pp. 1007-1018

Introducing a light-weight autonomic network middleware based on stigmergic mechanisms

Maurice MULVENNA*, Kevin CURRAN*,
Chris NUGENT*, Matthias BAUMGARTEN*

Abstract

Autonomous systems are capable of performing activities by taking into account the local environment and adapting to it. No planning is necessary therefore autonomous systems have to make the best of the resources at hand. Locality in this case is no longer geographical but rather the information and applications on the boundary of the autonomic communicating element which may be distributed over a wide area. The aim of autonomous communication systems is that they exhibit self-awareness properties, in particular self-contextualisation, self-programmability and self-management. In this paper, we identify the need for autonomous systems, their architecture, the path of evolution from traditional network elements, the need for open networks and future trends within autonomic computing.

Key words: Droit des télécommunications, Droit de la communication, Réglementation télécommunication, Droit européen, Droit Concurrence.

TITRE FRANÇAIS

Résumé

The sector-based model stemming from the regulatory framework for telecommunications, was extended to electronic communications, a term which takes into account the convergence of the telecommunications, broadcasting and IT sectors. At the same time, this model tends to dissolve in the general competition rules which are supposed to replace the sectoral regulation. Already, this regulation applies exclusively to operators having a significant market power on the relevant markets. But this competition model itself, is subordinated to general interest objectives, as universal service or cultural diversity. Thus, a new evolution is initiated, where the objectives of contents regulation could help to redefine some aspects of network regulation.

Mots clés : Telecommunication law, Communication law, Telecommunication regulation, European law, Competition law.

* Faculty of Engineering, University of Ulster, School of Computing and Mathematics – Newtownabbey, BT37 0QB, UK Northern Ireland, Email: {md.mulvenna, kj.curran, cd.nugent, m.baumgarten}@ulster.ac.uk

Sommaire

I. INTRODUCTION

Networks are becoming more complex due to the sheer number and variety of devices becoming attached; therefore they need to be enriched by some form of increased intelligence in the network. This is where the promise of autonomous systems comes into play. Paul Horn of IBM Research first suggested the idea of autonomic computing on 15 October 2001 at the Agenda conference in Arizona. The need centers around the exponential growth of networking complexity. Autonomous systems are capable of performing activities by taking into account the local environment and adapting to it. The most common definition of an autonomic computing system is one which can control the functioning of computer applications and systems without input from the user, in the same way that the autonomic nervous system regulates body systems without conscious input from the individual. Thus, we attempt here 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. One of the main drivers indeed behind autonomous computing is that Industry is finding that the cost of technology is decreasing, yet IT costs are not. In addition, 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, optimize and maintain these complex systems.

Autonomic computing systems will manage complexity, possess self knowledge, continuously tune themselves, adapt to unpredictable conditions, prevent and recover from failures and provide a safe environment [1]. 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 what runs 24x7 and its optimal performance. The autonomic system will change anything necessary so as to keep running at optimum performance, in the face of changing workloads, demands and any other external conditions it faces. Modern systems may also contain large amounts of different variables/options/parameters which a user is age to change to optimize performance. Few people however know how to use these and even fewer know how to get them exactly right to get 100% performance. An autonomic system could continually monitor and seek ways of improving the operation efficiency of the systems in 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. Also, 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 unauthorized access. Self-protected systems will anticipate pro-

blems based on constant reading 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 [1]. These characteristics stated above all come to together to help a system run more efficiently while reducing costs due to less human input [2].

For instance, in the scenario of a systems administrator having to move from server to server and from desktop to desktop installing software and generally correcting problems. Evolving autonomic computing technologies are attempting to relieve this burden from systems administrators by providing technologies and tools that enable applications, systems and entire networks to begin to manage themselves. By "manage themselves", we mean systems will have the ability to reconfigure, optimize, protect and repair themselves; however the administrator will not have to give up total control of the system. An autonomic system can help the administrators deal with software installation by being aware of what is needed to run and to install those components which need installing. It should obviously be also aware of what applications are installed on the system already and how to avoid or resolve any conflicts that would arise once installed. This type of system would constantly monitor itself for problems and should a problem arise then the fault is sought and corrected.

