or BGP [125]) and intra-domain (e.g., Open Shortest Path First or OSPF [110]) routing protocols. Its monitoring component learns about changes that occur within a domain so that the management system and the managed nodes share the same topological view of their network.

2.4.2 Segregation within domains to enhance scalability

Scalability remains one of the outstanding issues for the QoS research community as the growth of the Internet and the World Wide Web has significantly increased the amount of online information and services available. As with the PSTN, the key to solve the scalability problem in the Internet is to maintain strict levels of hierarchy, commonly referred to as **core**, **distribution** and **access** levels. This helps to limit the degree of meshing among nodes. While the core portion of the hierarchy is generally considered the central portion, or backbone of the network, the access level represents the network edge.
The core router’s primary task is to achieve packet switching rates that are as fast as possible. This is done by assuming that all security filtering has already been performed and that the packet to be switched had been forwarded to the core router because it conformed to the access and transmission policy of the network. On the other hand, access routers have to terminate a number of customer connections, typically of lower speed than the core routing systems. One could identify QoS as almost synonymous with policy, i.e., how certain types of traffic from specified points of origin are treated in the network. It is critical to understand the impact of implementing policy at various points in the network topology. In most cases, a much larger percentage of traffic transits the network core than transits any particular access point (or network edge), so implementing policy in the network core has a higher degree of impact on a larger percentage of the traffic.

Nonetheless, it may also be argued that less information pertaining to a single user’s flow is available at the core, and thus, customer-specific policies are more difficult to implement, i.e., policies still need to be enforced at network edges. This is a service engineering decision and how an ISP addresses it will be implicitly related to the advent of policy translation initiatives.
Best-effort data services provide limited revenue-growth potential for ISPs. However, by implementing end-to-end QoS controls, they can offer bundled services supporting data, audio, and video traffic, and create multiple revenue streams from their network. Standards compatibility and the adoption of next-generation technologies are enabling operators to provide measurable Service Level Agreements (SLAs)\textsuperscript{10} across the cable network and support real-time applications such as voice-over-IP (VoIP), streaming video, and gaming and entertainment applications. ISPs can also support robust business applications for their corporate end-users, including VPNs, corporate video, and remote access.

There is also an inherent problem in the way the SLA parameters are measured by IP service networks. Today, the most common method of service measurement utilises the built-in counting mechanism within the IP router, where each router tracks the number of dropped packets. However, this only provides measurements within the IP backbone; and there is no end-to-end calculation. How can the ISP commit to quality on the ‘last mile’ if the local loop (which is under the network operator’s control) is not part of the IP backbone? This question inadvertently brings the focus back to the issue about enforcing polices at the core and edges, hence necessitating three levels of policy granularity: end-to-end, network-level and element-level.

Network latency (or transfer delay) is also similarly measured from edge-to-edge, not from end-to-end, where most of the congestion frequently occurs. End-users, however, demand end-to-end service. Service availability is also measured with outdated technology. For example, sending an ICMP ‘ping’ request from the edge router to customer-premise equipments (or vice versa) is the usual assessment for availability.

\textsuperscript{10} Muller [111] described the information used to manage the networked computing environment at multiple levels. At the highest level there are SLAs that describe services provided to end-users, both in qualitative and quantitative measures. At the lowest level of information is the device configuration information that is specific to each device, which describes how and what the device is to do on the network.
This technique only tests the line, though, and not the end-to-end service. Therefore an IP services network must somehow provide a way of measuring an end-to-end SLA.

2.4.3 Agency relationships

Proceeding with an abstract view of the Internet takes us a step closer to viewing the Internet as a multi-agent system [153]; where a number of agents interact with one another, typically via a signalling phase, *i.e.*, exchange of control messages. As was noted in §2.2.3, it is necessary to address a paradoxical scenario that emerges from this manner of interaction, whereby negotiations for a collusive relationship occur between competing entities. ISPs collude in order to provide comprehensive connectivity to their end-users and compete when they share a flat-rate uplink.

Methods of research in science can usually be grouped in terms of inductive or deductive process [22]. While induction is the discovery of patterns in empirical data, deduction involves specifying a set of axioms and proving consequences that can be derived from those assumptions. The core of our creative work (in Chapter 4) makes use of what Axelrod calls the third approach of ‘doing science’, *i.e.*, agent-based modelling. Because the study of a large number of actors and the changing patterns of interaction often gets too difficult for a mathematical solution, an alternative tool is computer solution. Unlike the method of deduction, we do not prove theorems and unlike the method of induction, our simulated data come from our specified rules (stemming from our assumptions) rather than direct measurements of the real world.

Designers of multi-agent systems make use of a large number of languages and formalisations in order to encompass all aspects of these systems. Ferber 0[54] grouped these languages depending on the level of abstraction employed to analyse the subject matter. Implementation languages are used for programming the multi-agent system. Communication languages provide for interactions between agents. Describing the
behaviour of agents adds details that are necessary for comprehending the system. Languages for representation of knowledge allow agents to reason and to make predictions about the future on the basis of the data available to them.

![Diagram of agency relationships](image)

**Figure 8: Agency relationships**

To explain why it is fair to view the Internet as an example of a multi-agent system, we first assume that an agent is synonymous with an ISP. Adapting from Jennings’s interpretation [84], in Figure 8, the ISPs are able to act in an environment (i.e., the Internet) where different ISPs would have different ‘spheres of influence’, in the sense that they will have control and management responsibilities over different parts of the environment.

The Internet flows traverse several different network domains. Each ISP would prefer to manage its own network resources and enforce its own internal traffic engineering policies. When a domain routes transit traffic, resources are being consumed. Therefore, some domains might be willing to route some types of traffic but not others. Ideally, a domain should only have to reveal simple delivery commitments to its peering domains. As such, we observe that the relationships among these domains are characterised by the need for competition and cooperation without a common trusted
agent. Our approach of using agent interaction to investigate inter-domain issues is thus justified.

A domain must first establish the end result of its intended negotiation with its peers. Thus, by adapting from Moore’s model [109] of single- and multi-ability agents, it is noted that an Internet domain (administered by a single ISP) can provide end-to-end service to an end-user (i.e., comprehensively meet connectivity and resource demands) if and only if it is the sole provider than interconnects both the source and destination of a flow.

Taking it a step further, a set of domains can similarly provide end-to-end service if and only if it is mutually known that at least a member of this set can meet the requirements at each domain en route from source to destination. In order to negotiate effectively, a domain must be able to maintain a model of its own knowledge and objectives as well as to reason with a peer’s knowledge and objectives. When domains are non-collaborative (i.e., bearing in mind the competitive nature of the ISP industry), the process of negotiation is an iterative exchange of proposals towards reducing conflict and promoting the achievement of their respective individual objectives. The element of trust is implied in our model when, over repeated encounters in negotiation, domains establish an analogue to the social notions of reputation and credibility.

The starting point of decision making theory involves the distinction between choices made by the decision maker, and the options imposed on it by the environment. Wooldridge [153] formalised the environment and actions for multi-agent systems succinctly. It is first assumed that the environment may be in any of a finite set $E$ of discrete, instantaneous states:

$$E = \{e, e', \ldots\}. \quad (1)$$
This can be viewed as the dynamic nature of the Internet (or any subset of it, 
*e.g.*, within sub-networks or bottleneck points), taken as a single unit of environment 
with various congestion levels. The ISPs, in turn, playing the role of agents in this 
environment, are assumed to have a repertoire of possible actions available to them and 
each action that they choose to take will have varying effects on the state of the 
environment. Let

\[ Ac = \{a, a', \ldots\} \]

be the finite set of actions available to ISPs. As a result of each action, the environment 
can respond with a number of possible states. However, only one state will actually 
result, and on the basis of this resultant state, one of the agents will again choose 
another action and so on.

Thus, we can then look at a ‘run’ as a specific temporal instance that captures 
the snapshot of the Internet, which gives us a sequence of interleaved environment 
states and actions:

\[ r : e_0 \xrightarrow{a_0} e_1 \xrightarrow{a_1} e_2 \cdots \xrightarrow{a_{n-1}} e_n. \]

This simple abstraction puts us in the right setting for the task of establishing 
norms in Chapter 4, where actions (whether to punish or not), would determine the level 
of congestion at a given network bottleneck (*i.e.*, environment).

### 2.5 Comparison to the PSTN

PSTN environments have much more effective service management systems 
than do IP environments. The telephone network is more homogeneous than the 
Internet. In the Internet an ISP has greater reliance on another than even telephone 
companies do in the post-deregulation period. In general, we observe that in 
telecommunication networks, the traditional approach to quality of service (QoS)
requires end-users to tightly specify or control their traffic in return for tight QoS guarantees, while the operator turns away excess demand. The Internet offers the other extreme of accepting every demand, giving a vanilla ‘best effort’ QoS and relying on users behaving ‘nicely’ in order to ensure ‘fairness’.

In considering today’s Internet protocol (IP) service provisioning initiatives, it is worth noting its similarities to the fundamental philosophies that underpin those of the PSTN. For instance, the intelligent network (IN) is an architectural concept standardised by ITU-T\textsuperscript{11} and the European Telecommunications Standards Institute (ETSI) that, in principle, allows rapid and simple introduction of new telecommunication services in the network, which are essentially the same motivations for Active Networking technology in the context of IP networks.

On the PSTN, the Signalling System 7 (SS7) establishes the information required to set up and manage telephone calls in a separate network from the one where the telephone call is made on. SS7 uses out-of-band signalling, where control information travels on a dedicated 56 or 64 Kbps channel. We observe that this approach is akin to today’s resource reservation approach for IP packets (cf. RSVP in §2.7.2), in that control messages propagate through the intended service path before actual data packets travel from source to destination.

Thus, we see the impetus for network programmability brought on by the need for efficient service deployment, first in the voice network, and now in data networks. The traditional end-to-end model of interaction in the network is also evolving towards an alternative scenario where the network infrastructure can play a more flexible and proactive role. This is expected to create a global shared network infrastructure with new value-added services for all participants (i.e., end-users, service providers and

\textsuperscript{11} The ITU Telecommunication Standardisation Sector (ITU-T) is one of the three sectors of the International Telecommunication Union (ITU).
network providers). The drive here is to formulate a method where stakeholders of the Internet (at least) adopt a more cooperative approach and agree on a common platform for negotiation; and beyond which, they are free to make use of different strategies based on the constraints of the negotiation protocol.

The view adopted in this thesis is consistent with that of the Telecommunications Management Network’s (TMN) official definition [79], *i.e.*, service management is defined as a task that is responsible for all negotiations and resulting contractual agreements between a (potential) customer and the service(s) offered to this customer. In contrast with the TMN, the approach developed here is that the agents that represent an administrative domain (*cf.* the concept of inter-domain managers in §§6.3 and 6.4) negotiate for resources as required per service on behalf of their customers. The rationale is that these agents are expected to have some knowledge of a domain’s priorities and policies and allocate bandwidth with respect to those policies. It is believed that independent traffic labelling by end-users presents a scalability problem; and is unlikely to be sufficient because this would also mean that end-users would need to know their entire domain’s priorities and current network (link and transit node) resource consumption in order to always mark their traffic appropriately.

In accord with the telecommunications information networking architecture (TINA) principles [137], the programmable networks (*cf.* §3.1.3) paradigm also aims to define reusable components from which new services can be built. Both architectures also provide technology-independent mechanisms to establish, modify and release basic connectivity and resource reservations. The IEEE P1520 Work Group [77], which is essentially a programmable networks initiative, address the use of distributed computing (*e.g.*, Open Distributed Processing or ODP and Common Object Request Broker

---

12 We refer to the customer as end-users throughout this thesis (*cf.* §2.2).
Architecture or CORBA) to overcome the inability of centralised systems to scale when
the number of services and service sessions escalate – in the same way as how TINA
defines a distributed processing environment (DPE) to provide support for network
nodes to locate and interact with each other.

2.6 Service types

The dichotomy between high-level services that are evident to the end-user
(e.g., video conferencing, VoIP, web hosting) versus basic services that are diaphanous
to the end-user (e.g., routing, DNS) is loosely described. The former will rely on various
forms of the latter, i.e., for a high-level service to be properly deployed there is a need
for the basic services to be operating effectively. Our understanding of basic services is
directly consistent with network availability, i.e., fundamental transporting of packets
from one host to another thus enabling end-to-end communication. For instance, routing
and scheduling are essential to establish the minimum QoS required to run a video-
conferencing service. The advent of Active Networking technology meant that services
would be deployed in a more sophisticated fashion. For instance, content providers can
now reserve computational resources on specific active nodes that suit the requirements
of the offered service. Thus we feel that the service negotiation protocol would benefit
from a consideration of parameters like CPU time and memory.

The definition of ‘service’ in this thesis is in line with that of the FAIN service
model (see A.7). The underlying entity in this service model is the service component,
which is a piece of self-contained software that represents the smallest unit of
deployment and management. Thus, a service is referred to as a unit of functionality that
a content provider intends to offer its end-users (customers). This functionality is
realised by a combination of one or more service components. Service components may
be recursively composed of sub-components. The goal of this approach is to enable
flexible deployment and management of services at a suitable level of granularity, as well as to deal with services running in multiple execution environments (EEs, see §3.2 for description of the active node architecture) with varied capabilities and underlying software technologies on possibly heterogeneous active nodes.

2.7 QoS approaches

It is the view expressed here that all quality of service (QoS) initiatives can be loosely categorised into three approaches. Firstly, we have alternating physical paths. In this context, a backup path is established in case the primary path fails. We can also imagine traffic engineering efforts that create paths with different profiles used by source-based routing, where best effort service is provided via the low-bandwidth path and high priority traffic gets routed through the high-speed path.

The other two approaches are grouped according to the targeted OSI layers, i.e., link-layer mechanisms (e.g., ATM and frame relay) and network layer mechanisms like IP. In the Internet, when an end-to-end path does not consist of a single pervasive data-link layer, a packet will inevitably traverse links that cannot provide any type of differentiated service at the link layer, rendering an effort to provide QoS solely at the link layer an inadequate solution.

2.7.1 Current design issues

Recent approaches to differential QoS rely on segregation mechanisms provided by the data link or network layer. Packet flows are physically differentiated by means of tagging the respective header bits (e.g., differentiated service, MPLS) or creating virtual paths (e.g., ATM). This is, in fact, what is known as the ‘network problem’, i.e., how to appropriately mark packets. This is in contrast with the ‘user problem’ that focuses on designing good strategies (see Chapter 4) to achieve desired user-specific goals.
Varying opinions exist on where service differentiation can be provided most efficiently within the network topology. We concur with Ferguson [55] that the most appropriate place to provide differentiation is within the common denominator, where common is defined in terms of the pervasiveness of end-to-end deployment in today’s networks. In this vein, the issue is down to observing the most prevalent technology in the end-to-end traffic path for a packet, and the TCP/IP suite is the obvious choice.

![Figure 9: A good reason why service differentiation should be targeted at the OSI network layer – the pervasiveness of IP](http://www.cisco.com/warp/public/cc/pd/iosw/prodlt/moqcs_wp.pdf)

Figure 9: A good reason why service differentiation should be targeted at the OSI network layer – the pervasiveness of IP

In stating the obvious, routers are special-purpose computers used to interconnect networks and are the building blocks of the Internet. However, since they are busy handling network flows, such computers do not include application programs. They only have special-purpose software related to the job of interconnecting networks. Some router vendors have enhanced this software in order to provide guaranteed QoS. Nonetheless, whether these extended mechanisms are needed to provide QoS is a constantly debated issue. One opinion is that fibres and wavelength-division multiplexing (WDM) will make bandwidth so abundant and cheap that QoS will be automatically provided [82]. This view is indeed worth its weight seeing as how larger

---

ISPs are simply peddling as fast as they can to meet the growing demand for bandwidth by simply throwing bandwidth at congestion problems (cf. Global Crossing’s large investment in optical fibres). Nonetheless, we feel that a smarter way of working around the problem is to derive a more sophisticated manner of policing between different traffic flows that access an upstream network, as we further articulate in §4.3.2.

When the network biases available resources toward QoS traffic, non-QoS traffic receives resource levels (presuming constant available underlying resource), which in turn leads to an average lower service profile [55]. If the subscription fee remains the same, the resultant situation can be deduced as ‘constant pricing, but with declining expectation of service levels for the non-QoS customers’. What considerations have been given to non-QoS customers to avoid ending up in such a predicament? This question interestingly brings forth a solution that in fact, eases congestion by means of differential pricing, called the Paris Metro Pricing (PMP) approach (we further elaborate this in §5.2).

### 2.7.2 Integrated services

The philosophy behind the integrated services (IntServ) model [39] is that routers are required to reserve resources in order to provide per-flow QoS assurances with dynamic resource reservation. Applications requiring guaranteed or controlled-loaded service must set up paths and reserve resources before transmitting their actual data. In this context, there is a need to provide policy control of individual flows, and regulate their ability to reserve network resources. RSVP [40] was invented as a signalling protocol to reserve resources.

In the current Internet architecture, packet flow-related state information is stored in the end points in what Clark [45] calls a ‘fate-sharing’ approach to network reliability. The IntServ/RSVP strategy for providing QoS presents an evolution to this
architecture, where flow-specific states are required in the intermediate routers. Since the amount of state information increases with the amount of flows, a huge storage and processing overhead is placed on the router. Thus, the IntServ/RSVP architecture does not scale well in the Internet core.

2.7.3 Differentiated service

In differentiated service [34], sophisticated classification, marking, policing, and shaping operations are only needed at the boundary of the networks. Differentiated service is aimed at traffic aggregates that may not correspond to fine-grained flows. ISP core routers only need to have behaviour aggregate (BA) classification. This is advantageous for ISPs because ISP networks usually consist of boundary (edge) routers connected to customers and core router/switches interconnecting the edge routers. Differentiated service overcomes many of the issues that prevented the wide-scale deployment of the integrated services model. It allows the main forwarding path through the core network to be simple, and evolve separately from the policy and admission issues that are pushed to the edge of the network.

The main concept to pick up from the differentiated service architecture is the interaction between its defined domains. This focuses on bandwidth allocation between neighbouring differentiated services networks. There exist \textit{a priori} bilateral business relations between BBs of adjacent domains before end-to-end resource allocation can be set up. Real-time signalling is used only to confirm or activate the availability of pre-negotiated bandwidth, and to dynamically readjust the allocation amount when necessary (according to pre-negotiated policies).

In terms of scheduling algorithms to implement differentiated service, the fundamental element of a best effort, single quality network environment is the first-in-first-out (FIFO) queue. To enable preferential treatment to different flows, traffic
aggregation is conducted and resources are allocated according to a pre-determined ratio. Optimising bandwidth usage, say, for each traffic tranche means that the level of improved performance of one tranche will be matched by an equal degradation of performance by another traffic profile that traverses a network node.

In riposte to the perceived drawback in scalability scaling, the bandwidth broker (BB) mechanism [48] provides admission control and configures the edge routers of a single administrative network domain. For inter-domain network management, a specific contract between peered domains comes into place. These contracts are used by the BBs as input for their admission control decisions.

2.7.4 **MPLS – layer 2.5**

The motivation for multi-protocol label switching (MPLS) is to use fixed length label to decide packet handling [69]. It has fixed length label to decide packet handling and the header is encapsulated between the link layer header and the network layer header. Labels can be piggybacked by routing protocols for faster packet classification and forwarding. With MPLS, label switching paths (LSPs) are set up between each ingress router and egress router pair. Today, MPLS is mainly a backbone technology that enables a scalable routing, as well as a VPN solution. MPLS mainly resides within the core of the IP network backbone. Nonetheless, it is noted that MPLS *per se* does not provide QoS. It is fundamentally an approach used to separate traffic, which indirectly contributes to QoS.

2.8 **Summary**

In describing the problem space we recognised that the Internet has evolved towards a service delivery platform, where best-effort delivery does not sufficiently serve end-user needs. We listed the efforts by the research community to address this issue and took the opportunity outline our motivation for this thesis. We also elaborated
the ownership aspects of various segments that make up the Internet, and explained how Active Networking technology may fit into the current stakeholder hierarchy.

The complex web of relationships between these stakeholders naturally gives rise to a paradoxical scenario where ISPs need to cooperate with one another to provide end-to-end connectivity whilst actively competing for their share of end-users. To set the scene for the rest of the thesis, we further provide definitions for domains and route categorisations. Formalising the Internet and its stakeholders as a multi-agent system, we hope to gradually move on to Chapter 4, where we discuss the interactions between domains in an abstract context - as agents that react within a defined environment.

Because the study of a large number of actors with changing patterns of interactions often gets too difficult for a mathematical solution, an alternative tool is computer simulation. When agents adopt adaptive rather than optimising strategies, deducing the large-scale effects is often impossible, thus simulation becomes necessary. The simulated data will come from a rigorously specified set of rules that defined by the problem space in this chapter as well as the boundaries in a commercial context.

For completeness, we acknowledge that the PSTN has long addressed various topics in service management. Thus, we compared and contrast the different paradigms that exist between voice and data networks, before finally moving on to briefly describe the well-documented QoS efforts in IP networks. From the review, it is our opinion that because services traverse networks with different speeds and technologies, reservation between two neighbouring routing domains is vital in ensuring that bandwidth-sensitive services run properly. We address this issue in §6.4.
Chapter 3

The impact of Active Networks

This chapter helps us understand the impact of Active Networking technology with regards to inter-domain service management, and we begin by comparing the Active Networking paradigm against today's IP networks. We note that in the late 90s, the interest that was generated by the potential outcome of successful deployment of active and programmable networking technology clearly inspired cutting edge research initiatives on both sides of the Atlantic ocean within the DARPA^{14} and EPSRC^{15} programs.

This chapter was written within the initial period of our research with two main objectives in mind: to familiarise ourselves with Active Networking concepts; and to briefly investigate the key drivers for the following period of research. It is pertinent that we understand the additional currency for resource reservation, \textit{i.e.}, node computational time for setting up execution environments (EE) as per user.

3.1 Active Networks

3.1.1 Objective

Active Networks were first described in Tennenhouse and Wetherall's seminal paper [146], where the authors postulated two key benefits from this approach: that it would enable a range of new applications, specifically new Internet services, which would leverage computation within the network; and that it would accelerate the pace of

\[^{14}\text{http://www.sds.lcs.mit.edu/darpa-activenet/}\]
\[^{15}\text{http://www.cogs.susx.ac.uk/projects/safetynet/prognet/}\]
innovation by decoupling services from the underlying architecture. Indeed, after more than approximately seven years of research and prototype development, it may be resolved that Active Networking technology will play a more prominent role in the innovation of ‘on-the-fly’ service creation. It is thus concurred that networks of the future must be more facile, more quickly tailored to specific requirements.

Brunner [42] interpreted Active Networking as a means to enable the network to play the role of a computer, which is simultaneously shared among many parties. This is a paradigm that effectively allows a party to install and run a service on a network in a similar way as installing and executing a program on a computer. Users can program the network by injecting their program applications. These programs travel inside network packets and are executed in intermediate active nodes, thus resulting in the modification of their state and behaviour. This could either bring about a new protocol deployment, or simply a single active component deployed for a one-off use.

3.1.2 Implementation

There are currently two approaches to the realisation of Active Networks, i.e., the integrated programmable switch (out-of-band) approach and the discrete capsule (in-band). The programmable switch approach maintains the existing packet format, and provides separate mechanism that supports dynamic installation of extensions to the routers by downloading programs (i.e., programs and data are carried separately). Packets passing through these nodes request processing by one of the previously installed protocols.

Taking it a step further with the capsule approach, active miniature programs that are encapsulated in transmission frames replace the passive packets of present day architecture. Here, every message is a program. Packets carry the protocol (in the form of a code) with them as they travel throughout the network, and network nodes process
the packet by executing the accompanying code. These active packets are intercepted and executed at each node along the path.

However, Alexander [18] proposed a hybrid system that will include legacy, programmable switch and capsule features in his doctoral thesis. Initially, the switch starts off as a limited programmable switch. It then loads in a core functionality that is likely to include the other mode. This important conclusion confirms that the distinction between the out-of-band and the in-band models is a distraction rather than a central issue in Active Networks. To provision the underlying active node resources for service creation, the Active Network provider (ANP, as explained in §2.2.2) must first ‘extend’ the node in anticipation of future in-coming packets, as how the EE extends node operating system (NodeOS) functions.

3.1.3 Open interfaces in programmable networks

Active Networking demands the equivalent of an open interfaced operating system (OS) to be realised on the network hardware before new applications can be built upon a network node. This is a requirement akin to having an open and programmable Java-based IOS++[16] for routers. That OS will have to support dynamic network service creation, deployment, execution and management in a fully distributed manner. Compared to the out-of-band approach, the programmable network initiatives [32] aim at defining open interfaces for network elements, thus allowing the creation and deployment of new services in the network by exposing the underlying resources, mechanisms, and policies. This definition is in contrast to the earlier one on Active Networks, which highlighted the concept of injecting code into the network to dynamically customise the behaviour and capabilities of network nodes.

---

[16] The Internetwork Operating System (IOS) is an operating system from Cisco that is the primary control program used in its routers.
Calvert [43] viewed programmable networks from the point of network application programming interfaces (APIs), which define a virtual machine that interprets a specific language. The network API for the Internet Protocol (IP) comprises the language defined by the syntax and semantics of the IP header and its effects on the routers on the network; hence in traditional networks the virtual machine is fixed, and the expressive power of the language is limited. Thus, Active Networks can be seen as providing a ‘programmable’ network API: the IP header represents the input data to a virtual machine, whilst packets in Active Networks contain programs as well as input data.

The programmable networks approach, as it is being pursued in the industry, e.g., IEEE P1520 [77] and Nortel’s Open IP [96], involves a more restrictive programming environment, in which the network’s signalling and control functions are programmable, but the data transfer functions are fixed. In contrast, Active Networks permit applications to customise both the control plane and data plane.

![Diagram of data, control and management planes within an active node]

Figure 10: Illustration of the data, control and management planes within an active node

The concept of function planes mean that different sets of functions make use of all the basic services that the active node has to offer. The data plane makes use of the node for actual movement of data traffic and operations occurring real-time on ‘packet path’. On the other hand, the control plane manipulates the active node for signalling in order to set up and tear down reservations. It is crucial for synchronisation and
orchestration of events and is less time-critical, e.g., the RSVP signalling protocol and exchange of routing information like the BGP. The management plane uses Active Networking technology to improve network management process in a specific manner, where typical operations involve fault, configuration, accounting, performance and security (FCAPS) management.

3.1.4 Comparison

‘Passive’ or legacy routers (which are simply packet-forwarding engines) interconnect the hosts and perform computations up to the network layer. In today’s Internet, routers examine the destination address field of the IP header along with the internal routing tables to determine to which neighbour they should forward the IP packet. The extent of user control over the network’s behaviour is thus limited to the range of values that can be assigned to that field in the IP header. These packet-switched networks are closed, vertically integrated systems, with functions rigidly built into embedded software. As such, one will observe that the preponderance of the Active Networking technology lies in the flexibility to create new services.

It is further observed that the central feature that distinguishes a programmable router from configurable ones (i.e., passive routers) is the programming model used. A configurable engine focuses on establishing a maximum set of high-level features that can be activated as single atomic actions. In contrast, a programmable router focuses on identifying a minimal set of primitives from which one can compose an indefinite spectrum of high-level features (e.g., Parlay APIs for service creation).

In short, because traditional networks do not perform computation on network packets, the only shared resource is network bandwidth. Active Networks present an impact in congestion management as congestion now occurs due to contention for CPU time and bandwidth as a result of these two resources being shared.
Another foreseeable impact is that the network bandwidth in active nodes may end up being under-utilised. Our reason is that potential delays at a CPU queue may usurp the processing time for packet routing. As such, the number of packets transmitted over the uplink per unit time is reduced, causing the bandwidth of that output link not to be optimally utilised.

3.2 Node architecture

The active node architecture identifies three layers of code running on each active node.

Calvert’s generic model perceives an Active Network as consisting of a set of nodes (not all need to be ‘active’) connected by a variety of network technologies. Each active node runs a node operating system (NodeOS) and one or more execution environments (EEs). At the lowest level, the NodeOS is responsible for allocating and scheduling the node’s resources (e.g., link bandwidth, CPU cycles, and storage) among the various packet flows that traverse the node. The NodeOS essentially provides a set of APIs to the EE for accessing an active node’s resources.

Each EE defines a particular programming model for writing active applications. It implements a virtual machine that interprets active packets that arrive at
the node and may be thought as extending the NodeOS ‘upwards’ into user space. Each EE implements a set of abstractions using the ‘building blocks’ as provided by the API. To create a service, the content provider (i.e., an ‘inhabitant’ of the EE) manipulates these abstractions as appropriate. Leading examples of EEs include ANTS [150], PLAN [72], and CANES [29].

To enable the co-operation of functionality found in different EEs, the notion of virtual environments (VEs) was introduced. In the FAIN active node reference architecture [28], EEs are treated as technologies (e.g., Java Virtual Machines) used to implement services that, in turn, may operate entirely in one of the three planes: control, management, or transport (also frequently referred to as data or forwarding). A set of EEs is encapsulated by Virtual Environments (VEs). VEs that are connected together provide a proper virtual private network (VPN) on top of the network infrastructure. The VE is an abstraction that is used for the purpose of partitioning the resources of the active node such that different communities of users can stay isolated from each other. A VE is owned by a content provider and may group several EEs assigned to that particular content provider. Thus, a VE is the ‘front door’ to the services running in the EEs of a content provider. There is one special privileged VE owned by the node provider (i.e., the ISP) which oversees the basic node services, like EE management, VE management, bandwidth management, etc. The privileged VE is the main entry point to an active node. When a content provider owns multiple VEs spread over several network nodes the VEs form a virtual Active Network. In order to identify which VEs belong to a virtual network they are tagged with a unique virtual network identifier.

The net effect of this proposed architecture for active nodes, which would be sitting among other passive nodes on the Internet, is the need for a more predictable processing time for each flow profile. While previous research initiatives [88] have repeatedly stressed that multimedia applications require relatively high networking
bandwidth and processing power in the end systems, our view is that the successful implementation of Active Networking technology would mean further processing power required for intermediate nodes.

3.2.1 Active node resource

Present day research on Internet Economics [102] strives to put a value on bandwidth usage. Although not a ‘public resource’, a flat-rate pricing model for Internet has a behaviour that approximates one. After all, there is nothing to stop an end-user from ‘eating up’ its upstream provider’s resources, hence causing the other end-users to suffer degradation in their perceived QoS. The ISPs will face the possibility of a resource allocation problem unless end-users are charged per flow (e.g., electricity usage). As such, it is worthwhile to investigate a similar pricing mechanism (cf. Chapter 5) for CPU cycles, where the arbitrage theory in financial markets is adapted.

Galtier et al [60] looked at the problem of expressing meaningful processing requirements among heterogeneous nodes in an Active Network. They suggested that the sender of an active application include: (1) a matrix of average CPU utilisation time for each state change; and (2) its corresponding probability for each change of state\(^\text{17}\). Thus, in order to reserve CPU time, the content provider (using an ANP’s resources) must indicate the beginning and the end point for its state transitions within an active node. In an inter-domain context, the ANP must negotiate (by presenting (1) and (2)) with its neighbour before committing to offer the active infrastructure for end-to-end service to the content provider.

