Challenges and research directions in autonomic communications

Kevin Curran*, Maurice Mulvenna and Chris Nugent

Faculty of Engineering, University of Ulster, Northern Ireland, UK
E-mail: kj.curran@ulster.ac.uk E-mail: md.mulvenna@ulster.ac.uk
*Corresponding author

Alex Galis

Department of Electronic and Electrical Engineering, University College London, Torrington Place, London WC1E 7JE, UK
E-mail: a.galis@ee.ucl.ac.uk

Abstract: The increasing amount of traditional network services may still not fulfil the requirements of ever-demanding applications and must therefore be enriched by some form of increased intelligence in the network. This is where the promise of autonomous systems comes into play. Autonomous systems are capable of performing activities by taking into account the local environment and adapting to it. No planning is required; therefore autonomous systems must optimise the use of the resources at hand. This paper clearly identifies the need for autonomous systems in networking research, anticipated architectures, characteristics and properties, the path of evolution from traditional network elements and the future of such systems.

Keywords: autonomous systems; autonomic communications; autonomous network architectures; service-aware networks.

Reference to this paper should be made as follows: Curran, K., Mulvenna, M., Nugent C. and Galis, A. (2007) 'Challenges and research directions in autonomic communications', *Int. J. Internet Protocol Technology*, Vol. 2, No. 1, pp.3–17.

Biographical notes: Kevin Curran BSc (Hons), DPhil, SMIEEE, MBCS CITP, MACM, MAIAA, MIEE, ILTM is a Lecturer in Computer Science at the University of Ulster. He has published over 170 publications to date in the field of distributed computing, especially, emerging trends within wireless ad-hoc networks, dynamic protocol stacks and middleware. He is a member of the Editorial Committee of the *Journal of Universal Computer Science (J.UCS)*.

Maurice Mulvenna is a Senior Lecturer in Computer Science at the University of Ulster's School of Computing and Mathematics. He has a research career spanning over 17 years in Academia and Industry. He is principal investigator and grant holder in research grants worth around $6M in the areas of context-aware computing and artificial intelligence. He serves on the international programme committees of conferences including *International Conference on Smart Homes & Health Telematics 2006*, *IEEE Pervasive Computing 2006*, *IEEE-ACM Web Intelligence (2004–2006)* and *ACM Workshop on Continuous Archival & Retrieval of Personal Experiences (2005–2006)*. He is a member of the BCS, IEEE and ACM.

Chris Nugent is a Senior Lecturer at the University of Ulster, in Jordanstown. He is an active researcher and a member of numerous journal and conference committees. His research interests include artificial intelligence; medical informatics; multivariate data analysis and neural networks. Within these areas he has published over 120 papers in journals, books and conferences and has recently co-edited a text in Personal Health Management Systems.

Alex Galis is a Visiting Professor in the Networks and Services Research Group of the Department of Electronic and Electrical Engineering, University College London. He was Principal Investigator of the MISA and FAIN projects and leading the UCL activities in the MANTRIP, WINMAN, HARP, CONTEXT, E-NEXT, M-CDN, AMBIENT NETWORKS projects – part of the European IST research programme. He has published four books *Fast and Efficient Context-Aware Services* (John Wiley and Sons, 2006), *Programmable Networks for IP Service Deployment* (Artech House, 2004), *Deploying and Managing IP over WDM Networks* (Artech House, 2003), *Multi-Domain Communication Management* (CRC Press, July 2000), and over 100 publications in the field of networks and services and distributed systems.

1 Introduction

On 22 May 1973, a young man called Bob Metcalf authored a memo that described 'X-Wire', a 3 Mbps common bus office network system developed at Xerox's Palo Alto Research Center (PARC). It was also called *the* Alto Aloha Network protocol, but it became more commonly know as *Ethernet*. There are few networking technologies from the early 1970s that have proved to be so resilient. Metcalf deservedly went on to everlasting fame in the networking community (Metcalf, 1995) and also founded the global company 3COM, and his story could have ended here. However, in 1995, he predicted that the internet would self-destruct from overload (Metcalf, 1995). His argument was that network load (messages, packets) are growing exponentially, while network bandwidth (fibre capacity, switch performance) is growing linearly and at some point, these two curves cross and the result being that demand will exceed capacity. This prediction has not entirely come true. The anecdote is used here not to contradict a great researcher, but rather to serve as a warning to those of us who think that we know how a complex entity such as the internet is about to behave in the near or distant future.

Modern networks offer end-to-end connectivity. However, the increasing amount of traditional, offered services may still not fulfil the requirements of ever demanding distributed applications and must therefore be enriched by some form of increased intelligence in the network. This is where the promise of autonomous systems comes into play. Autonomous systems are capable of performing activities by taking into account the local environment and adapting to it. No planning is necessarily required; therefore autonomous systems simply have to make the best of the resources at hand. Locality in this scenario is no longer geographical, but rather the information and applications depend on the boundary of the autonomous communicating element, which may be distributed over a wide area.

One of the main drivers indeed behind autonomous systems is that the industry is finding that the cost of technology is decreasing; yet Information Technology (IT) costs are not (i.e., *IT gap*). Also, as systems become more advanced, they tend to become more complex and increasingly difficult to maintain. To complicate matters further, there has been, and for the foreseeable future, will be, a scarcity of IT professionals to install, configure, optimise and maintain these complex systems.

One other important driver behind autonomous system is the increasing network operational costs and the cost of introduction of new services (i.e., *Service Gap*). The rapid deployment of IT infrastructure technology and hardware, and its accelerating increase in performance (each generation improves by a factor of 1.5–2 per annum), gives rise to new challenging problems. On the one hand, the availability of computation performance and network bandwidth at a reasonable cost stimulates demand for products with more functionality and the demand for increasingly powerful services. On the other hand, developers cannot keep up with the demand for software, resulting in an ever-increasing Service Gap: the time lag and high cost of adapting/engineering end-system software, and the diversion of highly qualified staff to 'administrative' functions.

The main reason for this Service Gap, and the mismatch between technological potential and its realisation as software and services, is to be found in the separation between computational and communication resources and in the lack of focus on complexity. Complexity is currently the main factor preventing highly qualified workers from being productive, since many of them are occupied with system administration, configuration and maintenance, resulting in costs but no return of investment. At the same time, complexity limits developers' productivity, since it increases project timescales, and encourages specialisation and inflexibility. It is widely agreed that the high costs of the initial project stages and its associated administrative overheads, inhibit the future growth of IT and its advance into new areas. If software and services are to progress at a faster pace than computational performance and bandwidth, which is the only way to narrow the Service Gap, managing complexity becomes the key issue. In part, because of the Service Gap, these deficiencies remain to be addressed, and their solution using traditional technology is not getting nearer.

Therefore, the aim of autonomic systems is to reduce the amount of maintenance and management needed to keep systems working as efficiently as possible, as much of the time as possible. That is, it is about making systems self-managing for a broad range of activities. Trends in network design, which support the need for more 'open networks', include the increasing popularity of component architectures that reduce development time and offer freedom with choice of components. This allows alternative functionality to be deployed in various scenarios to combat differing Quality of Service (QoS) needs. Another trend is introspection, which provides run-time system information allowing applications to examine their environment and act accordingly. The middleware provides an infrastructure for building adaptive applications that can deal with drastic environment changes. The internet with its multiple standards and interconnection of components such as decoders, middleware, databases, etc., deserves more than a plug, try and play mentality. The introduction of mobility increases the complexity, owing to the proliferation in possible actions. A key goal for the next generation internet is to provide a principled means of allowing the underlying infrastructure to be adapted throughout its lifetime with the minimum of effort, thus the principles of autonomic computing provides a means of coping with change in a computing system, as it allows access to the implementation in a principled manner. This paper is an attempt to more clearly identify the need for autonomous systems, their architecture, the path of evolution from traditional network elements and the future of such systems.

2 The nature of the beast

The internet is comprised of close to a billion daily users, each of whom can potentially communicate. Hosts can be anything from desktop computers and WWW servers, to non-traditional computing devices, such as mobile phones, surveillance cameras and web TV. The distinction between mobile phones and Personal Digital Assistants (PDAs) has already become blurred with pervasive computing being the term coined to describe the tendency to integrate computing and communication into everyday life. New technologies for connecting devices like wireless communication and high bandwidth networks make the network connections even more heterogeneous. Additionally, the network topology is no longer static, owing to the increasing mobility of users. Ubiquitous computing is a term often associated with this type of networking (Schmidt et al., 2002b; Tanter, 2002). Thus a flexible framework is necessary in order to support such heterogeneous end-systems and network environments.

The internet is built on the DARPA protocol suite Transmission Control Protocol/Internet Protocol (TCP/IP), with IP as the enabling infrastructure for higher-level protocols such as TCP and the User Datagram Protocol (UDP). The IP is the basic protocol of the internet that enables the delivery of individual packets from one host to another. It makes no guarantees about whether or not the packet will be delivered, how long it will take, or if multiple packets will arrive in the order they were sent. Protocols built on top of this add the notions of connection and reliability. One reason for IP's tremendous success is its simplicity. The fundamental design principle for IP was derived from the 'end-to-end argument', which puts 'smarts' in the ends of the network – *the source and destination network hosts* – leaving the network 'core' dumb. IP routers at intersections throughout the network need do little more than check the destination IP address against a forwarding table to determine the 'next hop' for an IP datagram (where a datagram is the fundamental unit of information passed across the internet). However, the protocols underlying the internet were not designed for the latest generations of networks, especially those with low bandwidth, high error losses and roaming users, thus many 'fixes' have arisen to solve the problem of efficient data delivery (Saber and Mirenkov, 2003).