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 due 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, the need for open networks, the role of middleware and the future of such systems.

II. MIDDLEWARE

Messaging middleware is an inter-connection middle-tier solution that simplifies the technical challenges and effort required in sharing of information and processes. Messaging middleware architectures provide interfaces between applications, allowing them to send data back and forth to each other synchronously and/or asynchronously. This structure of middleware can be classified as a Messaging Oriented Middleware (MOM) [3]. Message-oriented middleware offers many advantages over other such middleware technologies such as Remote Procedural Call (RPC) including time independence of components, where the message sender and recipient do not have to be online at the same time, since MOM queues messages when their recipients are not available and location independence of components, which permits the ability to transfer either sender or receiver from one computer to another without bringing the system down due to its communication via topics (or channels).

The Common Object Request Broker Architecture (CORBA) is an object-oriented architecture, and a language-independent as well as vendor independent architecture based on RPC. The Object Management Group (OMG) is a consortium of hardware, software and end-user companies formed to create the CORBA architecture [4]. CORBA specifies the architecture of an

Object Request Broker (ORB) regulating interoperability between objects and applications. CORBA is used in the development of distributed applications that provides a standard mechanism for defining the interfaces between components. The Object Request Broker (ORB) is the software that manages communication between objects and is fundamental to the CORBA architecture. An ORB is a software component whose purpose is to facilitate communication between objects. It does so by providing a number of capabilities, one of which is to locate a remote object, given an object reference. A client application can be written in C++ while communicating with a server written in Java via the CORBA architecture. The CORBA specification does not address implementation details and leaves many areas undefined [5]. It is also fundamentally based on a blocked synchronous RPC model rather than an asynchronous Message-Oriented Middleware (MOM) model, which hinders the creation of real-time systems. The Distributed Component Object Model (DCOM) is a relatively robust object model that enjoys particularly good support on Microsoft operating systems because it is integrated with all versions of Windows since Windows 95. COM and DCOM are best considered as a single technology providing a range of services for component interaction, from services promoting component integration on a single platform, to component interaction across heterogeneous networks. COM and the DCOM extensions are merged into a single run time providing both local and remote access. ActiveX components are reusable and can be seen as the building blocks for providing the "glue" that connects components together based on COM and DCOM model. Java RMI is part of the Java Development Kit (JDK) which provides the core communication layer for any Java object wishing to communicate over an easy-to-use distributed network. RMI applications are often comprised of two separate programs: a server and a client. A typical server application creates some remote objects, makes references to make them accessible, and waits for clients to invoke methods on these remote objects (See Figure 1).

A typical client application gets a remote reference to one or more remote objects in the server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth. Java RMI runs atop of the JRMP protocol that is for the construction of pure Java networked systems. Java RMI uses a combination of Java serialisation and the Java Remote Method Protocol (JRMP) to turn standard method invocations into remote method invocations.

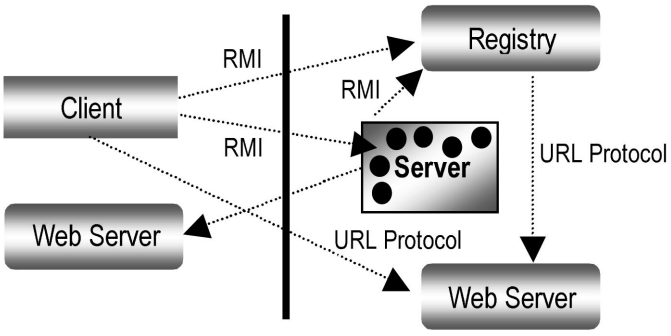

FIG 1. – RMI interconnection of the Client & Server.