Current routers provide relatively fixed set of processing functions for different types of IP packets. Since the per-packet processing requirement is well known, and the

\(^{17}\) The behaviour of a an active application within an EE can be viewed as a series of transitions between specific states where each state is a NodeOS system call.
router (obviously) knows its own processing capability, it is possible to estimate the CPU process time for a certain type of traffic flow.

In Active Networks, as long as a new packet class can be comprehended by the EE, it can be injected into and run on a node. In such cases, the active router would not know the amount of CPU time required for processing this new class. This mandates the need for a requestor (e.g., the ANP on behalf of the content provider) to indicate to the requestee the average CPU time required for each active application as well as the variance of the requested time. For this information to be meaningful to the requestee, the type of statistical distribution assumed for the variance should be mentioned as well, e.g., exponential, gamma or log normal. The Xenoserver project [58], on the other hand, relies on economic feedback to users to control application resource consumption. The distributed resource management setting investigated how an entity may delegate process time control over short timescales but which requires re-authorisation beyond a certain resource budget.

Banga et al [25] proposed a ‘resource container’ that groups an end-user’s network bandwidth, CPU cycles, and memory buffers such that input-output buffers used to queue messages and CPU cycles consumed are ‘charged’ to the container. Thus each end-user’s service flow has a container. Each container contains a single thread pool that is initialised when the container is created. Several parameters are specified when creating a thread pool, including the maximum number of threads in the pool, the scheduler to be used, the cycle rate at which the pool is allowed to consume the CPU, the maximum length of time a thread can execute between yields, the stack size for each thread, etc.

---

18 Since an application that requires 100 CPU seconds, say, to execute on a 10 million instructions per second (MIPS) node processor might require only 10 CPU seconds on a 100 MIPS node processor. MIPS is typically associated to the Dhyrstone [148] benchmark. Refer to [60] for a comprehensive list of variability in CPU time usage for different types of node technology.
The key issues are briefly summarised as follows. When a packet arrives at an active node, it enters the CPU queue, where it waits for the CPU to execute its code as contained within the data portion of the packet. After code execution, the packet enters the output queue to be routed to the next hop. When computationally intensive functions are deployed at active nodes, the CPU would naturally spend most of its time processing the packets. Since all active packets that enter a node would first enter the CPU queue before being forwarded to the next hop, delays are likely to occur, adding on to the more familiar factors for congestion, i.e., limited bandwidth. The net effect is further degradation of overall performance — an irony, seeing that our initial motivation is to improve service provisioning, which relies greatly on a more efficient performance by the intermediate network nodes.

3.2.2 Active packets

An Active Network packet consists of two parts: the header and the payload. The header of the packet contains information about: the protocol being used, the target handlers (entity that would process the packet on intermediate nodes), the sender, the receiver, and device drivers. A target handler can be an EE, a classic IP router stack or a dynamic handler, for example a flow.

Active Networks Encapsulation Protocol (ANEP) [20] is the protocol used in the case when the target is an execution environment. Currently, each implemented EE has a specific ANEP identifier assigned to it. While the header has a specific format, the payload is of arbitrary content. The payload may contain data or pieces of code or both. Its length can vary from zero to the maximum length that will keep the whole packet within the length allowed by the underlying transport service’s maximum transfer unit.
3.3 Management of active nodes

Based on experiences in FAIN, there are two main areas of interest in the combined research area of network management and Active Networking: the need for a management system that can manage an Active Network; and the role of an Active Network to reduce the load on any management system and by doing so, the management process is improved in a specific manner compared to non-active approaches.

Since the former represents a slight evolution in the simple IP forwarding paradigm, the capabilities and responsibilities of routers are increased. This calls for corresponding enhanced requirements when managing Active Networks. Currently, every node on the Abone [29] runs Active Networking daemon 'anetd'. It is an experimental software designed to support the configuration management, operation and control of Active Networks. There are three important aspects associated with the management of Active Networks: node resource management, configuration and management of active packets that essentially results in the monitoring and debugging the active messages flowing through the network [42].

On the other hand, as stated earlier, network management has frequently been mentioned within the research community as a natural application for Active Networking. For instance, Bell Labs used ABLE [93] to demonstrate congestion avoidance as a proof of concept for benefits of Active Networking, while on-going work at GE CRD [61] developed Active Network algorithms that allow active, intermediate network nodes to predict their own behaviour and that make it possible for network equipment vendors to provide active logical processes that accomplish prediction for their devices. This innovation will make possible predictive network configuration and management. The SmartPackets initiative at BBN Technologies [134] looked into diagnostic reporting and fault detection.
Why should active nodes be managed differently? The answer should first deal with the comparison between the conventional (SNMP-based) management (it is irrelevant whether it is policy-based or not) of active and passive nodes. Conventional element-level administration usually handles configuration and fault management manually, using generic remote access capabilities (e.g., telnet in Unix), rather than a specialised protocol. Some management applications do use SNMP to access and manipulate settings. However, configuration management functions are often too complex to handle via SNMP and are thus accomplished by scripts at the element under the control of the network-level management station.

Consequentially, it is intuitively extrapolated from this explanation that the use of unstructured and ad-hoc manual approaches in traditional element management is a significant barrier to the efficient, secure and robust operation of Active Networks. As such Active Networks need to be managed differently.

When considering the limitations of current management technologies in handling Active Networks, consider using SNMP to manage an Active Network. Some requirements are necessary. When an active element is loaded into a node, it is necessary to load respective instrumentation (e.g., specific agent software to interrogate the MIBs) and MIB components and integrate these with the element management software. It is also necessary to load similar MIB components into the management station to enable management application tools to access the enhanced feature on the active node. These dynamic changes at both the element management level and the network management level will have to be synchronised and coordinated with the dynamic changes in the Active Network.

In contrast with passive, traditional network applications, which are entirely separated from management software, active applications will need to integrate monitoring and control capabilities. The traditional approach to network software
design has been to incorporate function-specific monitoring and control capabilities with every protocol/network-system. Active Networks require an integration of management mechanisms and application software. This requirement has also been reflected by the management-by-delegation architecture of having an elastic server and elastic processes [67] running as agents on managed elements. In FAIN, the management nodes were developed in parallel with the core objective of active node development, and element-level management functionality was physically located as close as possible to the active node.

3.3.1 State-of-the-art review

There are several research projects that cover the management of Active Networks. Some of these projects use programmable network techniques so as to achieve more efficient management. Some of these research projects are reviewed in order to demonstrate the wide range of solutions proposed.

ABone [29] is a DARPA funded virtual testbed for the Active Networks research program. It is composed of a set of computer systems configured into virtual mesh of active nodes. The ABone nodes are administered locally, but can be used by remote users to start up execution environments (EEs) and launch active applications (AAs). Each core ABone node is configured with seven Unix specific accounts. Each account runs an instance of anetd – the Active Network management daemon. These daemons allow remote EE and AA developers to install, configure and control EE instances in these nodes. Anetd performs two major functions: deployment, configuration and control of network software, in particular EE prototypes; and demultiplexing of Active Network packets encapsulated using the Active Network Encapsulation Protocol (ANEP) to multiple EEs located on the same network node.
The Active Bell Labs Engine (ABLE) [93] initiative proposes a novel Active Network architecture, which primarily addresses the management challenges of modern complex networks. Its main component is an active engine that can be attached to any IP router to form an active node. The active engine is designed and implemented to execute programs that arrive from the network. The engine facilities and executed programs are oriented to the monitoring and control of the attached router. The active code is implemented in Java and active packets are also encapsulated in a standard ANEP header over UDP. ABLE offers an efficient access to the local state of the router, a secure system to modify the router behaviour as well as easy to use programming abstractions and interfaces.

The Active Virtual Network Management Prediction (AVNMP) [61] algorithm is a proactive management system. It provides the ability to solve a potential problem before it impacts the system by modelling network devices within the network itself and running that model ahead of real time. Predictions range from network performance to possible network or node faults. Such a proactive management approach is particularly useful in many applications. For example, in the case of handovers in a mobile environment, if the handover is prepared in advance, the service quality degradation is minimised. Similarly in QoS-sensitive applications, particularly those that are affected by an excessive or variable delay, the management system can avoid congestion before it actually happens.

The system is composed of different types of active node with different targets. Some active nodes realise predictions based on the information they have and publish it on the network. These predictions can either be about the network or about an offered service. Then a second type of active nodes captures these predictions and introduces them into the management algorithms which have been implemented. The algorithms basically compare the actual state of the network with previous predictions. If a
previous prediction was incorrect the configuration actions caused by this prediction are removed from the network. This correction is done through special kinds of messages called 'anti-messages'.

The team at BBN technologies developed Smart Packets [134] to focus on applying Active Networks technology to network management and monitoring without placing undue burden on the nodes in the network. The management applications developed are oriented to diagnostic reporting and fault detection. The framework is based on active packets carrying programs that are executed at nodes on the path to one or more target hosts. Smart Packets programs are written in a tightly-encoded safe language (spanner) specifically designed to support network management and avoid dangerous constructs and accesses. The spanner code is obtained after compiling the program written in a high-level programming language specifically created for the project, called sprocket. Smart packets are generated by management or monitoring applications and are encapsulated in ANEP. The ANEP daemon is responsible of receiving and forwarding smart packets correctly. Security is achieved through the limitations imposed by the tightly-encoded safe language and through a prudent execution of smart packets code: if the virtual machine does not know how to proceed with the code, then it stops the execution. Additionally, further security checks are realised such as user authentication and data integrity checks.

The main objective of the Smart Environment for Network Control, Monitoring and Management (SENCOMM) [80] framework is to implement a network control, management and monitoring environment using Active Networks. SENCOMM continues from Smart Packets since it reuses much of the Smart Packets system. User-written network management and monitoring programs generate smart probes, which are encapsulated in ANEP frames. The probes are demultiplexed to the local SENCOMM Management EE, which injects the smart probes into the network. A probe
can be sent to be executed only at the destination or at every active node running the SENCOMM Management EE, and measurements and control operations might be taken in a single packet’s traversal of the network. The probe contains directives to access loadable libraries of functions on the node, registers to receive incoming packets that meet a filter specification, and optionally inject the packet back into the network. Probe packets can be sent either to unicast or multicast addresses. The information content returned by probes to the management centre can be tailored in real-time to the current interest of the centre.

The Virtual Active Network (VAN) management framework [42] allows customers, on the one hand, to access and manage a service in a provider's domain, and, on the other hand, to outsource a service and its management to a service provider. VAN supports generic, i.e., service-independent, interfaces for service provisioning and management, and customized service abstractions and control functions, according to a customer's requirements. Only two types of EE exist in the management architecture: the management EE that works on the management plane, and the service provider EE that works on the data transfer as well as on the control plane. The tasks of the management EE are limited to node configuration and the management of virtual Active Networks in the Active Network provider’s domain. Note that in this context VAN management means the creation, modification, monitoring, and termination of virtual Active Networks. The management EE is not concerned with the management of active services running in the virtual Active Networks. In the VAN architecture, a service and the corresponding service management run in the same instantiation of a service provider EE.

3.3.2 Analysis of Active Network management using a constitutional framework

In this subsection, we discuss the roles involved if active nodes are to be managed using a policy-based approach (cf. §6.1) by presenting its analogy to a generic
constitutional framework. The Alfebite project\(^{19}\) represents perhaps the most relevant and interesting recent initiative in this area of work, where they investigated the application of formal models of norm-governed activity to the management and regulation of interactions between the Internet (which they referred to as Universal Information Ecosystem) stakeholders.

The powers created under the constitution are distributed among three main arms of a government – namely, legislature, executive and judiciary. While the legislature is charged under the constitution to make laws, the executive is charged with the implementation or execution of such laws. The judiciary on the other hand is to serve as an arbiter to interpret and declare what the law should be whenever there is a constitutional dispute between the other two branches.

With regards to the executive branch, numerous executive organs could be classified on the basis of different criteria according to the: territorial sphere of their competence; range and the character of their prerogatives; method of their formation; and decision-taking process.

\(^{19}\)http://www.iis.ee.ic.ac.uk/~alfebiite
The legislature is state descriptive; the Judiciary is state prescriptive while the Executive is stateless. The three types of agents can be identified as: constative, performative, normative. A performative agent can claim that it has carried out some action - executive function. A constative agent can judge if the action has been carried out - judicial function. A normative agent determines which performative agent should do what task and which constative agent should verify it - legislature function.

For relevance and context, consider the following example of a service-provisioning model, which illustrates how the agents could interact. After negotiating with the relevant ANP, policies (to be further explained in §6.1) are specified for the node resources that may be utilised at any single instance of time for particular classes of users. These policies hold the legislative role. Thus, the legislature also allows for error margins in excess use of bandwidth, CPU cycles, etc. The policies actually in place at a particular node will vary. Implicit in the design of the management system are norms of behaviour expected from the content provider. The local policies within an
administrative domain can be adapted to the behaviour of the content provider at a particular node (or maybe a set of nodes) when the active node and its management system are bootstrapped.

Thus, in the management of active nodes, various parties are allowed to perform computation as well as communication tasks on the node, there exists several performative and constative agents. When an ISP attempts to effectuate a new set of policies, a performative agent initiates a process that will invoke a constative agent, *i.e.*, the ANP, to determine if the content provider would exceed its quota in provisioning for network resources via the policy rules. The only normative agents are the policy databases that store the local policies.

### 3.4 Expectations

To recap previous discussions in the context of network operators’ expectations of Active Networking technology, the following set of contention highlights informal discussions with several network operators, who were fellow partners in the FAIN project, *i.e.*, Deutsche Telekom, France Telecom and KPN.

#### 3.4.1 Speeding service deployment and service customisation

Due to the vertically-integrated nature of IP routers, the deployment of new services and novel features in the functionality of network elements by the network operator requires a lengthy standardisation phase and technology diffusion phase before being operational in its network. Standardisation is a necessary step in network design to ensure interoperability, as network’s utility increases with the number of interconnected nodes. Since today’s Internet architecture mandates the implementation of IP in all routers; and requires up to 8 years [135] for the process of standardisation to move on to development and finally to the deployment process (*e.g.*, IETF to Cisco to ISPs), it is inflexible and evolves too slowly. By enabling networks to be programmed
based on standardised interfaces, Active Networks can also be used to modify an existing service on run-time, thereby implicitly reducing vendor dependence. This property is useful for an operator to offer personalised customer services. There are various ways to pass the user requirements and the modifying code to the appropriate nodes and resident services, e.g., by means of (router) plug-ins [47]. An interesting example of service customisation using Active Networks is multicasting [88].

3.4.2 Leveraged network and service management

Key mechanisms to achieve scalable solutions and cost reductions in terms of network and service management include outsourcing and distribution of management tasks. These aspects will be illustrated by reflecting upon the deregulated wholesale telecommunications market. From a regulator’s point of view, ex-PNOs that now dominate the network provider market are forced to open their network infrastructure to third-party service providers on the basis of the European Union’s Open Network Provision [145] directive. This directive mainly addresses the traditional telephony networks (or POTS) but its applicability and relevance to the data communications infrastructure in general is to be anticipated because many (if not all) NSPs either have affiliations or play the role of the operators. The Active Networking concept allows a network operator to delegate full management responsibility to the third party service providers, thereby complying with regulatory demands and simultaneously avoiding the management overhead.

3.4.3 Diversification of services and business opportunities

Due to the new regulatory environments and market demands, networks operators need to invent new services and break free from their traditional limits in terms of the types of services they can provide. Existing networks have limitations for introducing new types of services. For example, an operator can provide video-on-
demand services with an acceptable QoS based on resource reservation [40], but problems arise if the operator wants to extend the capability of its network to support more users or customers. In addition, providing VPN services to client companies with varying QoS requirements is not a trivial task although operators see this as a reasonable business opportunity. By adapting network services to the needs of target applications operators achieve fine-grain control and management for service provisioning.

3.5 A new role for stakeholders

Attention is drawn to the fact that an end-to-end resource reservation will include domains that have both active and passive sub-networks. Should traffic traverse an NSP's (cf. §2.2 for reminder on definitions for stakeholders) domain, since the network within forms the core of the Internet (i.e., with high-speed forwarding of aggregated traffic), it is fair to assume that the nodes are predominantly passive; hence NSPs do not play the role of an Active Network provider (ANP). The ANP provides facilities for the deployment and operation of active components into the network. It provides Virtual Environments (VE), which are essentially logical spaces for the content providers to deploy service.

As explained in §3.1, based on these functionalities, an ISP will play the role of an ANP, particularly since it offers the ISP an opportunity of a new revenue stream. The ISPs provide Active Networking technology on top of the basic network connectivity to these content providers. A content provider composes services and deploys these service components in the network via the ANP. Even in the case of the simple scenario in the case study in §6.5, the video traffic is effectively propagated across administrative domain boundaries.

20 Specifically higher-level services like teleconferencing and video-on-demand, as opposed to basic services (see our distinction of service types in §2.6).
3.6 Access and allocation strategies in a multi-user system

Access strategies are meant to ensure that if a packet such as an IP datagram travels from one portion of a network to another, then it has some legitimate business there. Access control policies are bound to evolve incrementally over the system life cycle, especially in large systems. The ability to modify a policy to meet the changing needs of an organisation is an important benefit of role-based access control (RBAC) [53]. To date, this topic has not received much formal attention. A notable exception is the work of Giuri and Iglio [64], which defined a formal model for constraints on role activation.

When several different administrative regions are involved, or when the security policy imposes different constraints as a packet traverses a succession of domains, then several edge routers will have to cooperate to enforce the policy. As such, it is important to determine the filtering decisions of individual routers, and these decisions can only be based on local information. While this thesis does not specify the router configuration files that will implement this functional behaviour; it stipulates the logical effects that those configuration files should achieve.

More importantly, the emphasis of this work is to use a mathematical approach to generalise domain access control policies. The generic definitions shown are equally applicable for accessing computational resources on an active node. In complex societies where there exist ample opportunities for collusion, defection and contention, the policies defined in §A.2 model possible strategies for systems that intend to promote their respective goals. These systems can be directly mapped to the agents that take part in the basic norms and metanorms games (see §4.3.2), except that the simulation in Chapter 4 is centred about observing the dynamic nature of the agents' environment when they are allowed to vary their range of attributes and actions.
Apart from access control policies, we have also defined resource allocation policies formally using first order logic. However to preserve the flow of this thesis, further details can be found in A.2 and §A.3. Sergot’s work [128] represents a much evolved and advance work in this respect using modal logic.

3.7 Service examples

In trying to understand the impact brought forth by the Active Networking technology, two use cases that may be taken up by future research initiatives in the area of secure service provisioning were conceptualised. Some new services, e.g., video-on-demand, require precise guarantees from the network both with regard to the timeliness of the arrival of their data and its correctness. Intuitively, apart from over-provisioning, the other alternative is through the reservation of resources on the set of network elements across which data flows. These resources constitute a connection and the means for establishment of a connection is often termed signalling.

In general, two visions of multi-service networks can be distinguished: those that are extensions of telephony and those that are extensions of data transfer between computers. The latter approaches are based on the IP suite, while the former ones assume the ATM standards. With the help of Active Networking technology, the scenario derived in the next section takes the middle path of leveraging on IP’s flexibility and robustness as well as ATM’s precise guarantees about network resource usage. It is further noted that the ATM standards have defined a succession of control primitives between the user and the ingress/egress switch, and between switches.
3.7.1 Scenario for service provisioning using the active node

The ABLE active router [93] below is ready to intercept any packets (with the help of the egress router\(^1\)) from a customer in order to set up a ‘reverse path’ high-speed route for the reply (in the form of a video stream) from the web server. With no known routes in the beginning (except for the final destination, i.e., the web server), the customer (who had earlier signed an SLA with the service provider) sends off a request (Sequence 1 in Figure 8 above). This request will be picked up by an egress router, which forwards it to the active router (2). The egress router and the active router form the Injector system.

![Diagram of reverse-path VPN provisioning using Injector and Interceptor Systems.](image)

The active router strips this request data packet (3), and appends control information that tells the ingress\(^2\) router in the Interceptor system (of the target

---

\(^1\) An egress router (see §2.4.2) is a router through which a data packet leaves one network for another network. Here, the egress router belongs to the end-user’s local service provider and is the final default node before a packet, originating from this service provider’s domain, gets send off into the Internet.

\(^2\) An ingress router is a router through which a data packet enters a network from another network.
network, *i.e.*, domain 2) to forward the active packet to the active router situated near the web server in domain 2. The control information for the data packet is basically translated\(^{23}\) from the policy-based element-level network manager that manages the active node. The active router programs the egress router and ATM Switch 1 using SNMP ‘set’ commands (*4.1 and 4.2*). This sets up and prepares the two machines in anticipation of the return stream of video data.

Subsequently, UDP packets with embedded control data are sent on a reserved port number to the ingress router via the ‘simple’ Internet with its destination field remaining unchanged (*5, then 6*). When the packet arrives at the ingress router, as expected, the router forwards (*7*) the packet to the active router. The diverter in the active node collects it, opens a file and starts writing the Java class contained in the packet to it (*8*). Within the active router, control information is stripped while data is returned (*9*) to the ingress router.

The control information is decoded to generate a Java class; and the Java class is loaded by a ‘fork’ed and ‘exec’ed Java Virtual Machine (JVM). The JVM starts by invoking a SecurityManager and the class being written to the file is loaded. When the class has loaded, the active session begins and the ingress router and ATM Switch 2 are programmed using SNMP ‘set’ commands (*10.1 and 10.2*). The ingress router finally forwards the end-user’s request data to the web server (*11*). The video request will then be transported to the end-user’s PC via this newly established VPN via the ‘fast’ Internet (*12, 13, 14, 15, and 16*) with an implicit higher QoS.

This suggestion of a use case represents early efforts in the development of a useful and novel scenario that takes advantage of the Active Networking technologies.

\(^{23}\) The Ponder language ([46], also see 6.1.1.2) provides a common means of specifying security policies that map onto various access control implementation mechanisms for firewalls, operating systems, databases and Java.
The need for a resilient trust model between the components in the case study is recognised and it is impossible to realise a system without being security-aware.

3.7.2 A secure framework for remote access

This next use case establishes an IPSec tunnel for a user, currently away from office, who wants to access the office network resources (e.g., mail server, and web cache). The key objective in this scenario is to establish end-to-end authentication and to enable the user to securely access the 'office' network without compromising the 'home' network she is logging in from. Firstly, the user sends off data packets (1), addressed to the active router in the office.

These packets are picked up but not recognised by the firewall, and so are forwarded (2) to the adjunct active router. The active router then strips this request data packet, and appends control information that basically 'tells' the firewall in the 'office network' to forward the active packet to the adjunct active router on that side (3.1). The active router also programs the firewall and the IPSec Gateway using SNMP 'set' commands (3.2 and 3.3). This sets up new routes and prepares these two machines in
anticipation of the control data returning from the office. The active router returns (4) the request data packet to the firewall and subsequently, UDP packets with embedded control data sent on a reserved port number to the IPSec Gateway with its destination field remain unchanged (5). Note that this is the only port that can be used for communication between the firewall and IPSec Gateway, and is also known as the Active Networking port.

When the packet arrives (after 6, and 7) at the ordinary router, as expected, it simply forwards (8) the packets to the active router. This packet contains 'clear' data now, since it is already out of the IPSec tunnel. It can be read as control data for the Active Networking port, which (as stated earlier) was the only port allowed to be open. The diverter in the active node collects it, opens a file and starts writing the Java class contained in the packet to it (9.1). The active router also checks the current state of policies that confirms that this user can only use the Samba server, mail server, and web proxy (neither telnet nor FTP allowed).

Within the active router, control information is stripped and the ordinary router is enhanced (9.2) with firewalling capabilities. The active router implements the policies on this ordinary router as firewall rules, thus effectively programming the firewall. This set of firewall rules determines how much of the resources on the private networks can be used by the user, hence helps to keep this private network at the user's office safe. Once the firewall is set up, the data can now be processed. The active router forwards (10) the request data packet to the 'office' network. For instance, this might be a Samba request to mount a disk drive, so the disk drive control information comes back.

The active router intercepts the packet on the way back (11) from the office network and inserts (12) further control data to instruct the active router on the user's end how to program the firewall. This step is essential because the user neither knows nor fully trusts that there are no intruders within the office network. The intruder might
have 'hacked' into the Samba file server, and sees a request coming in from the user. Consequently, the 'hacker' could telnet into the user's current host system. However, the active router is not on the 'hacked' Samba server. This 'independence' allows the former to send trusted, safe control packets back to set up the firewall on the user's side to make it safe during the subsequent interaction between the user (at the remote location) and the private network in the office (13, 14, 15).

The same logic process (16) occurs on the active router, and (step 17) is similar to (step 9.2), albeit with different policies taking effect as firewall rules. The active router finally forwards the data packet from the office network to the firewall, which in turn sends it to the user (18, 19). The user is safe if she calls from her telephone at home because no other machines can talk to her. But if she joins another network, other networks might try to use her machine as a router, hence gets access to her IPSec gateway. The firewall set up in (step 17) will prevent this from happening.

Physically, the firewall, the active router, and the IPSec gateway can all be on the same machine. The user could have a certificate on a floppy disk and as such, will be able to start up the IPSec, i.e., it does not need to check with the CA.

3.8 Summary

This chapter condensed an exhaustive literature survey on the impact of Active Networking technology on traditional IP networks. The main objective of this technology is to accelerate the pace of service deployment by decoupling services from the underlying architecture - not unlike that of PSTN's intelligent network (IN) approach. To understand the innovation involved, the fundamental differences between active and passive networks were compared. Perhaps the most important facet of the

---

24 There are two sets of policies: the first being 'who can join which network' (see §3.6), i.e., there might be many different 'home networks'; while the other being 'who can use what, and when' (see §A.2), i.e., resource allocation strategies. The former is relevant to (step 9.2), while the latter is for (step 17).

25 A certificate authority (CA) is an authority in a network that issues and manages security credentials and public keys for message encryption.
Active Networking paradigm is the ability for software code (that effectuates a service) to be executed on the network routers en-route from source to destination. This effectively meant that future QoS initiatives for networks that include active nodes must also reserve node computational resources, i.e., CPU time, on top of the usual communication resources like bandwidth, jitter and delay.

The impact on network management was also discussed and an explanation was given as to why active nodes may need to be managed differently. Having enjoyed various discourses on the evolution of IP networks in view of Active Networking technology with network operators during project meetings, it was concluded that the incumbents would like to: speed up service deployment and service customisation; efficiently leverage network and service management tasks; and provide additional revenue streams for ISPs. The latter implies a new role for current stakeholders of the Internet.

Opening the network to various parties with different permissions on what they can do on the underlying resources implies that access control and trust management issues must be carefully dealt with. Active Networking technology offers a fundamental change in networking architecture as useful computation may be migrated onto a network node. This means that resources in an Active Network node include both CPU and bandwidth and thus, resource access techniques used for traditional networks does not directly extend to Active Networks.

As such, a set of access strategies was presented, which may be used by active node owners, using first-order propositional logic. To put together what was learnt from the literature survey, two service examples that may be taken up by future research initiatives in the area of secure service provisioning were proposed. By presenting multiple admission strategies, the dynamic runtime environment brought forth by allowing multiple content providers to customise their access to network nodes, i.e., the
VEs, was highlighted and is expected to accommodate the rapid evolution and deployment of networking technologies. This helps to provide the increasingly sophisticated services demanded by Internet users by means of resource reservation.
Chapter 4

Norms in agent interaction

… the rational herdsman concludes that the only sensible course for him to pursue is to add another animal to his herd… this is the conclusion reached by each and every rational herdsman sharing a commons. Therein is the tragedy.

Garrett Hardin, Science Magazine, 1968

In this chapter [141], the interaction between sub-networks that compete for resources of an upstream domain is described. ISPs need to establish a more regulated manner of providing access to the end-user traffic, especially the multimedia streams. When a coordinated activity takes place without a proper central authority to comprehensively police the behaviour of participating agents, the resulting environment is attributed to the existence of norms. The cooperative relationship required for providing end-to-end service is also discussed and the norms that might develop within these contexts are examined. The objective is to stabilise these norms.

The approach is to specify how agents interact and to observe properties that occur at the level of the whole society. In practice, policies and protocols may dictate the manner in which agents interact; and by observing patterns of behaviour, any alignment of dominant strategies is abstracted. A norm exists in a given social setting to the extent that individuals usually act in a certain way and are often punished when seen not acting this way [22]. This is viewed as an emergent pattern of interaction that is perceived to be ‘correct’ by the community of equals that comprise this group, and the goal is to establish this property in the context of a congestion management framework.
4.1 Understanding multi-domain cooperation and competition

Each time we send an email or download a web page, packets will usually traverse more than one administrative domain. This demonstrates the cooperative relationship inherent between domains. ISPs exchange routing information with each other to enhance end-user coverage.

![Diagram showing cooperation between domains A, B, and C.](image)

**Figure 15: Cooperation in providing end-to-end service**

Best-effort routing using any of the exterior routing protocols (i.e., the de facto suite being BGP\(^{26}\)) might mean that a packet is destined to travel from A to E, transiting at B or C (see Figure 16). As with other forms of cooperation, there is a risk of **defection**. In a multi-domain environment, the traffic path depends as much on an ISP's policy (or strategy, as we represent in §A.11) as it does on resource availability. Thus, defection may mean that (although B has peering or transit arrangements with A) B advertises a higher hop count than what it actually takes to get to E; hence packets from A choose to go via C.

---

\(^{26}\) Details of BGP are beyond the scope of this thesis; for context, see Rekhter and Li's work in [125].
What is the temptation for B’s non-cooperation? Defection could either manifest from a strategic action on behalf of an ISP or an error in the BGP protocol implementation (hence explaining the advertisement of a higher hop count to a destination). This strategic action relates to basic services like routing. In terms of resource negotiation for a (high-level) service provisioning (cf. the WebTV case study in Chapter 6) it is noted that defection does not imply being fully non-cooperative, but ‘partial cooperation’ instead. For example, the temptation there is to offer less than the expected amount to meet the end-user’s end-to-end demand for CPU time, say, during the negotiation phase. The implicit nature to defect is derived from a separate relationship where providers compete against each other from a commercial perspective.

ISPs would need to compete with each other for an upstream provider’s transit bandwidth and processing time when they want to route traffic to a destination beyond their respective reach. However, with reference to the bi-directional, dotted arrows in
Figure 17, it makes little sense for two neighbouring ISPs to route each other’s traffic via an upstream provider’s trunk if the source and destination for a given packet flow is within their respective domains. Thus, if the traffic volume between these two ISPs is high, then it is more efficient for them to peer with each other.

![Figure 17: Competition in resource contention, coupled with the benefit of cooperation](image)

This leads to the conclusion that bilateral peering arrangement may only be adopted if the ISPs have approximately the same size, technology and end-user base. This cooperative relationship between them will then approximate that in Figure 15, such that the end-users will realise the benefits of positive network externalities. Positive network externality occurs when end-users reach a larger set of other end-users and information sources and vice versa; while a negative network externality occurs when an additional end-user causes network congestion, e.g., at bottlenecks.

In brief, negative network externalities give rise to the competitive relationship between ISPs while positive network externalities encourage different ISP domains to interconnect, hence cooperate. It is observed that cooperation is the primary relationship that leads to competition. For instance, multiple hosts must first interconnect to establish the Internet before naturally creating rivalry when resources become scarce.
Within an environment for cooperation, if the network externality is symmetrical, then a peering arrangement exists, while an asymmetrical network externality that exists between two ISPs (e.g., a regional and a local one) will naturally establish a transit (i.e., customer-provider) arrangement.

