MAGNET: QoS-based Dynamic Adaptation in a Changing Environment

Patty Kostkova, Steve Crane, Julie McCann and Tim Wilkinson
High Performance Extensible Systems Centre
City University
London, UK
{patty, jsc, jam, tim}@sarc.city.ac.uk

Abstract

The rapid growth of mobile computing, encouraged by ever-decreasing size and weight of hardware, has led to a spectrum of computing environments. Use of portables while in transit (e.g., on a train or a plane), or when movement is in the nature of business (e.g., travelling salesmen) has become a commonplace and is no longer remarked upon.

Common to all mobile users is a frequently-changing working environment. In order to utilize local resources fully, mobile users have to adjust their quality of service (QoS) requirements perceived according to local conditions. It enables them to fully utilize resources available in their current working environment. Hitherto, mobile users lack infrastructural support for such dynamic resource management.

This paper describes MAGNET, a model of dynamic information exchange. Applied as a resource manager, it permits dynamic update of resource information and user-defined QoS-based resource selection, enabling transparent reconfiguration.

1 Introduction

The enormous growth of mobile computing in recent years has been driven by three factors: improvements and affordability of wireless communication, decreasing hardware size and weight, and other, qualitative, advances in hardware technology (e.g., the invention of the colour LCD display). Owing to a combination of these advances, portable computers are frequently used during transport (e.g., on a train or plane), or when movement is the nature of business (e.g., mobile salesmen, members of staff working in different company branches, etc.) A characteristic of the portable is that it is stationary but transportable, usually running classical operating systems and applications, but constrained by hard-disk space and battery life [14].

In this paper, we focus on the problems encountered by mobile users who frequently change their computing environment, rather than lower-level problems due to changes in physical location and degree of connectivity. The fundamental problem imposed by the variation in computing environments is lack of support for dynamic adaptation to the change in resource configuration.

In this paper we describe MAGNET, an information pool which provides an environment for resource location based on a user-defined QoS description. We illustrate its utility by an example of a mobile user working on a portable in a changing environment.

The next section discusses our motivations in greater detail, focussing on the

characteristics of the mobile environment and support for dynamic information[1] update. Section 3 presents the MAGNET design. In section 4, we demonstrate the role of MAGNET in a typical 'mobile office'. Section 5 summarizes related work in comparison to our approach. Section 6 describes MAGNET's current status and discusses directions of future reseach. Finally, section 7 contains our concluding remarks.

2 Motivations

In this section we describe the mobile computing environment and our motivations to investigate services supporting dynamic information exchange. Unlike the classical QoS approach dealing with unreliability of wireless computing, we extend the definition of QoS to all 'characteristics' of resources (e.g., QoS provided by a printer can be described in terms of: black&white printer versus colour one, laser printer versus matrix or ink-jet ones, resolution, speed etc.) Finally, we elaborate the requirements of the information-sharing infrastructure.

2.1 Characteristics of Mobile Environment

By 'mobility' we mean frequent changing of users' physical location which cause the volatility of information such as the location of the nearest server, local hardware environment, and location-dependent information (e.g., the local time, a useable ISP, etc.).

We distinguish several degrees of connectivity forming a wide *spectrum of communication environments* extending from: totally disconnected, through weakly-connected (e.g., low-speed IR networks), to fully-connected by Ethernet or optical fibre networks (FDDI, ATM) [3].

Recent research has focused on adaptable systems which react to achieve a predefined level of QoS. This was to overcome the problems associated with mobile communications systems. However, in recent years we have observed enormous improvements in speed, reliability and coverage of wireless communications resulting in the need for reactive services decreasing. Expecting this process to continue, weakly-connected (via dial-up) systems will no longer suffer from significantly low bandwidth, high error-rates, speed fluctuation and frequent disconnection. Consequently, this trend has decreased the demand for so-called *reactive services* [13] which deal with the rapid fluctuations in QoS of the underlying communication infrastructure of weakly-connected networks [1]. On the other hand, this issue of QoS is still important for specific applications which require continuous data transmission (e.g., runtime multimedia applications). We complement that research by exploring the advantages to be gained from the use of reactive services which support configuration, reconfiguration and location dependent data requests to achieve QoS at a higher-level.