Mobility requires adaptability, meaning that systems must be location-aware and situation-aware taking advantage of this information in order to dynamically reconfigure in a distributed fashion (Solon et al., 2005). However, situations in which a user moves an end-device and uses information services can be challenging. In these situations, the placement of different cooperating parts is a research challenge. The heterogeneity is not only static, but also dynamic as software capabilities, resource availability and resource requirements may change over time. The support system of a nomadic user must distribute, in an appropriate way, the current session among the end-user system, network elements and application servers. In addition, when the execution environment changes in an essential and persistent way, it may be beneficial to reconfigure the cooperating parts. The redistribution or relocation as such is technically quite straightforward, but not trivial. On the contrary, the set of rules that the detection of essential and persistent changes is based on, and indeed the management of these rules is a challenging research issue, which to date has not been solved by the traditional 'smarts in the network' approach.

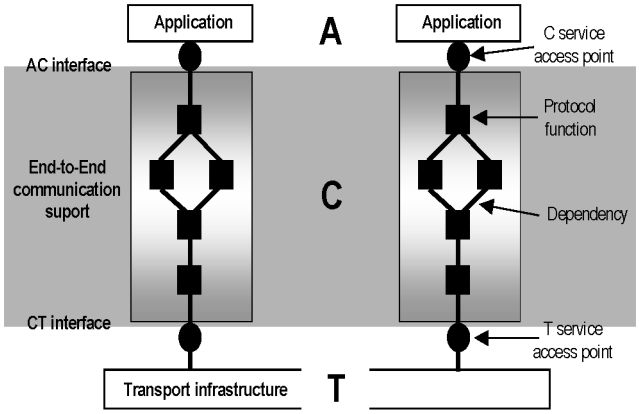
3 The traditional approach

A bare bones traditional communication system can be seen as consisting of three layers, as illustrated in Figure 1. End systems inter-communicate through layer T, the transport infrastructure. The service of layer T is a generic service corresponding to layer two, three or four services in the OSI Reference Model. In layer C, the end-to-end communication support adds functionality to the services in layer T. This allows the provision of services at the layer A for distributed applications (A–C Interface). Layer C is decomposed into protocol functions, which encapsulate typical protocol tasks such as error and flow control, encryption and decryption, presentation coding and decoding among others. A protocol graph is an abstract protocol specification, where independence between protocol functions is expressed in the protocol graph. If multiple T services can be used, there is one protocol graph for each T service to realise a layer C service. Protocol functions (modules) can be accomplished in multiple ways, by different protocol mechanisms, as software or hardware solutions with each protocol configuration in a protocol graph being instantiated by one of its modules (Plagemann et al., 1995).

Layering is a form of information hiding where a lower layer presents only a service interface to an upper layer, hiding the details of how it provides the service. A traditional network element such as the above could form part of the architecture of an adaptable middleware. Here the flexible protocol system could allow the dynamic selection, configuration and reconfiguration of protocol modules to dynamically shape the functionality of a protocol in order to satisfy application requirements or adapt to changing service properties of the underlying network. Some uses that these dynamic stacks may be used for could include increasing throughput where environmental conditions are analysed and heuristics applied to decide if change would bring about optimal performance. Many such dynamically reconfigurable conventional middleware systems exist (Becker et al., 2003; Blair et al., 2001; Ledoux, 1999), which enable systems to adapt their behaviour at runtime to different environments and application requirements. The resource restrictions on mobile devices prohibit the application of a fully-fledged middleware system, therefore one traditional approach is to restrict existing systems and provide only a functional subset (Object Management Group, 2002), which leads to different programming models or a subset of available interoperability protocols. Another option is to structure the middleware in multiple components, such that unnecessary

functionality can be excluded from the middleware dynamically. One such example is the Universally Interoperable Core UIC (Roman et al., 2001), which is based on a micro-kernel that can be dynamically extended to interact with various middleware solutions, but the protocol is determined prior to communication and dynamic reconfiguration is not possible. However, even in the case of most existing dynamically reconfigurable middleware, which concentrate on powerful reconfiguration interfaces – the domain that they are applied in is simply too narrow (e.g., Multimedia Streaming). It seems that *future proofing* for future uses is not built in. It must be noted that the authors are not claiming that this is trivial, rather an alternative approach for handling change in complex networks seems called for.

Figure 1 Three layer model



Source: Plagemann et al. (1995)

4 The need for 'open networks'

The internet at present is limited as it attempts to cater for the masses. Its rich end system functionality leads to high management overhead with much manual configuration, diagnosis and design (Clark et al., 2003). The next generation internet must obviously be 'backward' compatible, but it should seek to possess the ability to 'know' what it is being asked to do. It should also have a high-level view of its design goals and the constraints on which configurations are acceptable.

In other words, we need to separate the functional requirements of the network (what it does) from the non-functional requirements (how it does it). The goal is to overcome the limitations of the black box approach to software engineering, and to open up key aspects of the network infrastructure. This must, however, be achieved in such a way that there should be a principled division between the functionality they provide and the underlying implementation. The former can be thought of as the base interface of a module and the latter as a meta-interface. This allows a principled means of achieving an 'open' network. In an open network, the benefits are that it can bring modifications or extensions to itself by virtue of its own computation. It can 'think about itself', thus giving the possibility to enhance adaptability and to better control the applications that are based on top of it. By possessing the ability to 'think', we mean that significant benefits can be achieved in terms of monitoring key events (inspection), adapting components to changing circumstances (adaptation), and reconfiguring systems to meet new requirements (extension) (Blair, 1998).

For this, two different levels are needed: a base-level related to the functional aspects, i.e., the code concerned with computations about the underlying network, and a meta-level handling the non-functional aspects, i.e., the code supervising the execution of the functional code for each node (Villazón, 2000). An open network therefore naturally supports inspection, and adaptation for all applications residing on top. The inspection process takes place in the middleware; more specifically, it is realised by means of Meta-Object Protocols (MOP). Meta-objects within the Middleware framework are able to inspect the behaviour of an underlying application object known as base-objects, by viewing them as an abstract process. A meta-object is also able to control the behaviour of its base-objects by implementing crucial strategy issues such as fault tolerance or security. A MOP defines the set of meta-objects, as well as the interactions between them, thus representing the way applications are executed (Truyen et al., 2005). Therefore applications are the ability to observe the occurrence of arbitrary events in the underlying network, and ultimately allow each application to adapt the internal behaviour of the system, either by changing the behaviour of an existing service (e.g., tuning the implementation of message passing to operate more optimally over a wireless link), or dynamically reconfiguring the system (e.g., inserting a filter object to reduce the bandwidth requirements of a communications stream). Such steps are often the result of changes detected during inspection.

An open network as proposed here is similar to the MOP defined by Maes (1987). The concept of open implementation has been investigated by a number of researchers, most notably (Kiczales et al., 1997; Coady and Kiczales, 2003). In networking, where applications can adapt the end-to-end path to particular requirements using code within intermediate proxies, reflection is an interesting mechanism that can be exploited to dynamically integrate non-functional code to an active network service. An increasing number of algorithms used in classical network models or classical distributed systems have been adapted to take into account benefits of reflective code such as Active Multicast (Lehman and Tennenhouse, 1998) and Adaptive Routing (Curran and Parr, 2004). Thus, the flexibility of the active model is exploited, but on the other hand, the complexity of software design is increased. As a result, the composition of active components becomes difficult and service designers integrate in the code aspects that are not directly related to the core functionality of each component (Villazón, 2000). For example, tracing the behaviour of active packets and examining how they interact with various execution environments is a

non-functional aspect that crosscuts the original design of the component and is often integrated in several parts of the software. The slotting-in of such code, in most cases implies halting the component, integrating the changes, recompiling and redeploying the new component. This leads to a need for a clean solution for structuring services in order to separate those orthogonal aspects. The original problem, for instance with tracing packets above, is that relevant trace code is spread in several parts of the component, implying modification of the component in many places. Tracing can be seen as non-functional (or orthogonal, i.e., that which is other than core functionality of the component); therefore the non-functional code should be delegated to the meta-objects and the core-code to the base objects. This allows each component to be implemented without taking non-functional aspects into account, thus creating components with cleaner and more manageable code (Villazón, 2000).