Légende française

The middleware discussed here are based on the Remote Procedural Call (RPC) model which as the name suggests, amounts to simply calling procedures that live elsewhere. In general, a request is sent to a remote system to execute a designated procedure, using arguments supplied, and the result returned to the caller. Parameters can be established for the remote procedure and values can be returned from it. This provides a sort of poor man's application partitioning, letting one place parts of your application on other hosts or processors. A chief disadvantage of using RPC-based middleware protocol is that most RPCs execute synchronously therefore a call to an RPC will usually block application execution until the RPC returns. The main problem with the above conventional middleware systems (i.e. CORBA, RMI and DCOM) is that they have been developed to provide a homogeneous access to remote entities independent of operating systems. Typically, these middleware try to provide as much functionality as possible, which leads to very complex and resource consuming systems, which are not suitable for many networked devices. Approaches to solve this problem do exist but it must be remembered that conventional middleware was mainly designed for stable network environments, in which service unavailability is a rare event and can be treated as an error as opposed to autonomic systems which have been developed with such scenarios in mind.

III. THE HETEROGENEOUS NATURE OF MODERN NETWORKS

The distinction between mobile phones and personal device assistants (PDA's) 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, due to the increasing mobility of users. Ubiquitous computing is a term often associated with this type of networking [6]. 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 Internet Protocol is the basic protocol of the Internet that enables the unreliable 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 [7].

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 [8]. However, situations, in which a user moves an end-device and uses information services, can be challenging. In these situations the placement of different co-operating 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 co-operating 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.

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 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 analyzed and heuristics applied to decide if change would bring about optimal performance. Many such dynamically reconfigurable conventional middleware systems exist [9, 10, 11 and 12] which enable systems to adapt their behavior at runtime to different environments and applications requirements. The resource restrictions on mobile devices prohibit the application of a full-fledged middleware system therefore one traditional approach is to restrict existing systems and provide only a functional subset 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 [13] 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 that an alternative approach for handling change in complex networks seems called for.

IV. OPENING UP THE NETWORK

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 [14]. 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 [6]. This allows a principled means of achieving an "open" network. In an open network, the benefits that 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. 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 [15]. An open network therefore naturally supports inspection, and adaptation for all applications residing on top. Applications should be able to observe the occurrence of arbitrary events in the underlying network and ultimately allow each application to adapt the internal behavior of the system either by changing the behavior 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 autonomic network would 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. Flexible end-to-end protocols are configured to include only the necessary functionality required to satisfy an application QoS requirements for the particular connection. Some uses that dynamic stacks may be used for include increasing throughput where environmental conditions are analysed and heuristics are applied to decide if change would bring about optimal performance; interoperability in that dynamic stacks can simplify the interoperability process, by allowing code for protocol stacks to be written once and placed on repositories where they can be downloaded onto end systems so they can adopt the same stack; Security can be increased at run-time, for example, when an intrusion detection system dynamically responds to unusual behaviour and robustness, where faulty components can be detected and replaced to improve robustness (e.g. a mobile computer connected to an Ethernet LAN may automatically detect that its wired connection is broken forcing a switch to a Wireless LAN or GSM connection. Here it is profitable to dynamically load a new stack module optimised for the different characteristics of wireless connections).

Figure 2 depicts a typical layered protocol graph where a proxy is residing on the fixed network and a mobile device is receiving a multimedia stream. When a network device is switched from e.g., a low speed cellular network to a high speed wireless LAN, a session manager will detect that the underlying network has changed. The protocol stack control mechanism will install a new stack module, (e.g. which might contain an algorithm for conversion of a colour video stream to a monochrome video stream).