4.2 A basic game theoretic approach

The problem space can either be viewed using the cooperative approach or a non-cooperative approach, which are essentially different ways of looking at the same game. The non-cooperative theory is strategy-oriented and studies the players’ input. The cooperative theory, on the other hand, looks at the expected output.

4.2.1 Reasoning about strategies between two ISPs

By explaining the simultaneous 2-player Prisoner’s Dilemma [23], the interaction between two ISPs can be better understood. In each round, a player has the option to play C (i.e., to cooperate) or play D (to defect). If both cooperate, both earn as payoff a ‘reward’ R, which is larger than the payoff P, the punishment, which they receive if they both defect. But if one player opts for D while the other chooses C, then the defector receives a payoff T (the ‘temptation’), which is larger than R, while the cooperator’s payoff S (for ‘sucker’) is even smaller than P.
In this game, it is assumed that all the players have complete information. This means that each player's payoff function (*i.e.*, the function that determines the player's payoff as a result of actions chosen by both players) is common knowledge\(^{27}\) among all players. Looking at the payoff strategy in Figure 18, a rational player will always play a strictly dominated strategy. Consider the reasoning taken by player 1: if player 2 defects, the obvious action by player 1 is to defect as well, instead of cooperating; that is, player 1 obtains 1 point as opposed to zero. If player 2 cooperates, the obvious action by player 1 is (again) to defect, simply because the latter chooses to obtain 5 points instead of 3.

Consider a thought experiment in which an entire population consists of programmed players. Each of these automata is firmly wedded to a fixed strategy and will either always cooperate or always defect. Looking at two ISPs at any instance, it is reminded that the total payoff will depend on the other players encountered and thus, the composition of the population. For repeated instances of the game, as we have just reasoned, the strategy of defection will eventually swamp the population.

\(^{27}\) Common knowledge cannot be attained in a system where communication is not guaranteed [50].
However, in our context of the Internet, in particular its stakeholders, the same two providers will inevitably interact not just once but frequently. As such, the strategies for the repeated game should obviously change in response to what happened in previous rounds of the game, just as how an ISP will begin to review its own strategy (if not its SLA) with a peer should the latter continually fail to neither keep up its end of the bargain on basic connectivity nor honour its active node resource reservation.

Countless strategies for the repeated version exist [102], and none serves as a best reply against all opponents. Consider the following explanation. If an ISP decides to always cooperate, then its neighbouring domains will do best by always defecting. However, should the ISP decide to ‘cooperate until its peer defects and never to cooperate again’, then the neighbours should be careful not to spoil the partnership. For example, the neighbours must resist the temptation to reject a request for guaranteed resource when a prior agreement has been achieved to adhere to the negotiation protocol (cf. §6.4).

In the language of Prisoner’s Dilemma, the motivation to defect and grab five points instead of three will be more than offset by the expected loss in the subsequent rounds where the ‘cheating neighbours’ cannot hope to earn more than one point. In end-to-end resource reservation, this would mean that once an ISP fails to do its part in a cooperative coterie, peers may reciprocate its greed with constant defection, hence it is unable to offer a better SLA to its own end-users. In other words, it may benefit in the short term because this allows more bandwidth within its own domain to be allocated for other premium services, but in the longer term, it suffers because of its repeated inability to get a peer domain to ACK its BID (again, cf. § 6.4.2) signals for resource reservation.

In conclusion, by extracting from Axelrod’s result [23] when he conducted round-robin tournaments of the repeated Prisoner’s Dilemma on his computer, we
proceed to promote the Tit-for-Tat strategy as the basic principle in a negotiating strategy for computational and communication resource reservation in a peer domain. This strategy starts with a cooperative response and then always repeats the opposing player’s previous move.

We noted earlier that in a game of complete information the players’ payoff functions are common knowledge. Nonetheless, implicit in this assumption is that each ISP knows exactly how it benefits the other ISP if the latter were to defect. We argue that this may not be the case; and in addition, the game does not model how an ISP would respond when more than one peer domain sends a BID message during instances when resources are limited, i.e., when the network is congested.

### 4.2.2 Games of incomplete information

We proceed further with games of incomplete information where a player is uncertain about another player’s payoff function. We propose to incorporate the notion of pricing and a ‘platform’ for enabling auctions, where each bidder’s willingness to pay for a good is unknown to the other bidders and the utility derived by a player is also unknown to the others. Ideally, we want different ANPs to classify their own traffic loads, ‘honestly’ bid for their required consumption, and police each other. In the following scenario, we first assume that the downstream ISPs that are contending for upstream resource will cooperate by adhering to the credit system based on the metanorm payoff matrix (cf. Figure 21, as we further elaborate about metanorms in §4.3.3).

Having agreed upon a protocol for inter-domain resource allocation (see §6.4.2), we analyse the problem of non-cooperative flow control. This general setting has multiple end-users transmitting packets through an upstream bottleneck (see Figure

---

28 Recall that we introduced the Active Network provider (ANP) as role that can be played by any ISP that has active nodes within its network; hence we use the acronyms interchangeably depending on the context.
19), competing for bandwidth within the upstream provider’s domain and CPU time if Active Networking capabilities are required. **The abstraction for this analysis is sufficiently general for consideration of inter-ISP flows.**

While Active Networks offer flexibility in tailoring network services to applications, one of the drawbacks (as we have explicitly noted in §3.1.4) is the possible degradation in overall application performance when multiple traffic streams from various content providers simultaneously contend for resources at an active node. An ISP’s network is susceptible to ill-behaved packet sources that can generate packets at high rates and effectively seize an unfair share of the CPU time, as well as bandwidth. In traditional networks, Shreedhar and Varghese [133] worked on a packet scheduling algorithm that aimed to isolate each network flow from misbehaving flows. Parekh and Gallagher [118] ensured service discipline by means of fair queuing techniques. The central theme of these approaches stems from the basic need to obtain fair allocation of resources among contending flows.

If a real-time, operating system (RTOS) were to run on the network node where the bottleneck is formed, then we may allocate periods of guaranteed processing time (c.f. §3.2.1) on a per-flow basis. However, this would mean periods when processing time is over-allocated, thus giving rise to inefficient usage of valuable computational resource time, which is not unlike a case of resource provisioning. A more common scheduling mechanism is that of a Unix-like operating system. Each flow is allocated a fixed time slice. Problems arise when an incoming active packet flow is too large. The allocated time could be insufficient to complete the task, thus forcing the running application to an abrupt end. This may cause the dropping and resending of the application stream.

---

29 Over the past 5 years, larger ISPs have simply peddled as fast as they could to meet the growing demand for bandwidth by simply throwing bandwidth at congestion problems. Today, the dire financial positions of Worldcom and Global Crossing illustrate, to a certain degree, the fallacies of over-provisioning network resources.
Figure 19: Customer flows coming in at higher speed than ISP’s uplink, where the incoming traffic is over-subscribed

The bottleneck is a first-in-first-out (FIFO) queue with $N$ end-users competing for the use of a resource at a bottleneck. In this competitive setting, the ISPs are egoistic in that their utility function reflects only their own performance, and not that of the network as a whole. Let $S_n$ be the set of admissible strategies $s_n$ for end-user $n$, so that the product set of admissible strategies by all the other contending end-users is given by

$$S_{-n} = \bigotimes_{m \neq n} S_m.$$  \hspace{1cm} (4)

Using a game theoretic approach, the best-reply correspondence can be represented in following relation,

$$R_n : S_{-n} \rightarrow S_n.$$  \hspace{1cm} (5)

This represents the strategy that an ISP will adopt, given what the others are doing. Each end-user computes its best-reply where its utility is maximised. $R_n$ seeks to maximise $n$’s utility. We then ask whether these end-users will reach an equilibrium point, where dominant strategies emerge. By intuition, for two end-users, Nash

\textsuperscript{30} For simplicity, we unify both communication and computational resource.
equilibrium (cf. A.1) occurs when one end-user's best reply to another will also yield
the recipient's best reply. Using simulation (cf. §4.3.4), we show that these strategies,
which vary in terms of vengefulness and boldness, will self-regulate towards relatively
low levels of boldness if metanorms are adopted at bottlenecks.

4.2.3 Formalising the problem space: tragedy of the commons

We now prove that egoistic strategies will only lead to congestion, hence
further delays. To back up our rationale for introducing the concept of metanorms at
network bottlenecks, we consider the oft-cited abstract problem of Hardin's [71] bucolic
notion of the commons in analogy, and adapt Gibbons's [63] reasoning into a network
context. Consider \( n \) ISPs competing for processing time on an active node processor.
We denote the number of packets the \( i^{th} \) ISP sends by \( c_i \) and the total number of packets
to be processed by the node as \( C = c_1 + \cdots + c_n \). Traditionally, the only billing model
supported on the Internet for ISPs is a flat-rate model. The connection price today is
only based on the connection speed and does not take account of node processor load
usage. It is to be expected that each ISP will attempt to maximise its own traffic flow
through the node.

The utility derived by an ISP, on each of the packet it sends, when a total of \( C \)
packets are being processed on the node is \( u(C) \) per packet sent. Since a packet flow
would require a certain amount of computation time to complete its task on an active
node, there are a maximum number of packets that can be efficiently processed by the
node. Formally,

\[
C_{\text{max}} : \begin{cases} 
  u(C) > 0, & \text{if } C < C_{\text{max}} \\
  u(C) = 0, & \text{if } C \geq C_{\text{max}}
\end{cases}
\]  

This implies that for an under-utilised node processor, the packet streams will
utilise plenty of processing speed within their allocated VE to service their computation,
such that adding one more causes little problem for the processor. However, if the node is congested with heavy traffic flows from many ISPs \(i.e., C\) is just below \(C_{\text{max}}\); then adding one more packet dramatically harms the rest, causing \(u' < 0\), \textit{i.e.}, negative utility. This is attributed to the fact that active packets are repeatedly resent from source to destination because the intermediate active nodes fail to completely process them due to scarce processor time. Graphically,

![Graph of ISP utility versus the overall total number of packets on processor](image)

Each ISP simultaneously chooses its own strategy for sending their aggregated traffic streams. A strategy for ISP \(i\) is the choice of the number of packets \(c_i\) to be processed by the node. We assume that the strategy space \([0, \infty)\) covers all the choices that could possibly be of interest to the ISP, \textit{e.g.}, higher level of boldness (cf. §4.3.2) means sending more packets. Practically, \([0, C_{\text{max}}]\) would also suffice. The payoff to ISP \(i\), when the numbers of packets injected by the other ISPs are \((c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n)\) can be given by \((\text{total no. of packets}) \times (\text{utility per packet}) - (\text{total no. of packets}) \times (\text{cost of sending a packet}), \text{i.e.},\)

\[
c_i u(c_1 + \cdots + c_{i-1} + c_i + c_{i+1} + \cdots + c_n) - pc_i. \tag{7}
\]

Now consider the situation when the ISP \(i\) already has a fixed number of packets processed on the node and attempts to add another single packet such that \(c_i^*\)
must maximise its payoff. The utility derived by the ISP is \( u(c_i + c_i^*) \) and the cost is \( p \).

The harm to its existing packets is \( u'(c_i + c_i^*) \) per packet, or \( c_i u'(c_i + c_i^*) \) in total. The other ISPs choose \((c_1^*, \ldots, c_n^*)\). If \((c_1^*, \ldots, c_n^*)\) is to be a Nash equilibrium\(^{31}\), then the first order condition for this optimisation problem is given by

\[
u(c_i + c_i^*) + c_i u'(c_i + c_i^*) - p = 0, \tag{8}\]

where \( c_i^* = (c_i^* + \cdots + c_{i-1}^* + c_{i+1}^* + \cdots + c_n^*) \). Substituting \( c_i \) as \( c_i^* \), and taking consideration of all the ISPs’ respective optimal conditions yields

\[
u(C^*) + \frac{1}{n} C^* u'(C^*) - p = 0, \tag{9}\]

where \( C^* = (c_1^* + \cdots + c_n^*) \) and represents the total number of packets processed by a single node processor in the Nash equilibrium. On the contrary if we attempt to maximise the net benefits for every ISP’s flow on a bottleneck node at any instance of time, the ‘social optimum’, denoted by \( C^{**} \), should naturally solve

\[
\max_{0 \leq C \leq \infty} Cu(C) - Cp \tag{10}
\]

As before, the optimisation problem will have a condition as follows:

\[
u(C^{**}) + C^{**} u'(C^{**}) - p = 0 \tag{11}\]

Comparing equations (9) and (11), suppose \( C^* \leq C^{**} \). Then, \( u(C^*) \geq u(C^{**}) \) because of the inverse relationship between the total number of packets and the utility derived by each ISP. Likewise, \( u'(C^*) \geq u'(C^{**}) \). Finally, \( \frac{1}{n} C^* < C^{**} \). This makes the left-hand side of equation (9) strictly exceeding the left-hand side of equation (11), which is impossible

\(^{31}\) In general, we will say that two strategies \( s_j \) and \( s_2 \) are in Nash equilibrium (see A.1 for further explanation) if:

1. under the assumption that ISP 1 adopts \( s_j \), ISP 2 can do no better than to adopt \( s_2 \); and
2. under the assumption that ISP 1 adopts \( s_2 \), ISP 2 can do no better than to adopt \( s_j \).
since both equal zero. Thus, we conclude that if all ISPs are selfish in nature (and fail to consider the externalities, \( i.e. \), the effect of its actions on the processor), the processor will be over-utilised because \( C^* > C'^* \), \( i.e. \), the total number of packets that can be processed by the router when considering all the ISPs’ respective optimum condition is more than those when considering the social optimum. This results in a higher number of incomplete processing, as well as a higher number of dropped packets and retransmission.

The owner of ‘commons’ usually adopt a ‘hands-off’ approach towards managing the conflicts between the inhabitants of the area, and so, we take cue from this idea to provide a framework and mechanism for discovery of sufficient information to prove and punish the ‘defectors’, as we shall see in the remaining parts of this chapter.

4.3 Stabilising norms

4.3.1 Current flow control strategies

Jain et al [83] clearly differentiated between flow control and congestion control. The former is taken to avoid buffer overflow at the destination between two directly connected nodes, while the latter addresses a configuration that includes a sub-network, where the source must not only obey the directives from the destination, but also from all the other nodes en-route. We are interested in addressing the latter.

A sending end-user is able to detect the loss of a packet by means of the lack of a positive acknowledgement being received within the timeout period. In TCP congestion avoidance [81], users determine their rate of sending packets into the
network by observing positive acknowledgements and packet losses. They increase\textsuperscript{32} the rate of sending when loss levels are low and decrease it when congestion is inherent in the communication channel. Traffic from applications that are able to modify their data transfer rates according to the available bandwidth within the network is termed elastic. Apart from the TCP example, a data link layer example is the available bit rate transfer (ABR) capability of ATM networks [78]. Now, with the addition of active queue management (\textit{e.g.}, RED) to the Internet infrastructure, where routers detect congestion before the queue overflows, routers are no longer limited to packet drops as an indication of congestion. Routers can instead set the Congestion Experienced (CE) codepoint in the IP header of packets from ECN-capable [124] transports.

Looking at other networking protocols, the Ethernet does pretty well in tackling congestion, partly because of the adoption of a norm in the form of the carrier sense multiple access collision detect (CSMA/CD). Under the Ethernet CSMA/CD media access process, any host on a CSMA/CD LAN can access the network at any time, provided that it listens and detects that there is no traffic on the network before it starts to transmit. As a contention-based environment, the Ethernet allows any station on the network to transmit whenever the network is ‘quiet’. Back-off algorithms determine when the colliding stations should retransmit. How then do we encourage this type of ‘discipline’ between contending agents such that a more organised norm persists at access networks where end-users compete for a fixed amount of resource, or at inter-domain bottlenecks where different ISPs complete for an uplink facility?

\textsuperscript{32} Using the slow-start algorithm, each time the sender receives an acknowledgement from the receiver, congestion window is increased by one segment size – this effectively doubles the transmission rate for each round-trip time.
4.3.2 The norms game in the context of access networks and bottlenecks

Axelrod [22] first introduced the norms game, where he proposed two dimensions in a player's strategy. Boldness is the propensity for defecting, while vengefulness is the propensity for punishing a defector.

We assume here that the players that represent ISPs are not fully rational, but rather more likely to use trial-and-error for their decisions. In other words, ISPs will adopt strategies that work well, and discontinue actions that turn out badly for them. To simulate this adaptive behaviour, we use a genetic algorithm (cf. §4.3.4).

If a player's boldness is higher than the probability of being seen, a player will defect and use up a large chunk of the shared bandwidth (or take up a bigger slice of the node processing time), i.e., a player's boldness can be directly mapped to the total link utilisation by an ISP. Despite the punishments, consistent with Axelrod's simulation, our results show that the tendency for players to be bold is still high, regardless of how high we bias the vengefulness level to start off the game. Over time, the population stabilises towards one with average vengefulness but high boldness, which gives rise to network congestion (see result in §4.3.4).

An opportunity to defect is accompanied by a known chance of being observed (S). Boldness (B) and vengefulness (V) define a player's strategy. If B > S, the player
will defect, hence obtains a payoff of 3 credits. Every other player in the group gets 'hurt' by 1 credit. Should the defector be spotted, it shall be heavily punished at a cost of 9 credits, while the punisher, in turn, incurs a cost of 2 credits for its enforcement action.

End-users generate packets to be dispatched into the network. While limits on the size of packets are governed by the IP protocols, in principle, an end-user may choose to set packet sizes on a per packet basis. It is assumed that all packets have a given fixed size and we are interested in the rate of sending packets into the communication channels, measured in packets per second.

4.3.3 Metanorms as a means to stabilise norms

Axelrod further proposed the concept of metanorms as an effective way to get a norm started and to protect it once it is established such that an entire system becomes self-stabilising. However, it is not our motivation to have a group completely organise and govern itself – as noted earlier, we do expect minimal administrative responsibilities from the upstream provider, *i.e.*, at least to provide a framework for discovering the relevant parameters that determines the 'behaviour' of fellow members of the group. In this context, the administrator only has to keep track of the virtual credits given to end-users who pay a flat rate for Internet access. Thus, given shared resource, we naturally expect temptations for the sub-networks to 'monopolise the use of collective goods'.

The correlation between vengefulness against someone who defects versus the vengefulness against non-punishers remains speculative, but simulation results show that should this approach be successful in a practical case, establishing computational and communication QoS may be further simplified using the Paris Metro Pricing mechanism [115] (see §5.2).
In practice, logs at the access router for each sub-network may be used to find out which downstream sub-network defects (i.e., using up a lot of bandwidth or taking a huge slice of processing time). Over time, a sub-network (e.g., transit ISPs) gets ‘frustrated’ and starts enforcing norms within its downstream users. This practice will propagate all the way down to per-user granularity of metanorm enforcement. The sub-networks can either choose to punish the non-punishers and incur a further cost of 2 credits or do nothing about it. As with the basic norms game, this depends on a sub-network’s vengefulness and the non-punishers who get punished will suffer a cost of 9 credits (see Figure 21).

If end-users were simply Web browsing, there would be no noticeable deterioration in the maximum speed of the line, due to burstiness. However, if end-users are transferring large amounts of data such as MP3 files through peer-to-peer file sharing, or streaming video content, then congestion can quickly become obvious as the connection becomes saturated. The latter is our context of defection. Thus, the sub-network’s objective is to push traffic off its domain into the next, costing minimum credits. If it starts defecting, its credit level will drop and would need to be replenished, i.e., purchased from the uplink provider.

The metanorms approach forces agents who have spotted defection to punish the act. As discussed in §3.6, this would potentially represent a complex society that may need to account for collusion in defection. In §A.2, strategies were formally defined so as to generalise relationships between the members of the society. In §4.3.4, we allow for a variation in the cooperation-defection parameter in order to simulate and investigate the dynamic nature of evolving strategies.
4.3.4 Simulation

In our simulation using Java\textsuperscript{33}, the basic norms game was adapted to reflect the bottleneck at the upstream ISP’s uplink. Boldness, in our context, means the likelihood that sub-networks will push high traffic volumes through the uplink. The strategies of the sub-networks for the initial population are chosen at random. For each generation (as indicated by different shades of plot in Figure 22), the scores are calculated using the payoff explained in §4.3.3. The sub-network with the most successful boldness-vengefulness strategies, \textit{i.e.}, one standard deviation ($\sigma$) above the average ($\mu$) score for that population, will be retained as well as having its strategy replicated; while those between $\mu$ and ($\mu + \sigma$) will simply be retained. The other sub-networks (also known as ‘agents’ in the multi-agent analysis in §2.4.3 or ‘players’ in the metanorms game in §4.3.3) in the simulation are eliminated, while new ones are regenerated randomly to maintain the total population of 20 agents.

At each generation, our approach is to allow for stochastic strategies (depending on the values of B and V) that respond to the probability of being ‘seen’ and the vengefulness of other players, by changing only their statistical propensity to punish. This means that different ISPs would end up with strategies that are not obliged to always respond in a same way to a given round of the game. For example, a player with a randomly generated boldness level of 4/7 will only defect if the probability of being seen is lower than 4/7. We feel that such uncertainty simulates the inevitable errors and inconsistencies that occur during actual inter-ISP interactions.

The expected output from the $\texttt{getBinary()}$ method implemented is a 3-bit number that would represent numbers from 0 to 7. The Math class in Java was used to randomly generate a fraction (for each bit) between zero and one. If the random fraction

\textsuperscript{33} A full listing of the source code used is available in §A.2.
is below 0.5, it is taken to be zero; otherwise, it is taken to be one. So if the binary number generated, is 101, say, the final output is 5 out of a possible 7 (hence 5/7).

It was decided that each bit has a 1% chance of ‘flipping’ to represent the notion of mutation for the next generation of numbers. This is done so that we allow for new values for each trait (i.e., B and V) to evolve over time. Thus, if the second bit in the output 5 in the subsequent generation flips, the value of the trait becomes 7, which means that agent turns into a very vengeful (or bold) entity compared to an average one.

Looking at our results, consistent with those of Axelrod’s, the first thing to happen is the fall in boldness level. Once the boldness level fell, the vengefulness level dropped as well. This enabled an evolution towards bolder sub-networks, thus giving rise to channel congestion (see Figure 23). We extrapolated this result by mapping the property of boldness against the propensity for an agent to send large amounts of traffic upstream, i.e., a high level of boldness indicates that a large amount of packets will ultimately be dropped. We make an assumption that the maximum capacity of a channel is arbitrarily equal to 75 percent of the total of maximum boldness of the 20 sub-
networks. This assumption is based on the fact that in practice, it is unlikely for a provider to completely over-engineer a network to the point that there will never be more aggregated traffic demand than the network can accommodate. Thus if the total value of boldness is more than 75%, we take that to mean that packets will be dropped in an actual network bottleneck. As can be seen in Figure 23, the plot exceeds the maximum capacity, when there is no further incentive to punish the 'defectors'.

![Figure 23: Temporal adaptation of channel utilisation with no incentive to punish 'defection'](image)

In the basic norms game, the sub-networks are not motivated to punish a non-punisher (i.e., a defection). The metanorms approach provides more incentive for being vengeful. A drop in the level of boldness (see Figure 24) justifies this incentive. As with the simulation in the basic norms game, we repeat the scoring and evolution for 500 generations. The result shows a distinctive drop of boldness level that stays at that low level. This implies that the sub-networks become more disciplined when they start policing each other, not only by punishing the defectors, but also to punish the non-punishers.
In Figure 25, where we use boldness to represent bandwidth usage per sub-network for the premium channel, the drop in total channel utilisation is consistent with the low level of boldness in the metanorms game.
We repeated the metanorms experiment by biasing the random trait value generator to come up with agents that have higher boldness and lower vengefulness. The result is depicted in Figure 26.

![Figure 26: Group dynamics revert to high average vengefulness and low average boldness even after starting of the metanorms game biased with low vengefulness and high boldness.](image)

As can be seen, the metanorms approach helps to establish a proactive and stable community that lets the agents self-govern each other so that the average boldness level for the whole community is maintained at a low level. We clearly observe this phenomenon as shown by the migration of agent traits from the bottom right position of the diagram towards the top left corner, where the final average after 500 generations (as depicted by the red dot) is just slightly over 1/7 in boldness although it started off with a heavily biased boldness of almost 7/7. This stability was successfully maintained by the relatively high level of vengeance (i.e., approximately 55/70) although we biased the first generation to have zero vengeance.
4.3.5 Practical implementation

Leading off from this encouraging result demonstrated by the simulation, we then address how this idea can be practically implemented. In the real world: how does one see defection; how does one identify who defected. When a fellow player sees this defection, she punishes the defector by sending a report to the administrator. How do we avoid reckless accusations of defection? How do we know if an ISP has not punished a non-punisher?

When a sub-network notices a delay when using the shared medium, it interrogates the access router for information with regards to the number of packets sent upstream by other sub-networks. We use the function provided by Linux iptables, where the sub-network can learn about source and destination addresses of the defecting traffic flow as well as the packet count for a time period. If it is shown to be high, the sub-network then plays the role of the 'punisher' (see Figure 30). For this purpose we have written our own MIB module that monitors the rules within netfilter/iptables\textsuperscript{34} so that SNMP can be used to probe the router.

Originally developed as a firewall subsystem in the past Linux kernels, for our purpose, netfilter's most useful function is, in fact, the packet filtering subsystem. Each protocol defines a set of 'hooks' (IPv4 defines 5), which are well-defined points in a packet's traversal of that protocol stack. At each of these points, the protocol stack will call the netfilter framework with the packet and the hook number. Parts of the kernel can register to listen to the different hooks for each protocol. So when a packet is passed to the netfilter framework, it checks to see if anyone has registered for that protocol and hook. We use this method to listen at designated interfaces in order to learn about the size of packets that is being transmitted by a particular destination. Because the netfilter/iptables project developed at a rapid pace, we took the liberty to write our own

\textsuperscript{34} http://www.netfilter.org/
MIB module, *i.e.*, the iptable MIB (see §A.2), so that downstream machines may use
SNMP to query the router for the requisite MIB objects that would give them an
indication of the 'defecting' packet flows that may have caused a bottleneck congestion.

![Diagram](image)

Figure 27: An abstraction of the departmental research network\textsuperscript{35}; iptable rules are inserted at
'george' so that 'jorg' and 'wilson' can use SNMP to probe each others' traffic through 'george'.

To listen at the designated interface, we first insert some packet matching rules
on the IP table (see §A.5 for basic iptable commands) for 'george', which would act as
the bottleneck node for 'wilson' and 'jorg'. A table contain chains, and within the filter
table are the INPUT, FORWARD, and OUTPUT chains. We define our own chains to
filter the traffic data from sub-networks, *i.e.*, 'jorg' and 'wilson'.

\textsuperscript{35} Our experimentation network at UCL is part of the FAIN pan-European testbed, one of the pioneering
Active Networks that interconnect both active and passive nodes (belonging to the consortium's partners)
using IP tunnels over the public Internet.
To generate a stream of predictable traffic, we use the ‘ping’ application to send ICMP\textsuperscript{36} ECHO\_REQUEST packets from ‘wilson’ to reblochon.ee.ucl.ac.uk, which we use to simulate a gateway to the Internet.

\textsuperscript{36}The Internet Control Message Protocol (ICMP) is a message control and error-reporting protocol between a host server and a gateway to the Internet.
The implementation of our idea terminates at this point. We will present henceforth the design philosophy of the metanorms system. There are various implementation approaches that can be used to realise this system. By presenting a generic design, we establish useful starting points for future work by practitioners.

![Diagram](image)

Figure 30: Mapping the metanorms game into a network context

### 4.3.6 Reasoning about defections between ISPs

In the literature of epistemic logics [50], the universal modal operator $\Box$ is usually called $K$. If $n$ agents are taken into account, a family of subscripted operators $(K_1, K_2, K_3, \ldots, K_n)$ is considered, where, $K_i p$ is read as 'domain $d_i$ knows $p$'. A domain’s knowledge implies that of the ISP’s administrator. A player $i$ knows $\psi$ in world $s$ of structure $M$ exactly if $\psi$ is true at all worlds that $i$ considers possible. The Kripke structure $M$ for $n$ players is a pair $(S, \Pi)$, where $S$ is a set of possible worlds and $\Pi$ abbreviates a sequence of sets of pairs of elements of $S$. $\Pi$ can be seen as an equivalence relation on $S$. A statement like $K_i \psi$ is read as 'player I knows $\psi$'. Formally, $(M, s) \vDash K_i \psi$ iff $(M, t) \vDash \psi \forall t$ such that $(s, t) \in \Pi_i$. 
In this section, the discovery process for each ISP is analysed with regards to their defecting peers, i.e., fellow ISPs who are causing congestion at the upstream bottleneck. The Kripke structure is viewed as a labelled graph, i.e., a set of labelled nodes connected by directed labelled edges. The nodes are the states of $S$, while the label of state $s \in S$ describes which primitive propositions are true and false at $s$. The edges are labelled by sets of ISPs; and the label on the edge from $s$ to $t$ includes $i$ if $(s, t) \in \Pi_i$. The proposition $p$ is represented as 'ISP 3 has defected' (and has sent\(^{37}\) a high volume of traffic upstream'). Suppose that $S = \{s, t, u\}$ and $p$ is true at states $s$ and $u$, but false\(^{38}\) at $t$. ISP 1 cannot distinguish $s$ from $t$ (so that $\Pi_1 = \{(s, s), (s, t), (t, s), (t, t), (u, u)\}$), and ISP 2 cannot distinguish $s$ from $u$ (so that $\Pi_2 = \{(s, s), (s, u), (t, t), (u, s), (u, u)\}$). This simple example is captured in the graph below.

It is noted that the self-loop at each edge labelled by both 1 and 2 (representing ISPs 1 and 2) denotes reflexive relations for $\Pi_1$ and $\Pi_2$. The edges have an arrow in each direction because $\Pi_1$ and $\Pi_2$ are symmetric.

ISP 1 will not know if $p$ is true because it considers both worlds, $s$ and $t$, possible since it possess insufficient information to distinguish whether the actual world is $s$ or $t$. On the other hand, ISP 2 does know that ISP 3 has defected since in both

\(^{37}\) This analysis is equally valid if the form of defection is caused by an ISP who does not punish and defector.

\(^{38}\) This would mean that the suspicion (that some other peer ISP is defecting) arising from a deteriorating quality of bandwidth is proven to be unfounded.
worlds that the former considers possible at \( s \) (i.e., \( s \) and \( u \)), it can conclusively determine that \( p \) is true. This situation where ISP 2 appears to be more informed than ISP 1 arises in our context because it has probed the SNMP agent for the relevant MIB OIDs (see A.2) that we have defined and implemented on the upstream bottleneck (i.e., the machine ‘george’ in §4.3.5).

Next, taking this analysis a step further for the metanorms context, it follows that in state \( s \), ISP 1 knows that ISP 2 knows whether or not ISP 3 has defected because in both worlds that ISP 1 considers possible in state \( s \) and \( t \), ISP 2 knows the validity of \( p \). In practice, if the congestion does not ease up, ISP 1 may punish ISP 2, i.e., a scenario where the non-punisher is punished for not cooperating and not abiding to the metanorm. By way of contrast, although in state \( s \) ISP 2 knows that \( p \) is true, it does not know that ISP 1 does not know this fact. Again, we reason that this is the case because in one world that ISP 2 considers possible, namely \( s \), ISP 1 does not know this fact. We shall thus enhance the discovery of the non-punishers by elaborating the design of the metanorms system in the following section. This analysis can be summarised using the following statement: \( (M, s) \models p \land \neg K_1 p \land K_2 p \land K_1(K_2 p \lor K_2 \neg p) \land \neg K_2 \neg K_1 p \).