5 The need for service-aware networks

The aim of service-aware networking to improve the capability of a network infrastructure (Bastide et al., 2000; Beddus et al., 2000; Suzuki and Suda, 2004; Veytser, 2004; Biswas et al., 2000) in order to better support services (in terms of resource management, content distribution, flexibility, programmability, protocol adaptation, reusability, scalability, reliability, security, etc.) is an always timely objective. A first attempt to solve the problem by using the principle of separation of concerns was already included in the OSI model and involved placing service support functions in the top three layers (session layer, presentation layer, application layer). However, the market chose for the simplicity of TCP and against OSI. Choosing a rather primitive version of network infrastructure has in some sense reopened the area of service support middleware. CORBA (Bastide et al., 2000; CORBA Components, 2005; Vinoski, 2000) was a major step in this direction as was Java Remote Method Invocation (Java RMI) (Franti and Stal, 1998). Java RMI enables a programmer to create distributed Java-based to Java-based applications and services, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. DCOM (Franti and Stal, 1998) is Microsoft's product for distributed component support. Java-based network execution environments provide the mechanisms for network deployment and management of service components (Galis et al., 2004).

In the same direction, 'service architecture' is a concept, which mainly tries to exploit modular design and reusability, but complex service architectures, like TINA (TINA-C, 2005) have in practice met lukewarm reception, probably owing to their complexity and the lack of timely production of support tools. The Parlay API enables both third parties and network operators to build new applications that rely on real-time control of network resources. The 'Java APIs for integrated networks' (JAIN) Community has defined a Java version of the Parlay API to bring the benefits of the Java language to the Parlay API (Beddus et al., 2000).

One approach towards service-aware networks is based on Overlays (Ratnasamy et al., 2001; Ratnasamy, 2002, Rowstron and Drushel, 2001a; Ritter et al., 2005; Subramanian et al., 2004; Braynard et al., 2002; Abraham et al., 2003; Andersen et al., 2001). Overlay networks strive for a clear border (Castro et al., 2002; Bassi et al., 2002; Clarke et al., 2003; Gao and Steenkiste, 2004; Ghosh et al., 2000), below which, any trick is permitted to provide the overlay abstraction, and above which, applications should not have to care about the underlying 'real' network infrastructure. In overlay networks, a set of nodes (servers, end-user equipment, etc.) and virtual links, not directly related to the underlying network topology, are involved in specific applications. The overlay traffic traverses through the overlay nodes and virtual links. Therefore, an overlay network acts as a specialised middleware. Two elements have to be considered:

- techniques to efficiently map the overlay abstraction to the underlying resources
- the management of the overlay, i.e., the mapping policy, configuration and reconfiguration, performance monitoring, etc.

The beauty of overlays is that they can be customised for a single service or a service family, thus creating a variety of overlays. One way to create multiple overlays is by inheritance, i.e., network instances are generated from a parent virtual network by inheriting signalling protocols and communication services. This method is used in Genesis (Campbell et al., 1999).

The separation between services (applications) and network infrastructure results in some loss of interaction (and control) between the two layers; the active network technology offers some remedy to this inefficiency. Active and programmable networks (Wetherall et al, 1998; Galis et al., 2004; Biswas et al., 1998, 2000; Brunner and Stadler, 2002; Calvert et al., 1998; Kirstein et al., 2002) can be used in conjunction with overlays for the benefit of applications. Active networks are 'active' in two ways: routers and switches within the network can perform computations on user data flowing through them; and users can 'program' the network, by supplying their own programs to perform these computations. Active and programmable networks (Calvert etal., 1998; Galis et al., 2004; Raz and Shavitt, 2000; Smith et al., 1999; Moore et al., 2001; Schmidt et al., 2002a; Tsarouchis et al., 2003; Vicente, 2001; Vivero et al., 2002; Wang and Touch, 2002; Wetherall et al., 1998; Yang et al., 2003) can rely on active packets (i.e., carrying executable code) or active nodes (the code resides in the nodes, but it is initiated by commands in the packets), or both. The pre-1999 situation is exposed in detail in Montz's report (Montz et al., 1995), in which network management (Tullmann et al., 2001), quality of service (Yan and Mabo, 2004), reliable multicasting, web caching, congestion control and security have already been identified as applications, which can benefit from the active network

approach. Since then, quite a few active network approaches and related products have appeared. The Darwin (Chandra et al., 2001) and FAIN (Galis et al., 2000) projects rely on code that is sent by applications or service providers to network nodes to implement customised management of their data flows. Darwin code is executed on designated routers and can affect resource management through a control Application Programming Interface (API) (Montz et al., 1995; Chandra et al., 2001). X-Bone (Smith et al., 1999) discovers, configures, and monitors network resources to create overlays over existing IP networks (Stoica et al., 2001a). Anetd (2001) is software that supports the configuration and operation of the Active Networks Backbone (Berson, 2002). ANON (Schmidt et al., 2002a) is more suitable for overlays than Anetd, with which it has certain similarities, but it relies on existing products, i.e., on Lynx for web access, on PGP for encryption and on the UNIX toolbox. Network management has always been an important application for active networks (Eaves, 2002). ABLE (2000) is an active engine, which can be used to allow fast and easy deployment of distributed network management applications in IP networks (NetBSD, 2000). In the itmBench (Veytser, 2004), traffic control applications could be developed on one node or across an overlay of nodes, either using a kernel API so they run in kernel space, or using a user-space API so they run in user space. Active node scalability and the related API are presented in Wang et al. (2002). The active network API of CANES (Bhattacharjee et al., 2002) is used for composing complex services out of components.

An application area that has recently received strong attention in the areas of service engineering and ambient intelligence is context awareness (Salber et al., 1999; Dey, 2001; Dey and Abowd, 2000; Chen and Kotz, 2000; Chen et al., 2003; Korpipää et al., 2003; Serrat et al., 2004; Yang and Galis, 2003; Yang et al., 2003; Xynogalas et al., 2004). Context-aware service provision over an active network infrastructure is a new area (Yang et al., 2003, Pascoe et al., 1999). Programmable networks are not only capable of providing additional control to the service layer, but they can also provide network-related context.

A large number of peer-to-peer overlay network designs (de Meer and Tutschku, 2002; de Meer et al., 2003; Schollmeier, 2001; Stoica et al., 2001a, 2001b; Yang et al., 2002; Singla and Rohrs, 2002; Braynard et al., 2002) have been proposed recently, such as CAN (Abraham et al., 2003), Chord (Stoica et al., 2001a), Freenet (Clarke et al., 2000), Gnutella (Gnutella, 2001; Ly et al., 2002; Klingberg and Manfredi, 2002; Anonymous, 2001; Harvey, 2004; Stokes, 2003), Pastry (Rowstron and Druschel, 2001b), Salad, Tapestry, Viceroy, SkipNet (Ghosh et al., 2000). The main feature of peer-to-peer networks is that they enable service components deployment (e.g., content services and CDN – Content Distribution Networks) in a flexible, scalable and decentralised way. A key function that these networks enable is a distributed hash table, which allows data to be uniformly diffused over all the participants in the peer-to-peer system. The basic approach in systems like Chord and Pastry is to diffuse content randomly throughout an overlay in order to obtain uniform, load-balanced, peer-to-peer behaviour. In Chord, Pastry and Tapestry the expected number of hops between any two communicating nodes scales with log N. In SkipNet, data is uniformly distributed across a well-defined subset of the nodes in a system, such as all nodes in a single building. Overlays have also appeared over grids (Andersen et al., 2001; Keahey et al., 2002; Korpela et al., 2001). Keahey et al. (2002) presents a dynamic overlay of active network routers to accomplish scalable time management in a grid environment, while Anderson (Andersen et al., 2001) deals with topology awareness.

Related work in the area of generic interfaces, protocols and frameworks that facilitate the creation of generic overlay networks and peer-to-peer applications, has developed the Opus overlay utility (Braynard et al., 2002), and the JXTA technology (SUN Microsystems, 2005). Opus is a large-scale overlay utility service that provides a common platform and the necessary abstractions for simultaneously hosting multiple distributed applications. Opus allocates available nodes to meet the requirements of competing applications based on dynamically changing system characteristics. Opus offers QoS characteristics by using Service Level Agreements to dynamically allocate utility resources among competing applications.

Introduced by Sun Microsystems, JXTA technology is a set of open, generalised peer-to-peer protocols that allows any connected device (cell phone to PDA, PC to server) on the network to communicate and collaborate. The main requirement based on which JXTA was developed is interoperability among devices and networks. Both technologies, although they define a generic methodology for creating a variety of overlay applications do not provide or utilise information regarding the status of the underlying network.

Issues to be explored in the service-aware networks include service discovery (Gao and Steenkiste, 2004), self-management and other autonomic behaviour (Suzuki and Suda, 2004), resource optimisation, quality of service (Abraham et al., 2003; Subramanian et al., 2004; Yan and Mabo, 2004) survivability (even in the presence of malicious adversaries (Abraham et al., 2003), service roaming and service federation (Machiraju et al., 2003), Quality of Context (QoC) (Buchholz et al., 2003) and service composition (Ocampo et al., 2005).

6 The autonomic communications solution

The first main aim of autonomous computing and communication systems is that they manage complexity, possess self-knowledge, continuously tune themselves, adapt to unpredictable conditions, prevent and recover from failures, and provide a safe environment (Murch, 2004; ACF, 2005).