[1] By 'information' we mean the data characterizing components of an application required to satisfy a request whether it be a request; for a location-dependent resource (e.g., printer) or for location dependent data (e.g., tourist information).

2.2 Support for up-to-date information exchange

The improvement in low-level wireless communications motivate us to focus on a residual problem — service support for *dynamically-updated* information for applications requiring location and time-aware information (such as tourist navigation software running on personal digital assistants (PDAs), and portable applications using local resources in different offices, etc.) The crucial problem for portables no longer is the fluctuation in the quality of connection to their home services (while dialing-in) but an inability to adapt dynamically to the resource configuration in environments where they have just arrived. In order to fully utilize access to local resources, current mobile users have to manually adjust their QoS requirements to local conditions.

Resource QoS changes in two dimensions. Firstly, it varies with time (e.g., fluctuation of local throughput and error-rate of networks, contention for devices such as printers), secondly, it changes with location (e.g., there is a colour ink-jet printer in this new office, which a portable should be able to use).

Due to performance restrictions and the requirement of heterogeneity, computing environments are no longer course-grained and monolithic. System elements at all levels (hardware, software, and data) are becoming fine-grained, componentised and more independent of one another. This environment, supported by dynamic services, enables applications to tailor the environment using only the required components. This is essential in a mobile computing environment where computer power is constrained by the capacity of the battery.

2.3 Information-sharing Infrastructure Requirements

Above we described the changing spectrum of environments and identified the lack of service support to meet the needs of dynamic mobile users. In order to provide an information-sharing infrastructure (which we term a 'pool') enabling users to exchange up-to-date information (which we term a 'tuple'), we need to define its high-level requirements.

To achieve full generality, the infrastructure should not define any format for, or pose any restrictions on, information placed into the pool. The pool should be distributed, in order to prevent a centralized element becoming a bottleneck of the entire system. In addition, it must be scalable to the extent required by mobile applications (e.g., mobile users travelling into different company branches — presumably abroad — still require access to the pool information).

High volatility of information in mobile environments imposes a fundamental requirement on mobile applications — support for dynamic information update. Classical two-party 1:1 interaction (such as in a traditional client-sever systems, where one service satisfies a particular request) is not suited to the multi-component nature of dynamic mobile applications. That is, they require *multi-party communication* (m:n interaction, where m sources communicate with n sinks) enabling dynamic service choice.

Finally, in order to enable the actual communication in the pool, tuples must interact — *match*. If requirements can be expressed exactly then 'white-pages' matching can be carred out. Two tuples are termed matching if, and only if, their corresponding actual paramenters are of equal type and value, and formal parameters are of equal type. White-pages matching is unsuitable for situations where requirements on resources are parametrized, requiring 'yellow-

pages' matching (that is, two tuples match if, and only if, all structured corresponding parameters 'match', where the matching function has user-defined semantics) [12]. This level of freedom leads to a crucial requirement on the dynamic information exchange service: in addition to providing fundamental white-pages matching, it must also provide a general mechanism for incorporating user-defined matching algorithms. Current research has not addressed this issue as yet.

3 Dynamic Information Exchange Infrastructure Design

The infomation exchange framework, called MAGNET, consists of three key elements: *information pool, matching functions,* and MAGNET Server. The information pool, the data structure is the shared tuple repository enabling data to be inserted and withdrawn. Application-specific trading functions match service offers against requests. Access to each pool is controlled by a MAGNET Server which communicates with users, with other MAGNET Servers, and is responsible for distribution and scalability.