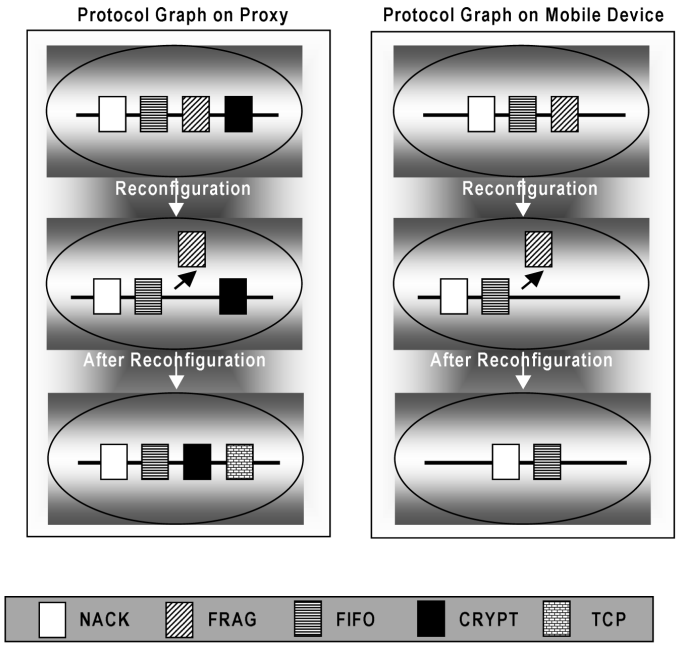

FIG 2. – Streaming Video Application Object Graph.

Légende française

An open network as proposed here is similar to the Meta-object Protocol (MOP) defined by Maes [16]. The concept open implementation has been investigated by a number of researchers, most notably at Xerox PARC [17, 18]. 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 [19] and Adaptive Routing [20]. Thus, the extreme flexibility of the active model is exploited, but on the other hand, the complexity of software design is increased. As a consequence, the composition of active services becomes very difficult and service designers integrate in the service code some aspects that are not directly related to the main functionality of the service itself. For example, tracing the activity of active packets and analyzing how they interact with the different execution environments is a non-functional aspect that cross-cuts the original design of the service and that is often integrated in several parts of the software. Furthermore, the insertion of such code, in most of cases implies stopping the service execution, integrating the modifications, recompiling and re-deploying the service over the network. This leads to a clean solution for structuring services in order to separate those orthogonal aspects. Dynamic integration of non-functional aspects is a notion that can be exploited in a context of adaptive media streaming. One of the major advantages is that the information obtained from the execution of a

service can be gathered and combined with similar information handed over several execution environments at the node level in order to enhance the overall service management, independently from the network level service code [15].

V. A LIGHT-WEIGHT AUTONOMIC NETWORK

We propose Autonomic Network Elements which singularly and collectively utilise the current situation of their context in order to become situation aware. Context here might be a mobile user (e.g. a person in distress) requiring a certain quality of service over a GPRS network (perhaps urgently) or a network which is fighting of a distributed denial of service attack. Therefore, to become aware of the situation 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 (or nodes) should ideally exhibit what we term *triple A* 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 functionally independent (*Autonomic*).

We believe that the coordination activities among nodes can occur via stigmergic mechanisms [21]. The concept of stigmergy was introduced by Pierre-Paul Grassé [22] in the 1950's to describe the indirect communication taking place among individuals in social insect societies. Grassé 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 organized 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 [23].

The stigmergic approach is more light weight than the "Knowledge Plane" approach [14]. 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 [24, 25] 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) [26]. Swarm Intelligence is the property of a system whereby the collective behaviors 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 centralized control or the provision of a global model. Thus global robust adaptive self-organizing behaviour can be made emerge in systems of a large number of lightweight agents that indirectly interact via the mediation of an environment agent [27] 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 light-weight 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) [28]. PMML is an XML-based standard developed by the Data Mining Group[1] 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 A* 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 for example is the detection by this engine of network trends and subtle changes in data flows [29].

In [30] the concept of a pulse monitor has be been explored, which is basically an extension of the fault tolerant heartbeat monitor mechanism to incorporate reflex urgency levels and health check summary information. In theory, this technique could be modified for an autonomic framework by implementing a "heartbeat" mechanism into any knowledge entity (ACE) such that at given intervals relevant health based information are send to all other entities and/or to a central monitoring facility. The same could be used vice versa diffusing behavioural or contextual information to relevant knowledge entities.