- The autonomic nervous system frees our conscious mind from self-management and is the fundamental point of autonomic computing, thus 'freeing' up system administrators and normal users from the details of system operation and maintenance. If a program can deal with these aspects during normal operation, it is a lot closer to providing users with a machine that runs 24×7 and has optimal performance. The autonomic system will change anything necessary to keep running at optimum performance, in the face of changing workloads, demands and any other external conditions it faces. It should be able to cope with software and or hardware failures whether they are because of an unforeseen incident or malicious acts.
- Installing and configuring systems can be extremely time consuming, complex and can be open to human error, no matter how qualified the administrator is. Autonomic systems could configure themselves automatically by incorporating new components seamlessly.
- Modern systems may contain large amounts of different variables/options/parameters, which a user is asked to change to optimise performance. Few people, however, know how to use these and even fewer know how to modify them to attain 100% performance. An autonomic system could continually monitor and seek ways of improving the operation and efficiency of the systems in terms of both performance and/or cost. It is faster at this than a person and is able to dedicate more time to finding ways of improving performance.
- Autonomic systems are designed to be self-protecting, able to detect hostile or intrusive acts as they occur and deal autonomously with them in real time. They can take actions to make themselves less vulnerable to unauthorised access. Self-protected systems will anticipate problems based on constant readings taken on the system, as well as being able to actively watch out for detailed warnings of attacks from internet sources. They will take steps from such reports to avoid or mitigate them (Murch, 2004).

The characteristics stated above, all come together to help a system run more efficiently while reducing costs owing to less human input.

Autonomic computing

The IBM Autonomic Computing Toolkit[1] enables developers to add self-configuring and other autonomic capabilities to their software systems. The Autonomic Computing Toolkit is a collection of technologies, tools, scenarios, and documentation that is designed for users wanting to learn, adapt, and develop autonomic behaviour in their products and systems. Microsoft aims to develop self-healing, autonomic computing under its Visual Studio product line and presently claim to be in the process of software releases designed to reduce data centre complexity.

Autonomic communications

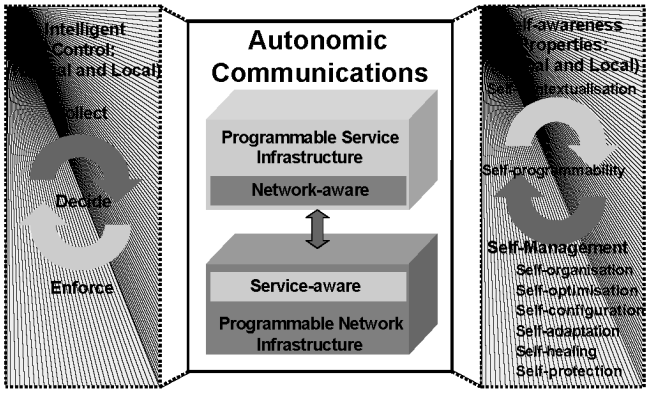
The key aim of autonomous communication systems is that they exhibit self-awareness properties, in particular self-contextualisation, self-programmability and self-management (i.e., self-optimisation; -organisation; -configuration; -adaptation; -healing; and -protection) as depicted in Figure 2.

- *Self-contextualisation.* According to (Dey and Abowd, 2000; Dey et al., 1999; Dey, 2001), context is any information that can be used to characterise the situation of an entity (a person or object) that is considered relevant to the interaction between a user and an application. A context-aware system (Bucholtz et al., 2003; Chen and Kotz, 2000; CoolTown, 2004; Fang and McDonald, 2002; Gray and Salber, 2001; Kanter, 2002; Chen et al., 2003) is capable of using context information, ensuring it successfully performs its expected role, and also maximises the perceived benefits of its use. Nevertheless, this is a user-centric view and reflects the fact that most research on context and context-awareness up to now has been focused on 'user context'. In contrast, a new generation network gives context a much broader scope and renders it universally accessible as a basic commodity provided and used by the network. In this way context becomes a decisive factor in the success of future autonomous rule-based systems adaptive to changing conditions. As such, contextualisation is a service/software property (Crowcroft et al., 2003; Brown et al., 1997; FIPA, 2002; Gray and Salber, 2001; Kanter et al., 2000; Kindberg and Barton, 2002; Kitamura et al., 2001). Self-contextualisation is the ability of a system to describe, use and adapt its behaviours to its context; meanwhile, it does not have to be aware of any other form of context knowledge. However, a context-aware system is a system that acts based on knowledge of a certain context. Network context for supporting service/software components should be made available, so that multiple service/software components may take advantage of the available network context. In order to do so in the complex environment of the large and heterogeneous internet, the service/software component must be equipped with certain self-management capabilities. Once a service/software component becomes context-aware (Serrat et al., 2004; Yang et al., 2003; Yau and Karim, 2003, 2004; Salber et al., 1999; Korpipää et al., 2003; Mendes et al., 2003; Perkins, 2002; Samann and Karmouch, 2003), it can make use of context information for other self-management tasks that depend on context information.
- *Self-programmability.* Recent research on distributed systems and network technologies has focused on making service networks programmable. The objectives of programmable service networks are to take advantage of network processing resources and to promote new service models allowing new business

models to be supported (Galis et al., 2003, 2004; Gelas and Lefevre, 2002; Leslie et al., 2000; Vicente et al., 2000). The resulting service models do not, however, target development of services, but, rather, their deployment. Dynamic service programming applies to executable service code that is injected into the autonomic system's elements to create the new functionality at run-time. The basic idea is to enable third parties (users, operators, and service providers) to inject application-specific services into the Autonomic System's platform. Applications and services are thus able to utilise required network support in terms of optimised network resources, and as such, they can be said to be network-aware, i.e., a service-driven network. Self-programmability means that network programmability is following autonomous flows of control, triggered and moderated by network events or changes in network context. The network is self-organised in the sense that it autonomically monitors available context in the network, and provides the required context and any other necessary network service support to the requested services, and self-adapts when context changes.

- *Self-management.* Management is an essential topic when dealing with utilisation of network context information for supporting services (Brunner et al., 2001; Galis et al., 2004; Fonseca et al., 2001, Goldszmidt and Yemini, 1995; Schwartz et al., 1999; Sloman and Lupu, 1999; Sloman, 1994; Yemini et al., 2000; de Vergara, 2003) and contextualised services (DeVaul et al., 2001; Gray and Salber, 2001; Yang and Galis, 2003; Yang et al., 2003). Managing context from the perspectives of the context information provider means dealing with a number of processes related to manipulation of network context information. For instance, the creation, composition and inference of context of diverse quality; also QoC-based storage, distribution and caching are relevant. Clearly, a context source must be trustworthy and the information it provides must be sufficiently precise for the task in hand, and, in this way, the new concept of QoC becomes important.

Figure 2 Autonomic systems: self-properties and intelligent control loop



A set of eight capabilities can be considered to characterise an autonomous system. Among these autonomous capabilities, self-optimisation, self-organisation, self-configuration and self-adaptation are highly relevant. Moreover, the management of network context information must be addressed in the framework of the autonomous computing paradigm. This means that a key element for contextualisation is the addition of intelligence and self-management capabilities to facilitate network context self-management and thus eliminating unnecessary multi-level configurations as in conventional hierarchical management systems. With such embedded intelligence, it is only necessary to write or specify the high-level design goals and management constraints, so that the network and service overlay should make the low-level decisions on its own. The system should reconfigure itself according to changes in the high-level requirements (i.e., use of cognitive and knowledge network principles). This requires the ability to express rules within each configuration level and also between levels. Currently, network management faces many challenges: complexity, data volume, data comprehension, changing rules, reactive monitoring, resource availability, and others. Self-management aims to automatically perform these tasks, including the following:

- *Self-optimisation.* In the large and heterogeneous internet, heterogeneous and distributed network context information and resources and their availability are rapidly changing. There is a need for an autonomous tool for consistent monitoring and control of network context information and resources, so that service/software components may be executed or deployed in the most optimised fashion. Autonomic systems must seek to improve their operation every time. They must identify opportunities to make themselves more efficient from the point of view of strategic policies (performance, cost, etc.).
- *Self-organisation.* Network elements and context information and resources are distributed across heterogeneous networks. In order for services to make use of this distributed information and resources, they must be structured or referenced in an easy-to-access-and-retrieve structure in an automatic fashion. All these network context information and resources must be autonomously organised and reserved through a service layer. The autonomous structuring of network context information and resources is the essential work of self-organisation.
- *Self-configuration/self-adaptation.* It enables autonomous structuring of network context information and resources, making them available to services. User services and the underlying supporting services must be reconfigured in order to make use of new network context information and resources. The new network context information and resources also trigger changes such as reconfiguration in the network context-aware overlay. Self-configuration is therefore desirable. Autonomic Systems aim at developing and assessing a

novel open programmable infrastructure for enabling self-configuration. Autonomic Systems must configure themselves in accordance with high-level policies representing service agreements or business objectives, rules and events. When a component or a service is introduced, the system will incorporate it seamlessly, and the rest of the system will adapt to its presence. In the case of components, they will register themselves and other components will be able to use it or modify their behaviour to fit the new situation.