3.1 Information pool

The design of the information pool was inspired by a communication environment called the tuplespace. Tuples (record-like data structures) can be dynamically inserted in, or removed from, a shared data structure, the tuplespace. This is typically distributed, and can be accessed by various applications. Actual communication occurs by exchanging information in tuples that *match* (currently, white-pages matching is only supported) [8].

The fixed semantics of the classical operations defined in the tuplespace paradigm are too restrictive to satisfy requirements of mobile applications. Nevertheless, MAGNET's information pool has properties of the classical tuplespace: *decoupling* of the communicating parties, *free-naming*, *asynchronous* and *multi-party communication*, and, finally, *stateless* character of tuples.

The key feature, which our information pool shares with tuplespace, is the *decoupling* of the communicating parties (traditionally client and server). Any server component can generate a tuple of interest to any client component, which allows communication to proceed anonymously. This essential characteristic enables the pool to support *free-naming* (that is, communication can be performed without previous knowledge of the peer's actual identity). If naming is required, it can be encapsulated as a parameter of a tuple.

Asynchronous communication is another characteristic of the model, sparing applications unnecessary synchronization. The information pool acts as a shared 'noticeboard' enabling *multi-party communication* of dynamically-formed groups of components — any component can join or leave a group at any time without affecting other parties.

At the pool level, all tuples are stateless — state is encapsulated within the tuple as a parameter. Therefore, there is no need for the pool to provide a state-maintenance mechanism (e.g., checkpointing). This concept improves the generality and clarity of the system while contributing to system reliability.

In contrast to the classical tuplespace, MAGNET permits *time-constrained* operations to be performed on applications that are unwilling to wait until a required item becomes available. This feature contrasts to classical tuples which persist forever even if the required matching tuple has not been inserted.

In addition to user-originated binding, MAGNET provides a framework for *third-party* binding. It allows; in exceptional cases; a 'human manager' to interfere with the matching process, in order to enforce a binding which could not have been established. In addition to classical operations for *insertion* and *removal*, MAGNET provides a mechanism for the actual communication — *the matching* of tuples.

3.2 Application-Specific Trading

MAGNET, as a general framework, does not restrict the format of data items placed into the pool. In the previous section 2.3, we discussed the need for white-pages and yellow-pages matching which leads to one of MAGNET's key addvantages: the implemention of QoS-based user-defined functions. To define a universal unambiguous matching function for unconstrained parametrized matching is impossible because data parameters (e.g., resource features) cannot be linearly ordered without application-defined preferences. For this reason, QoS matching functions must be provided by the applications themselves.

MAGNET solves this problem by enabling a parametrized matching function to be dynamically 'inserted' into the pool. However, enabling user-defined extensions to execute their code on the information pool adds the problem of data security in the pool, and it now becomes impossible to estimate a time in which a match can be achieved. As for the security, section 3.4 discusses this issue in more detail. The problem of time complexity of user-defined matching functions has been theoretically proven as algorithmically unsolvable (it is equivalent to the halting problem). Therefore, it is impossible for MAGNET to incorporate a 'finality check' of user-defined functions before they are inserted into the information pool. We approach this issue by enabling parallel processing of matching functions (each function is activated as an independent thread of control) preventing other applications from being blocked by incorrectly defined functions.

In order to make the construction of the matching function more tractable, MAGNET encourages applications to *predefine* the matching functions typically required by the parties involved (users requiring resources, tourists asking for information, etc). Fundamental predefined QoS matching functions are: *first-fit* (first appropriate matching tuple is returned), and *best-fit* (the most suitable tuple according to QoS requirements is returned). Applications with requirements not covered by predefined functions have to implement matching functions themselves. MAGNET treats both classes of matching functions (predefined and user-defined) equally.

3.3 The MAGNET Server

The MAGNET Server is an active process controlling the information pool. By exporting the pool operations (e.g., insert, withdraw, etc.), it defines a further set of supported operations available to users. In addition, it handles

time-constrained operations (for example, it resumes blocked components when their timeout expires), and implements the appropriate strategy which manages QoS matching functions (their activation, parallelization and coordination), and plays an essential role in the distribution of the information pool.