Finally, the area of game theory offers a number of concepts that could be used for different aspects of knowledge networks. Defined as *"… the study of the ways in which strategic interactions among rational players produce outcomes with respect to the preferences (or utilities) of those players, none of which might have been intended by any of them."* [31] combines three important concepts that are particular relevant for autonomous behaviour, namely: (a) strategic interactions, (b) preferences, (c) intentions. In context with autonomic computing game theory studies mechanism that enable decision making based on:

- the interaction among individual components (a), independent of the fact that they may have been intentionally or not;
- individual preferences that may be available at different levels of granularity (b);
- high level goals as set out by an underlying concepts (c).

The last concept is of particular interest because of the fact that, although any decision making process is driven by dedicated intentions its resulting decision is not necessarily limi-

1. http://www.dmg.org/

ted to those intentions. All of the concepts above are relevant when analysing behavioural patterns and [32] has stated that "*game theory is a universal language for the unification of the behavioral sciences*". While this may seems a bit extraordinary it is reasonable as game theory has been used successfully in countless games often attempting to replicate, predict and react on behavioural patterns.

VI. CONCLUSION

The aim of autonomic computing is to reduce the amount of maintenance needed to keep systems working as efficiently as possible, as much of the time as possible. 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 flexibility with increased choice of components. This allows alternative functionality to be deployed in various scenarios to combat differing 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. Autonomous systems are capable of performing activities by taking into account the local environment and adapting to it. We propose here a light-weight autonomic network model where more informed semantic forms of self-organisation can be reached. We believe that the coordination activities among nodes can occur via stigmergic mechanisms thus we introduce the concept of autonomic network knowledge which can leverage the traditional concept of 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.

Manuscrit reçu le
Accepté le

REFERENCES

[1] MURCH (R.), Autonomic Computing. *IBM Press*, Prentice Hall PTR, ISBN: 013144025x, 2004.

[2] KEPHART (J.), CHESS (D.), The Vision of Autonomic Computing, *IEEE Computer*, **36**, n° 1, pp. 41-50, January 2003.

[3] LOYALL (J.), GOSSETT (J.), GILL (C.), SCHANTZ (R.), ZINKY (J.), PAL (P.), SHAPIRO (R.), RODRIGUES (C.), ATIGHETCHI (M.), KARR (D.), Comparing and Contrasting Adaptive Middleware Support in Wide-Area and Embedded Distributed Object Applications. Proceedings *of 21st IEEE International Conference on Distributed Computing Systems (ICDCS-21*), April, Phoenix, AZ, 2001.

[4] OMG, Object Management Group. The Common Object Request Broker*: Architecture and Specification, Revision 3.0*, July 2002

[5] GOKHALE (A.), NATARAJAN (B.), CROSS (J.) AND SCHMIDT (D.), Towards Dependable Real-time CORBA Middleware, Cluster Computing: *the Journal on Networks, Software, and Applications Special Issue on Dependable Distributed Systems*, January 2002

[6] TANTER (E.), VERNAILLEN (M.), PIQUER (J.), Towards transparent adaptation of migration policies. *Proceedings of EWMOS 2002*, Chile, pp. 34-39, 2002.

[7] SABER (M.), MIRENKOV (N.), 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, 2003.

[8] SOLON (A.), MC KEVITT (P.), CURRAN, (K.), TeleMorph: Bandwidth determined Mobile MultiModal Presentation. *Information Technology and Tourism*, **7**, No. 1, pp. 33-47, February 2005, ISSN: 1098-3058, Cognizant Publishers, USA.

[9] CURRAN (K.), PARR (G.), 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 2004, ISBN: 3-540-23423-3, p. 313, Lecture Notes in Computer Science, Springer-Verlag Heidelberg, ISSN: 0302-9743.

[10] BLAIR (G.), COULSON (G.), ANDERSEN (A.), The design and implementation of OpenORB version 2, *IEEE Distributed Systems Online Journal*, **2**, n° 6, pp. 45-52, 2001.

[11] BECKER (C.), SCHIELE (G.), GUBBELS (H.) ROTHERMEL (K.), 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 2003.

[12] LEDOUX (T.), OpenCorba: A reflective Open Broker. *Proceedings of the 2nd International Conference on Reflection'99*, pp. 197-214, Saint Malo, France, 1999.