### 4.3.7 Enterprise view of the metanorms system using UML use cases

The Unified Modelling Language (UML) is a standard notation for the modelling of real-world objects as a first step in developing an object-oriented design methodology. It enables system designers to create blueprints that capture their visions in a standard, easy-to-understand way and to communicate them to others. Through a series of UML diagrams, we provide the basis for the development of a metanorm system.
Figure 31: Actors in the system

Figure 32: Relationship between the Checker and the Defector

Figure 33: Relationship between the Accomplice and the Defector
As a practical example, to enforce metanorms MIBs could be used to represent information such as the identity of the punisher, whether a punisher has probed the iptable MIB (hence would have known of the existence of a defection), and whether a fellow sub-network has further punished non-punishers. To validate a punishment, a sub-network simply sets a MIB variable on the access router. Apart from SNMP, practitioners may also opt to use the Common Object Request Broker Architecture (CORBA) technology to pass punishment parameters to a database. The architecture has an implicit capability to recognise which entity has or has not punished and defector, having probed the database. Java RMI also offers similar functions, and is a more sensible prototype approach if we know in advance that our system is to be fully implemented in Java.

4.4 Summary

Ceterus parabus, when separate entities (represented as agents in our analysis) have unlimited access up to the maximum throughput of their shared media, the classic problem of the commons have been proven to exist. This externality is derived from the fact that we have in consideration a packet-switched network that sits on a shared-media technology. Every extra packet that an agent sends imposes a cost on all other agents because the resources that the former dominates are also available to the others and as
Internet users are well accustomed to, this cost can come in the form of delays and dropped (lost) packets. It is naturally concluded that this occurrence is due to the lack of incentive to economise on usage.

We have seen countless management approaches for addressing traffic congestion. In this chapter, it was shown that a more effective way is to let the parties involved in congestion police each other and the upstream provider only provide a framework for discovery of facts (e.g., resource usage, and action by the contending parties) pertaining to a specific bottleneck. In order to provide end-to-end service, we need to coordinate at points of conflict; and the task of coordinating a multi-domain task without a central authority that polices proceedings would benefit from an element of predictability. To achieve predictability, we explored the interaction within a system and observe if there is an alignment of dominant strategies. In our results, we showed that this required alignment does exist and stability was achieved. This was made possible because we successfully encouraged participating entities to adhere to the system by enforcing a punishment scheme, called metanorms.
Chapter 5

Pricing node resources

‘Res tantum valet quantum vendi potest’
(A thing is worth only what someone else
will pay for it)

Latin maxim for fair asset prices

In our view, discussions on inter-domain service management would be incomplete without a mention on pricing. It is clear that the traditional PSTN offers a mature and sophisticated pricing mechanism for the services it provides, while the pricing framework for Internet services has not evolved greatly beyond the ‘flat rate’ and ‘usage based’ dichotomy. In principle, most backbone and regional network traffic (see §2.4.2) move over leased phone lines (which are increasingly fibre optic in nature); thus, at the physical layer, the underlying technology that supports both the Internet and the PSTN remains identical. The fundamental difference, however, lies in how these lines are used by the Internet providers and telephone operators. Indeed, the use of connectionless packet-switched service by the former (compared to the latter’s circuit-switched service) has clear and distinctive implications for pricing and the efficient use of network resources.

In this chapter, we adopt a pricing approach that takes account of both guaranteed and best effort service by adapting the arbitrage theorem from financial theory. One of our intentions in this brief chapter (as well as in the preceding one) is to

---

39 We do note, however, that these two pricing frameworks may not be directly comparable to each other since the PSTN architecture for network operators in different countries are more ‘regional’ in nature, while IP enjoys a more consistent and ‘global’ adoption.
try to synthesise insights from different disciplines to address problems in data networks.

5.1 Internet economics and the relevance of financial instruments

Many issues in Internet economics are not new [104]. Accounting and pricing issues are obviously key concerns for operators and ISPs. Kleinrock [91] once asked, 'how does one introduce equitable charging and accounting scheme in a network system as heterogeneous as the Internet?' It is felt that Active Networking initiatives further compound this complexity by adding an additional parameter of network resource that needs to be priced, i.e., node processing time. Since content providers would need to reserve VEs (cf. §3.2), ANPs (cf. our definition of stakeholders in §2.2) could benefit from a pricing approach that accounts for CPU time. The Active Networking research community has, by and large, focused on technical issues with regards to the design of the node operating system and execution environments. We complement Shenker's call [132] for new research directions by presenting, in our knowledge, the first framework for pricing computational resources.

In the world of financial derivatives, options give the owner the right to purchase or sell an asset. The use of derivatives is proposed in order to protect against node resource unavailability due to congestion. The pricing of options and related instruments [33] has been a major breakthrough in financial theory and as such, we hope to leverage its use in communication networks, particularly pricing limited node resources in a fair manner. As such each 'market participant' can have its own strategy for trading CPU time. Participants try to maximise their utility function subject to their respective budget constraint. Recall that one of the underlying objectives in the thesis is to control the contention for Active Network resources, such that a service element is provided with a superior level of network resource, while the other service element does
not obtain an unfair allocation of resources. Putting a fair price of node resources contributes to this cause.

5.2 Incorporating the Paris Metro Pricing (PMP) idea

It is believed that, in practice, the choice of what is considered ‘optimal’, even from a system viewpoint, is heavily influenced by the end-users’ utility functions. As such, it is best to let the network determine a fix price, and allow the sub-networks to respond and place demands upon the network. This is expected to provide a set of behaviours, from which dominant strategies may be extracted\(^*\). The term ‘price’ is suggestive of a monetary transaction, but it need not involve money. Indeed, even in the case of exchange of money, the actual charging mechanism is separate from the control signals, and the entire framework would most probably involve pre-purchase of credits, as was suggested in the payoff matrix in §4.3.2.

The fundamental issue to be addressed here is the allocation of limited resource, \(i.e.,\) having multiple sub-networks that attempt to maximise the usage of their upstream provider’s uplink as well as the computational resource; and any solution would need an explicit charging mechanism to prevent ‘tragedy of the commons’ (cf. §4.2.3). In Odlyzko’s Paris Metro Pricing (PMP), providers do not provide formal quality of service (QoS) guarantees [115] and packets are still handled on a best effort basis. The network is logically partitioned into separated channels. One of the unused bits in IPv4 is suggested to separate the flows into premium channel and best-effort channel. Physically, no traffic engineering is performed. Differentiation in congestion levels is provided by the price discrepancy between the two channels. The simple rationale is that when the premium channel gets congested, end-users will realise that it is not worth

\(^{40}\) As what was done in §4.3.4
paying for the quality that they are getting, so they drop to the best-effort channel. This self-regulating behaviour is expected to restore the differential in QoS.

### 5.2.1 Relevance to Chapter 4

Summarising §§4.3.3 and 5.2, the traffic from a sub-network may stay in the premium channel: if it pays more (hence increases the credits); or if it does not defect, \textit{i.e.}, self-imposed discipline on sending traffic through provider’s uplink. With regards to the metanorms game, when a sub-network has been punished too much, its credits will drop. The provider must allow the sub-networks to replenish credits by paying more. As a mechanism to avoid dropping to the best-effort channel in the middle of a transmission, a derivative market can be established at the queue. The downstream sub-network can exercise call options when credits drop below a certain level. Semret’s \cite{130} (cf. §5.4.3) approach was adapted to enable an end-user to pay a reservation fee, which gives the right to buy extra credits at any time in the future at its bid price for the premium channel to stay within it.

To enhance the stability of the premium channel, we propose that the self-policing mechanism of metanorms be used within the premium channel. Thus, traffic from a sub-network may stay in the premium channel: if it pays more; or if it does not defect, \textit{i.e.}, self-imposed discipline on sending traffic through provider’s uplink. In effect, PMP enforces two different prices for two separate instances of the same product with the hope that the end-user would align their network usage based on the price discrepancy. How do we set a ‘fair’ price differential between the basic channel and the premium channel?

### 5.3 Mathematical representation of resource price

Computational and communication resources are represented by a vector of prices denoted by $S$, which groups all network resources under a single symbol, \textit{i.e.},
In this pricing model, it is implicitly assumed that a varied ratio of bandwidth and CPU cycle, say, will give rise to different levels of utility, e.g., a reservation array that has high bandwidth and low CPU time slice will have active packets promptly transported across links but the processing of the packets will be slower and the allocated buffer may overflow. On the other hand if a domain has reserved low bandwidth with high CPU time slice, the active packet flow form that domain will naturally have its data processed quickly and forwarded to the output interface of an active node but having to wait for transmission across the transmission channel.

Fair pricing for the respective resources would mean that the various sub-networks would not benefit from getting a better price for a resource in relation to the other resources. In the language of pricing financial assets, equilibrium pricing methods mean that fair prices of assets has been achieved (i.e., arbitrage opportunity has been eliminated), and this is what we target in pricing node computational resources in lieu of current link communicational resources. Thus, the central thesis of our approach is that once arbitrage opportunities have been eliminated, our node (or link) resource has been fairly priced. We subsequently prove the reciprocal relationship between price and utility.

We further let vector $W$ denote all possible, mutually exclusive, states of the world.

$$ W = \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} $$
Each \( w_i \) represents a distinct outcome that may occur. As the level of network usage and congestion fluctuate, the resource prices would also vary over time. Here, we assume that there are a finite number \( K \) of such possible states.

With \( N \) different types of resources under consideration, \( u_{ij} \) denotes the utility gained by one unit of resource \( i \) in state \( j \). Thus, we can condense our current notation, based on (12) and (13), as follows.

\[
U = \begin{bmatrix}
  u_{11} & \cdots & u_{1K} \\
  \vdots & \ddots & \vdots \\
  u_{N1} & \cdots & u_{NK}
\end{bmatrix}
\]  

(14)

This matrix can be considered in two ways. Each row may be seen as representing utility gained from one unit of resource in different states of the world. Conversely, we can say that each column of \( U \) represents utilities from different mix of resources in a given state of the world.

5.4 ‘Asset classes’ for active nodes

5.4.1 Resource pools

Apart from bandwidth as a communication resource, Active Networking brings forth a new ‘commodity’ to be traded – computational processing time. In addition, we have memory and persistent storage. We adopt the model described by the Active Network NodeOS workgroup [122], where the five core abstractions defined by the NodeOS – EE (see Figure 35) interface are thread pools, memory pools, channels, files and domains. To avoid confusion with our definition of network domains (cf., §2.4), we will refer to it as flows.
Active packets that arrive on an input channel are processed by the EE using threads and memory allocated to a specific flow. The packets are later sent to the output channel. Thus, a channel consumes network bandwidth as well as CPU cycles and memory buffers. A dedicated thread pool takes charge of packets across a flow’s channels and the cycles that they consume are charged to that pool. Likewise, the input-output buffers used to queue packets on a flow’s channels are allocated from (and charged to) the flow’s memory pool. It can be concluded that the flow is a container that encapsulates resources used across both the NodeOS and an EE on behalf of a stream of packets coming from a particular domain.

The thread pool is the primary abstraction for computation. Each flow is allocated a thread pool to process its packets. Each flow bears several characteristics, namely, maximum number of threads and the rate at which the pool consumes the node CPU. The NodeOS is expected to pre-allocate threads and to be able to increase its specifications when required. The memory pool is used to implement packet buffers and to hold EE-specific state. It combines the memory resources for one or more flows and makes those memory resources available to all threads associated with the flows.

How then do we put a price on these resources? We next take cue from the arbitrage theorem in financial markets.
5.4.2 Arbitrage theorem

We first consider a case where a domain is generally interested in three dimensions of resources from its neighbour. We categorise the first dimension as basic connectivity, where the former only utilises the latter's routing table to get to further next-hops – as is common in today's Internet. The utility is dependent on the network state at a particular instance of time. The requesting domain will accept any given throughput on behalf of its end-users, as it is well aware that the service obtained is best effort and will pay $B(t)$. Secondly, we want to put a price on guaranteed high communication bandwidth; CPU time slice; and memory. Here, the end-user clearly expects a guaranteed utility as befits the price it pays for, i.e., $G(t)$. These prices vary with demand [38].

The third category relates to Semret's [130] work in establishing a derivative market in admission control. In his proposal, the user pays a premium $C(t)$ to obtain an option contract that gives her the right to buy the capacity at any time in the future up to a specified duration $T$ at her strike price $X$ if $G(t)$ rises above $X$. The option will expire worthless if $G(t)$ stays below $X$, i.e., in periods of low congestion, by the end of $T$.

The prices for these resources will form the following vector.

$$S_i = \begin{bmatrix} B(t) \\ G(t) \\ C(t) \end{bmatrix} \quad (15)$$

Due to the variability in network conditions, significant research has been carried out in the area of pricing network resources using market mechanism [51] [104]. The 'payoffs' (i.e., utility) will be grouped in a matrix $U$ as in equation (14), as discussed earlier. The utility gained from paying $B(t)$, a subscription-like fee, will be fixed in that the low-level services (cf. §2.6 for service definition) that enable end-to-end connectivity are the
only certainties that the ISP provides its peer with. We represent this constant level of utility as $b$. High volumes of traffic will potentially suffer delays.

When the requesting domain pays $G(t)$, it knows for sure that its packets will be transported using the premium channel at a predetermined bandwidth. Within an active node, its VE will also be assured to get hold of a predetermined amount of CPU time. Intuitively\(^{41}\), since $G(t)$ will fluctuate, with constant resource levels being reserved, the utility\(^{42}\) would be inversely proportional to the price direction, i.e., better value for each unit of resource when its price drops (see Figure 36). The utility for guaranteed resources over time is represented by $g_1(t + \Delta)$ and $g_2(t + \Delta)$ for a price increment and decrement respectively.

![Figure 36: Inverse relationship between utility and price for each unit of resource](image)

The ‘market value’ of the call option (i.e., the right to buy capacity) $C(t)$ will change in line with the movements in the underlying resource price $G(t)$. Thus, the following matrix gives $U_t$.

\[
U_t = \begin{bmatrix}
    b & b \\
g_1(t + \Delta) & g_2(t + \Delta) \\
c_1(t + \Delta) & c_2(t + \Delta)
\end{bmatrix}
\]  

(16)

\(^{41}\) As will be seen in equations (23) and (24), this relationship is proven to be true.

\(^{42}\) Note that the key difference with the notation used within the financial markets is the use of utility to represent the throughput of packets, hence the sender’s satisfaction. In financial markets, it is simply the price paid for a stock, say, factored with the uptick or downtick over time.
5.4.3 Adapting from financial theory

**Theorem [112]:** Given $S_t$ and $U_t$ as defined in equations (15) and (16); and given that the two states have positive probabilities of occurrence; setting $\Delta$ to 1; and normalising the basic connection fee to 1,

(1) if positive constants $\psi_1, \psi_2$ can be found such that resource prices satisfy

$$
\begin{bmatrix}
1 \\
S(t) \\
C(t)
\end{bmatrix} =
\begin{bmatrix}
b & b \\
g_1(t+1) & g_2(t+1) \\
c_1(t+1) & c_2(t+1)
\end{bmatrix}
\begin{bmatrix}
\psi_1 \\
\psi_2
\end{bmatrix},
$$

(17)

then there are no arbitrage opportunities, and the resources are fairly priced\(^{43}\); and

(2) if there are no arbitrage possibilities, then positive constants $\psi_1, \psi_2$ satisfying the above equation can be found.

In practice, it is obvious that this relation cannot be observed since $g_1(t+\Delta)$ and $g_2(t+\Delta)$ are 'possible' future values of the underlying resource, where only one of them, *i.e.*, the one that belongs to the state that is realised, will be attained. Working through the theorem, it is easy to see that if a guaranteed bandwidth provides a subnetwork/domain with a utility of 1 in state 1, and 0 in state 2, then $S(t) = (1)\psi_1$. This simply implies the notion that the requesting domain is willing to pay $\psi_1$ units of credits for an 'insurance policy' that offers one unit of account in state 1 and nothing in state 2. Similarly, $\psi_2$ indicates how much the requesting domain would like to pay for the insurance that pays 1 in state 2 and nothing in state 1. Clearly by spending $(\psi_1 + \psi_2)$ credits, that domain can guarantee 1 unit of account in time $(t+1)$, regardless of the state of the network in the future.

\[^{43}\text{The mapping between the actual price and our notion of 'credits' is beyond the scope of this thesis.}\]
5.4.4 Pricing the options

Quite often, one can imagine that a domain may not want to reserve any resources in advance in its neighbouring domain and would be perfectly happy with best-effort service by paying the basic peering fee. Yet, a more attractive strategy is to enable the former the right to request resource guarantees ‘on-the-fly’ by going long\(^{44}\) on, say, a CPU time call option. This way, the requesting domain can be certain that the VE created will have its expected CPU time slice should it decide to exercise the option.

While Semret’s work, to our knowledge, represents the seminal initiative in providing bandwidth guarantees by means of a derivative instrument, our approach is to use the arbitrage theorem to unify best-effort service, guaranteed service and the options instrument to establish a fair price. We also introduce the notion of incorporating the reservation of computational resource in view of Active Networking technology.

In considering (again) equation (17), we multiply the first row of \(U\) by the vector of \(\psi_1, \psi_2\), we get

\[
1 = b\psi_1 + b\psi_2. \tag{18}
\]

We move on to define

\[
\bar{P}_1 = b\psi_1 \tag{19}
\]

\[
\bar{P}_2 = b\psi_2
\]

We set the constant level of utility \(b\) to be positive (naturally likewise for the credits and prices) and in view of equation (18),

\[
0 < \bar{P}_1 \leq 1
\]

\[
\bar{P}_1 + \bar{P}_2 = 1. \tag{20}
\]

\(^{44}\) Taking a ‘long’ position on a call option means to purchase it; as opposed to a ‘short’ position, which indicates the position taken by the domain owner.
So \( \tilde{P}_i \) is a positive number and should sum up to 1. This enables us to interpret it as a probability associated to a state under consideration. In financial theory for pricing assets, \( \{\tilde{P}_1, \tilde{P}_2\} \) are known as synthetic probabilities \([112]\); and we will use this term. As we have adapted from the arbitrage theorem, the positive constants \( \psi_1, \psi_2 \) only exists if a fair price is obtained. Since \( \tilde{P}_i = b \psi_i \), the synthetic probabilities that we have just defined will also exist if there are no ‘mispriced’ resources.

Three separate equalities are represented within equation (17), namely,

\[
1 = b \psi_1 + b \psi_2 \tag{21}
\]

\[
G(t) = \psi_1 g_1(t + 1) + \psi_2 g_2(t + 1) \tag{22}
\]

\[
C(t) = \psi_1 c_1(t + 1) + \psi_2 c_2(t + 1) \tag{23}
\]

When we multiply the right hand side of equations (22) and (23) by \( b/b (= 1) \); and replace \( b \psi_i, i = 1,2 \), with their corresponding synthetic probabilities \( \tilde{P}_1, \tilde{P}_2 \), the prices of guaranteed resources \( G(t) \) and its options \( C(t) \) are

\[
G(t) = \frac{1}{b} \left[ \tilde{P}_1 g_1(t + 1) + \tilde{P}_2 g_2(t + 1) \right], \tag{24}
\]

\[
C(t) = \frac{1}{b} \left[ \tilde{P}_1 c_1(t + 1) + \tilde{P}_2 c_2(t + 1) \right], \tag{25}
\]

respectively. We consider the terms within the square brackets as a type of expected value, \( i.e., \) the weighted average of possible values from different states of the world. All the terms within the square brackets are multiplied by \( 1/b \), which accurately proves our earlier intuition depicted in Figure 36.
5.5 A numerical example and basic computations

To gain a better perspective of our mathematical representation, we put some numbers into the equations and illustrate our concept of fair pricing by means of achieving the no-arbitrage condition. If we set the cost of guaranteed bandwidth\(^45\) \(G(t)\) at 20 credits, say, depending on the supply and demand at an instance of time, \(G(t)\) may then assume two possible values in the following instant:

\[
G_1(t + 1) = 25, \quad (26)
\]

and

\[
G_2(t + 1) = 15, \quad (27)
\]

thus giving rise two to potential states of the world.

We also have the option that gives a sub-network the right purchase more bandwidth or CPU time and we set the strike price at 22.5 credits, and assume that the option contract will expire in the following period. Further, we put a charge of 1 credit on best-effort service that will provide a utility of 10%, say, of the paid credit. Leaving \(C(t)\) unspecified, we will obtain the following:

\[
\begin{bmatrix}
1 \\
20 \\
C
\end{bmatrix} = \begin{bmatrix}
1.1 & 1.1 \\
15 & 25 \\
0 & 2.5
\end{bmatrix} \begin{bmatrix}
\psi_1 \\
\psi_2
\end{bmatrix}.
\quad (28)
\]

If arbitrage is not possible, multiplying the dividend matrix with the vector of \(\psi_i\)'s yields three equations:

\[
1 = (1.1)\psi_1 + (1.1)\psi_2 \quad (29)
\]

\[
20 = 15\psi_1 + 25\psi_2 \quad (30)
\]

---

\(^{45}\) This example can equally be used to illustrate best-effort and reserved computational resources, relative to its associated 'derivative' product, \(i.e.,\) the right to reserve and purchase resources on-the-fly later.
Solving equations (29) and (30) gives \( \psi_1 = 0.272 \) and \( \psi_2 = 0.636 \). Thus, given the credits that we initially assign to the prices of best-effort and guaranteed resources, we then arrive at a fair price \( C(t) \) for the option to purchase further guaranteed resources as 1.59 credits.

5.6 Summary

We feel that fair allocation of resource prices presents the best way to achieve fairness in resource provisioning. End-users wishing to access a particular resource in a neighbouring domain must pay the access price worth the 'credit' currency of the target domain. The research conducted in this direction has been actively carried out in the past 5-7 years and various techniques and approaches have been proposed. In this chapter, we offered an alternative view to the subject matter, by incorporating the arbitrage theorem in financial markets to unify price levels for best-effort and guaranteed resources, as well as the right to extend resource reservation.

Furthermore, traditional carriers have been trading bandwidth amongst themselves for years, in the broadest sense, by buying and selling point-to-point SONET/SDH circuits. The carriers are accustomed to other forms of third-party representation as they pay agents to resell almost all their services at the retail level, and they continue to dabble at arbitrage with tiny fractions (less than 1%) of their international voice traffic on the 'minutes markets' through exchanges like Arbinet-thexchange, Band-X and The Global TeleExchange (GTX) [38]. In view of this, we felt that it was only natural to extend the underlying concepts of pricing financial assets into our task of establishing a fairer framework for allocating network assets.
As a caveat, we do feel that price transparency and competition will potentially flatten prices and as such, we imagine that ISPs may have their reservations about bandwidth trading, and thus, our approach.
Chapter 6

Inter-domain management in FAIN

We have thus far looked at service management in IP networks and discussed how the advent of Active Networking technology may impact the status quo. We then added value to the current initiatives by looking at the abstract problem of inter-administrative domain interactions from a multi-agent perspective. Finally, we will now describe lessons learnt from practical experiences of service deployment and management in Active Networks gained from the IST 5th Framework project, called Future Active IP Networks (FAIN). The agents described in Chapter 4 are now physically represented by the inter-domain manager (IDM).

The core of this chapter presents inter-domain management issues in the context of the FAIN project. Assumptions are first made about the management architecture as well as the network elements that are being managed. An overview of the software components within the FAIN network-level management system are also presented here because as the project's network-level management work area leader, I provided significant contribution to the design of the system. The role of the IDM is highlighted for reservation of computational and communicational resources to enable end-to-end service deployment, particularly when requests occur between different administrative entities. The implementation issues for the IDM as part of a contribution to the FAIN project work is explained.

More importantly, the ideas that extend beyond the scope of the project are presented. In particular, a signalling protocol for negotiating node and link resources prior to service provisioning is suggested and defined. This idea is validated by
incorporating the protocol into the FAIN demonstration environment, which was successfully shown at the final project technical verification meeting. The case study gives an idea of how the suggested inter-domain protocol may fit into an Active Networking environment within today’s Internet architecture (assumptions of which are described in §2.2).

6.1 State-of-the-art reviews

This subsection is broadly presented in two main categories in relation to the chapter. In §6.1.1, we look at policy-based management initiatives, which was the central idea behind the network management work in FAIN, while in §6.1.2, we summarised the inter-domain management considerations, which was the key responsibility that was taken up as part of our contributions to the project (see [114], [142], [144] for details).

6.1.1 Policy-based network management

In their efforts to diversify and enhance product range and offerings, vendors are reinventing network management, transforming its role from passive network monitoring to proactive QoS and network service level agreement provisioning by means of policy-based network management (PBNM) software. The purpose of a policy system is to manage and control a network as a whole, so that network operations conform to the business goals of the organisation that operates the network [138]. Ultimately, achieving such control requires altering the behaviour of the individual entities that comprise the network. To illustrate the concept of high level, ‘business policies’, some examples include:

All routers will run code version 6.2;

On-site contractors will all have special security restrictions on their ports;
Apply special forwarding to all ports whose customers have paid for premium service;

Each of these policies could represent an action applied to hundreds of thousands of configuration variables. In order to automate this practice, how does the network manager implement business policies across their network? How can the standard protocol that enforces these policies on all of their devices, regardless of the vendor, be derived? These are high-level questions that drove most of the research initiatives in this field. As a measure of complexity of these tasks, it is noted that the Internet Engineering Task Force (IETF) has set up no less than 3 work groups to address the questions raised. Further key issues for these initiatives lies in the development of well-defined, and elaborate information models in order to simplify the translation of policies from one level to another. The policy schema shown in §A.9 adheres to the IETF information model.

6.1.1.1 A generic policy-based management architecture

The Policy Working Group of the IETF is chartered to define a scalable and secure framework for policy definition and administration. The main goal is to support QoS management. This group has defined a framework for policy-based management that defines a set of components to enable policy rules definition, storing and enforcing. It identifies two primary main components by their functionality, i.e., a Policy Enforcement Point (PEP) and a Policy Decision Point (PDP).

http://www.ietf.org/html.charters/policy-charter.html
The generic model in Figure 37 illustrates the relationship between the PEP, which acts as the 'policy target'; the PDP, which operates as a 'policy consumer'; and the policy repository, which stores all the policies created for the administrative domain that it presides over.

The PEP is the component that actually encounters the data packets and is responsible for enforcement and execution of policy actions. It would typically be co-located with the packet-forwarding component of an access router or a network server.

The PDP is the component that is responsible for determining what actions are applicable to which packets via the use of policies (hence 'policy consumer'). A PEP may query the PDP to make decisions on its behalf on the occurrence of specific events, such as the arrival of a new reservation request or a data packet. In more complex systems, each PDP contains at least two types of component: the condition and action interpreters. These components provide action and condition processing logic for those policy types that are handled by the PDP. Each PDP has at least one instance of each type but can be dynamically extended to accommodate more interpreters that are capable of processing new actions and conditions conveyed by the policies.
The policy repository, a storage facility where the policies defined for the domain are stored, may be located in a single physical site within the policy domain, or it may be replicated at several devices. The repository could be a database, a flat file, an administrative server or a directory server. The communication between the PDP and the policy repository may be performed via several protocols, depending on the nature of the policy repository. Implementation-wise, when the policy repository is a network directory, LDAP\textsuperscript{47} is the protocol of choice for most vendors; and when the policy repository is a database, standard SQL\textsuperscript{48} queries can be used to access the policies stored at a repository.

The use of a repository like LDAP is important to support reusability of data across managed entities, as well as to allow an administrator to edit existing management data. In addition to being stored in a repository, the policy data must be moved to where it will be used, e.g., when the PDP needs to make a critical decision on behalf of the PEP. Information distributed from a centralised repository also aids in ensuring consistency of information throughout the managed environment.

Finally, policies are stored in the repository by means of a policy management tool. The policy management tool must provide functions that validate and ensure that policies stored in the repository are well-formed, mutually consistent and can be satisfied by the network.

6.1.1.2 Policy translation

There is a requirement for device-independent, 'lower-level' policies to control and manage heterogeneous networks. Policy translation holds the key to a more scalable policy-based approach to network management because a standardised policy

\textsuperscript{47} Lightweight Directory Access Protocol (or LDAP) is a protocol for accessing online directory services and runs over TCP (see http://www.openldap.org/)
\textsuperscript{48} Structured Query Language is used to communicate with a database. According to the American National Standards Institute, it is the standard language for relational database management systems.

127
information model must be used to help bridge the conceptual gap between a human policy maker and a network element that is configured to enforce the policy.

The policies to be deployed on the network elements must also cooperate with each other to accomplish required functions. It is observed that policies are high-level concepts that are translated into terms that relate more specifically to the actual infrastructure being managed [152]. For example, there are high-level policies, such as

Manager M must protect Dept D files from loss due to fire or media failure;

and low level goals derived from it, such as:

Domain BackupSW(read, write) Domain D files;

This low-level policy is one of several in a set which implements the high-level one by means of regular back up.

Policies can be seen as a constraint on how a system should work or how people should use the system. Goh [66] defines the constraint at an abstract level and refines it so that it is meaningful in terms of the real systems and the various locations and organisations in which they exist. Moffett and Sloman [106] envisaged a relationship between policies at different levels of a policy hierarchy, in which

Figure 38: Abstraction levels to aide policy translation
partitioning, refinement and delegation of responsibility could be shown to achieve the high-level policy. Thus, it seems useful to regard high-level policies as requirements, and the low level ones to be an implementation of those high-level policies. The standard techniques of a system life-cycle process can then be used to derive the implementation from the requirements [94].

Since higher- and lower-level policies may not necessarily correspond one-to-one, a higher-level network policy may have to be translated into two or more lower-level polices; or, two or more cooperating higher-level policies may have to be translated into one lower-level policy. Interestingly, Kanada [85] suggested that low-level policies deployed on devices could be regarded as the building blocks of a higher-level policy, and terms the former transformation as policy division, while the latter transformation is aptly termed policy fusion. However this idea seems to suggest that the process of policy division is the inverse of policy fusion.

Since policies (particularly those that deal with security issues) should ideally be strongly worded, unambiguous yet specific, it is suggested that a form of feedback mechanism, resulting from the enforced actions, be implemented in order to modify and improve the actual actions that take place. This would help to gradually eradicate unforeseen effects of policy enforcement. Research initiatives are driven to answer several questions, namely: is it possible to control action Y by adjusting policy X? Is the network-level view of a policy, which is an intuitive, high-level perspective of topology, connectivity, and end-to-end service objectives, sufficient to reconcile the feedback information in order to ‘re-author’ the policies?
The notion of a feedback relationship between a network element and a management system is an elegant approach, especially to automate correction of unforeseen results in policy enforcement.

There are often 'simplistic', high-level claims [147] that company objectives are translated into network-wide policies within the administrative domain on the necessary bandwidth, communication resources, and topology requirements, which are, in turn, translated again to node policy rules that are specific to the required behaviour of the node to manage its local resources. What a policy is has never been consistently defined, mostly because everyone has preconceived ideas about what the policies are. As such, different policy languages are derived, e.g., Keynote [36] and Ponder [46] initiatives. Thus it is also useful to analyse and unify the common properties of these languages as a first step towards a more standardised policy notation, which would help ease policy translation complexities.