In order to achieve the scalability and tuple information sharing, the information pool is physically distributed into 'local pools'. The semantics of the distribution is defined by the actual application, and is irrelevant for high-level MAGNET design. Local pools can be disconnected, or connected to other local pools. Each one is running its MAGNET Server which communicates with other MAGNET Servers in order to achieve data location and distribution transparency in the information pool.

To join the information pool, the MAGNET Server of a disconnected unit must locate other MAGNET Servers with which to communicate. Intercommunication between MAGNET Servers is implemented using a multicast group.

Figure 1 illustrates the high-level MAGNET structure: the information pool with tuples placed into it, three predefined matching functions, and the MAGNET Server controlling it. For clarity, the physical distribution of the information pool and the MAGNET Server is not visible at this level. MAGNET forms an environment for communication between three components advertising the information (such as resource offers, tourist information etc.), and two components requesting the information (such as requests for resources, tourists information queries etc.)

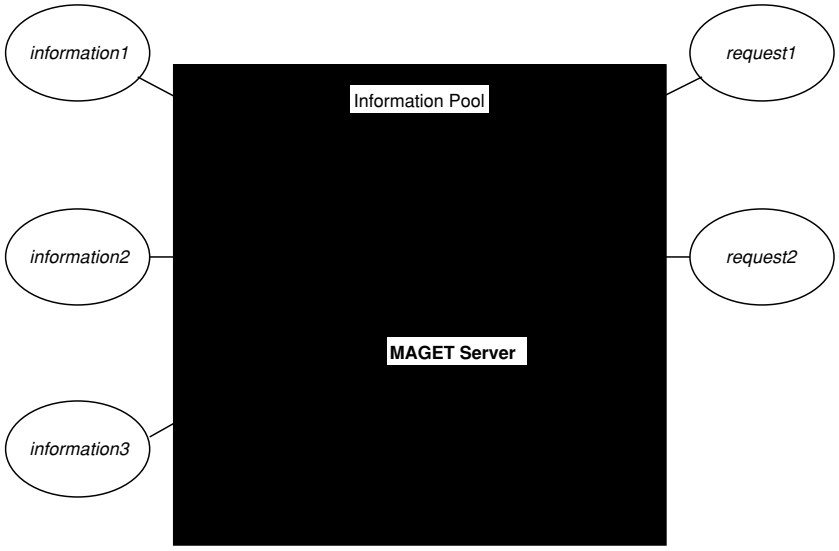

Figure 1: The MAGNET Structure

3.4 Security Issues

As all interconnectivity between system components is achieved through the information pool, security is an issue. The pool itself does not provide any semantics for security and therefore it is possible for untrusted matching functions to corruptly use the information in the pool. Furthermore, unrestricted infor-

mation pool scaling can result in undesirable users joining the pool through an unauthorized MAGNET Server.

Information in tuples is the responsibility of the user therefore security issues are addressed at the user level. There are two mechanisms which facilitate the protection of tuples in the information pool. Firstly, they can encapsulate a 'signature' parameter in their tuples, therefore matching with non-signed tuples will not be achieved. Secondly, user-defined matching functions can also incorporate a protection policy.

4 Example: The Mobile Office

In this section, we demonstrate the role of MAGNET on the problem of a typical mobile user requiring dynamic adaptation to different local environments (a base office, a train, and another branch of the company).

4.1 Problem Description

Users, such as travelling salesmen, work on portable computers and often travel between their base office and other branches of the company. They may also use the portable while commuting on a train. Furthermore, in other branches, they can give a presentation, or attend a seminar, both using the same portable.