[13] ROMAN (M.), KON (F.), CAMPBELL (R.), Reflective Middleware: From your desk to your hand. *IEEE Distributed Systems online Journal. Special issue on Reflective Middleware*, July 2001.

[14] CLARK (D.), PARTRIDGE (C.), RAMMING (C.), WROCLAWSKI (J.), A Knowledge Plane for the Internet, *Proceedings of the 2003 ACM SIGCOMM Conference*, Karlsruhe, ACM Press, pp. 3-10, 2003.

[15] VILLAZÓN (A.), A Reflective Active Network Node. *Proceedings of the Second International Working Conference on Active Networks (IWAN 2000)*, pp. 120-132, Tokyo Japan, October 2000.

[16] MAES (P.), Concepts and experiments in computational reflection. *OOPSLA'87*, pp. 147-155, 1987.

[17] KICZALES (G.), LAMPING (J.), VIDEIRA (L.), MENDHEKAR (A.), MURPHY (G.), Open Implementation Design Guidelines, *Proceedings of International Conference on Software Engineering*, Boston Ma, May pp. 87-96, 1997.

[18] COADY (Y.), KICZALES (G.), Back to the Future: A Retroactive Study of Aspect Evolution in Operating System Code *In Proceedings of Aspect Oriented Systems Development AOSD 2003*, pp. 138-146, 2003.

[19] LEHMAN (L.), TENNENHOUSE (D.), Active reliable multicast. In *IEEE INFOCOM '98*, San Francisco, pp. 34-46, March 1998.

[20] FRY (G.), WEST (R.), Adaptive routing of QoS constrained media streams over scalable overlay topologies. *In 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, May 25-28, Le Royal Meridien, Toronto, Canada, 2004.

[21] HOLLAND (O.) (1996), Multi-agent systems: lessons from social insects and collective robotics, AAAI Spring Symposium on Adaptation, Coevolution and Learning in Multiagent Systems, March 25-27, Stanford, 1996

[22] GRASSÉ (P.), 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*, **6**, pp.41-81, 1959.

[23] THERAULAZ (G.), BONABEAU (E.), A Brief History of Stigmergy, *Artificial Life*, **5**, n° 2, pp. 97-116, 1999.

[24] RATSANAMY (S.), GHT: A Geographic Hash Table for Data-Centric Storage, *1st ACM Int.l Workshop on Wireless Sensor Networks and Applications*, Atlanta, Georgia, USA, September 2002.

[25] ROWSTRON (A.), DRUSCHEL (P.), 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 2001.

[26] BONABEAU (E.), THERAULAZ (G.), Swarm smarts. *Scientific American*, pp. 72-79, March 2000.

[27] PARUNAK (V.), BRUECKNER (S.), SAUTER (J.), Digital Pheromones for Coordination of Unmanned Vehicles. *Workshop on Environments for Multi-agent Systems (E4MAS)*, LNAI 3374, Springer Verlag, 2004.

[28] GROSSMAN (R.), BAILEY (S.), RAMU (A.), MALHI (B.), CORNELISON (M.), HALLSTROM (P.), QIN (X.), The Management and Mining of Multiple Predictive Models Using the Predictive Modeling Markup Language (PMML). *AFCEA 1999 Conference*, San Diego, CA, USA, 1999.

[29] BLACK (M.), HICKEY (R.), (2003). Learning classification rules for telecom customer call data under concept drift. *Soft Computing*, **8**, No. 2, pp: 102-108, 2003.

[30] STERRIT (R.), GUNNING (D.), MEBAN (A.), HENNING (P.), Exploring Autonomic Options in an Unified Fault Management Architecture through Reflex Reactions via Pulse Monitoring. *Proceedings of IEEE Workshop on the Engineering of Autonomic Systems (EASE 2004) at the 11th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2004)*, Brno, Czech Republic, 24-27 May, Pages 449-455, 2004.

[31] http://plato.stanford.edu/entries/game-theory/

[32] GINITS (H.), *Game Theory Evolving.* Princeton: Princeton University Press; ISBN: 0691009430, 2000.