- *Self-healing.* Autonomic Systems will detect, diagnose and repair problems caused by network or system failures. Using knowledge about the system configuration, a problem-diagnosis embedded intelligence would analyse the monitored information. Then, the network would use its diagnosis to identify and enforce solutions or alert a human in the case of no solutions being available.
- *Self-protection.* There are two ways in which an Autonomic System must self-protect. It must defend itself as a whole by reacting to, or anticipating, large-scale correlated problems arising from malicious attacks or cascading failures that remain uncorrected by self-healing measures.

The realisation of self-awareness properties revolves around increasing the level of automation of the intelligent control loop (Figure 2) described as *Collect-Decide-Enforce*.

- 'Collect' is about monitoring that allows constructing a picture about the surrounding environment in order to build self-awareness.
- 'Decide' involves inference and planning. The former refers to a process whereby the problem is diagnosed based on the collected information while the latter refers to the process whereby a solution is selected.
- 'Enforce' comprises deployment, which adds functionality by means of new components, and configuration, which changes the existing functionality by means of programmability of networks.

The realisation of this intelligent loop eventually leads to a distributed, adaptive, global evolvable system capable of fostering continuous changes as depicted in Figure 2.

7 Building upon nature

It is proposed here that Autonomic Network Elements should singularly and collectively utilise the current situation of their context (i.e., become situation aware). Context here might be a mobile user (e.g., Doctor) requiring a certain quality of service over a GPRS network (perhaps urgently) or a network, which is fighting off a distributed denial of service attack. Therefore, to become aware of the situation, in which each node finds itself, there must be some form of global knowledge available to 'one and all'. Knowledge should only reside where it is useful, however, knowledge may be useful in multiple locations, therefore, the rapid distribution of knowledge to 'trouble spots' is an important issue in an autonomic network.

Autonomic communication elements (aka nodes) should ideally exhibit what we term *triple behaviour*. They should be Alert, Aware and Autonomic. That is to say that they should be spontaneous/Ready for Action (*Alert*) (situationally aware of the current situation (*Aware*) and, of course, functionally independent (*Autonomic*).

We believe that the coordination activities among nodes can occur via stigmergic mechanisms (Holland, 1996). The concept of stigmergy was introduced by Grassé (1959) in the 1950s to describe the indirect communication taking place among individuals in social insect societies. Grasse showed that the regulation and coordination of the building activity of termite nests do not depend on the workers themselves, but is mainly achieved by the nest: a stimulating configuration triggers a response of a termite worker, transforming the configuration into another configuration that may trigger in turn another, possibly different, action performed by the same termite or any other worker in the colony. The most important feature to understand is how local stimuli are organised in space and time to ensure the emergence of a coherent adaptive structure, and to explain how workers could act independently, yet respond to stimuli provided through the common medium of the environment of the colony (Theraulaz and Bonabeau, 1999).

The stigmergic approach is more lightweight than the 'Knowledge Plane' approach (Clark et al., 2003). The 'Knowledge Plane' is considered as an additional network layer between the network and the application layer, and it is the place in which nearly all network control activities take place. The knowledge plane is populated by heavyweight agents, managing and exchanging knowledge about the current state of the network, and that directly enact forms of control over both network and application components. Using a stigmergic approach allows individual nodes to both handle and manage knowledge without the necessity of a global network plane. An autonomic knowledge network in contrast to an overlay network in peer-to-peer environments (Ratsanamy et al., 2002; Rowstron and Druschel, 2001a) does not simply transport data and messages, but rather exists to support nodes with a situationally aware contextual intelligent update, in order to adapt in the best way possible.

A similar method is Swarm Intelligence (SI) (Bonabeau and Theraulaz, 2000). SI is the property of a system whereby the collective behaviours of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge. SI provides a basis with which it is possible to explore collective (or distributed) problem solving without centralised control or the provision of a global model. Thus global robust adaptive self-organising behaviour can be made to emerge in systems of a large number of lightweight agents that indirectly interact, via the mediation of an environment agent (Parunak et al., 2004), by depositing and

by sensing 'pheromones'. A new type of cognitive stigmergy thus arises and this can build on the intelligent network of knowledge, thus leading to a more informed method of self-organisation.

This stigmergic lightweight network memory could be a machine-understandable XML-based syntax, comprising different standards that maintain high semantic integrity and coherence for the data and knowledge such as the Predictive Modelling Mark-up Language (PMML) (Grossman et al., 1999). PMML is an XML-based standard developed by the Data Mining Group[2] with the aim of aiding model exchange between different model producers and between model producers and consumers. PMML provides the first standard representation that is adhered to by all the major data-mining vendors. The use of PMML within the network context effectively decouples the self-adaptive engine from the producer of the knowledge that it uses.

Any such memory would in essence be a collection of rule sets that can maintain network policies, as well as behavioural descriptions and policies, and meet the *triple behaviour* as previously discussed. Through introspection and mediation, each node can self-adapt to improve performance depending on the context and needs of use. In order to execute the behavioural knowledge, a scalable high-performance engine is required. This is similar in construct to a recommender engine, in that it is constantly updating the rule bases based upon the application of predictive algorithms on network behavioural data. A key component is the detection by this engine of network trends and subtle changes in data flows, for example (Black and Hickey, 2003).

8 Conclusion

The key aim of autonomous communication systems is that they exhibit self-awareness properties, in particular self-contextualisation, self-programmability and self-management. Autonomous systems are capable of performing activities by taking into account the local environment and adapting to it. We believe that a promising avenue of research for coordination activities among nodes can occur via methods that build upon techniques found in nature, for example, stigmergic mechanisms. Further, the concept of autonomic network knowledge can leverage the traditional concept of stigmergy into a concept of 'cognitive' stigmergy where activities can be driven, not simply by reacting to a local concentration of meaningless pheromones, but can be driven by the actual knowledge represented by the network of knowledge. Through introspection and mediation, each node can self-adapt to improve performance depending on the context and needs of use. Thus, through a lightweight autonomic network model, more informed semantic forms of self-organisation could be reached by autonomic network elements collectively utilising the current situation of their context with knowledge residing where it is useful, often in multiple locations allowing rapid distribution of knowledge to 'trouble spots'.

These are possible research directions, worthy of exploration in the emerging new area of autonomic communications. While bio-inspired methods show much promise, it yet remains to be seen if key issues such as scalability, performance global-to-local coherence and other issues can be addressed realistically.

References

Introduction

Autonomic Communication Forum (ACF) (2005) http://www.autonomic-communication.org.

Becker, C., Schiele, G., Gubbels, H. and Rothermel, K. (2003) 'BASE – a micro-broker-based middleware for pervasive computing', *Proceedings of the IEEE International Conference on Pervasive Computing and Communication (PerCom)*, Fort Worth, USA, July, pp.122–132.

Black, M. and Hickey, R. (2003) 'Learning classification rules for telecom customer call data under concept drift', *Soft Computing*, Vol. 8, No. 2, pp.102–108.

Blair, G. (1998) 'The role of open implementation and reflection in supporting mobile applications', *Proc. IEEE Workshop on Mobility in Databases and Distributed Systems (MDDS'98)*, Vienna, August, IEEE, pp.42–48.

Blair, G.S., Coulson, G. and Andersen, A. (2001) 'The design and implementation of OpenORB version 2', *IEEE Distributed Systems Online Journal*, Vol. 2, No. 6, pp.45–52.

Bonabeau, E. and Theraulaz, G. (2000) 'Swarm smarts', *Scientific American*, March, pp.72–79.

Clark, D., Partridge, C., Ramming, C. and Wroclawski, J. (2003) 'A knowledge plane for the internet', *Proceedings of the 2003 ACM SIGCOMM Conference*, Karlsruhe (D), ACM Press, pp.3–10.

Coady, Y. and Kiczales, G. (2003) 'Back to the future: a retroactive study of aspect evolution in operating system code', *Proceedings of Aspect Oriented Systems Development AOSD*, Boston, Massachusetts, pp.138–146.

Curran, K. and Parr, G. (2004) 'Introducing IP domain flexible middleware stacks for multicast multimedia distribution in heterogeneous environments', *MATA 2004 – International Workshop on Mobility Aware Technologies and Applications*, Florianopolis, Brazil, 20–22 October, ISBN: 3-540-23423-3, p. 313, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, ISSN: 0302-9743.

Grassé, P-P. (1959) 'La Reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs',. *Insectes Sociaux*, Vol. 6, pp.41–81.

Grossman, R., Bailey, S., Ramu, A., Malhi, B., Cornelison, M., Hallstrom, P. and Qin, X. (1999) *The Management and Mining of Multiple Predictive Models Using the Predictive Modeling Markup Language (PMML)*, AFCEA 1999 Conference, San Diego, CA, USA.

Holland, O.E. (1996) *Multi-agent Systems: Lessons from Social Insects and Collective Robotics*, AAAI Spring.

Kiczales, G., Lamping, J., Videira Lopes, C., Mendhekar, A. and Murphy, G. (1997) *Open Implementation Design Guidelines, Proceedings of International Conference on Software Engineering*, Boston Ma, May, pp.87–96.

Ledoux, T. (1999) 'OpenCorba: a reflective open broker', *Proceedings of the 2nd International Conference on Reflection'99*, Saint Malo, France, pp.197–214.