Currently, the user has to perform manual adaptation to the local computing environment. That is, having arrived to an office, the user has to find out information about local resources (printers, file systems, etc.) then reconfigure and reboot the portable. Also the QoS requirements need adjusted according to characteristics of the available resources. Futher, the user needs to carry out extra functions, e.g., to check the e-mail, he has to login remotely into his base office mail server, he cannot just use mail without explicitly stating where the mail is stored. Ideally, the user wants to take the portable, connect it into the new environment and continue working without reconfigurating, rebooting or running special applications (e.g., for network, printers setup, etc). The next few sections describe our solution to this.

4.2 MAGNET support for Dynamic Resource Reconfiguration

In this section, we illustrate how MAGNET ensures transparent adaptation to diverse local environments saving the user from manual reconfiguration. The example environments illustrate the functionality of a portable operating in the base office, on a train, and in a remote office. Further, we discuss these environments in terms of two resources; printers and file systems.

For simplicity, we assume a command, PRINT, which sends a job into a print queue; and a command, MAIL, which enables the reading of a personal mail-box. (In this example we focus on remote sharing of the mail-box, and omit sending e-mail, which involves other issues, such as support for buffering messages written while the portable was disconnected, connecting to local mail servers etc.) At this level, we assume security and protection checks are not an issue. For lucidity, illustrating figures presented below, are only schematic and expose only the relevant resources which we discuss.

4.2.1 In a home office

When the user connects the portable to the network, the MAGNET Server running on the portable establishes a communication with other MAGNET Servers enabling access to local resources, (for example, a printer, or local file system into which the portable file system is mounted). Performing a PRINT command sends the job into the local printer queue as predefined by the user printer matching function. Correspondingly, the MAIL command initializes the mail matching function and locates the mail-box in the file system enabling it to be accessed. Figure 2 illustrates the base office configuration.

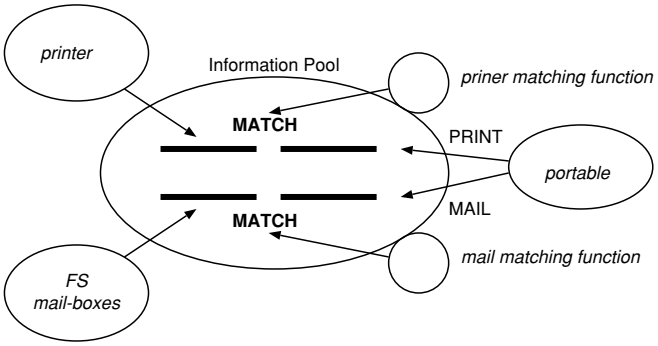

Figure 2: The MAGNET configuration in the base office

4.2.2 On a train

On a train, the portable is disconnected from the information pool and relies on the local resources only (e.g. local file system). When a PRINT or MAIL command is issued the print and mail matching functions detect that requested resources cannot be used. That is, tuples matching the request cannot be found in the local information pool. Consequently, the commands are rejected when a timeout is set by the user-defined print or mail matching functions expire. Figure 3 represents the situation.

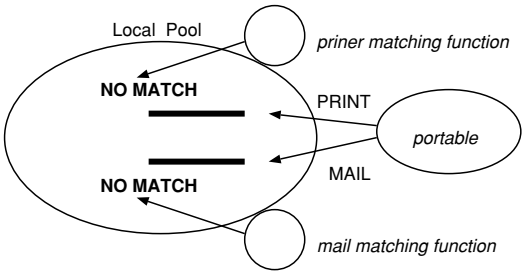

Figure 3: The MAGNET configuration on a train

4.2.3 Remote branch

Finally, in another branch of the company, the user connects the portable to the local network. The initialization process (connecting into the information pool) is performed in the same manner as the base office. Only, in this example, the

user-defined matching function invoked by the PRINT command matches the request with a colour ink-jet printer available there, instead of an ordinary printer. This process demonstrates adjusting QoS requirements according to the local resource availability. That is, a range of accepted requirements in the matching function is incorporated (a black&white printer is considered acceptable, but the colour ink-jet printer is prefered). A MAIL command invokes the matching function requesting the user's base office mail-box. To enable the mail matching function to find the requested tuple (the base office file system containing the user's mail box) the remote MAGNET Server communicates with the base office MAGNET Server. Once the mail-box is located, transparent access via the network is enabled. (Service infrastructure support enabling the transparent access is not predefined by MAGNET). Figure 4 illustrates this case.