6.1.1.3 Current policy-based research projects

When used in the context of computer systems, a 'policy' refers to any system configuration that controls its behaviours (e.g., security policies, quality of service policies, etc.) by effectuating a definite goal, course or method of action to guide and determine present and future decisions. While IETF's initiatives have been articulated in
§6.1.1.1, the Distributed Management Task Force (DMTF)\textsuperscript{49} Policy Model (cf. [107], [108]), on the other hand, provides a common framework for specifying system behaviours that are both sufficiently abstract to be independent of implementation-specific details and scalable enough to configure large, complex computer systems, \textit{i.e.}, the DMTF Policy Model is a specific model for expressing such policies in a general and wide-encompassing way. Developed jointly by the IETF and the DMTF, the Policy Model is an object-oriented model that enables constructing policy rules in the form of the event-condition-action (ECA) paradigm [151]. The ‘condition’ term is a Boolean expression used to specify the rule selection criteria. These criteria may include temporal conditions (\textit{i.e.}, when the rule is applicable), scoping conditions (\textit{i.e.}, on which devices do the rule apply) and state-related conditions (\textit{i.e.}, under what circumstances should the action(s) of the rule be attempted).

Apart from these standardisation activities, many research projects have also covered the field of policy-based management. As we have just noted, a relevant one includes the Ponder project. The Ponder project [46] developed one of the first technology- and manufacturer-independent policy-based management frameworks. It was well accepted within the research community and its results have been used in many research projects that explore the use of policy-based management. Ponder defines a language and framework for specifying security policies that map onto various access control implementation mechanisms for firewalls, operating systems, and databases using the Java programming language. It supports obligation policies that are event-triggered, condition-action rules for policy based management of networks and distributed systems. Ponder can also be used for security management activities such as registration of users; as well as logging and auditing events for dealing with access to critical resources and security violations. Other main concepts of the language include

\footnote{http://www.dmtf.org}
roles to group policies relating to a position in an organisation, relationships to define interactions between roles and management structures to define a configuration of roles and relationships pertaining to an organisational unit such as a department.

The Jasmin project [103] aims to enhance the distribution and invocation of network management scripts with distributed network management applications. The implementation supports multiple languages and run-time systems. A set of classes were added to support policy-based configuration management of Linux differentiated service nodes. In particular, general policy management language extensions, domain specific policy management language extensions and drivers realising the mappings between domain specific policies and the underlying device-level mechanisms, have been realised.

One of the first projects to work with policies in Active Networks was the Seraphim project [100]. It enables the extension of its security mechanisms by allowing the active code to dynamically install its own application-specific security functions. These code fragments, which are encapsulated inside active packets, have been named active capabilities (AC). An AC is able to carry the active code, the security policies customised for a particular application as well as the code required for making policy decisions.

Kanada [86], as mentioned earlier, suggested a method for the dynamic extension of a policy-based management system by means of policies in Active Networks. The Policy Extension by Policy (PxP) proposal is limited to the extension method, and must be included in a separate management architecture. The method defines two types of policies for realising this extension, i.e., Policy Definition (PD) policies and Policy Extension (PE) policies. On the one hand, PD policies allow a user to add a new type of policy into the Policy Server to specify the correct syntax and restrictions. Subsequently through the PE policies, users can specify the corresponding
methods for translating the new policies types into commands on different types of
network nodes. Both PD and PE policies are defined either by network operators or an
e external application.

The architecture where this extension method has been conceived is the general
policy-based management architecture that contains a graphical user interface (GUI), a
policy manager (or policy server), a database and policy agents.

When a user introduces a new policy, the policy manager verifies the (syntactic
and semantic) correctness of the policy with the information contained in the
corresponding PD policy. The policy agent translates the new policy into managed
device commands according to the instructions contained within the corresponding PE
policy. The way policies should be translated is described within PE policies by means
of templates. These templates are completed with the policy information using ‘fillers’.
The ‘fillers’ specify what information should be retrieved from the policy to complete
the template. A program interpreter is included inside each policy agent to evaluate
‘fillers’ and this allows specification of certain processing of the policy data before
inclusion in the template.

Fonseca [57] proposed a framework to allow the interoperability between
different ISP management domains satisfying end-to-end requirements given by users.
The proposed framework extends the policy-based management framework proposed by the IETF by including capsules for the communication between the different components of this framework. Capsules represent user requirements and are used for service negotiation and network elements configuration. They have defined three types of capsules: one to request decisions from the PEP to the PDP, one for notify decisions from the PDP to the PEP and a third one to negotiate between ISPs.

Kato [87] proposed a management framework designed to reduce management traffic by allowing network elements to take decisions. This is done by defining active packets that may contain policy parameters and code, which are executed inside network elements. This allowed network elements to take autonomous, intelligent decisions. The policy is edited in the GUI and executed in the Active Program Execution System (APES). APES is responsible for carrying out the actions specified by the policy at a given time, e.g., contacting another APES. Programmable packets are autonomously routed through all nodes to be managed.

We suggest viewing a policy-controlled network as a state machine, where policies control the state of a network element at any given time. Policy is applied using a set of policy rules. Naturally, each policy rule consists of a set of conditions and a set of actions. It could thus be interpreted that a policy is a specification of behaviours or actions to be taken in the managed environment; and as such, may be reactive (require an external event to trigger them) or may be proactive (e.g., policies are invoked as a consequence of temporal information).

The classes comprising the Policy Core Information Model [107] are intended to serve as an extensible class hierarchy for defining policy objects that enable service developers and network administrators to represent different types of policies. Policy rules may be aggregated into policy groups. These groups may be nested, to represent a hierarchy of policies. The set of conditions associated with a policy rule specifies when
the policy rule is applicable. The set of conditions can be expressed as either an ORed set of ANDed sets of condition statements, otherwise known as the Disjunctive Normal Form (DNF), or an ANDed set of ORed sets of statements, also known as the Conjunctive Normal Form (CNF). The individual condition statements can also be negated. If the set of conditions associated with a policy rule evaluates to TRUE, a set of actions may be executed such that the current state of the policy object is either maintained or changed to a new state.

6.1.2 Inter-domain management

Our approach is inspired by the 2-bit differentiated service architecture proposed by Nichols et al [113], which highlighted essential properties for reservation mechanisms, namely, to enable the dominant Internet traffic, e.g., web surfing, e-mailing and file-transfer, to remain best effort. Premium services like on-demand video streaming and teleconferencing are the types of applications that we want to allocate differentiated treatment. We also concur with the views that multilateral agreements rarely work. Instead, end-to-end services need to be constructed out of purely bilateral agreements between two consecutive ISPs. Like the bandwidth broker, the IDM manages the aggregated traffic per domain via a policy-based approach and are able to negotiate the required state information with its peers in handling end-users’ packets coming from within its domain.

We observed two possible approaches for implementing an entity for brokering end-to-end resource allocations. The first one, which we termed the agent-based approach, adopts a physically separated agent for each administrative domain for aggregating reservation sessions. Variations of this agent-based approach have been described by Schelen and Pink [129], and Blake et al [34]. The advantage is that it removes (control) message-processing burden from the routers. The disadvantage is that
synchronising reservation information (in view of further negotiations) among routers may be complex.

Baker et al [24] and Guerin et al [68] have proposed a **router-based approach** by modifying RSVP to support scalable reservation, where the processing logic stays within the router, as opposed to a separate machine. As in our approach, bandwidth brokers [113] are agent-based approaches for differentiated services while router-based approached usually necessitate string per-flow state on the forwarding nodes itself. In Li and Rekhter's provider architecture for differentiated services and traffic engineering (PASTE) [99], they used MPLS for traffic aggregation and RSVP to set up the label switched paths (LSPs). Traffic classes are split into best effort, priority, and network control. Consistent with our assertion when we summarised Chapter 2, they recognised that because most flows on the Internet today traverse multiple ISPs, an architecture that enables creation of inter-domain SLA is required. It is further felt that computational resources must be reflected in the SLA as well as the negotiation parameters, together with the usual communication aspects like bandwidth, delay and jitter.

### 6.2 The FAIN management system

It is important that we first set out our key assumptions and definitions to facilitate comprehension of the subsequent issues described in this chapter. Firstly, each element management system (EMS) only manages a single node – active or passive. Here, we think in terms of individual network elements, typically from a single vendor. Secondly, a network management system (NMS) looks over these EMSs, while the inter-domain manager (IDM) is located within the NMS. Network managers are systems we most often associate with general network status. The ubiquity of the simple network management protocol (SNMP) enables a wider range of vendor scope.
When a content provider intends to set up a virtual environment (VE) on a neighbouring ANP’s infrastructure, the neighbouring NMS first receives XML-formatted policies from the ANP of the requesting domain, i.e., where the content provider resides on. Network-level policies are processed by the NMS’s policy decision points (PDPs), and are then sent to the NMS’s policy enforcement points (PEPs) to translate (see §6.1.1.2) and map them into (potentially more than one set of) element-level policies. When the EMS receives element-level policies, it performs similar processes, except that the element-level PEP resides within the management VE (cf. §3.2) on an active node. The PEP finally executes enforcement actions on the managed node according to the policies, i.e., setting access rights on the nodes for the use of the requisite resources. While the FAIN project stopped at only managing active nodes, we extended the initiative to enable and integrated policy for management of passive nodes. If the managed node is a passive router, then the EMS PEP resides on the EMS itself and simply makes use of SNMP ‘set’ commands to change the router’s configuration. See §A.8 for the QoS XML schema that reflects our work.

50 The policies are defined and categorised according to the semantics of management operations, e.g., QoS and VPN, and are accordingly processed by dedicated Policy Decision Points (PDPs) and Policy Enforcement Points (PEPs).
The main components of the policy-based network management architecture are briefly described in the following sub-sections before we proceed to §6.3 for further discussions about the inter-domain management functionality.

6.2.1 PDP manager

The PDP Manager is responsible for forwarding policies to the appropriate PDPs. If the PDP for the policy is not installed, the PDP Manager will request the active service provisioning (ASP) system (see §A.7) to download and install it, thereby extending the management functionality of the system as required. This component also acts as a control point, as it has all the necessary information to understand the semantics of policy processing. For instance, consider two different sets of policies to be deployed, whereby the latter is only deployed if the former is successfully enforced. In this case, the PDP Manager keeps the second one in a dormant state, until it receives notification of successful enforcement for the first policy.
6.2.2 Policy decision point (PDP)

The PDP's main function is to check for possible syntactic and semantic conflicts [101] in policies. The PDP also decides when and where a policy should be enforced, which mandates the need for the PDP to interact with the monitoring system. It needs to forward decisions to the correct PEP components for enforcement. The FAIN management architecture accommodates different types of PDP, i.e., the QoS (represented by the white box in Figure 42), Delegation (represented by the lighter grey box) and the Monitoring (represented by the darker grey box) PDPs. In order to reach a decision, they sometimes interact with other components that assist the PDPs in making a decision, e.g., the resource manager for admission control.

6.2.3 Policy enforcement point (PEP)

Each type of PDP has several PEP counterparts. Network-level policies are translated by the network-level PEP into element-level ones, and are then sent to the pertinent element-level PDPs that reside within the EMSs. This translation takes account of the topological information of the EMS PDPs, i.e., policy translation is associated to the location where policies are to be distributed. Similarly, element-level PEPs enforce the policies sent by the EMS PDPs by mapping them onto the FAIN active node open interfaces. The use of open interfaces is consistent with the objectives of P1520 [77] and allows all PEPs across the network to share the same view of the nodes' control interfaces, thus making them node (hence platform) independent.

6.2.4 Service manager

The service manager is responsible for setting up a VAN in response to a service request. It receives, as an input, context information that specifies service start and end points. It uses this information together with the topological requirements (see

---

51 In FAIN, we assume that VAN creation is always driven and preceded by a request for service deployment.
§A.8) imposed by a service, which it retrieves from the ASP, to generate the relevant policies for the service. After a VAN has been successfully created, the service manager instructs the ASP to trigger the deployment of the requested service.

6.2.5 Resource manager

The resource manager maintains a view of the connectivity between network elements and the availability of resources within a domain. This sub-component is mainly used during the creation of a new VAN for service installation. During these processes, the resource manager has to communicate with both the QoS PDP and the ASP. For each node, it stores the available properties such as its public IP address, the private address that the node may have inside the FAIN testbed, the types of VEs that are supported by the particular node, and the links attached to the node. For each link, the available properties are the IP addresses of the start and end nodes, the total capacity of the link and the current available bandwidth of the link (cf. §A.8 for the interface definition language or IDL representation of property structures).

6.3 Inter-domain manager

6.3.1 Overview

To implement end-to-end guaranteed service across the Internet, it is inevitable that the traffic flow for the (negotiation) control signals and packet flow for the service must propagate across different domains. We had earlier (cf. §2.4.1) defined a domain as a collection of nodes by a single administrative entity where security and management policies are uniformly applied. The different administrative domains under discussion are owned by separate organisations. The IDM is a software agent that represents a domain for route discovery and resource negotiation in order to achieve a consistent agreement for service provisioning among all the domains en-route from the
required source to destination. The NMS must abstract the service request of an end-user (via the content provider) within its domain to present it to the target domain. For this reason, we assume that both source and destination domains use the FAIN PBNM; and as such, will understand the service parameters for a particular request.

6.3.2 Design and implementation

When the NMS receives a service deployment requested by the content provider that involves a target node (i.e., the content provider’s customer/end-user) beyond its domain, the IDM will contact its peer IDM in that target domain in order to negotiate service deployment on one or more of their active nodes.

![Figure 43: RMI communication using stub and skeleton](image)

We adopt the Java Remote Method Invocation (RMI) as the communication channel for the distributed system. The basic assumption in RMI is that all participating programs are written in Java. This makes for a simpler framework for prototyping our ideas. We use Java's socket classes to handle communication between distinct processes, i.e., the negotiation protocol. The communication between IDMs requires three features that are not provided by a connectionless protocol: IDMs that send BID and ACK messages (see §6.4) require confirmation that their request has been received; IDMs that receive these control signals need the ability to validate (and request retransmission) of a request; and finally, both negotiating IDMs need the
communication mechanism to preserve the order in which the negotiation messages are sent.

For these purposes, the protocol implementation is naturally helped out at transport layer by the transmission control protocol (TCP), \textit{i.e.}, it gives us a conceptual communication model (between IDMs) akin to a ‘direct conversation’ rather than a ‘courier service’. Because TCP/IP networking is almost always implemented as part of the underlying operating system, using RMI means that Java, which uses its libraries to access the operating system’s TCP/IP functionality, automatically provides us with a reliable negotiation channel.

In our implementation (see §A.10 for source code), we established a repository that maps destination sub-network addresses of a domain to an IDM, \textit{i.e.}, a ‘many-to-one’ relationship. As such, an ANP must register a list of destination addresses within its domain on a repository that has a well-known address, so that this repository can provide a discovery mechanism for mapping destination addresses to their respective IDM.

The communication interface for IDMs makes use of sockets. The Socket class in Java enables a single connection between two known, established processes. In order to proceed with negotiation, two communicating IDMs must have created instances of Socket. This happens when an IDM registers at the repository to set up the domain to listen to peers. The other important class, the ServerSocket class, manages initial connections between a client IDM and a server IDM, \textit{i.e.}, when the requesting domain first connects to its target domain using an instance of Socket, it first communicates with an instantiation of ServerSocket. This marks the beginning of the handshaking phase, which initiates the resource negotiation phase.
Within an NMS, the IDM interfaces with two other key components, i.e., the resource manager and the service manager. The effect of considering an inter-domain view is shown in Figure 44. The resource manager will invoke the `relay (Context, ServiceDescriptor)` method on the IDM in Domain A. Context represents the ingress IP and the destination IP of the reserved route. For ServiceDescriptor, it would be sufficient to input the service component name. The resource manager should extract this information from the VANPath (see §A.8) information that it has. The intra-domain process is stalled at this point. The IDM in Domain A will request the IDM in Domain B to extend the VAN in the neighbouring domain. The service manager’s `deployService (ServiceName, VANNode[], Credential)` method will be invoked by the IDM in Domain B. The VANNode (see §A.8) array is a pair of ingress-destination IP address abstracted form the context information obtained from the resource manager.

![UML sequence diagram](image)  

Figure 44: UML sequence diagram depicts interaction between the subcomponents of the IDM with the other key components within the NMS, i.e., the resource manager and the service manager
6.3.3 IDM enhancement

In FAIN, the IDM implementation relied on a separate software component known as the resource manager (see §6.2.5) to provide static information about the current computational resource availability. The implementation read from pre-defined information within a text file. We extended the IDM function with SNMP management capability so that the IDM can play the role of a manager in the manager-agent relationship between the management station and the managed device.

We basically enabled the IDM to poll for MIB variables on a managed device to discover current levels of resources usage within its own domain. This allows the IDM to make a more informed choice when deciding on whether to accept or reject a request for bandwidth, say, reservation. We make use of the well-defined MIB as described in RFC 2790.

The Host Resources MIB [148] defines a uniform set of objects useful for the management of host computers. The term ‘host’ is construed to mean any computer that communicates with other similar computers attached to the Internet. Although this MIB does not necessarily apply to devices whose primary function is communications services (e.g., software routers that run variants of Unix), such relevance is not explicitly precluded. Thus, the IDM makes use of this MIB to interrogate the routers within its domain to find out the current resource consumption in order to respond to resource requests by its peers. The enhanced IDM functionality was utilised in the case study in §6.4.4.
6.4 Inter-domain resource-centric negotiation protocol (IRNP)

6.4.1 Motivation

We now proceed to articulate a protocol for resource negotiation in a multi-domain setting. This requires that the ISP network administrator recognise and observe this negotiation protocol in order to promote a stable and orderly manner for (high-level, see §2.6) service provisioning and resource reservation. The expected final result is that an end-to-end path will be established between the end-user and the content provider’s media source. All nodes en-route would then have adequately provisioned resources as required by a service. This path is created based on the underlying connectivity of the network, i.e., the low-level service that holds the Internet together. Resource allocation will be achieved via a ‘chained’ peer negotiation, as opposed to having a centralised entity to arbitrate the end-to-end provisioning exercise. This chain depends on the underlying routing connectivity. If two domains exchange route reachability, then there is a link - hence a two-node chain is formed. The main objective of this protocol compared to its predecessors is to agree upon the pertinent active nodes and its corresponding resources to be reserved by a peer domain.

Traditional approaches (e.g., TCP), which describe protocols in terms of action sequences, limit the flexibility of the agents in executing the protocols. In contrast, we use an approach for specifying protocols where the content of an action can be captured, thus introducing the notion of negotiation within the signalling phase. Capturing the intrinsic meaning of the actions and explicitly representing them as part of the protocol brings in flexibility to the protocol, permitting the agents to reason about their and others’ behaviour during the execution of the protocol, and enabling them to modify their actions as best suits them.
Active Networking increases the complexity and customisation of the computation that is performed within the network interposed between the communicating end points. Prior inter-domain reservation approaches do not address this aspect. Moreover, since neither IntServ nor RSVP [39] deals with negotiations between providers, the problem is amplified when crossing domain boundaries. For these reasons, our service management protocol is designed such that RSVP may be used for intra-domain reservations (no negotiations required), while inter-domain reservation protocols set up coarsely-measured reserved flows between domains. Intra-domain issues are beyond the scope of this thesis, and we will assume that a reservation could be accomplished within a domain as and when needed.

6.4.2 Assumptions

The negotiations will be constrained by a set of parameters (see Table 2) to ensure that all entities involved in the ‘bid-ask’ (see §6.4.3) process have a common understanding of the negotiation semantics. In addition, this would mean that the overall objective here is not to pursue agreement on the resource parameters (i.e., resource profile) to be reserved, but on their absolute or relative values. A further assumption in this work involves the understanding that the negotiation process will only involve two entities at any single session, thus mimicking the simple client-server paradigm of interaction in network theory. It is also easy to see that a basis for any negotiation is initiated by a client entity.

It is also assumed that reservations are additive, although in practice we note that statistical multiplexing could be used. The process of negotiation will be dictated by the protocol and defines the rules of encounter between IDM's. Beyond this, given our protocol, there exist various strategies that can be adopted by IDM's to maximise its utility while considering any externalities that may arise. The properties of this protocol should include: optimised joint utility, stability, and guaranteed conclusion. Based on
our results from the metanorms simulation, a protocol will be stable if it provides all entities with an incentive to act rationally. An agreement should ideally be reached in all negotiations. We aim to provide a richer set of behaviours and optimisations that can be constructed from a simple framework by providing the option for negotiation.

6.4.3 The protocol

Network protocols are designed so that different entities on the network can interoperate with each other. IRNP’s objective is to allow two administrative domains to adaptively peer with each other so that the end-user needs for each service can be customised and negotiated as needed. An important property of a protocol is the flexibility for all entities to express preferences about which other entities they interact with. Our protocol will permit all parties to express choice and to enforce their local polices in order to serve their interest.

There are three basic sessions of interactions between the negotiating entities (implemented as IDMs), i.e.; (i) to achieve an agreement; (ii) to modify an agreement; and (iii) to terminate an agreement. The termination session is a simple single-step process. The modification session is similar to the first step of attempting to achieve agreement, except that negotiation begins with the previous set of agreed values between the two relevant entities. Before the negotiation process iteratively converges towards an agreement, the ‘client’ and ‘server’ IDMs will take turns to refresh their preferred values for the resource profile.

Before proceeding with the examples, to help summarise the messages that can be sent by the negotiating agents using this protocol, see Table 1. In general, BID and ASK messages can be sent to-and-fro between two negotiating entities until either party agrees to the last revised offer of resource values. In order for either party to

---

52 Each time, the IDM server will respond to the IDM client’s request by revising all the preferred values for the end-user’s resource profile.
signal towards an end of negotiation and to have finite\textsuperscript{53} steps in the negotiation, the BID and ASK messages is further refined to FIN_BID and FIN_ASK respectively. This forces a party to decide on whether to finally accept or reject (using REJ messages). There are essentially two ways that the negotiation process may end: when either party send a REJ message at any point of the process; or when an ACK message is sent in response to another ACK message. The latter is ensued by the actually process of setting up the respective network nodes with the requested service, while the former signifies the end of the reservation sequence without reaching an agreement.

<table>
<thead>
<tr>
<th>IDM</th>
<th>Messages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INIT_NEG</td>
<td>Starts off the negotiation process and incorporates the first BID values for the requested resources</td>
</tr>
<tr>
<td>Client</td>
<td>BID</td>
<td>Sent in response to an ASK message; refreshes the ASK values</td>
</tr>
<tr>
<td></td>
<td>FIN_BID</td>
<td>Also sent in response to an ASK message, but forces the server to either accept (via the ACK message) or reject (via the REJ message) the values proposed in this message</td>
</tr>
<tr>
<td></td>
<td>REJ</td>
<td>Ends the negotiation process; sent in response to an ASK or a FIN_ASK message</td>
</tr>
<tr>
<td></td>
<td>ACK</td>
<td>Agrees on the last revised set of resource values</td>
</tr>
<tr>
<td></td>
<td>ASK</td>
<td>Sent in response to either the INIT_NEG or the BID message; revises the values bade by the client IDM</td>
</tr>
<tr>
<td>Server</td>
<td>FIN_ASK</td>
<td>Also sent in response to a BID message, but forces the client to either accept (via the ACK message) or reject (via the REJ message) the values proposed in this message</td>
</tr>
<tr>
<td></td>
<td>REJ</td>
<td>Ends the negotiation process; sent in response to a BID or a FIN_BID message</td>
</tr>
<tr>
<td></td>
<td>ACK</td>
<td>Agrees on the last revised set of resource values</td>
</tr>
</tbody>
</table>

Table 1: Description of messages for the inter-domain resource-centric negotiation protocol (IRNP)

\textsuperscript{53} The server side of this protocol can specify the maximum number of BID-ASK interactions before a negotiation draws to a conclusion.
We first consider a reservation sequence where traffic is only required to hop between two separate domains. Before two IDMs negotiate, they must have had a prior agreement to enable each other to send a request for resource reservation on behalf of their end-users and subsequently to 'haggle' for the required volumes of resource. This could be done at the SLA definition phase, e.g., both parties should agree that each request for resource is negotiable, and so it is clear that this protocol takes place after the static process of establishing an SLA has been conducted. Negotiations occur in response to a service request that demands for guaranteed levels of resources beyond a domain. There must be at least one router in a domain that understands IRNP so that it can divert the initial request (i.e., INIT_NEG messages) for resources to the IDM. The control packets may be marked using the IP precedence bits in the IP header.

![Sequence of signalling messages to conduct negotiation as dictated by the inter-domain resource-centric negotiation protocol (IRNP).](image)

To begin a negotiation, the client sends an INIT_BID message to request for resource. The resource profile allows for values to be proposed for various node and link 'assets' (cf. §5.4), thus setting up a 'bargaining' environment between the client IDM and the server IDM. The server will respond to the request by sending an ASK message that basically revises all its preferred values for the resource profile, relative to those proposed by the client. If the client is happy with the values in ASK, as returned by the server, it then confirms acceptance of the server's ASK by sending an ACK to
the server. This means that the client has agreed to the resource levels offered by the server.

| Pricing        | • flat-rate  
|               | • usage-based 
|               | • quality    |
| Quality of service | • best-effort 
|                 | • specified  |
| Resource       | • memory – bytes 
|                | • link bandwidth – bits per second 
|                | • node processor time – milliseconds |
| Duration       | • until-further-notice 
|                | • specified  |

Table 2: Information model for negotiation parameters

Next, we show the negotiation sequence where a traffic flow needs to hop across two domains, which would mean involving at least three separate domains. Our reservation algorithm reserves resources along the existing route set up by whichever underlying routing protocol in place. The ACK message will traverse the exactly same path as that by the BID message but in opposite direction. The context of our perceived agreement is a network of IDMs that have a means for conducting several synchronised phases of information exchange. After that, they must all agree on some set of core information, e.g., time and duration of reservation\(^{54}\).

Figure 47: Sequence of signalling messages to conduct negotiation when reservation is required across two (or more) other domains

\(^{54}\) Level of commitment on value of resources vary for each domain
6.4.4 Discussion

Pan [116] accurately pointed out that the reservation path in an inter-domain environment depends as much on an ISP’s policy as it does on resource availability. We concur with this rationale because we feel that a routing domain en-route might have the necessary bandwidth and node processing ability to support a VE, but chooses not to allocate them to the requestor due to its own reasons. Thus, as in Pan’s BGRP implementation, it is the responsibility of the resource requestor to probe the network to ensure end-to-end routing path for its reservation.

It is the neighbouring IDM’s discretion to issue a reject message (i.e., an END message sent prematurely, prior to any negotiation) back to the original sender. As such, we can naturally see that there is a need for the IDM to reconcile two sets of information: a domain-wide policy (stored within the IDM’s repository) that reflects its strategy; and actual resource availability based on its polling results (cf. §6.3.3). The policy reflects the strategy that a domain chooses to adopt with regards to its neighbours.

In essence, while we have described and implemented the IRNP with a signalling protocol in mind, on reflection, it may be more accurately seen as a management application. For the objectives that we were pursuing, various other approaches, which may prove more efficient, exist. The novel idea of adding a negotiation aspect as well as flexibility to the multi-domain resource reservation process, nonetheless, still stands.

6.5 Case study

This case study was presented during the final FAIN technical verification meeting on the project testbed, which is one of the pioneering Active Networks that interconnect both active and passive nodes (belonging to the consortium’s partners)
using IP tunnels over the public Internet. We logically divided our testbed to simulate
different administrative domains. The final demonstration was well received by the
three experts in the verification panel. A segment of this sub-section represented an
input to Chapter 18 (WebTV Scenario) of the FAIN book [59], where we were the
chapter authors.

![Diagram showing the FAIN pan-European testbed](image)

**Figure 48:** The FAIN pan-European testbed logically divided to represent to separate
administrative domains

### 6.5.1 Motivation and requirements

The main objective of this scenario is to deploy service components, *i.e.*, the
duplicator and the transcoder\(^{55}\), so that different formats of video streams can be
converted to suit an end-user's needs. End-users first subscribe to the WebTV service
by contacting the WebTV server. If an end-user uses a terminal that is not capable of
correctly displaying the video stream, *e.g.*, she may be using a handheld device with low
processing power and a low access bandwidth, the WebTV provider (*i.e.*, the content

\(^{55}\) The transcoder is a software component to convert a video format
provider in our definition) can individually select and process the video stream destined to this end-user by deploying an audio/video transcoder in the network so that the video stream can be received by the handheld device in an appropriate format. Related work includes Keller's multicast video distribution architecture involving knowledgeable active routers [88], Fankhauser's WaveVideo [52], and Hoffman's layered transmission scheme for media stream [73].

As the various types of end-user devices for Internet access grow, so do the different protocols for handling multimedia streams. If we were to rely on the lengthy process of standardisation to move on to development and finally to the deployment process, the product lead-time would inevitably be unacceptable for any business model. Thus, we set out a practical framework here that may help a network node understand new protocols as and when required, as long as an Active Networking environment is available, i.e., the right execution environments (e.g., a Java Virtual Machine) are present to run a software component that can enforce these protocols.

This case study also helps to highlight the management aspect of the service provisioning exercise by illustrating the communication between two IDMs when allocating resources between neighbouring domains.

The ANP at Domain 1 must ensure that the administrator of Domain 2 would adequately provision the VE resources on the active node to allow the WebTV provider's (i.e., a content provider) transcoder to run (see Figure 49). The ISP in Domain 2 is the connectivity provider to (probably) a subset of the WebTV provider's end-users. Content providers need to make sure that their ISPs have established contracts\(^{56}\) with neighbouring ISPs to encompass the geographical spread of their targeted client base. This is typical of today's inter-domain requirements except that in

\(^{56}\) Peering or transit, as described in §2.2.3, as well as the agreement to negotiate computational and communication resources prior to service provisioning.
our context, the ISP en-route (if a service traffic flow needs to be processed on intermediate nodes) and the destination ISP (if a service traffic flow needs to be processed in ingress nodes of the destination domain) must also be ANPs, i.e., computational resources need to be reserved.

Legend

- Virtual active network (VAN)
- Virtual environment (VE)
- Management instance (MI) for WebTV content provider

Figure 49: Video-on-demand delivery

There are basically two options for the content provider to propagate its reservation sequence throughout the Internet domains in which it wants to set up a VE. Firstly, the content provider could separately negotiate with each ANP the parameters for resource reservation per domain via the ANP’s network-level management system. However, this is a rigid approach that would require static allocation of all its resources before a service is rolled out.

Thus, we propose the alternative approach such that an ANP could set out its requirements to its neighbouring ANP, which would, in turn, negotiate (see 6.4.2) the desired VE parameters with its neighbouring domains, in order to satisfy the ‘least

---

57 Our inter-domain negotiation (see §6.4) only reserves resources along the existing route set up by whichever underlying routing protocol in place.
The end result for both options is a virtual Active Network (VAN) [42] established for the content provider that encompasses multiple administrative domains, achieved by means of the IDM located within the two neighbouring FAIN PBNM systems. Setting up a VAN involves the alignment of requested and available resources. Thus, before an ANP can commit to the requested resource profile, it must be able to ascertain that the requests adhere to its local policies and resource availability.