Lehman, L. and Tennenhouse, D. (1998) 'Active reliable multicast', *IEEE INFOCOM '98*, San Francisco, March, pp.34–46.

Maes, P. (1987) 'Concepts and experiments in computational reflection', *OOPSLA'87*, pp.147–155.

Metcalf, B. (1995) 'Predicting the internet's catastrophic collapse and ghost sites galore in 1996', *InfoWorld*, December 4.

Murch, R. (2004) *Autonomic Computing*, IBM Press, Prentice Hall PTR, ISBN: 013144025X.

Object Management Group (OMG) (2002) *The Common Object Request Broker: Architecture and Specification*, Revision 3.0, July.

Parunak, V., Brueckner, S. and Sauter, J. (2004) 'Digital pheromones for coordination of unmanned vehicles', *Workshop on Environments for Multi-agent Systems (E4MAS)*, LNAI 3374, Springer-Verlag, New York, USA.

Plagemann, T., Saethre, K.A. and Goebel, V. (1995) 'Application requirements and QoS negotiation in multimedia systems', *Proceedings of Second Workshop on Protocols for Multimedia Systems*, Salzburg Austria, October, pp.123–132.

Ratsanamy, S., Karp, B., Estrin, D., Shenker, S. (2002) 'GHT: a geographic hash table for data-centric storage', *1st ACM Int.l Workshop on Wireless Sensor Networks and Applications*, Atlanta, Georgia, USA, September.

Roman, M., Kon, F. and Campbell, R. (2001) 'Reflective middleware: from your desk to your hand', *IEEE Distributed Systems online Journal*, Special issue on Reflective Middleware, July, Vol. 2, No. 5, pp.32–40.

Rowstron, A. and Druschel, P. (2001a) 'Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems', *18th IFIP/ACM Conference on Distributed Systems Platforms*, Heidelberg (D), November, pp.86–94.

Saber, M. and Mirenkov, N. (2003) 'A multimedia programming environment for cellular automata systems', *DMS'2003 – The 9th International Conference on Distributed Multimedia Systems*, Florida International University Miami, Florida, USA, September 24–26, pp.104–110.

Schmidt, D., Natarajan, B., Gokhale, A., Wang, N. and Gill, C. (2002a) 'Tao: a pattern-oriented object request broker for distributed real-time and embedded systems', *IEEE Distributed Systems Online*, Vol. 3, No. 2, February, pp.1–20.

Solon, A., Mc Kevitt, P. and Curran, K. (2005) 'TeleMorph: bandwidth determined mobile multimodal presentation', *Information Technology and Tourism*, February, ISSN: 1098-3058, Cognizant Publishers, USA, Vol. 7, No. 1, pp.33–47.

Tanter, E., Vernaillen, M. and Piquer, J. (2002) *Towards Transparent Adaptation of Migration Policies*, Position paper submitted to *EWMOS 2002*, Chile, pp.34–39.

Theraulaz, G. and Bonabeau, E. (1999) 'A brief history of stigmergy', *Artificial Life*, Vol. 5, No. 2, pp.97–116.

Villazón, A. (2000) 'A reflective active network node', *Proceedings of the Second International Working Conference on Active Networks (IWAN 2000)*, Tokyo, Japan, October, pp.120–132.

Service-aware networking

Bastide, R., Palanque, P., Sy, O. and Navarre, D. (2000) ''Formal specification of CORBA services: experience and lessons learned', ACM SIGPLAN Notices, *Proceedings of the 15th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, Vol. 35, No. 10, October, pp.92–104.

Beddus, S., Bruce, C. and Davis, S. (2000) 'Opening up networks with JAIN parlay', *IEEE Communications Magazine*, Vol. 38, No. 4, April, pp.136–143.

CORBA Components (2005) *Full Specification v3.0: Document – formal/02-06-65*, http://www.omg.org/technology/documents/corba_spec_catalog.htm.

Franti, P and Stal, M. (1998) 'An architectural view of distributed objects and components in CORBA, Java RMI and COM/DCOM', *Software-Concepts and Tools*, Springer-Verlag, Vol. 19, No. 1, June, pp.14–28.

Suzuki, J. and Suda, T. (2004) 'Middleware support for super distributed autonomic services in pervasive networks', *Proc. of the International Symposium on Applications and the Internet, SAINT 2004*, Tokio, Japan, 26–30 January, pp.375–381.

TINA-C (2005) http://www.tinac.com.

Truyen, E., Robben, B., Kenens, P., Matthijs, F., Michiels, S., Joosen, W. and Verbaeten, P. (2005) *Open Implementation of a Mobile Communication System*, Dept. of Computer Science – K.U.Leuven Technical Report; www.cs.kuleuven.ac.be/~eddy/mp/smove.html.

Vinoski, S. (2000) 'Introduction to CORBA', *Proceedings of the 22nd International Conference on Software Engineering*, Limerick, Ireland, pp.826–827.

Overlay and peer-to-peer networks

Abraham, I., Awerbuch, B., Azar, Y., Bartal, Y., Malkhi, D. and Pavlov, E. (2003) 'A generic scheme for building overlay networks in adversarial scenarios', *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, 22–26 April, Nice, France, pp.212–218.

Andersen, D.G., Balakrishnan, H., Kaashoek, M.F. and Morris, R. (2001) 'Resilient overlay networks', *Proc. 18th ACM SOSP*, Banff, Canada, October, pp.131–145.

Anonymous (2001) *Gnut: Console Gnutella Client for Linux and Windows*, http://www.gnutelliums.com/linux_unix/gnut/.

Bassi, A., Beck, M., Moore, T. and Plank, J.S. (2002) 'The logistical backbone: scalable infrastructure for global data grids', *Proceedings of Asian Computing Science Conference*, Hanoi, Vietnam, Lecture Notes in Computer Science, Vol. 2550/2002, Springer, December 4–6, pp.1–12.

Braynard, R., Kostic, D., Rodriguez, A., Chase, J. and Vahdat, A. (2002) 'Opus: an overlay peer utility service', *Proceedings of the 5th International Conference on Open Architectures and Network Programming (OPENARCH)*, June, New York, USA, pp.3–21.

Campbell, A.T., de Meer, H.G., Kounavis, M.E., Miki K., Vicente, J. and Villela, D.A. (1999) 'The genesis kernel: a virtual network operating system for spawning network architectures', *Proc. IEEE OPENARCH'99*, New York, March, pp.156–165.

Castro, M., Druschel, P., Kermarrec, A., Rowstron, A. (2002) *One Ring to Rule Them All: Service Discovery and Binding in Structured Peer-to-Peer Overlay Networks*', SIGOPS, France, September.

Clarke, O.I., Sandberg, B.W. and Hong, T.W. (2000) 'Freenet: a distributed anonymous information storage and retrieval system', *Workshop on Design Issues in Anonymity and Unobservability*, July, pp.311–320.

de Meer, H. and Tutschku, K. (2002) 'Dynamic operation in peer-to-peer overlays', Poster Presentation Supplement to the *Proceedings of Fourth Annual International Working Conference on Active Networks*, Zurich, Switzerland, December 4–6, pp.183–192.

de Meer, H., Tutschku, K. and Tran-Gia, P. (2003) 'Dynamic operation in peer-to-peer overlay networks', *Praxis der Informationsverarbeitung und Kommunikation, PIK Journal*, Special Issue on Peer-to-Peer Systems, June, pp.65–73.

Gao, J. and Steenkiste, P. (2004) 'Design and evaluation of a distributed scalable content discovery system', *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1, January, pp.54–56.

Ghosh, A., Fry, M. and Crowcroft, J. (2000) 'An architecture for application layer routing', *Active Networks*, May, pp.71–86.

Gnutella (2001) http://www.gnutelliums.com/.

Harvey, N.J.A. and Munro, J.I. (2004) 'Deterministic SkipNet', *Information Processing Letters*, Vol. 90, No. 4, May, pp.204–208.

Keahey, K., Fredian., T., Peng, Q., Schissel, D., Thompson, M., Foster, I., Greenwald, M., McCune, D. (2002) 'Computational grids in action: the national fusion collaboratory', *Future Generation Computer Systems*, Vol. 18, No. 8, October, pp.1005–1015.

Klingberg, T. and Manfredi, R. (2002) *The Gnutella Protocol Version 0.6 Draft*, Gnutella Developer Forum, http://groups.yahoo.com/group/the_gdf/files/Development/.

Korpela, E., Werthimer, D., Anderson, D., Cobb, J. and Lebofsky, M. (2001) 'SETI@home: massively distributed computing for SETI', *Computing in Science and Engineering*, Vol. 3, No. 1, January, pp.78–83.

Ly, Q., Ratnasamy, S. and Shenker, S. (2002) 'Can heterogeneity make gnutella scalable?', *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, March, pp.94–103.

Machiraju, V., Sahai, A. and van Moorsel, A. (2003) 'Web services management network: an overlay network for federated service management', *IFIP/IEEE Eighth International Symposium on Integrated Network Management*, 24–28 March, pp.351–364.