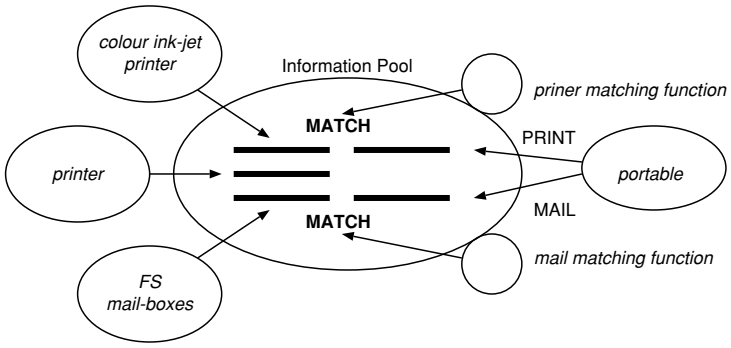

Figure 4: The MAGNET configuration in remote branch

4.3 Other issues

To implement the examples discussed in previous sections MAGNET's resource reconfigurator must employ flexible lazy binding. This is, it only requests resources at the moment they are actually required. Which also improves efficiency and flexibility of the matching process.

Furthermore, matching functions can be defined so that the QoS corresponds with the more disconnected environment, (e.g., an automated battery saving regime is activated, when the matching function discovers the absence of a mains power supply, etc.)

The issue of locality is explicitly expressed in the matching functions (e.g., the function matches a local printer, instead of a remote one). Furthemore, the added advantage of the pool is that the local pool is searched first, therefore implicitly the local resource is chosen.

Problems concerning the infrastructural communicational support for adaptability (support for IP address migration, compatibility of file systems and network protocols, etc.) are beyond the scope of MAGNET, and in this example we assume they are not an issue. We aim to address this issue in our further research.

5 Related Work

The construction of systems using components, and the use of flexible communications mechanisms, are two areas well-explored by contemporary research. Nevertheless, systems which provide the functionality required by mobile applications to allow dynamic reconfiguration and adaptability, have not been widely investigated.

Linda was the first system to support a generative communication model [8], providing several important features (see section 3), but its fixed tuple format and semantics do not provide the flexibility we require. A question-based system OSPREY [5], which was motivated by Linda, implemented application-server coupling using tuple-based interaction. It added a level of flexibility through utilizing a result-based tuple naming scheme and by replicating tuples over many nodes, but it did not address issues concerning user-defined matching.

Blair et.al. [2] investigated the tuplespace approach to QoS support in a mobile environment. It extends the traditional tuplespace with QoS management providing support for monitoring and adaptation for applications using heterogeneous networking environments. Unlike MAGNET, its emphasis is on QoS monitoring and adaptation to changes in network connectivity, and not user-defined QoS-based resource allocation and configuration.

Support for non-stop dynamic data object communication has been explored in "The Information Bus Architecture" [6], based on principles such as self-describing, anonymous communications and minimal semantic communication protocols. Nevertheless, this approach restricts data access to subject-based addressing, unlike MAGNET which permits both white-pages and yellow-pages naming.

The problem of dynamic adaptation to a change in environment has been successfully addressed by the Personal Computer Memory Card International Association. PCMCIA ethernet cards can be add and removed from the system without powering-off or rebooting the computer. The Linux *kernel daemon* is another successful attempt, enabling operating system kernel adaptation by adding or removing modules transparently on demand [10]. Incidently, both these dynamic adaptation approaches can be implemented using MAGNET (they are additional examples of dynamic resource reconfiguration).