6.5.2 Use case

The WebTV provider intends to establish a VAN between, say, a sub-network A in Domain 1 and another sub-network B in Domain 2 (see Figure 49). In FAIN, we create a VAN ad-hoc, in response to a service deployment request. The WebTV provider inputs the service-specific topology into a registry within the active service provisioning (ASP) component (with reference to Appendix A.7, this is known as the ‘service release’ phase). The service topology contains both the computational and communication resource requirements of service components that make up a service (e.g., bandwidth, processor time, memory, and duration of resource usage).

The deployment of the service component is triggered when the resource manager (see §6.2.5), upon receiving input in the form of QoS policies (i.e., XML documents translated from SLAs). With further reference to Figure 49, the QoS policy provides connectivity information: source IP (i.e., sub-network A) and destination IP (i.e., sub-network B). This information, together with the service topology from the ASP, and the network topology that the resource manager deduces via the PBNM’s monitoring component, enables the resource manager to recommend exactly which physical nodes to deploy code modules in order to realise end-to-end active service.
When the resource manager detects that sub-network B is not within Domain 1, but is in Domain 2 and only accessible via the edge router, then the IDM comes into play. The resource manager will supply the IDM with the next-hop IP address of the egress router of that domain, which effectively points to the ingress router of the target domain. The IDM in the originating domain knows that it can fulfill WebTV provider's (connectivity and resource) requirements from sub-network A to the edge (egress) router, but cannot meet the requirements from edge (ingress) router to site B.

IDM 2 passes this information down to its corresponding RM and the provisioning process in the neighbouring domain proceeds locally as in an intra-domain scenario. IDM 2 negotiates with IDM 1 based on what it knows from its own domain and what is required by Domain 1.

6.5.3 Management by delegation

In recent years, new management paradigm proposals tried to overcome some of the key deficiencies of SNMP. In a related work, the Management by Delegation (MbD) [67] paradigm proposes a distributed hierarchy of managers that solves the problem of polling distance between the manager and the agent. MbD was expected to be a more scalable proposition when compared to the SNMP model because if data analysis is only conducted at the management station (as is the case for the latter), it will require data access and processing rates that do not scale up for large and complex networks (e.g., the Internet).

Recall that our scenario involved a content provider who intends to provide video streaming, which we referred to as the WebTV service, to a group of end-users regardless of the latter’s terminal capabilities (e.g., PC, mobile phone). As such, it will do two things: request an end-to-end link that includes active node resources; and advertise the service on a website, say. When an end-user decides to purchase the
service, a SIP negotiation [126] occurs between the end-user and the web portal. The portal triggers the deployment of the transcoder by contacting the ANP’s management station and providing it the parameters of the end-user (e.g., what video format it expects). The ANP will receive a QoS policy to be enforced on both the NMS and the EMS.

Since the active node resources can now be partially controlled by a set of different owners, namely the ANP, who owns the system, and the content providers who ‘rent’ VEs on the nodes, the management system must now equally evolve to allow the content providers to self-manage their allocated resources. Thus, we introduce the concept where the ANP, along with all the other ‘tenants’ of the active node, will have copies of the management software to manage their respective VEs\textsuperscript{58}. These copies are called **management instances**.

![Management instances illustrated on simplified FAIN management systems](image)

In order to explain the concept of management instances, we will now proceed to extrapolate from our earlier use case by tackling how the management system

\textsuperscript{58} The ANP controls the operations from the management VE on the active node. It has privileges akin to the 'root' user in the Unix operating system.
handles policies that are sent to it. It is essentially a variant of the customer network management (CNM) concept, which allows network configuration, performance, and fault information to be directly provided to end customers on a near real-time basis. Our management instance approach further enables the end user to install its own management components in the form of dedicated PDPs and PEPs, as we elaborate below.

As a result of an earlier SLA agreed between the ANP and the SP, policies are sent to the ANP's management instance. Consequently, the ANP's PBNM receives a QoS policy and enforces it on both the NMS and the EMS. This invokes the node management framework to create a new VE for the content provider. If the VE creation is done successfully, then the ANP further enforces a delegation policy through the NMS and the EMS, which will give the content provider the privilege to use the PBNM to manage the resources reserved on the active nodes. The ANP then creates a management instance in all the appropriate EMS stations for this WebTV-SP and assigns the access rights (based on the delegation policy) to the active nodes interfaces. The WebTV provider is now ready to configure its allocated resource by sending service-specific policies to its management instance.

The WebTV-SP will install the transcoder and duplicator service components using its instance of the PBNM. In addition, the WebTV-SP deploys service-specific policies in the QoS PDP of its management instance. In this manner, the WebTV-SP can define further service-specific policies that will be enforced in the active node. For example, using monitoring policies, the monitoring system can used for the reconfiguration of the transcoder at runtime when the access bandwidth changes dramatically and the end-user needs a different transcoding format on the video stream.

---

59 The 'service' in this case is obviously the WebTV service, which requires service components like the transcoder and the duplicator to be deployed on the VEs.
6.5.4 Added value

How does this case study add value to the current manner of providing streaming video? How has Active Networking (in particular, the out-of band approach) improved matters? Looking back at the WebTV use case in §6.5.2, the key contribution by network programmability stems from the ability of the WebTV provider to install software components ‘on-the-fly’ to accommodate the various formats of video data encoding techniques. Without this ‘convenience’, these software enhancements would have to be embedded within a passive router at the hardware manufacturing phase and would (most often) be proprietary in nature. This advantage gives service providers the freedom to test out new service offerings without having to go through a lengthy standardisation process. More importantly, should the venture not prove to be profitable, the sunk cost would be lower.

6.6 Summary

The service deployment and management approach presented in this chapter represents a mechanism for the Active Network provider (ANP) to establish and support a service-centric overlay-network, *i.e.*, a virtual private network that is comprised of strategically located active nodes. Within the FAIN framework the overlay network, referred to as a VAN, was able to create a ‘service channel’ that meets the requirements for a customer’s (*i.e.*, the WebTV / content provider) request to deploy services to its own end-users.

Having allocated resources for this purpose, the content provider is further delegated with the task of managing its own private network, thus extending from the concept of customer network management. This approach has an added advantage of allowing the content provider to dynamically extend its private infrastructure by means of adding new PDP/PEPs to manage newly-created service components within the
confines of its delegated rights on both the management station and the active node. While this idea and prototype implementation sounds interesting, there is a clear challenge for the ANP in finding the optimal VAN path.

Setting up a VAN involves the alignment of requested and available resources. However, before an ANP can commit to the requested resource profile, it must be able to ascertain that the requests adhere to the local policies. The added value that the IDM adds to this pre-SLA status is the ability to negotiate a lower resource value if it cannot fulfil the initial request. For this functionality, the IDM must be able to reason about requests.

We could have further introduced the concept of testimonials, i.e., how many other domain NMSs have acknowledged your previous track record of negotiations. There ought to be routine checks on testimonial data before negotiation commences. ANPs have a responsibility to adhere to submitting testimonial. Ignoring it will have a negative impact to interconnect with other domains – ANPs should avoid extending interconnection services to those ANPs that disclaim such responsibility. The various facets involved in using policies as a tool to enhance scalability of management tasks have been discussed. A policy system devised in our framework shifts the focus from configuring individual devices to setting policy for the network in aggregate and controlling domain strategy through network policy.

We concluded the chapter by restating how an out-of-band Active Networking approach could improve media streaming on the Internet by analysing the WebTV scenario, which was designed, implemented and demonstrated at the final FAIN project technical verification meeting.
Chapter 7
Conclusions

This thesis investigated how the cooperative approach to dealing with a potentially competitive situation may be adapted into a network context. It was observed that by allowing the participants of a network bottleneck to monitor and punish each other, while modelling them with different combinations of boldness (i.e., the propensity to defect) and vengefulness (i.e., the propensity to punish) levels, a more predictable and stable form of interaction may be created in the long run. Practical implementations of this concept were also developed within a real network bottleneck environment and the design issues for further enhancement were discussed. Although the results explicitly refer to a network bottleneck as a test environment, it is felt that we may equally adopt this approach in the context of access networks, as explained in §7.2.1.

Having established stable multi-domain interactions, the next key aspect of service management that was dealt with included the resource reservation and allocation phase. The main emphasis was to include computational resources as a parameter for end-to-end negotiation, and we presented a framework that was integrated into a larger scale implementation in the form of the FAIN project. In our experience of adapting conventional management and control tasks for Active Networks, we opined that unless the research community can guarantee higher performance for active nodes, it would be difficult to justify a significant shift from traditional to programmable networks, especially in the network core. Thus, Active Networking may, at best, be deployed at the network edges, where the performance requirement is not crucial. To complete the
service management discussion, we presented and briefly analysed a possible approach that can be used for pricing best-effort, reserved, and optional resources, which was adapted from financial theory.

7.1 Comparison to related work

In a related work by Gibbens and Key [62], based on similar tenets of game theory (i.e., to alleviate the tragedy of the commons), the end-user’s strategy for resource optimisation was investigated. The key similarity to our work is their use of control signals [124] that indicate a cooperative approach, where user behaviour is constrained, and policed. Instead of using congestion signals, we used direct commands to the upstream router to punish a peer that causes congestions at the bottleneck.

Semret introduced an auction-based approach to the pricing of edge-allocated bandwidth in differentiated services Internet [97]. The work investigated the feasibility of maintaining stable and consistent SLAs across multiple networks where demand-driven dynamic allocations are made only at the edges. He further went on to establish a derivative market in admission control [130].

In Pan’s Border Gateway Reservation Protocol (BGRP) proposal [116], he divided the Internet into stub and transit domains and had sink-trees built for each stub domain in order to aggregate traffic. Reservations from other providers following inter-domain routing path to a destination provider’s network form a tree, rooted at the destination provider’s border router. One of the most challenging aspects on provider-level signalling is that the protocol needs to be applicable and scalable to potentially all network providers in the Internet.
7.2 Future work and open issues

The Internet is, in fact, a loose amalgamation of computer networks run by different organisations. Most technical decisions are made by small communities of volunteers, e.g., the IETF, who set standards (as seen in the RFCs frequently cited in this thesis) for interoperability. These standards implicitly enforce cooperation amongst different computer networks. The work presented in this thesis requires a further commitment for cooperation in that the sub-networks are expected to proactively monitor their peers' usage of their shared resources.

When discussing service provisioning on the Internet, the issue of its quality must inadvertently be answered. Similarly, one must also address the array of service types that can be provided by the ISPs. What can one do by having intelligence in the network to improve QoS to users, and resource utilisation for network operators and service providers? In view of next generation networks, will services, networks and devices remain integrated as in today's cellular networks? If not, what interfaces will be needed and how will money flow across them?

When the network biases available resources toward QoS traffic, non-QoS traffic receives resource levels (presuming constant available underlying resource) that lead to an average lower service profile. What considerations have been given to non-QoS customers to avoid ending up with such a predicament? What mechanism will prove most efficient for the deployment of new architectures, services, and protocols? This will also bring forth the programmable router paradigm that will focus on identifying a minimal set of primitives from which one can compose an indefinite spectrum of high-level features (e.g., Parlay APIs for service creation). Is this a move forward for routers, or an overwrought description that outweighs the cost of security and complexity that comes with it?
The research community would be interested to finally see how initiatives in Active Networking technology would translate into a viable technology in a commercial sense. In particular, how do we bridge the Active Networking research agenda with ‘real’ requirements by carriers/operators and ISPs for provisioning and managing IP networks? This implicitly implies that Active Networking technology is a means to an end and not the ‘end game’ itself, i.e., the added value of its innovation. What are the constraints that are imposed by Active Networks that might limit their usefulness?

There is also a need to discuss the impact of the new service model, brought on by Active Networking technology, upon existing operators and providers. How will existing operators open up their networks for service providers to enable this task? Since many proponents of this technology have begun to agree that it fits in well as an network-edge technology, can it help an IP service provider commit to quality on the ‘last mile’ since the local loop is not part of the IP backbone?

In our opinion, it remains to be seen exactly how active routers might be deployed in a commercial environment and how this might affect the current ISP market. The ISP service profile can be broadly characterised as access service and carriage. Due to performance constraints in view of the present status of Active Network research, it is deemed more feasible for this technology as the enabling vehicle for service provisioning (hence resource reservation is required before hand) and differentiation at the network edges (i.e., to help ISPs enhance their access technologies).

To meet security requirements, a policy-based control of who can access specific systems, what they can do within them, and when they are allowed access are among issues that should be addressed in the future. On a grander scale, questions to be looked at include: How does the operator allocate its resources? How does the operator
ensure that the service providers are not overstepping the boundaries of the allocated resources within the node? What methodology can be used to arbitrate disputes?

To reiterate the problem in hand, there are often 'simplistic', high-level claims that company objectives are translated into network-wide policies within the administrative domain on the necessary bandwidth, communication resources, and topology requirements, which are, in turn, translated again to node policy rules that are specific to the required behaviour of the node to manage its local resources. What do these (before-after) refinements look like? Can a formalised description of policies bring us a step closer towards identifying the granularity of detail at each level of management abstraction?

Which applications will come up in the near and further future, which impacts will they have on the networks and what is their economic viability? Which kind of intelligence is needed in the networks for these applications, what are the pros and cons of centralised and distributed intelligence for them? What are the pros and cons of distributed and centralised networks in half-a-, 1, 2, 5 year’s time? What is the viability of frameworks like differentiated services and protocols such as COPS, LDAP, and IPv6 for the future development of networks and management of applications? And again, how does Active Networking technology impact them?

In multi-domain, end-to-end resource reservations, can a path be found through the network that can provide the CPU time required by an application, while also meeting the conventional QoS parameters like bandwidth, jitter and delay? Can we associate CPU time reservation with a measure of cost? Can an Active Network provider express all these requirements to query its neighbours for multiple paths, each with an associated cost?
7.2.1 ADSL contention ratio

We digress slightly on a separate but relevant context. The idea of 'contention ratios' is not old; e.g., most ISPs do not have enough modems for every member. So, if fate was particularly cruel and every member of Freeserve, say, decided to dial in to their ISP, then a lot of them would not be able to get connected. In a perfect world, they would have a 1:1 contention ratio but this would be prohibitively expensive, so a lower ratio is chosen, perhaps 50:1 - and so it is with ADSL. Bandwidth is finite and BT is depending on the fact that it is very unlikely that all its subscribers will connect simultaneously and this allows it to install less bandwidth than all its subscribers. If the number of users gets a bit high, it can fall back on intelligent caching to improve perceived data throughput. With relatively few ADSL subscribers, users are unlikely to experience a loss of bandwidth but as the number of subscribers rises, we will see a drop in bandwidth.

As with any other Internet connection, the overall transmission speed is limited to that of the weakest link in the network 'chain' that connects an end-user to whatever website or other resource. In taking this approach, an ISP cannot then guarantee its end-users any specific speed of connection, but instead must rely on the diversity factor, i.e., the fact that not all end-users will be using all of their connection bandwidth all of the time. It is recognised that this is a far more effective way of utilising scarce and expensive bandwidth, and this is what we refer to as 'contention'. The ADSL line from the end-user's property terminates at an ADSL card at the local exchange, and the DSLAM unit combines all the ADSL data onto a single pipe, which carries all Internet traffic to and from the exchange. Contention therefore occurs within the DSLAM unit, with all the ADSL users in an exchange competing for the pipe; hence the possibility for using our metanorms approach for contention management.
7.2.2 Final word

While we have presented various insights to address some of the questions posted earlier in this section, we hope that practitioners and developers will help us to bridge the discrepancy between the pragmatism of the industry and the optimism of scientific researchers. Using our approach, we hope to convince ISPs that we will help establish a more autonomous framework that will increase their efficiency in managing end-user traffic. Secondly, with a fair pricing mechanism in place as well as a multi-domain service provisioning protocol (both of which take account of unique computational resources for Active Networking), ISPs will thus be motivated to introduce node programmability for its end-users as this will increase the product range that it can offer them (mainly for content providers). Router vendors should also be interested to enhance their product MIB to enable new actions to be performed in network management. In our context, this implies the discovery of metanorms parameters.

In terms of the immediate research circle, previous researchers who have worked on the relevant fields would be interested to see how their initial (seminal in some cases) works have been enhanced. For instance, Axelrod's metanorms game would have been fully implemented in a networking context, while Semret would see that his initial notion of offering bandwidth derivatives has been expanded and unified to include a pricing mechanism that takes account of best-effort and guaranteed service on top of a derivative product. Finally, M.Sc and M.Eng project students who want to explore and synthesise insights from different disciplines to address problems in data networks will find a project in this area an interesting starting point.

In essence, it is felt that the Internet landscape has indeed grown more complicated, coming a long way from the days when it was simply a collaborative research project where stakeholders held on to common goals. Today, a stakeholder
holds an influential place within the Internet simply by owning proprietary switching
technology, \textit{e.g.}, Cisco. In the longer term, it is foreseeable that the ‘hourglass model’ of
the protocol stack may deviate from the original design. This model depicts IP as the
lone protocol that sits at the ‘waist’, hence acting as the common service bearer with
different application-centred software residing above and a variety of transport media
located below it. Deviations from this model are caused by the introduction of numerous
other concepts and technologies (at the ‘waist’) that perform functions other than IP
forwarding. Active Networking technology contributes to this so-called deviation by
diluting the significance of IP as the single necessary feature of all communications
sessions. Instead of concentrating diversity and function at the end systems, Active
Networking technology seeks to spread intelligence throughout the network. Is this a
good or bad development? Should we acknowledge this evolution as a normal entropy
that besets all large, engineered systems over time, or strive to redirect research
initiatives back towards its original ‘hourglass’ design? While we have described,
discussed and suggested plausible benefits that may be reaped from Active Networking
technology in Chapter 3, we have also introduced an approach where we push the
responsibility of attaining discipline (within congested systems) into the hands of the
Internet stakeholders in Chapter 4. Alternatively, one could envisage enforcing order by
adopting a fair approach for pricing resources, as suggested in Chapter 5.

In concluding this thesis, we restate the main ideas for service management.
Firstly, we suggested using Odlyzko’s Paris Metro Pricing idea to establish quality
differentiation between channels. Secondly, we used Axelrod’s metanorms approach to
establish order within a channel. Thirdly, financial derivatives, \textit{i.e.}, options, were
incorporated to provide guarantee of continuous access within a channel before finally
adopting the arbitrage theorem to derive fair prices for computation and communication
resources per channel.
A.1 Nash equilibrium

Consider a normal-form game \( G = \{S_1, \cdots, S_n; u_1, \cdots, u_n\} \), with \( n \) players.

**Definition:** If \((s^*_1, \cdots, s^*_n)\) are Nash equilibrium strategies, then the \( i^{th} \) player’s best response, when the strategies specified for the \((n-1)\) other players is \( (s^*_1, \cdots, s^*_{i-1}, s^*_{i+1}, \cdots, s^*_n) \), is \( s^*_i \), i.e., \( u_i(s^*_1, \cdots, s^*_{i-1}, s^*_{i+1}, \cdots, s^*_n) \leq u_i(s^*_1, \cdots, s^*_{i-1}, s_i, s^*_{i+1}, \cdots, s^*_n) \). Suppose game theory offers strategies \((s^*_1, \cdots, s^*_n)\) as a solution to \( G \). If \((s^*_1, \cdots, s^*_n)\) is not the Nash equilibrium of \( G \); then, there exists some player \( i \), such that \( s^*_i \in S_i \) is not the best response to \( s^*_i \in S_i \), where \( s^*_i = (s^*_1, \cdots, s^*_{i-1}, s^*_{i+1}, \cdots, s^*_n) \). Consequently, \( \exists i, s^*_i \in S_i, u_i(s^*_1, s^*_i') \leq u_i(s^*_1, s^*_i) \).
A.2 Access control policies using first-order logic

Opening the network to various parties with different permissions on what they can do on the underlying resources means that access control and trust management issues must be carefully dealt with. The objective here is to generalise about a world that can be described in terms of non-empty sets of primitive propositions, represented in the Internet as an ISP’s admission policy. We will restrict ourselves to using first order propositional logic, where the formulas contain only primitive propositions \( \text{Prop} = \{p, q, \ldots\} \), together with the usual propositional connectives \( \neg \) and \( \land \) to denote negation and conjunction. However, there will be instances when we use the universal and existential quantifiers (i.e., \( \forall \) and \( \exists \) respectively).

There ought to be at least two different levels of granularity for these policies, i.e., the need for a semantic translation from policies to firewall rules, BGP Loc-RIB, and/or IPSec filter rules is important for effectuating the intended actions. The same concept applies to allocation policies. At node level, the resource ‘currency’ would include processor cycles, buffer spaces and link capacity. At network level, resource allocation decisions should consider parameters that would contribute to the quality of end-to-end connectivity. Consider a set of customers vying to reserve resources on an active node, owned by an ISP, in order to further provision service to their end-users. We iteratively derive policy statements to explore plausible enforcement options for admission policies.

Definition 1: The set \( S = \{s_1, s_2, s_3, \ldots, s_n\} \) represents the general set of ISPs who intend to access systems across administrative domains \( D = \{d_1, d_2, d_3, \ldots, d_n\} \) that lie across the Internet. There also exists a set of privilege users who owns the nodes. We identify them as administrators \( \Phi = \{\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_n\} \), and it comes naturally that administrators are domain owners and as such, perform administrative functions \( \alpha(d) \) on their domains. It also follows that there is only one such a role per administrative domain.

Definition 2: Within each administrative domain, we note a subset of the general group of users, who have satisfied a certain access control policy to enable the use of its node resources; and are, thus, known as authorised entities denoted by \( A = \{a_1, a_2, a_3, \ldots, a_n\} \), where \( A \subseteq S \), and \( S \setminus \Phi \) effectively gives \( \{x \mid x \text{ represents all the contending service providers}\} \).

In reaching an admission decision, the network management system currently considers the capacity of the network links and the node queues. The resources suffice because each packet sent through each router and link on the network receives identical processing. Once Active Networks are deployed, the situation changes because each packet might receive non-homogeneous processing. Network management systems need to take into account the CPU time requirements for new classes of packets injected into the packet to serve data flowing along a path. Admission control decision will then have to account for CPU capacity (in addition to the link and queue capacity) available in the network along the path that packets are expected to flow.

No policy: any entity may access system without discrimination. The node will advertise its routes to peers to such that traffic (transit included) may be directed via this node. Firewalls that protect the active node will allow packets to deploy new protocols on the node or standardised node APIs may be utilised. Formally,

\[
\forall x(x \in S \rightarrow x \in A)
\]  

(32)
This first (non-) policy is typical in the real world since best-effort service means that offers equal usage of its link and node resources to both paying (QoS-seeking) and non-paying (best-effort) customers.

**Tickets:** any entity may access system if it has a requisite credential. The owner of the system (root) awards the credential based on a quality the entity possesses (e.g., paid subscription, barter trades on network resources). The owner’s IPSec filter rules are appropriately configured to reflect this access control.

**Definition 3:** $C(p, q)$ represents a credential given by $p$ to $q$. Formally,

$$
\forall x (x \in \Phi \lor C(\Phi, x) \rightarrow x \in A)
$$

**Sponsor/guarantor:** any entity may access system if at least another authorised entity $a$ within the system can guarantee the former’s credibility. Obviously, the entity within the system would know the potential new entrant, and can vouch for the safety of the latter’s operations within the system.

**Definition 4:** $F(p, q)$ represents $q$ as $p$’s alliance, e.g., two organisations with prior bi-lateral agreements.

$$
\forall x (\exists y (F(x, y) \land y \in A \land y \neq x) \rightarrow x \in A)
$$

This policy probably offers too high a degree of flexibility to an authorised entity. The administrator has basically delegated its node access control to a trusted third party. We may further refine this policy by limiting the number of entities an authorised entity can further allow to access the system.

**Majority:** any entity may access system if a certain percentage $r$ of the existing entities within the system at that particular instance allows it. These entities may or may not know the potential new entrant but make decisions (to allow or deny) based on their satisfaction of current resource usage (i.e., refer to allocation policies below to reason about the semantics of negotiation protocols).

**Definition 5:** The satisfaction function $\sigma(p)$ returns a Boolean result based upon the satisfaction level of entity $p$.

$$
\forall x (\exists y \left(\sum_{n=1}^{y > r \cdot n[A]} \land \sigma(y) \land y \in A\right) \rightarrow x \in A)\text{, where } 0 \leq r \leq 1
$$

In view of (34) and (35), it may be fairly argued here that an entity that gains initial access to the system obtains de facto control, and thereby the benefit it can secure for itself. As such, apart from having a formal notation to identify the order of access, we also introduce additional restriction on these voting rights:

**Limited voting rights:** any entity that has already gained access to the system may vote to allow or deny a new entrant if it has not already done so for $w$ number of times. Note that this is not an access control policy but rather a constraint enforced upon an entity’s voting rights.

**Definition 6:** $V(p, q)$ denotes the number of times $p$ has voted on the decision on $q$’s access permission. If $q$ is left empty, i.e., $V(p)$, the default notation denotes the number of times $p$ has voted in total; and $A$ denotes the right to vote in the next decision-making process. Formally,

$$
\forall x (x \in A \land V(p) < w \rightarrow x \in A), \text{ where } w \in \mathbb{Z}^+
$$

The motivation here is to restrain reckless denial of access and, at the same time, ensure that current entities within the system can maintain the right to negotiate...
and carefully decide if their interests and satisfaction would be compromised in view of the impending new entrant to the system.
A.3 Allocation strategies for resource contention

In resolving resource contention, there ought to be at least two different levels of granularity for these policies, i.e., the need for a semantic translation from policies to firewall rules, BGP Loc-RIB, and/or IPSec filter rules is important for effectuating the intended actions. We present in this section a set of strategies that can be enforced by ISP administrators and we have formalised the exposition of these strategic policies using first order logic.

An ISP must recognise what is akin to the ‘theory of unlimited territorial integrity’ that forbids a country to alter the natural conditions of its own territory to the disadvantage of a neighbouring country [89]. On the other hand, according to the Harmon doctrine, which advocates the ‘theory of absolute territorial sovereignty’, where a country has absolute sovereignty over the area of any river basin in its territory. Evidently, one can foresee these doctrines in conflict, just as how the upstream ISPs has to ensure that its resource allocation policies are foolproof and will not compromise a sub-network’s interest in riposte to another sub-network’s benefit. The syntax and semantics of the policies described below extend from our definitions in §3.6.

**Definition 7:** Physical resources \( \psi \) can be divided between bandwidth \( B=\{b \mid b \) represents units of bandwidth\}, processor cycles \( P=\{p \mid p \) represents units of CPU cycles\}, and memory \( M=\{m \mid m \) represents units of memory space\}. Formally, \( B \subseteq \psi \), \( P \subseteq \psi \), \( M \subseteq \psi \), and \( B \cap P \cap M = \emptyset \).

The constraint for resources is represented by subscript notation, viz., \( b_{\text{max}} \in B \) indicates the total bandwidth that can be provided per node, \( p_{\text{max}} \in P \) indicates the total processor cycles that can be achieved by the node, and \( m_{\text{max}} \in M \) indicates the total disk space per node. The temporal limit for the duration of resource usage is represented by the function \( \zeta_{\text{max}} \).

**Definition 8:** Resources are further categorised into total resources that are still available \( \psi_{\text{avail}} \) and the total resource that have been allocated \( \psi_{\text{alloc}} \), where \( \psi_{\text{avail}} \land \psi_{\text{alloc}} = \emptyset \), and \( \psi_{\text{max}} = \{\psi_{\text{alloc}}, \psi_{\text{avail}}\} \) such that \( (\psi_{\text{alloc}} \rightarrow \emptyset) \rightarrow (\psi_{\text{avail}} \rightarrow \psi_{\text{max}}) \).

**Basic limits:** For the lower boundary, no entity can buy less than \( y \) amount of memory, \( z \) amount of CPU cycles and \( z \) amount of bandwidth. For temporal limit on each type of resource reservation, all entities must reserve a resource within a range of time limits. For the upper boundary, no entity can be allocated more than the total amount of available memory, CPU cycles and bandwidth.

**Definition 9:** The parameters for the allocate function are defined as \( \gamma(p, q, u, t) \) which describes a network operator \( q \in \Phi \) allocating an amount of resource \( u \in \{\psi \mid \psi \) represents the available resource\} to entity \( p \in A \) for a duration of \( t \in \{\zeta \mid \zeta \) represents the available duration for resource reservation\}. Formally,

\[
\forall x \in A, \forall y \in \Phi(\gamma(x, y, \psi, \zeta) \rightarrow ((\psi_{\text{min}} \leq \psi \leq \psi_{\text{avail}}) \land (\zeta_{\text{min}} \leq \zeta \leq \zeta_{\text{avail}})))
\]

**Minimum allocation:** Entities will always be allocated the least possible quantity of available resources.

---

60 First authoritatively stated by Judson Harmon, an American Attorney-General who made the declaration concerning the Rio Grande.

61 Representation of constraints without the 'max' subscript indicates instantaneous values. Intuitively, adding the 'min' subscript indicates minimum values for resource and period of subscription.

62 The minimum limit for resource requestors is imposed from a business aspect, whereby the revenue from resources provisioned should, at least, justify the cost (e.g., maintenance and management costs) of provisioning in order to satisfy the break-event point.
where $\psi'$ = alternative amount of resource to be allocated

**History:** No entity may request more than a certain percentage of its prior highest reservation parameter.

$$\forall x \in A, \forall y \in \Phi(\gamma(x, y, \psi, \zeta) \rightarrow (\neg \exists \psi'(\psi' \leq \psi))), \quad (38)$$

where $h(\psi) = \text{prior amount requested for resource } \psi$, and $0 < r < 1$

**Pareto-optimal**\(^{63}\): No entity can reserve so much resource such that the next request coming from any other entity is denied of taking equally as much. In other words, the remaining resource available should be at least equal or more than what is to be reserved. Intuitively,

$$\forall x \in A, \forall y \in \Phi(\gamma(x, y, \psi, \zeta) \rightarrow (\psi_{\text{avil}} \geq 2\psi)) \quad (40)$$

The consequence of this policy is that any entity may only reserve (at best) a third of allowable bandwidth. For example, consider a DS-3\(^{64}\) (or T3) link that is capable of carrying data at more than 40Mbps (equivalent to approximately 672 DS-0 channels at 64kbps). Should the transit ISP provision 20 percent, say, of the T3 link for premium IP traffic, the allowable bandwidth (i.e., the bandwidth that the resource allocation policy must account for) is 8Mbps, leaving 32Mbps for best-effort traffic. Thus, the first entity to reserve bandwidth may (at best) be able to ask for 40 DS-0 channels. Subsequent requests for bandwidth reservation would have decreasing upper limits. Unused portions of allowable bandwidth should be used for best-effort traffic until it is being reserved for.

**Ideal aspiration:** an entity is allowed to reserve the maximal available resource if no other entities access the system during that period of time. Thus, in the absence of other entities, the lone entity would experience an 'ideal aspiration' level.

$$\forall x \in A, \forall y \in \Phi((\gamma(x, y, \psi, \zeta) \wedge x = 0) \rightarrow (\psi = \psi_{\text{max}})) \quad (41)$$

However, consistent to other resource sharing predicaments, it is impossible to guarantee every entity its 'ideal aspiration' level as more entities gain access to a system.

---

\(^{63}\) Named after the work of Italian economist and sociologist Vilfredo Pareto (1848-1923), the Principal of Pareto Optimality is an evaluative principle that says, "The community becomes better off if one individual becomes better off and none worse off".