Ocampo, R., Cheng, L. and Galis, A. (2005) 'ContextWare support for network and service composition and self-adaptation', *IEEE MATA 2005 – Mobility Aware Technologies and Applications, – Service Delivery Platforms for Next Generation Networks*, Springer ISBN-2 553-01401-5, 17–19 October, Montreal, Canada, www.congresbcu.com/mata2005/.

Ratnasamy, S. (2002) *A Scalable Content-Addressable Network*, PhD Thesis, U.C. Berkeley, October.

Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S. (2001) 'A scalable content-addressable network', *Proceedings of ACM SIGCOMM*, San Diego, CA, September, pp.161–172.

Ritter, J. (2005) *Why Gnutella Can't Scale. No, Really*, http://www.darkridge.com/~jpr5/doc/gnutella.html.

Rowstron, A. and Druschel, P. (2001b) 'Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems', *International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, November, pp.329–350.

Schollmeier, R. (2001) 'A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications', *First International Conf. on Peer-to-Peer Computing (P2P2001)*, Linköping, Sweden.

Singla, A. and Rohrs, C. (2002) *Ultrapeers; Another Step Towards Gnutella Scalability*, Gnutella Developer Forum (http://groups.yahoo.com/group/the_gdf/files/Proposals/Ultrapeer/Ultrapeers_1.0_clean.html).

Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M., Balakrishnan, H. (2001a) 'Chord: a scalable peer-to-peer lookup service for internet applications', *ACM SIGCOMM '01*, San Diego, CA, September.

Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H. (2001b) 'Chord: a scalable peer-to-peer lookup service for internet applications', *Proceedings of ACM SIGCOMM*, August, San Diego, CA, USA, pp.149–160.

Stokes, M. (2003) *Gnutella2 Specification Document – First Draft*, Gnutella2 Web site (http://www.gnutella2.com/gnutella2_draft.htm).

Subramanian, L., Stoica, I., Balakrishnan, H. and Katz, R. (2004) 'OverQoS: an overlay based architecture for enhancing internet QoS', *Proc. 1st NSDI*, San Francisco, CA, March, pp.66–74.

SUN Microsystems (2005) *JXTA Technology*, http://www.sun.com/software/jxta/, March.

Yang, H., Meng, X. and Lu, S. (2002) 'Self-organized network layer security in mobile ad hoc networks', *ACM Workshop on Wireless Security (WiSe)*, Atlanta, USA, September, pp.11–20.

Service programmability and programmable networks

Anetd (2001) http://www.sdl.sri.com/projects/activate/anetd/.

Bhattacharjee, S., Calvert, K., Chae, Y., Merugu, S., Sanders, M. and Zegura, E. (2002) 'CANEs: an execution environment for composable services', *Proceedings of DARPA Active Networks Conference and Exposition*, San Francisco, pp.255–272.

Biswas, J. , Lazar, A., Huard, A. and Lim, K. (1998) 'The IEEE P1520 standards initiative for programmable network interfaces', *IEEE Communications, Special Issue on Programmable Networks*, Vol. 36, No. 10, October, pp.64–70, http://www.ieee-pin.org/.

Biswas, J., Vicente, J., Kounavis, M., Villela, D., Lerner, M., Yoshizawa, S. and Denazis, S. (2000) *Proposal for IP L-interface Architecture*, IEEE P1520.3, P1520/TS/IP013, http://www.ieee-pin.org/doc/draft_docs/IP/p1520tsip013.pdf.

Brunner, M. (2002) 'Tutorial on active networks and its management', *Journal Annals of Telecommunications*, Vol. 57, No. 6, pp.46–58.

Brunner, M. and Stadler, R. (2000) 'Service management in multi-party active networks', *IEEE Communications Magazine*, March, Vol. 38, No. 3, pp.32–40.

Calvert, K., Bhattacharjee, S. Zegura, E. and Sterbenz, J. (1998) 'Directions in active networks', *IEEE Communications Magazine*, Vol. 36, No. 10, October, pp.72–78.

Chandra, P., Chu, Y-h., Fisher, A., Gao, J., Kosak, C., Ng, T.S.E., Steenkiste, P., Takahashi, E. and Zhang, H. (2001) 'Darwin: customizable resource management for value-added network services', *IEEE Network*, Vol. 15, No. 1, January–February, pp.22–35.

Galis, A., Denazis, S., Brou, C. and Klein, C. (Ed.) (2004) *Programmable Networks for IP Service Deployment*, ISBN 1-58053-745-6, June, Artech House Books, p.450, www.artechhouse.com.

Galis, A., Gelas, J.P., Lefèvre, L. and Yang, Y. (2003) 'Programmable network approach to grid management and services', *International Conference on Computational Science*, LNCS 2658, ISBN 3-540-40195-4, Melbourne Australia, 2–4 June, pp.1103–1113, www.science.uva.nl/events/ICCS2003/.

Galis, A., Plattner, B., Smith, J.M., Denazis, S., Moeller, E., Guo, H., Klein, C., Serrat, J., Laarhuis, J., Karetsos, G.T. and Todd, C. (2000) 'A flexible IP active networks architecture', *Proceedings of International Workshop on Active Networks-Tokyo*, October, and in 'Active Networks', Springer-Verlag, ISBN 3-540-41179-8, pp.1–15.

Gelas, J-P. and Lefèvre, L. (2002) 'Toward the design of an active grid', *Lecture Notes in Computer Science*, Computational Science – ICCS 2002, Vol. 2230, April, p.578.

Kirstein, P., O'Hanlon, P., Carlberg, P. and Gevros, P. (2002) 'The radio active networking architecture', *dance*, *DARPA Active Networks Conference and Exposition (DANCE'02)*, San Francisco, CA, May, pp.394–402.

Leslie, I., McAuley, D., Black, R., Roscoe, T., Barham, P., Evers, D., Fairbairns, R., and Hyden, E. (2000) *The Design and Implementation of an Operating System to Support Distributed Multimedia Applications*, http://www.cl.cam.ac.uk/Research/SRG/netos/old-projects/nemesis/documentation.html.

Montz, A., Mosberger, D., O'Malley, S. and Peterson, L. (1995) *Scout: A Communications-Oriented Operating System*, IEEE HotOS Workshop, May, Orcas Island, WA, USA, May, pp.58–61.

Moore, J., Kornblum Moore, J. and Nettles, S. (2001) 'Scalable Distributed Management with Lightweight Active Packets',' *Technical Report MS-CIS-01-26*, Department of Computer and Information Science, University of Pennsylvania, September.

NetBSD (2000) http://www.netbsd.org.

Raz, D. and Shavitt, Y. (2000) 'Active networks for efficient distributed network management', *IEEE Communications Magazine*, Vol. 38, No. 3, March, pp.138–143.

Schmidt, S., Chart, T., Sifalakis, M. and Scott, A. C. (2002b) 'Flexible, dynamic, and scalable service composition for active routers', *Proc. Fourth Annual International Working Conference on Active Networks (IWAN 2002)*, Zürich, Switzerland, Lecture Notes in Computer Science 2546, Springer-Verlag, December, pp.253–266.

Smith, J., Calvert, K., Murphy, S., Orman, H. and Peterson, L. (1999) 'Activating networks: a progress report', *IEEE Computer*, Vol. 32, No. 4, April, pp.32–41, http://www.cs.princeton.edu/nsg/papers/an.ps.

Tsarouchis, C., Denazis, S., Kitahara, C., Vivero, J., Salamanca, E., Magana, E., Galis, A., Manas, J., Carlinet, L., Mathieu, B. and Koufopavlou, O. (2003) 'A policy-based management architecture for active and programmable networks', *IEEE Network*, Vol. 17, No. 3, May–June, pp.22–28.

Tullmann, P., Hibler, M. and Lepreau, J. (2001) 'Janos: a java-oriented OS for active networks', *IEEE Journal on Selected Areas of Communication*, Vol. 19, No. 3, March. http://www.research.att.com/~kobus/docs/tempest_small.ps.

Veytser, G., Bestavros, I., Zhang, M. and Chen, S. (2004) *itmBench: Generalized API for Internet Traffic Managers*, http://www.cs.bu.edu/groups/itm/itmBench/itmBench.pdf.

Vicente, J. (2001) *L-interface Building Block APIs*, IEEE P1520.3, P1520.3TSIP016, 2001. http://www.ieee-pin.org/doc/draft_docs/IP/P1520_3_TSIP-016.doc.

Vicente, J., Kounavis, M., Villela, D., Lerner, M., and Campbell, A. (2000) 'Programming internet quality of service', *3rd IFIP/GI International Conference of Trends toward a Universal Service Market*, Munich, Germany, September 12–14.

Vivero, J., Tan, A., Serrat, J., Salamanca, E., Galis, A., Kitahara, C., Tsarouchis, C. and Denazis, S. (2002) 'The FAIN management framework: a management approach for active network environments', *10th IEEE International Conference on Networks*, ICON 2002.

Wang, J-G., Li, Z-Z. and Kou, Y-N. (2002) 'Research and implementation of a scalable secure active network node', *Proc. of IEEE Intern. Conf. on Machine Learning and Cybernetics*, Vol. 1, November, pp.111–115.