The MAGNET approach has been adopted in a component based operating system called BITS exploring the construction of an entire operating system from independent components [4]. MAGNET is a dynamic resource manager in BITS which underlies a universal dynamically configurable computing environment [7].

6 Project Status and Future Work

We are adopting a combined approach to protection of pooled data. Currently, data can be protected by 'signatures' or by user-defined matching functions. We are investigating how MAGNET Servers can restrict or disable data sharing with untrusted local pools according to a user-defined policy.

MAGNET is part of PhD research, and is currently being implemented in the Regis [9] distributed programming environment.

7 Conclusion

This paper has examined the problem of adjusting the QoS where dynamic configuration is necessary in response to changes of user location, and computing environment. Through examples, we have shown that mobile users are the major category which benefit the most from such dynamic reconfigurable systems.

The paper has provided a flexible solution based on a current information exchange model, MAGNET. This has illustrated that a distributed information pool can provide dynamic matching of resource requests to service offers as a function of the user-defined QoS-based preferences and the current environment configuration. An automatically adapting portable computer allows the mobile user to concentrate on the task at hand without having to manually adapt the computer to frequent changes in the computing environment.

References

[1] N. Davies, G. S. Blair, K. Cheverst and A. Friday. *Supporting Adaptive Services in a Heterogeneous Mobile Environment.* In Proceedings of the 1st Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 8-9, 1994.

[2] G. S. Blair, N. Davies, A. Friday and S. P. Wade. *Quality of service support in mobile environment: an approach based on tuple spaces.* In Proceedings of the 5th IFIP International Workshop on Quality of Service, New York, USA, May 1997.

[3] G. H. Forman and J. Zahorjan. *The Challenges of Mobile Computing.* IEEE Computer, 27(4), pages 38-47, April 1994.

[4] P. Kostkova, K. Murray, T. Wilkinson. *Component Based Operating System.* 2nd Symposium on Operating Systems Design and Implementation, WIP session, Seattle, USA, October 1996.

[5] D. Bolton, D. Gilbert, K. Murray, P. Osmon, A. Whitcroft, T. Wilkinson, N. Williams. *A Question based approach to Open systems: OSPREY.* Internal TR, SARC, City University, London. March 1993.

[6] B. Oki, M. Pfluegl, A. Siegel, D. Skeen. *The Information Bus.* ACM SIGOPS European Workshop (SIGOPS '93), December 1993.

[7] P. Kostkova, T. Wilkinson. MAGNET*: A Virtual Shared Tuplespace Resource Manager.* In Proceedings of the 31st Hawaii International Conference on System Science, Hawaii, USA, January 1998.

[8] D. Gelernter. *Generative Communication in Linda.* ACM Transactions on Programming Languages and Systems, 7(1), pages 80-112, January 1985.

[9] J. Magee, N. Dulay, J. Kramer. *A Constructive Development Environment for Distributed Programs.* IEE/IOP/BCS Distributed Systems Engineering, 1(5), pages 304-312, September 1994.

[10] H. Storner. Linux kerneld mini-HOWTO. Electronic document available at http://www.image.dk/~storner/kerneld-mini-HOWTO.html, version 1.7, July 19, 1997.

[11] A.S. Tanenbaum. *Distributed Operating Systems*. Prentice Hall, Inc., 1995.

[12] A.P.M. Ltd. *The ANSA Reference Manual Release 01.00*. APM Cambridge Limited, UK, March 1989.

[13] N. Davies, S. Pink and G. Blair. *Services to Support Distributed Applications in a Mobile Environment*. 1st International Workshop on Services in Distributed and Networked Environments, Prague, Czech Republic, 1994.

[14] J.A. McCann, J.S. Crane. *Component DBMS Architecture for Nomadic Computing*. Submitted to BNCOD '98, Springer-Verlag, 1998.