\(^{64}\) North American digital carrier hierarchy (T-carrier)
A.4 UCL-IPTABLESv0-MIB.txt

-- Local variables:
-- mode: indented-text
-- End:

UCL-IPTABLESv0-MIB DEFINITIONS ::= BEGIN

IMPORTS
 OBJECT-TYPE, MODULE-IDENTITY, IpAddress, Integer32, Counter32
 FROM SNMPv2-SMI
 DisplayString, RowStatus, TruthValue, AutonomousType
 FROM SNMPv2-TC
 ucltsgExperimental
 FROM TSG-UCL-MIB;

-- This MIB module uses the extended OBJECT-TYPE macro as
-- defined in [9].

iptablesO MODULE-IDENTITY
   LAST-UPDATED "0212271303Z"  -- Fri, 27 June 2003 13:03:21 +0000
   ORGANIZATION "Network Services Group"
   CONTACT-INFO
      "Alvin Tan
       Postal: Department Electrical Engineering
       University College London
       Torrington Place
       London WC1E 7JE
       Tel: +44 709 2376326
       E-Mail: weaves@ee.ucl.ac.uk"

   DESCRIPTION
      "A MIB module to monitor the rules within NetFilter/IPTables."

::= { ucltsgExperimental 2 }

operation OBJECT IDENTIFIER ::= { iptablesO 1 }
chains OBJECT IDENTIFIER ::= { iptablesO 2 }
rules OBJECT IDENTIFIER ::= { iptablesO 3 }
traps OBJECT IDENTIFIER ::= { iptablesO 255 }

reload OBJECT IDENTIFIER ::= { traps 1 }
packet OBJECT IDENTIFIER ::= { traps 2 }

-- Operational status
-- Lists packets received and the status of the device

opHandle OBJECT-TYPE
   SYNTAX INTEGER {
      filter(1), -- filter table
      nat(2), -- nat table
      mangle(3) -- mangle table
   }
   MAX-ACCESS read-write
   STATUS current
   DESCRIPTION
      "Get a new handle for interrogation. Specify which table to
get the handle on."
::= { operation 1 }

opOther OBJECT-TYPE
   SYNTAX Counter32
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
      "Some other operation."
::= { operation 2 }

-- Chains table

chainsCount OBJECT-TYPE
   SYNTAX INTEGER
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
      "Number of chains."
::= { chains 1 }

chainsClear OBJECT-TYPE
clear(1) -- clear all chains

chainsTable OBJECT-TYPE
SYNTAX SEQUENCE OF ChainEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "The chains available."
::= { chainsTable 1 }

chainEntry OBJECT-TYPE
SYNTAX ChainEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "Chains chains description."
INDEX { chainIndex }
::= { chainsTable 1 }

ChainEntry ::= SEQUENCE {
  chainIndex      INTEGER,
  chainName       DisplayString,
  chainIsBuiltin  TruthValue,
  chainPolicy     DisplayString,
  chainPktCount   Counter64,
  chainByteCount  Counter64,
  chainRefs       INTEGER,
  chainRules      INTEGER,
  chainZero       INTEGER,
  chainStatus     RowStatus
}

chainIndex OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The index of this entry."
::= { chainEntry 1 }

chainName OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Name of the chain."
::= { chainEntry 2 }

chainIsBuiltin OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Whether the chain is a built-in one."
::= { chainEntry 3 }

chainPolicy OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS current
DESCRIPTION "If built-in, the policy of the chain."
::= { chainEntry 4 }

chainPktCount OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "If built-in, the packet count of the chain."
::= { chainEntry 5 }

chainByteCount OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "If built-in, the byte count of the chain."
::= { chainEntry 6 }

chainRefs OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "If not built-in, the number of references to the chain."
::= { chainEntry 7 }

chainRules OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of rules in the chain."
::= { chainEntry 8 }

chainZero OBJECT-TYPE
SYNTAX INTEGER {
reset(1) -- set to zero
}
MAX-ACCESS read-write
STATUS current
DESCRIPTION "Zero the counters of the chain."
::= { chainEntry 9 }

chainStatus OBJECT-TYPE
SYNTAX RowStatus
MAX-ACCESS read-create
STATUS current
DESCRIPTION "In the unlikely event that dynamically loadable chains
are implemented this would be how to add them."
::= { chainEntry 10 }

-- Rules table

rulesCount OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Count of all the rules of all tables."
::= { rules 1 }

rulesClear OBJECT-TYPE
SYNTAX INTEGER {
  clear(1) -- clear all rules
}
MAX-ACCESS read-write
STATUS current
DESCRIPTION "Clear all the rules."
::= { rules 2 }

rulesTable OBJECT-TYPE
SYNTAX SEQUENCE OF RuleEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "The rules available."
::= { rules 3 }
ruleEntry OBJECT-TYPE
SYNTAX  RuleEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "Rules rules description."
INDEX { ruleIndex, ruleChain }
::= { rulesTable 1 }

RuleEntry ::= SEQUENCE {
   ruleIndex  INTEGER,
   ruleChain  INTEGER,
   ruleSrc    IpAddress,
   ruleDst    IpAddress,
   ruleSrcMask IpAddress,
   ruleDstMask IpAddress,
   ruleInIface DisplayString,
   ruleOutIface DisplayString,
   ruleInIfaceMask DisplayString,
   ruleOutIfaceMask DisplayString,
   ruleProto INTEGER,
   ruleFlags INTEGER,
   ruleInFlags INTEGER,
   rulePktCount Counter,
   ruleByteCount Counter32,
   ruleTarget DisplayString,
   ruleZero INTEGER,
   ruleStatus RowStatus
}

ruleIndex OBJECT-TYPE
SYNTAX  INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The index of this entry."
::= ( ruleEntry 1 )

ruleChain OBJECT-TYPE
SYNTAX  INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The index of this entry."
::= ( ruleEntry 2 )

ruleSrc OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Source address."
::= ( ruleEntry 3 )

ruleDst OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Destination address."
::= { ruleEntry 4 }

ruleSrcMask OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Netmask for the source address."
::= { ruleEntry 5 }

ruleDstMask OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Netmask for the destination address."
::= { ruleEntry 6 }

ruleInIface OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Name of the in interface."
::= { ruleEntry 7 }

ruleOutIface OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Name of the out interface."
::= { ruleEntry 8 }

ruleInIfaceMask OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Mask for the in interface."
::= { ruleEntry 9 }

ruleOutIfaceMask OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Mask for the out interface."
::= { ruleEntry 10 }

ruleProto OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Prototype. Zero is any."
::= { ruleEntry 11 }

ruleFlags OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Flags of the rule."
::= { ruleEntry 12 }

ruleInvFlags OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Inversion flags of the rule."
::= { ruleEntry 13 }

rulePktCount OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Packet count."
::= { ruleEntry 14 }

ruleByteCount OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "Packet count."
::= { ruleEntry 15 }

ruleTarget OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The target of the rule."
::= { ruleEntry 16 }

ruleZero OBJECT-TYPE
SYNTAX INTEGER {
  zero(1) -- zero the counter
}
MAX-ACCESS read-write
STATUS current
DESCRIPTION "Packet count."
::= { ruleEntry 17 }

ruleStatus OBJECT-TYPE
SYNTAX RowStatus
MAX-ACCESS read-create
STATUS current
DESCRIPTION "In the unlikely event that dynamically loadable rules are implemented this would be how to add them."
::= { ruleEntry 18 }

END
### A.5 Basic iptables commands

The filter table listens on three hooks, thus providing three chains for packet filtering. All packets coming from the network and destined for the local box traverse the INPUT chain. All packets which are forwarded (routed) by us traverse the FORWARD chain (and only the FORWARD chain). Finally, the packets originating from the local box traverse the OUTPUT chain.

The basic syntax for an iptables command is:

```
iptables -t table -O operation chain -j target matches
```

The basic operations include:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-A</code></td>
<td>append rule</td>
</tr>
<tr>
<td><code>-I</code></td>
<td>insert rule</td>
</tr>
<tr>
<td><code>-D</code></td>
<td>delete rule</td>
</tr>
<tr>
<td><code>-R</code></td>
<td>replace rule</td>
</tr>
<tr>
<td><code>-L</code></td>
<td>list rules</td>
</tr>
</tbody>
</table>

The target that is common to all chains include:

<table>
<thead>
<tr>
<th>Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ACCEPT</code></td>
<td>accept the packet</td>
</tr>
<tr>
<td><code>DROP</code></td>
<td>drop the packet</td>
</tr>
<tr>
<td><code>QUEUE</code></td>
<td>queue packet to userspace</td>
</tr>
<tr>
<td><code>RETURN</code></td>
<td>return to the previous (calling) chain</td>
</tr>
</tbody>
</table>

Basic matches, common to all chains:

<table>
<thead>
<tr>
<th>Match</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-p</code></td>
<td>protocol (tcp/icmp/udp/...)</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>source address (ip address/masklen)</td>
</tr>
<tr>
<td><code>-d</code></td>
<td>destination address (ip address/masklen)</td>
</tr>
<tr>
<td><code>-i</code></td>
<td>incoming interface</td>
</tr>
<tr>
<td><code>-o</code></td>
<td>outgoing interface</td>
</tr>
</tbody>
</table>
A.6 Java source code for simulation

/**
 * RandomGeneratorBin.java
 * This program randomly generates numbers from 0 to 7 to represent binary numbers of the order 2^3.
 */

public class RandomGeneratorBin {
    public RandomGeneratorBin() {
        // code...
    }

    public int getBinary() {
        double val = 0;
        double rand = 0;
        int[] bit = new int[3];
        int i = 2;
        while (i != -1) {
            rand = Math.random();
            if (rand >= 0.5)
                bit[i] = 1; // 50/50 chance
            else bit[i] = 0;
            val += (new Integer(bit[i])).doubleValue() * Math.pow(2, (new Integer(i)).doubleValue());
            i--;
        }
        return (new Double(val)).intValue();
    }

    public int mutate(int v) {
        double val = 0;
        double rand = 0;
        int remain = 0;
        int quote = 0;
        int[] bit = new int[3];

        // first convert to binary
        while (v / 2 != 0) {
            for (int i = 0; i < 3; i++) {
                remain = v % 2;
                if (remain == 1)
                    bit[i] = 1;
                else bit[i] = 0;
                v /= 2;
            }
        }
        if (v == 1) bit[0] = 1;

        // each bit has 1% chance of 'flipping'
        int j = 2;
        while (j != -1) {
            rand = Math.random();
            if (rand >= 0.9) {
                if (bit[j] == 1)
                    bit[j] = 0;
                else bit[j] = 1;
                val += (new Integer(bit[j])).doubleValue() * Math.pow(2, (new Integer(j)).doubleValue());
            } else {
                val += (new Integer(bit[j])).doubleValue() * Math.pow(2, (new Integer(j)).doubleValue());
            }
            j--;
        }
        return (new Double(val)).intValue();
    }

    public static void main(String[] args) {
        // test
        RandomGeneratorBin rg = new RandomGeneratorBin();
        int x = rg.getBinary();
        System.out.println("Randomly generated number between 0-7, inclusive = " + x);
        System.out.println("Randomly mutated number between 0-7, inclusive = " + rg.mutate(x));
    }
}
import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ExperimentPlot extends JFrame {
    static double total = 0;
    static double average = 0;
    static double diffSquared = 0;
    static double sigma = 0;
    static int totalBoldness = 0;
    static int totalVengefulness = 0;
    static int noOfISPs = 20;
    static int noOfGenerations = 500;

    public ExperimentPlot() {
        super(" Norms Dynamics ");
        setSize(600, 600);
    }

    public void paint(Graphics g) {
        g.setColor(Color.black);
        for (int n = 0; n < 9; n++) {
            g.drawLine(150, 500 - n*50, 550, 500 - n*50); //horizontal grids
            g.drawLine(150 + n*50, 500, 150 + n*50, 100); //vertical grids
        }
        for (int n = 0; n < 8; n++) {
            g.drawString(n + "/I", 145 + n*50, 520); //horizontal labels
            g.drawString(n + "/", 115, 505 - n*50); //vertical labels
        }
        g.setFont(new Font("TimesRoman", Font.BOLD, 12));
        g.drawString("Basic Norms Game", 300, 50);
        g.setFont(new Font("TimesRoman", Font.PLAIN, 12));
        g.drawString("boldness", 350, 550);
        g.drawString("vengefulness", 20, 250);
        g.fillOval(4*50 + 145, 495 - 4*50, 10, 10);
        //creation phase
        Vector agents = new Vector();
        agents = create(noOfISPs);
        calcTotal(agents);
        //plot
        g.setColor(new Color(255, 255, 255));
        g.fillOval((new Double(divide(totalBoldness, agents.size())*5)).intValue() * 10 + 145, 495 - (new Double(divide(totalVengefulness, agents.size())*5)).intValue() * 10);
        g.drawString((new Integer(0)).toString(), (new Double(divide(totalBoldness, agents.size())*5)).intValue() * 10 + 152, 500 - (new Double(divide(totalVengefulness, agents.size())*5)).intValue() * 10);
        //first run
        Vector survivors = new Vector();
        survivors = run(agents);
        System.out.println("End of round 1");
        calcTotal(survivors);
        //plot
        g.setColor(new Color(240, 240, 250));
        g.fillOval((new Double(divide(totalBoldness, survivors.size())*5)).intValue() * 10 + 145, 495 - (new Double(divide(totalVengefulness, survivors.size())*5)).intValue() * 10, 10, 10);
        g.drawString((new Integer(1)).toString(), (new Double(divide(totalBoldness, survivors.size())*5)).intValue() * 10 + 152, 500 - (new Double(divide(totalVengefulness, survivors.size())*5)).intValue() * 10);
        //subsequent runs
        for (int i=1; i < noOfGenerations; i++) {
            survivors = run(survivors);
            System.out.println("End of round " + (i+1));
            for (int j=0; j < survivors.size(); j++) {
                calcTotal(survivors);
                g.setColor(new Color(240 - (240*i/noOfGenerations), 240 - (240*i/noOfGenerations), 240 - (240*i/(3*noOfGenerations))));
            }
        }
    }
}

//******************************************************
/*ExperimentPlot.java*/
/*This program simulates the basic norms game and*
/*plots the final scores*/
******************************************************
if (i == (noOfGenerations-1)) g.setColor(Color.red);
g.fillOval((new Double(divide(totalBoldness, survivors.size())*5)).intValue()*10 + 145, 495 - (new Double(divide(totalVengefulness, survivors.size())*5)).intValue()*10, 10, 10);
g.drawString((new Integer(i+1)).toString(), (new Double(divide(totalBoldness, survivors.size()))*5)).intValue()*10+152, 500 - (new Double(divide(totalVengefulness, survivors.size()))*5)).intValue()*10);
}

/***************************************************************************/
// To create agents that have two traits each (represent vengefulness and boldness), randomly scaled upwards from 0-10.
/***************************************************************************/
private static Vector create( int n ) {
  RandomGeneratorBin rg = new RandomGeneratorBin();
  Vector p = new Vector();
  for ( int i=0; i<n; i++ ) {
    int x = 0;
    int y = 0;
    while (x==0) { // enables skewing of boldness
      x = rg.getBinary();
    }
    while (y<4) { // start off with sufficient level of vengefulness
      y = rg.getBinary();
    }
    p.add(i, new Profile(x, y));
  }
  return p;
}

private static void calcTotal( Vector p ) {
  totalBoldness = 0; //reset
  totalVengefulness = 0; //reset
  for ( int i=0; i < p.size(); i++ ) {
    totalBoldness = totalBoldness + ((Profile)p.elementAt(i)).boldness;
    totalVengefulness = totalVengefulness + ((Profile)p.elementAt(i)).vengefulness;
  }
}

private static double divide( int m, int n ) {
  return (new Integer(m)).doubleValue()/(new Integer(n)).doubleValue();
}

private static Vector run( Vector p ) {
  for ( int i=0; i<p.size(); i++ ) { // first check who's in the game!
    ((Profile)p.elementAt(i)).addCredits(1000);
    RandomGeneratorBin rg = new RandomGeneratorBin();
    int seen = 0;
    while (seen==0) {
      seen = rg.getBinary(); // probability of being spotted
    }
    System.out.println("Probability of being seen is: " + seen + "/7");
  }
  System.out.println("Stats for this generation:";
  //total and average
  for ( int i=0; i < p.size(); i++ ) {
    System.out.println("Agent " + (i+1) + " : " +

total = total + (new Integer (((Profile)p.elementAt(i)).getCredits())).doubleValue();

System.out.println("Total credits = " + total);
average = total/p.size();
System.out.println("Average = " + average);

// std deviation
for (int i=0; i < p.size(); i++) {
    diffSquared = diffSquared + Math.pow((new Integer((((Profile)p.elementAt(i)).getCredits())).doubleValue() - average, 2));
}
sigma = Math.sqrt(diffSquared/(p.size()-1));
System.out.println("Standard deviation = " + sigma);
System.out.println("End of game. ");

// 'Stronger' agents, i.e., those with credits one
// standard deviation above the average, are retained
// and doubled; 'weaker' agents are replaced with new
// traits.

System.out.println("******** Generating new agents ********");
Vector v = new Vector();
RandomGeneratorBin r = new RandomGeneratorBin();
for (int i=0; i < p.size(); i++) {
    if ((((Profile)p.elementAt(i)).getCredits()) > (average + sigma)) {
        v.addElement(((Profile)p.elementAt(i)));
        v.addElement(new Profile(((Profile)p.elementAt(i)).boldness,
                                  ((Profile)p.elementAt(i)).vengefulness));
    } else if (((Profile)p.elementAt(i)).getCredits() <= (average + sigma) &&
                ((Profile)p.elementAt(i)).getCredits() >= (average - sigma)) {
        v.addElement(((Profile)p.elementAt(i)));
    } else if (((Profile)p.elementAt(i)).getCredits() < (average - sigma)) {
        System.out.println("( " + ((Profile)p.elementAt(i)).boldness + " , " + ((Profile)p.elementAt(i)).vengefulness + " ) eliminated");
    }
}
while (v.size() > p.size()) {
    for (int i = p.size(); i < v.size(); i++) v.removeElementAt(i);
}
if (v.size() < p.size()) {
    // if less than initially set, randomly generate some more
    for (int i = v.size(); i < noOfISP; i++) {
        int x = 0;
        int y = 0;
        while (x==0) {
            x = rg.getBinary();
        }
        while (y==0) {
            y = rg.getBinary();
        }
        v.add(i, new Profile(x, y));
        System.out.println("New agent " + (i+1) + ": " + ((Profile)v.elementAt(i)).boldness + " , " + ((Profile)v.elementAt(i)).vengefulness + "");
    }
} else {
    System.out.println("No need to generate new ones, we've got enough");
}
System.out.println("**** New agents + survivors after 1% chance of mutation ****")
for (int i = 0; i < v.size(); i++) {
    ((Profile)v.elementAt(i)).boldness = mutate(((Profile)v.elementAt(i)).boldness);
    ((Profile)v.elementAt(i)).vengefulness = mutate(((Profile)v.elementAt(i)).vengefulness);
    System.out.println("Agent " + (i+1) + ": " + ((Profile)v.elementAt(i)).boldness + " , " + ((Profile)v.elementAt(i)).vengefulness + ", credits = " + ((Profile)v.elementAt(i)).getCredits()));
}
System.out.println("No. of survivors and new agents: " + v.size());
public int mutate( int v ) {
    double val = 0;
    double rand = 0;
    int remain = 0;
    int quote = 0;
    int[] bit = new int[3];

    // first convert to binary
    while ( v/2 != 0 ) {
        for ( int i = 0; i < 3; i++ ) {
            remain = v % 2;
            if (remain == 1)
                bit[i] = 1;
            else bit[i] = 0;
            v /= 2;
        }
    }
    if (v==1) bit[0] = 1;

    // each bit has 1% chance of 'flipping'
    int j = 2;
    while ( j != -1 ) {
        rand = Math.random();
        if ( rand >= 0.9 ) {
            if (bit[j] == 1) bit[j] = 0;
            else bit[j] = 1;
            val += (new Integer(bit[j])).doubleValue() * Math.pow(2, (new Integer(j)).doubleValue());
        } else {
            val += (new Integer(bit[j])).doubleValue() * Math.pow(2, (new Integer(j)).doubleValue());
        }
        j--;
    }
    return (new Double(val)).intValue();
}  // end mutate()

public static void main( String[] args ) {
    ExperimentPlot ep = new ExperimentPlot();
    ep.addWindowListener(
        new WindowAdapter() {
            public void windowClosing( WindowEvent e ) {
                System.exit( 0 );
            }
        });
}
public void paint(Graphics g) {
    g.setColor(new Color(255, 255, 255));
    g.drawString((new Integer(0)).toString(), (new Double(divide(totalBoldness, agents.size()))*5)).intValue()*10 + 152, 500 - (new Double(divide(totalVengefulness, agents.size()))*5)).intValue()*10);
    g.setColor(new Color(240 - (240*i/noOfGenerations), 240 - (240*i/noOfGenerations), 240 - (240*i/(3*noOfGenerations))));
    if (i == (noOfGenerations-1)) g.setColor(Color.red);
    g.fillOval((new Double(divide(totalBoldness, survivors.size())))*5).intValue()*10 + 152, 500 - (new Double(divide(totalVengefulness, survivors.size()))*5)).intValue()*10);  
}  
  //subsequent runs  
  for (int i=1; i < noOfGenerations; i++) {
    survivors = run(survivors);
    System.out.println("End of round "+ (i+1));  
    for (int j=0; j < survivors.size(); j++) {
        calcTotal(survivors);
        g.setColor(new Color(240 - (240*i/noOfGenerations), 240 - (240*i/noOfGenerations), 240 - (240*i/(3*noOfGenerations))));
        if (i == (noOfGenerations-1)) g.setColor(Color.red);
        g.fillOval((new Double(divide(totalBoldness, survivors.size())))*5).intValue()*10 + 152, 500 - (new Double(divide(totalVengefulness, survivors.size()))*5)).intValue()*10);  
    }
}

/***************************************************************************/
/* To create agents that have two traits each {represent

* vengefulness and boldness), randomly scaled upwards
* from 0-10.

private static Vector create( int n ) {
  RandomGeneratorBin rg = new RandomGeneratorBin();
  Vector p = new Vector();
  for ( int i=0; i<n; i++ ) {
    int x = 0;
    int y = 0;
    while (x==0) { // enables skewing of boldness
      x = rg.getBinary();
    }
    while (y<4) { // start off with sufficient level of vengefulness
      y = rg.getBinary();
    }
    p.add(i, new Profile( x, y ));
  }
  return p;
}

private static void calcTotal( Vector p ) {
  totalBoldness = 0; //reset
  totalVengefulness = 0; //reset
  for ( int i=0; i < p.size(); i++ ) {
    totalBoldness = totalBoldness + ((Profile)p.elementAt(i)).boldness;
    totalVengefulness = totalVengefulness +
                        ((Profile)p.elementAt(i)).vengefulness;
  }
}

private static double divide( int m, int n ) {
  return (new Integer(m)).doubleValue()/(new Integer(n)).doubleValue();
}

private static Vector run( Vector p ) {
  for ( int i=0; i<p.size(); i++ ) { // first check who's in the game!
    ((Profile)p.elementAt(i)).addCredits( 1000 );
  }
  RandomGeneratorBin rg = new RandomGeneratorBin();
  int seen = 0;
  while ( seen==0 ) {
    seen = rg.getBinary(); // probability of being spotted
    System.out.println ( "Probability of being seen is: " + seen + "/7" );
    for ( int i=0; i < p.size(); i++ ) { // defect
      if ( ((Profile)p.elementAt(i)).boldness >= seen ) {
        ((Profile)p.elementAt(i)).addCredits( 3 );
        for ( int j=0; j < p.size(); j++ ) { // reaction by others
          if ( j != i ) { // defector obviously won't punish ownself
            ((Profile)p.elementAt(j)).addCredits(-1); // others hurt
          }
        }
      } // defense occurred, saw, but did not punish
      if ( ((Profile)p.elementAt(i)).vengefulness < 6 ) {
        // defection occurred, saw, but did not punish
        for ( int k=0; k < p.size(); k++ ) { // metanorms loop
          if ( ((Profile)p.elementAt(k)).vengefulness >= 6 && k != i && j != j && j != k ) {
            /*k != i && k != j because defector and non-punisher won't
                   * punish ownselves, while j != i to avoid punisher punishing
                   * defector second time */
            ((Profile)p.elementAt(j)).addCredits(-9); // punishment
            ((Profile)p.elementAt(k)).addCredits(-2); // enforcement cost
          }
        }
      }
    }
  }
}

188
//total and average
for (int i=0; i < p.size(); i++) {
    System.out.println( "Agent " + (i+1) + " : " + ((Profile)p.elementAt(i)).getCredits());
    total = total + (new Integer(((Profile)p.elementAt(i)).getCredits())).doubleValue();
}
System.out.println( "Total credits = " + total);
average = total/(p.size());
System.out.println( "Average = " + average);

//std deviation
for (int i=0; i < p.size(); i++) {
    diffSquared = diffSquared + Math.pow((new Integer(((Profile)p.elementAt(i)).getCredits())).doubleValue() - average, 2);
}
sigma = Math.sqrt(diffSquared/(p.size()-1));
System.out.println( "Standard deviation = " + sigma );
System.out.println( "End of game." );

="/***********************************************************************
 * 'Stronger' agents, i.e., those with credits one
 * standard deviation above the average, are retained
 * and doubled; 'weaker' agents are replaced with new
 * traits.
***********************************************************************/
System.out.println( "******** Generating new agents *********" );
Vector v = new Vector();
RandomGeneratorBin r = new RandomGeneratorBin();
for (int i=0; i<p.size(); i++) {
    if (((Profile)p.elementAt(i)).getCredits() > (average + sigma)) {
        v.addElement( ((Profile)p.elementAt(i)));
        v.addElement( new Profile(((Profile)p.elementAt(i)).boldness, ((Profile)p.elementAt(i)).vengefulness) );
        v.addElement( new Profile(((Profile)p.elementAt(i)).boldness, ((Profile)p.elementAt(i)).vengefulness) );
    }
    if (((Profile)p.elementAt(i)).getCredits() <= (average + sigma) && ((Profile)p.elementAt(i)).getCredits() >= (average - sigma) ) {
        v.addElement( ((Profile)p.elementAt(i)));
    }
    if (((Profile)p.elementAt(i)).getCredits() < (average - sigma) ) {
        System.out.println( "( " + ((Profile)p.elementAt(i)).boldness + " ,  " + ((Profile)p.elementAt(i)).vengefulness + " ) eliminated" );
    }
}
while (v.size() > p.size()) {
    for (int i = p.size(); i < v.size(); i++)
        v.removeElementAt( i );
}
if (v.size() < p.size()) {
    // if less than the no. of ISPs initially set, randomly generate some more
    for (int i = v.size(); i < noOfISPs; i++) {
        int x = 0;
        int y = 0;
        while (x==0) {
            x = rg.getBinary();
        }
        while (y==0) {
            y = rg.getBinary();
        }
        v.add(i, new Profile(x, y ));
        System.out.println( "New agent " + (i+1) + " : ( " + ((Profile)v.elementAt(i)).boldness + " , " + ((Profile)v.elementAt(i)).vengefulness + " )" );
    }
} else {
    System.out.println("No need to generate new ones");
}
System.out.println("New agents + survivors after 1% chance of mutation:");
for (int i = 0; i < v.size(); i++) {
    ((Profile)v.elementAt(i)).boldness = 189
mutate(((Profile)v.elementAt(i)).boldness);
((Profile)v.elementAt(i)).vengefulness =
mutate(((Profile)v.elementAt(i)).vengefulness);
System.out.println("Agent "+(i+1)+":
+((Profile)v.elementAt(i)).boldness + ", " +
+((Profile)v.elementAt(i)).vengefulness + "), credits = "+
+((Profile)v.elementAt(i)).getCredits());
}
System.out.println("No. of survivors and new agents: " + v.size());
// clear counters
total = 0;
average = 0;
diffSquared = 0;
sigma = 0;
// clear credits
for (int k = 0; k < v.size(); k++) {

((Profile)v.elementAt(k)).clearCredits();
}
return v;
}
public int mutate(int v) {
double val = 0;
double rand = 0;
int remain = 0;
int quote = 0;
int[] bit = new int[3];

// first convert to binary
while (v/2 != 0) {
    for (int i = 0; i < 3; i++) {
        remain = v % 2;
        if (remain == 1)
            bit[i] = 1;
        else bit[i] = 0;
        v /= 2;
    }
}
if (v==1) bit[0] = 1;

// each bit has 1% chance of 'flipping'
int j = 2;
while (j != -1) {
    rand = Math.random();
    if (rand >= 0.9) {
        if (bit[j] == 1) bit[j] = 0;
        else bit[j] = 1;
        val += (new Integer(bit[j])).doubleValue() * Math.pow(2, (new Integer(j)).doubleValue());
    } else {
        val += (new Integer(bit[j])).doubleValue() * Math.pow(2, (new Integer(j)).doubleValue());
    }
    j--;
}
return (new Double(val)).intValue();
// end mutate()
}
public static void main(String[] args) {
    ExperimentMeta ep = new ExperimentMeta();
    ep.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    });
}
}
A.7 FAIN active service provisioning (ASP) mechanism

Active Networking technology allows rapid deployment of new services that would otherwise have a long lead-time and possibly require installation of new hardware. Active Service Provisioning, or ASP for short, is understood in the context of the FAIN project as a system for deploying active services in the FAIN network. The 'on-the-fly' deployment process is usually seen as a number of preparatory activities before the phase of the service operation. Typical activities include releasing the service code, distributing the service code to the target location, installing it and activating it.

The relevance of this appendix is that the IDM (via the FAIN PBNM) helps negotiate resources for a service with the help of the work by the ASP work group in FAIN. In particular, we make use of the definition of service components and the relevant network resources that are required for end-to-end service deployment (see interface definition language in §A.8).

Figure 51 presents the identified use cases, their dependencies as well as the actors interacting with them. We explain the following use cases in order to help articulate the case study in §6.4.4:

Service release: A service is ‘released’ when the content provider makes the service meta-information and service code modules available to the ASP system.

Service deployment: Subsequently, the content provider may want to deploy this service so that it can be used by end-users. This means finding active nodes that are most suitable for the given service installation; determining a mapping of the service components to the available EEs of the targeted node; downloading the appropriate code modules, and finally installing and activating them.

Service removal: The content provider may request to remove a service from the environment in which it was deployed. The ASP identifies the installed service components and removes them from the EEs of the target environment.

Service withdrawal: A service released in an Active Network may be withdrawn so that is no longer available to be deployed. The ASP removes the service meta-information and discards the service code modules.

Service reconfiguration: Changes to the current configuration of a service may be requested. Reconfiguration may include modifying component bindings, deploying additional service components or redeploying components that have been already deployed.