Wang, Y-s. and Touch, J. (2002) 'Application deployment in virtual networks using the X-bone', *DANCE: DARPA Active Networks Conference and Exposition*, May, pp.484–493.

Wetherall, D., Guttag, J. and Tennehouse, D. (1998) 'ANTS: a toolkit for building and dynamically deploying network protocols', *Proc. of IEEE OPENARCH '98*, San Francisco, CA, April, pp.178–189.

Yan, B. and Mabo, R. (2004) 'QoS control for video and audio communication in conventional and active networks: approaches and comparison', *IEEE Communications Surveys and Tutorials*, http://www.comsoc.org/livepubs/surveys/public/2004/jan/bai.html.

Programmable system management

ABLE (2000) *The Active Bell Labs Engine*, http://www.cs.bell-labs.com/who/ABLE/.

Brunner, M., Plattner, B. and Stadler, R. (2001) 'Service creation and management in active telecom environments', *Communications of the ACM*, March.

de Vergara, J.E.L. (2003) *Specification of Network Management Information Models by jeans of Ontology and Knowledge Representation Techniques*, PhD Thesis, Spanish, Universidad Politecnica de Madrid, Madrid.

Fonseca, M., Agoulmine, N. and Cherkaoui, O. (2001) *Active Networks as a Flexible Approach to Deploy QoS Policy-Based Management*, HP-OVUA'01, Berlin, Germany, June, http://citeseer.nj.nec.com/483138.html.

Goldszmidt, G. and Yemini, Y. (1995) 'Distributed management by delegation', *Fifteenth International Conf. on Distributed Computing Systems*, Vancouver, June.

Schwartz, B., Jackson, A., Strayer, W., Zhou, W., Rockwell, D. and Partridge, C. (1999) 'Smart packets for active networks', *OpenArch '99*, March.

Sloman, M. (Ed.) (1994) *Network and Distributed Systems Management*, Addison-Wesley, Reading, MA.

Sloman, M. and Lupu, E. (1999) 'Policy specification for programmable networks', *International Working Conference on Active Networks (IWAN'99)*, Berlin, Germany, June–July.

Yemini, Y., Konstantinou, A. and Florissi, D. (2000) 'NESTOR: an architecture for NEtwork Self-managemenT and organization', *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 5, pp.758–766.

ContextWare networks

Berson, S., Braden, B., and Ricciulli, L. (2002) Introduction to the ABone, available at http://www.isi.edu/abone/DOCUMENTS /ABarch/February 2002.

Brown, P.J., Bovey, J.D. and Chen, X. (1997) 'Context-aware applications: from the laboratory to the marketplace', *IEEE Personal Communications*, Vol. 4, No. 5, October, pp.58–64.

Bucholtz, T., Kupper, A. and Schiffers, S. (2003) 'Quality of context information: What is it is and why we need it', *Proceedings of the 10th HP-OVUA Workshop*, Geneva, Switzerland, July, pp.112–120.

Chen, G. and Kotz, D. (2000) *A Survey of Context-aware Mobile Computing Research*, Technical Report, TR2000-381, Dept. of Computer Science, Dartmouth College, November.

Chen, H., Finin, T. and Joshi, A. (2003) 'An ontology for context-aware pervasive computing environments', *IJCAI Workshop on Ontologies and Distributed Systems*, IJCAI'03, August.

CoolTown (2004) home page. http://www.cooltown.hp.com/.

Crowcroft, J., Hand, S., Mortier, R., Roscoe, T. and Warfield, A. (2003) 'Plutarch: an argument for network pluralism', *ACM SIGCOMM 2003 Workshops*, August.

DeVaul, R.W. and Pentland, A.S. (2001) *The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications*, The Media Laboratory, MIT, Report available online at http://acg.media.mit.edu/ people/rich/.

Dey, A.K. (2001) 'Understanding and using context', *Journal of Personal and Ubiquitous Computing*, Vol. 5, No. 1, pp.4–7.

Dey, A.K. and Abowd, G.D. (2000) 'Towards a better understanding of context and context awareness, in Workshop on the What, Who, Where, When and How of Context-Awareness', affiliated with the 2000 *ACM Conference on Human Factors in Computer Systems (CHI 2000)*, The Hague, The Netherlands, April.

Dey, A.K., Salber, D., Abowd, G.D. and Futakawa, M. (1999) *An Architecture to Support Context-aware Applications*, GVU Technical Report Number:GIT-GVU-99-23.

Eaves, W., Cheng, L. and Galis, A. (2002) 'SNAP based resource control for active networks', *GLOBECOM - The World Converges Taipei*, Taiwan, ROC, November 17–21.

Fang, Y. and McDonald, A.B. (2002) 'Cross-layer performance effects of path coupling in wireless ad hoc networks: power and throughput implications of IEEE 802.11 MAC', *Proc. 21st IEEE International Performance, Computing, and Communications Conference*, April, Phoenix, Arizona, USA, pp.281–290.

Foundation for Intelligent Physical Agents (FIPA) (2002) *FIPA Quality of Service Ontology Specification*, Geneva, Switzerland, Specification number SC00094.

Gray, P.D. and Salber, D. (2001) 'Modelling and using sensed context information in the design of interactive applications', *Proc. 8th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI'01)*, May, Toronto, Canada, pp.317–328.

Kanter, T., Lindtorp, P., Olrog, C. and Maguire, G.Q. (2000) 'Smart delivery of multimedia content for wireless applications', *Mobile and Wireless Communication Networks*, pp.70–81.

Kanter, T.G. (2002) 'Hottown, enabling context-Aware and extensible mobile interactive spaces', Special Issue of *IEEE Wireless Communications and IEEE Pervasive on 'Context -Aware Pervasive Computing'*, October, pp.18–27.

Kindberg, T. and Barton, J. (2002) 'People, places, things: web presence for the real world', *Mobile Networks and Applications*, Springer Science, Vol. 7, No. 5, October, pp.365–376.

Kitamura, Y., Kasai, T. and Mizoguchi, R. (2001) 'Ontology-based description of functional design knowledge and its use in a functional way server', *Proceedings of the Pacific Conference on Intelligent Systems*, Seoul, Korea, pp.400–409.

Korpipää, P., Mäntyjärvi, J., Kela, J., Keränen, H. and Malm, E-J. (2003) 'Managing context information in mobile devices', *IEEE Pervasive Computing*, Vol. 2, No. 3, pp.42–51.

Mendes, P., Prehofer, C. and Wei, Q. (2003) 'Context management with programmable mobile networks', *IEEE Computer Communication Workshop*, Dana Point, CA, US, pp.132–140.

Pascoe, J., Ryan, N., and Morse, D. (1999) 'Issues in developing context-aware computing', *Proc. First International Symposium on Handheld and Ubiquitous Computing (HUC'99)*, Karlsruhe, Germany, pp.208–221.

Perkins, C. (Ed.) (2002) *IP Mobility Support for IPv4*, RFC3344, August.

Salber, D., Dey, A.K. and Abowd, G.D. (1999) 'The context toolkit: aiding the development of context-enabled applications', *Proc. CHI'99*, May, Pittsburgh, PA, pp.434–441.

Samann, N. and Karmouch, A. (2003) 'An evidence-based mobility prediction agent architecture', *Proc of the 5th Int. Workshop on Mobile Agents for Telecommunication Applications (MATA2003)*, Marrakesch, October, ISBN 3-540-20298-6 – Lecture Notes in Computer Science, Springer-Verlag.

Serrat, J., Serrano, J.M., Justo, J., Marín, R., Galis, A., Yang, K., Raz, D. and Sykas, E.D. (2004) *An Approach to Context Aware Services*, NOMS2004 19–23 April, Seoul, Korea.

Xynogalas, S.A., Chantzara, M.K., Sygkouna, I.C., Vrontis, S.P., Roussaki, I.G. and Anagnostou, M.E. (2004) 'Context management for the provision of adaptive services to roaming users', *IEEE Wireless Communications*, Vol. 11, No. 2, April, pp.40–47.

Yang, K. and Galis, A. (2003) 'Network-centric context-aware service over integrated WLAN and GPRS networks', *14th IEE International Symposium On Personal, Indoor And Mobile Radio Communications*, September, pp.854–858.

Yang, K., Galis, A., Serrat, J., Jean, K., Vardalachos, N. and Guo, X. (2003) 'Network-centric context-aware service over integrated WLAN and GPRS networks', *14th IEEE Conference on Personal, Indoor and Mobile Radio Communications*, PIMRC 2003, Vol. 1, 7–10 September, pp.854–858.

Yau, S.S. and Karim, F. (2003) 'A lightweight middleware protocol for ad hoc distributed object computing in ubiquitous computing environments', *Proc. 6th IEEE Intl. Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2003)*, May, pp.172–179.

Yau, S.S. and Karim, F. (2004) 'An adaptive middleware for context-sensitive communications for real-time applications in ubiquitous computing environments', *t Real-Time Systems, The International Journal of Time-Critical Computing Systems*, Kluwer Academic Publishers, Dordrecht, The Netherlands, Vol. 26, No. 1, January, pp.29–61.

Notes

[1]http://www-128.ibm.com/developerworks/autonomic/probdet1.html.

[2]http://www.dmg.org/.