Service update: The content provider may announce a new version of an already released service to an Active Network. The service code and metadata of the new version of the service have to replace the code and metadata of the old (updated) version.
Figure 51: Use case depicts the functions of the ASP mechanism. In FAIN the Active Network Provider is referred to as ANSP.
A.8 The interface definition language (IDL) for the ASP

```idl
#include "management.idl"

module org {
    module ist_fain {
        module network {

/** VE identifier. */
typedef string VeID;

/** VANNode identifier */
typedef string VANNodeID;

/** Sequence of VANNode identifiers */
typedef sequence<VANNodeID> VANNodeIDs;

/** Property structure. */
struct Property {
    string name;
    any value;
};

/** Sequence of properties. */
typedef sequence<Property> Properties;

/** Struct identifying a certain VAN node. */
struct VANNode {
    VANNodeID nodeID;
    Properties props;
};

/** Sequence of VAN nodes. */
typedef sequence<VANNode> VANNodes;

/** Struct identifying a certain link between two VAN nodes. */
struct VANLink {
    VANNodeID start;
    VANNodeID end;
    Properties props;
};

/** Sequence of VAN links. */
typedef sequence<VANLink> VANLinks;

/** Struct describing a particular VAN. */
/*struct VANInfo {
    VANNodes nodes;
    VANLinks links;
    VANNodes userNodes; // indicates the user (edge) nodes
};*/

struct VANPath {
    VANNodes nodes;
    VANLinks links;
};

typedef sequence<VANPath> VANPaths;

/** RequiredNode identifier */
typedef string NodeID;

/** the node properties required by a service to deploy */
struct NodeRequirements { // ServiceComponentRequirements /
    Properties props; /* name="INPUT", type=Boolean, semantic: true - INPUT node,
                         false-otherwise, name="OUTPUT", type=Boolean, semantic: true - INPUT node, false-
                         otherwise*/
};

/** Sequence of NodeRequirements. */
typedef sequence<NodeRequirements> NodeRequirementsList;

/** Struct identifying a required link between two required nodes. */
struct LinkRequirements { // ServiceComponentLinkRequirements /
    NodeRequirementsList
    Properties props;
    /* name="INPUT", type=Boolean, semantic: true - INPUT node, 
    false-otherwise, name="OUTPUT", type=Boolean, semantic: true - INPUT node, false-otherwise*/
};

/** Sequence of LinkRequirements. */
typedef sequence<LinkRequirements> LinkRequirementsList;
```

typedef sequence<LinkRequirements> LinkRequirementsList;

/** Struct describing a virtual node **/
struct Node {
    NodeID nodeID;
    NodeRequirements nodeRequirements;
};

/** Sequence of virtual nodes */
typedef sequence<Node> Nodes;

module asp {
    /** Struct describing the topological requirements of a service. The
    information is static and comes from the service descriptor. */
    struct ServiceTopologyInfo {
        LinkRequirementsList links;
        Nodes nodes; // indicates all nodes required by the
        service topology
    };

    /** Struct identifying a certain ServiceComponent. */
    struct ServiceComponentInfo {
        org::ist_fain::node::management::iComponentInitial initial;
        //org::ist_fain::node::management::tPortList ports;
    };

    /** Sequence of ServiceComponentInfos. */
typedef sequence<ServiceComponentInfo> ServiceComponentInfos;

    /** URL */
typedef string URL;

    /** Sequence of URLs */
typedef sequence<URL> URLs;

    /** Service name. */
typedef string ServiceName;

    /** Service component name. */
typedef string ServiceComponentName;

    /** Sequence of service component names. */
typedef sequence<ServiceComponentName> ServiceComponentNames;

    /** Identifier for service instances. */
typedef string ServiceInstanceID;

    /** Description of a service. */
typedef string ServiceDescriptor;

    /** Reference of a code module. */
typedef string CodeModuleRef;

    /** Identifier of a code module. */
typedef string CodeModuleID;

    /** Identifier for service component instances.
    * Includes implicitly the ID of the associated service instance.
    */
typedef string ServiceComponentID;

    /** IP Address of a node */
typedef string NodeIPAddress;

    /** Reference of a service (IOR) */
typedef string ServiceReference;

    /** Exception: service component not found. */
    exception ServiceComponentNotFound {
        ServiceComponentName serviceComponentName;
    };
}

194
/** Exception: service not found. */
exception ServiceNotFound {
    ServiceName serviceName;
};

/** Sequence of service component IDs. */
typedef sequence<ServiceComponentID> ServiceComponentIDs;

/** Exception: service instance not found. */
exception ServiceInstanceNotFound {
    ServiceInstanceID serviceInstance;
};

/** Exception: code module not found. */
exception CodeModuleNotFound {
    CodeModuleID codeModuleID;
};

/** Exception: service component instance not found. */
exception ServiceComponentInstanceNotFound {
    ServiceComponentID serviceComponentID;
};

/** Exception: service (component) installation failed. */
exception InstallationFailed {
    string reason;
};

/** Exception: service (component) instantiation failed. */
exception InstantiationFailed {
    string reason;
};

/** Exception: service (component) removal failed. */
exception RemovalFailed {
    string reason;
};

/** Exception: service component binding failed. */
exception BindingFailed {
    string reason;
};

/** Exception: service component unbinding failed. */
exception UnbindingFailed {
    string reason;
};

/** Exception: service mapping failed. */
exception ServiceMappingFailed {
    string reason;
};

/** Exception: service registration failed. */
exception ServiceRegistrationFailed {
    string reason;
};

/** Exception: ip address not found. */
exception IPAddressNotFound {
    string reason;
};

/* *
* Service inquiry interface.
* Works on the service descriptor, can be used before a service is
* instantiated.
*/
interface ServiceInquiry {

/** Return the names of the service components of a given service. */
ServiceComponentNames listServiceComponentNames
    (in ServiceName serviceName)
    raises (ServiceNotFound);

/** Return the runtime requirements of the given service component. */
Properties getServiceComponentRequirements
    (in ServiceName serviceName, in ServiceComponentName componentName)
    raises (ServiceNotFound, ServiceComponentNotFound);
/** Return the link requirements between the given two service components. */

LinkRequirements getServiceComponentLinkRequirements
(in ServiceName serviceName, in ServiceComponentName from,
in ServiceComponentName to)
raises (ServiceNotFound, ServiceComponentNotFound);

/** Return the topology of a service. */
ServiceTopologyInfo getServiceTopologyRequirements
(in ServiceName serviceName)
raises (ServiceNotFound, ServiceComponentNotFound);

/** Service management interface. */
interface ServiceManagement {

/** Register a service */
void registerService
(in ServiceName serviceName,
in ServiceDescriptor serviceDescriptor,
in URLs serviceURLs,
in org::ist_fain::tidentity who)
raises (ServiceRegistrationFailed);

/** Unregister a service */
void unregisterService
(in ServiceName serviceName,
in org::ist_fain::tidentity who)
raises (ServiceNotFound);

/** Calculate best candidate nodes to deploy service components of a service based on the provided VAN */
VANPath calculateBestCandidate
(in ServiceName serviceName,
in VANPaths vanPaths,
in org::ist_fain::tidentity who)
raises (ServiceNotFound, ServiceMappingFailed);

/** Instantiate a given service using the nodes of the provided VANPath. */
ServiceInstanceID createServiceInstance
(in ServiceName serviceName,
in Properties aConfiguration,
in VANPath vanPath,
in org::ist_fain::tidentity who)
raises (ServiceNotFound, InstantiationFailed);

/** Remove a running service instance. */
void removeServiceInstance
(in ServiceInstanceID serviceInstance,
in org::ist_fain::tidentity who)
raises (ServiceInstanceNotFound, RemovalFailed);

ServiceComponentID deployServiceComponent
(in ServiceName serviceName,
in Properties aConfiguration,
in ServiceDescriptor descriptor,
in URL codeModuleURL,
in VANPath vanPath,
in org::ist_fain::tidentity who)
raises (InstallationFailed);

/** Add a code module to the given service instance on the specified VAN node. */
// Requested by Epi
CodeModuleRef addServiceCodeModule
(in ServiceInstanceID serviceInstance,
in CodeModuleID codeModuleID,
in VANNodeID vanNodeID,
in org::ist_fain::tidentity who)
raises (ServiceInstanceNotFound, CodeModuleNotFound, InstallationFailed);

/** Remove a code module from the given service instance on the specified VAN node. */
void removeServiceCodeModule
(in ServiceInstanceID serviceInstance,
in CodeModuleID codeModuleID,
instance. */
ServiceComponentIDs listRunningServiceComponents  
(in ServiceInstanceId serviceInstance)  
raises (ServiceInstanceNotFound);
/** Return the IDs of the running service components of a given service */
ServiceComponentID getServiceComponentID  
(in ServiceInstanceId serviceInstance,  
in ServiceComponentName componentName)  
raises (ServiceInstanceNotFound, ServiceComponentNotFound);
/** Retrieve instanceID of a specific service component within a specific service */
ServiceComponentID addServiceComponent  
(in ServiceInstanceId serviceInstance,  
in ServiceComponentName componentName,  
in VANPath vanPath,  
in org::ist_fain::tidentity who)  
raises (ServiceInstanceNotFound, ServiceComponentNotFound,  
InstallationFailed);
/** Remove a service component instance from a given service service. */
void removeServiceComponent  
(in ServiceComponentID componentID,  
in org::ist_fain::tidentity who)  
raises (ServiceComponentInstanceNotFound, RemovalFailed);
/** Bind the given two service component instances using WP3 ports. */
void bindServiceComponents  
(in ServiceComponentID first, in ServiceComponentID second,  
in org::ist_fain::tidentity who)  
raises (ServiceComponentInstanceNotFound, BindingFailed);
/** Unbind the given two service component instances. */
void unbindServiceComponents  
(in ServiceComponentID first, in ServiceComponentID second,  
in org::ist_fain::tidentity who)  
raises (ServiceComponentInstanceNotFound, UnbindingFailed);
/** Bind the given service component instance to the specified WP3 port. */
void bindServiceComponentPort  
(in ServiceComponentID componentID,  
in org::ist_fain::node::management::tPortName cPortName,  
in org::ist_fain::node::management::tPort otherPort,  
in org::ist_fain::tidentity who)  
raises (ServiceComponentInstanceNotFound, BindingFailed);
/** Unbind the given service component instance from the specified WP3 port. */
void unbindServiceComponentPort  
(in ServiceComponentID componentID,  
in org::ist_fain::node::management::tPortName cPortName,  
in org::ist_fain::node::management::tPort otherPort,  
in org::ist_fain::tidentity who)  
raises (ServiceComponentInstanceNotFound, UnbindingFailed);
/** The network ASP manager supports both service inquiry and management. */
interface NetworkASPManager : ServiceInquiry, ServiceManagement {  
};
A.9 XML Schema for QoS policy for active and passive nodes

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!-- edited with XML Spy v 4.3 U (http://www.xmlspy.com) by UCL -->
  <xsd:annotation>
    <xsd:documentation xml:lang="en">Virtual En</xsd:documentation>
  </xsd:annotation>
  <xsd:element name="fainPolicyRule" type="fainPolicyRuleType" />
  <xsd:element name="CommonElements">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Caption" type="xsd:string" />
        <xsd:element name="Description" type="xsd:string" />
        <xsd:element name="CommonName" type="xsd:string" />
        <xsd:element name="PolicyKeywords">
          <xsd:simpleType>
            <xsd:list itemType="PolicyKeywordValue" />
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="PolicyActionKeys">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="CreationClassName">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="Key" type="xsd:boolean" fixed="true" />
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="PolicyActionName">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="Key" type="xsd:boolean" fixed="true" />
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="PolicyConditionKeys">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="CreationClassName">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="Key" type="xsd:boolean" fixed="true" />
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="PolicyConditionName">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="Key" type="xsd:boolean" fixed="true" />
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="PolicyVariableKeys">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="CreationClassName">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="Key" type="xsd:boolean" fixed="true" />
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="PolicyVariableName">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="Key" type="xsd:boolean" fixed="true" />
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```
```xml
  <xsd:complexType name="fainPolicyRuleType">
    <xsd:sequence>
      <xsd:element name="PolicyRoles" type="xsd:string"/>
      <xsd:element name="Enabled" type="xsd:decimal"/>
      <xsd:element name="PolicyDecisionStrategy" type="xsd:decimal" maxOccurs="3"/>
      <xsd:element name="PolicyRuleName" type="xsd:string" minOccurs="0"/>
      <xsd:element ref="CommonElements" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
```
<xsd:element name="RequestedBW" type="xsd:integer" />
  - <xsd:element name="ActionMode">
    - <xsd:simpleType>
      - <xsd:restriction base="xsd:integer">
        <xsd:enumeration value="0" />
        <xsd:enumeration value="1" />
        <xsd:enumeration value="2" />
        <xsd:enumeration value="4" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
</xsd:complexType>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="fainQoSCompAllocActionType">
  - <xsd:complexContent>
    - <xsd:extension base="fainSimplePolicyActionType">
      - <xsd:sequence>
        - <xsd:element name="VNIId" type="xsd:string" />
        - <xsd:element name="CompQoSClass">
          - <xsd:simpleType>
            - <xsd:restriction base="xsd:integer">
              <xsd:enumeration value="0" />
              <xsd:enumeration value="1" />
              <xsd:enumeration value="2" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        - <xsd:element name="EETechnology" type="xsd:string" />
        - <xsd:element name="ActionMode">
          - <xsd:simpleType>
            - <xsd:restriction base="xsd:integer">
              <xsd:enumeration value="0" />
              <xsd:enumeration value="1" />
              <xsd:enumeration value="2" />
              <xsd:enumeration value="4" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        - <xsd:element name="Credential">
          - <xsd:complexType>
            - <xsd:sequence>
              - <xsd:element name="Name">
                - <xsd:simpleType>
                  - <xsd:restriction base="xsd:string" />
                </xsd:simpleType>
              </xsd:element>
              - <xsd:element name="Password">
                - <xsd:simpleType>
                  - <xsd:restriction base="xsd:string" />
                </xsd:simpleType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        - <xsd:element name="RequestedBW" type="xsd:integer" />
        - <xsd:element name="CompQoSClass">
          - <xsd:simpleType>
            - <xsd:restriction base="xsd:integer">
              <xsd:enumeration value="0" />
              <xsd:enumeration value="1" />
              <xsd:enumeration value="2" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        - <xsd:element name="EETechnology" type="xsd:string" />
        - <xsd:element name="ActionMode">
          - <xsd:simpleType>
            - <xsd:restriction base="xsd:integer">
              <xsd:enumeration value="0" />
              <xsd:enumeration value="1" />
              <xsd:enumeration value="4" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="fainQoSAllocActionType">
  - <xsd:complexContent>
    - <xsd:extension base="fainSimplePolicyActionType">
      - <xsd:sequence>
        - <xsd:element name="VNIId" type="xsd:string" />
        - <xsd:element name="QoSClass">
          - <xsd:simpleType>
            - <xsd:restriction base="xsd:integer">
              <xsd:enumeration value="0" />
              <xsd:enumeration value="1" />
              <xsd:enumeration value="2" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        - <xsd:element name="RequestedBW" type="xsd:integer" />
        - <xsd:element name="CompQoSClass">
          - <xsd:simpleType>
            - <xsd:restriction base="xsd:integer">
              <xsd:enumeration value="0" />
              <xsd:enumeration value="1" />
              <xsd:enumeration value="2" />
              <xsd:enumeration value="4" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        - <xsd:element name="EETechnology" type="xsd:string" />
        - <xsd:element name="ActionMode">
          - <xsd:simpleType>
            - <xsd:restriction base="xsd:integer">
              <xsd:enumeration value="0" />
              <xsd:enumeration value="4" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:complexContent>
</xsd:complexType>
A.10 Java source code for the inter-domain manager (IDM)

The following notes were made for running the IDM, available in a file called readme.txt.

Instructions:

1. Compile the implementation files to generate the stubs. Do this on all the nodes that host IDMs. Best practice currently is to do (3) and (4) on the same machine, and run the test from another one. Once decided, on the RequestorClient file, change REPOSITORY_IP to the IP address of the repository's host.
   
   ```
   % rmic -v 1.2 RepositoryImpl RequestorImpl
   ```

2. Start up the RMI registry on all the nodes where the IDM is sitting on.
   
   ```
   % rmiregistry &
   ```

3. Run the repository server, providing the IP address of its host as an argument.
   
   ```
   % java RepositoryServer [repository_hostIP] &
   ```

4. IDMs must register reachable destinations (within their domain) on the repository, providing (in correct order) the IDM's host IP, the destination to be registered, and the repository's IP, as arguments.
   
   ```
   % java RequestorServer [IDM_IP] [destination_IP] [repository_hostIP] &
   ```

5. The test program is now ready to run. Currently, it is written to facilitate the Resource resource manager, i.e., the resource manager is supposed to enter six arguments when invoking the relay() method. The QoS parameters should be given as integers, currently arbitrarily good inputs approximately between 10 to 400.
   
   ```
   % java Tester [ingress_IP] [destination_IP] [bandwidth] [CPU time] [memory] [-r reserve] [-n negotiate] *but not both*
   ```

Miscellaneous:

- resource.txt is a dummy file to represent the monitored values of the peer's available resources.
- repository.txt simulates database entries on the repository, which maps destination addresses of a domain to the IDM in charge of a domain.
- parameters.txt is the output file used to show that a request has been relayed from one domain to another. It contains: ingress IP, destination IP, service name, and credential (basically content provider's name) and this is exactly the arguments for the deployService() method for the service manager. For integration, the request should be 'piped' to the service manager, instead of the file.

Versions:

12 February 2003: This folder contains a working version of the IDM.

26 February 2003: Negotiating function added. resource.txt is the dummy file to represent monitored values of the peer's available resource. Works locally.

3 March 2003: IDM works well stand-alone between two separate machines.
import java.rmi.*;
import java.rmi.server.*;

public class RequestorClient{
    private static final String REPOSITORY_IP = "10.0.4.1";
    //kindly change this to the IP address of the repository's host

    public RequestorClient() {}

    public void relay( ServiceDescriptor s, Context c, QoS q, String a ){
        //System.setSecurityManager( new RMISecurityManager() );
        //Contact the repository to get the pertinent IDM's location
        System.out.println( "Getting IDM-destination mappings from repository ")
        try{
            repository = ( Repository )Naming.lookup( "//" + REPOSITORY_IP + "/repository" );
            idm = repository.getLocation( c.destination );
        } catch(Exception e ) {
            System.out.println( "Error: " + e ) ;
        }
        try {
            req = ( Requestor )Naming.lookup( idm );
            if (a.equalsIgnoreCase( "-r" ) ) req.reserve( s, c );
            if (a.equalsIgnoreCase( "-n" ) ) System.out.println( req.trade( q ) );
        } catch(Exception e ) {
            System.out.println( "Error: " + e ) ;
        }
        System.exit(0);
    }

    private static Requestor req;
    private static Context context;
    private static ServiceDescriptor service;
    private static Repository repository;
    private String idm;
}

import java.rmi.*;
import java.rmi.server.*;
import java.io.*;

public class RequestorImpl extends UnicastRemoteObject implements Requestor {
    public RequestorImpl() throws RemoteException {)
        public void reserve( ServiceDescriptor s, Context c )
            throws RemoteException, FileNotFoundException {
                String target = null;
                try
                }

    private static Requestor req;
    private static Context context;
    private static ServiceDescriptor service;
    private static Repository repository;
    private String idm;
}
try {
    target = new PrintStream (
        new FileOutputStream (new File("parameters.txt")));
}

try {  
    try {
        target.println(c.ingress + "\n" +
        c.destination + "\n" + s.name + "\n" + s.provider + "\n");
    } catch (Exception e) {
        System.err.println(e.getMessage() + "\n");
        e.printStackTrace();
    }
}

public String trade(QoS q) throws RemoteException, FileNotFoundException {
    Negotiator n = new Negotiator();
    String s = n.bargain(q);
    return s;
}

public class RequestorServer {

    * private static final String REPOSITORY_IP = "192.168.1.15";
    * 192.168.1.15 is george, 128.40.40.22 is face,
    * 192.168.1.9 is ronnie, 128.40.40.93 is optimus;
    * if using this approach, replace args[2] with REPOSITORY_IP

    public static void main(String args[]) {
        * Note: takes in three arguments, the IDM's IP (args[0]) -
        * do NOT input 'localhost'; the final destination (args[1]);
        * and the repository's IP [args[2]], which is well-known to
        * all participating ANSPs

        try {
            System.out.println("Inputting IDM-destination mappings on the repository");
            repository = (Repository)Naming.lookup("//" + args[2] + "/" + "repository");
        } catch (Exception e) {
            System.err.println("Error " + e);
        }

        System.out.println("Constructing server implementations at a potential requestee...");
        RequestorImpl req = new RequestorImpl();
        printRegistry(args[0]);
        System.out.println("Binding server-side implementations to IDM");
    }

    import java.rmi.*;
    import java.rmi.server.*;

    public static void printRegistry(String args[2]) {
        try {
            repository.setLocation(args[0] + "/" + args[1] + "," + args[1]);
        } catch (Exception e) {
            System.out.println("Error " + e);
        }
    }

    public static void printRegistry(String args[0]) {
        System.out.println("Binding server-side implementations to IDM");
    }
}
registry...";
Naming.rebind("/" + args[0] + "/" + args[1], req); //change from
//args[1], the final destination is given as the logicalName
System.out.println("Waiting for invocations from peer IDM...");
}
catch(Exception e) {
  System.out.println("Error: + e);
}

private static void printRegistry (String s) {
  try {
    System.out.println("Current registry entries on: + s");
    String [] data = Naming.list("/" + s + "/");
    int i = 0;
    for (i=0; i<data.length; i++) {
      System.out.println(data[i]);
    }
  } catch(Exception e) {
    System.out.println("Error in listing registry entries: + e");
  }
}
private static Repository repository;

/*================================================================***/
<p>Title: Repository.java</p>
<p>Description: Interface for a database of destination network address mapped against URL where the corresponding IDM is located on. Provides methods for retrieving IDM location when given address; Also to input the IDM-destination mapping </p>
/*================================================================***/
import java.rmi.*;
import java.rmi.server.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.rmi.
import java.rmi.server.*;
public interface Repository extends Remote {
  public String getLocation(String g)
    throws RemoteException, FileNotFoundException;
  public void setLocation(String s)
    throws RemoteException, FileNotFoundException;
}
/*================================================================***/
<p>Title: RepositoryImpl.java</p>
<p>Description: Database of destination destination address mapped against IDM port. Provides method for retrieving port when given address.</p>
/*================================================================***/
import java.io.*;
import java.util.*;
import java.net.*;
import java.rmi.*;
import java.rmi.server.*;
public class RepositoryImpl extends UnicastRemoteObject implements Repository {
  public RepositoryImpl() throws RemoteException {
    //method to get the IDM's IP, given a destination address
    public String getLocation(String d)
      throws RemoteException, FileNotFoundException {
      String f = get();
      repValues = commaSplitter(f);
      int i = 0;
      for (i=0; i < repValues.length; i++) {
        if (repValues[i].equalsIgnoreCase(d))
          idm = "/" + repValues[i-1];
      }
public void setLocation( String d ) throws RemoteException, FileNotFoundException {
    String f, g;
    PrintStream target = null;
    f = get();
    System.out.println( "Repository entries before input by IDM: " + f );
    g = f.concat( "," + d );
    try {
        target = new PrintStream {
            new FileOutputStream ( 
                new File("repository.txt"))};
        catch ( Exception e ) {
            System.err.println( e.getMessage() + "n" );
            e.printStackTrace();
        }
        try {
            target.println( g );
            System.out.println("Repository entries after input by IDM: " + g);
        }
        catch ( Exception e ) {
            System.err.println( e.getMessage() + "n" );
            e.printStackTrace();
        }
        return;
    }

    private String get() throws RemoteException, FileNotFoundException {
        String line = "";
        BufferedReader repFile = new BufferedReader ( 
            new InputStreamReader ( 
                new FileInputStream ( 
                    new File("repository.txt"))));
        try {
            line = repFile.readLine();
        }
        catch ( Exception e ) {
            System.err.println( e.getMessage() + "n" );
            e.printStackTrace();
        }
        return line;
    }

    private String[] commaSplitter( String s ) {
        int loc = s.indexOf( ',' );
        if ( loc == -1 ) return null;
        Vector v = new Vector();
        while ( (( loc = s.indexOf( ',' ) ) != -1 ) {
            String bit = s.substring( 0, loc );
            v.addElement(bit);
            s = s.substring(loc + 1);
        }
        if ( s.length() > 0 ) v.addElement( s );
        String[] result = new String[v.size()];
        v.copyInto( (Object[]) result );
        return result;
    }

    String[] repValues;
    String destination, idm;
}

>Title: RepositoryServer.java </p>
<Description: </p>

import java.rmi.*;
import java.rmi.server.*;
public class RepositoryServer {

    public static void main( String args[] ) {
        // Note: takes in one argument, the repository's IP (args[0])
        try{
            //this try-catch block attempts to set up the repository to listen to peers
            System.out.println( "Constructing server implementations " );
            try {
                String[] data = Naming.list( "//" + args[0] + "/*destination_IP" );
                int i =0;
                for (i=0; i<data.length; i++) {
                    System.out.println( data[i] );
                }
            } catch ( Exception e ) {
                System.out.println( "Error in listing : " + e );
            }
            RepositoryImpl rep = new RepositoryImpl();
            System.out.println( "Binding server implementations to registry..." );
            Naming.rebind( "//" + args[0] + "/*repository", rep );
            System.out.println( "Waiting for method calls from the IDMs..." );
        } catch( Exception e ) {
            System.out.println( "Error: " + e );
        }
    }
}

import java.io.*;

public class Context implements Serializable {
    String ingress;
    String destination;

    public Context( String ingress, String destination ) {
        this.ingress = ingress;
        this.destination = destination;
    }
}

import java.io.*;

public class ServiceDescriptor implements Serializable {
    String name;
    String provider;
    String location;
    String component;

    public ServiceDescriptor( String name, String provider, String location, String component ) {
        this.name = name;
        this.provider = provider;
        this.location = location;
        this.component = component;
    }
}

Title: Context.java </p>
Description: Two end points of a VAN in a domain</p>
*******************************************************************************/

import java.io.*;

public class Context implements Serializable {
    String ingress;
    String destination;

    public Context( String ingress, String destination ) {
        this.ingress = ingress;
        this.destination = destination;
    }
}

/* ******************************************
Title: ServiceDescriptor.java */
Description: To emulate a service descriptor */
*******************************************************************************/

import java.io.*;

public class ServiceDescriptor implements Serializable {
    String name;
    String provider;
    String location;
    String component;

    public ServiceDescriptor( String name, String provider, String location, String component ) {
        this.name = name;
        this.provider = provider;
        this.location = location;
        this.component = component;
    }
}
import java.io.*;

public class QoS implements Serializable {
    String bandwidth;
    String cpu;
    String memory;

    public QoS(String bandwidth, String cpu, String memory) {
        this.bandwidth = bandwidth;
        this.cpu = cpu;
        this.memory = memory;
    }
}

import java.util.*;
import java.io.*;

public class Negotiator {
    public Negotiator() {
    }

    public String bargain(QoS q) throws FileNotFoundException {
        String finalStatus = "";
        BufferedReader fileInput = new BufferedReader(
            new InputStreamReader(
                new FileInputStream(
                    new File("resource.txt"))));
        String line = "";
        try {
            line = fileInput.readLine();
        }
        catch (Exception e) {
            System.err.println(e.getMessage() + "\n");
            e.printStackTrace();
            return null;
        }
        String[] monitoredValues = commaSplitter(line);
        String[] requestedValues = {q.bandwidth, q.cpu, q.memory};
        String[] resourceKey = {"bandwidth", "CPU", "memory"};
        String[] requestStatus = new String[3];
        int reqIndex = 0;
        //int monIndex = 0;
        for (; reqIndex < requestedValues.length; reqIndex++) {
            requestStatus[reqIndex] = (Integer.parseInt(requestedValues[reqIndex]) <
                Integer.parseInt(monitoredValues[reqIndex])?
                "Request for " + resourceKey[reqIndex] + " granted."
                : "Insufficient " + resourceKey[reqIndex] + ". Re-negotiate for " +
                monitoredValues[reqIndex] + "?";
            finalStatus = finalStatus.concat("\n" + requestStatus[reqIndex]);
        }
        System.out.println("Responded to peer on availability");
        return finalStatus;
    }

    private String[] commaSplitter(String s) {
        int loc = s.indexOf(',');
        if (loc == -1) return null;
        Vector v = new Vector();
        while ((loc = s.indexOf(',')) != -1) {
            String bit = s.substring(0, loc);
            v.addElement(bit);
            s = s.substring(loc + 1);
        }
    }
}
if (s.length() > 0)
    v.addElement(s);
String[] result = new String[v.size()];
v.copyInto(Object[] result);
    return result;
}

/*************************************************************************/

import java.io.*;

public class Tester {

    public static void main (String[] args)
        throws FileNotFoundException {
            context = new Context (args[0], args[1]);
            service = new ServiceDescriptor("SPI", "WebTV", "ingress", "transcoder");
            qos = new QoS(args[2], args[3], args[4]);
            RequestorClient rc = new RequestorClient();
            String action = args[5];
            rc.relay(service, context, qos, action);
        }

private static ServiceDescriptor service;
private static Context context;
private static QoS qos;
}
A.11 XML Schema for domain-wide strategies

```xml
<xsd:schema targetNamespace="http://www.ee.ucl.ac.uk/~atan"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ee.ucl.ac.uk/~atan DomainStrategy.xsd">
  <xsd:element name="Dimension" type="xsd:positive-integer"/>
  <xsd:element name="Strategy">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="InitialResponse" type="xsd:boolean"/>
        <xsd:element name="Cooperation" type="xsd:boolean"/>
        <xsd:element name="Defection" type="xsd:boolean"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TimePeriodConditionType">
      <xsd:sequence>
        <xsd:element name="TimePeriod" type="xsd:string"/>
        <xsd:element name="MonthOfYearMask" type="MoYMT" minOccurs="0"/>
        <xsd:element name="DayOfMonthMask" type="DoMMT" minOccurs="0"/>
        <xsd:element name="DayOfWeekMask" type="DoWMT" minOccurs="0"/>
        <xsd:element name="TimeOfDay" type="xsd:string" minOccurs="0"/>
        <xsd:element name="LocalOrUtcTime" type="LoUTType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:simpleType name="MoYMT">
    <xsd:restriction base="xsd:hexBinary">
      <xsd:length value="2"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="DoWMT">
    <xsd:restriction base="xsd:hexBinary">
      <xsd:length value="1"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="LoUTType">
    <xsd:restriction base="xsd:integer">
      <xsd:enumeration value="1"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="DoMMT">
    <xsd:restriction base="xsd:hexBinary">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="IPv4Addr">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="(\d{0,255})\.(\d{0,255})\.(\d{0,255})\.(\d{0,255})/\d{0,32}"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="Resource">
    <xsd:complexContent>
      <xsd:sequence>
        <xsd:element name="QoSClass" type="xsd:integer"/>
        <xsd:element name="RequestedBW" type="xsd:integer"/>
        <xsd:element name="RequestedCPU" type="xsd:integer"/>
        <xsd:element name="EEType" type="xsd:integer"/>
        <xsd:element name="ActionMode" type="xsd:integer"/>
      </xsd:sequence>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```
<xsd:element name="Pricing">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:restriction base="xsd:integer">
                <xsd:enumeration value="0"/>
                <xsd:enumeration value="1"/>
                <xsd:enumeration value="2"/>
            </xsd:restriction>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
A.12 Publications

The following is a list of publications achieved over the 4-year course of the Ph.D research period.


Bibliography and references


(http://www.ietf.org/rfc/rfc2205.txt)


[75] http://qbone.internet2.edu/


Li, T., and Y. Rekhter. A provider architecture for differentiated services and traffic engineering (PASTE), In IETF RFC 2430, 1998.


225