View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Ulster University's Research Portal

Autonomic Computing

Kevin Curran
University of Ulster, Ireland

Maurice Mulvenna
University of Ulster, Ireland

Chris Nugent
University of Ulster, Ireland

Matthias Baumgarten
University of Ulster, Ireland

INTRODUCTION

Modern networks offer end-to-end connectivity however, the increasing amount of traditional offered services may still not fulfill 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. Paul Horn of IBM Research first suggested the idea of autonomic computing on October 15, 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. No planning is required hence 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 on the boundary of the autonomic communicating element, which may be distributed over a wide area. 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.

BACKGROUND

Autonomous systems are capable of performing activities by taking into account the local environment and adapting to it. No planning is required hence 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 on the boundary of the autonomic communicating element, which may be distributed over a wide area. 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, self-organisation, self-configuration, self-adaptation, self-healing, and self-protection). 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. 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 problems 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. These characteristics previously stated all come to together to help a system run more efficiently while reducing costs due to less human input. An autonomic system for instance can

Copyright © 2008, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.

help IT 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 also be 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, and so forth, deserves more than a plug, try, and play mentality. 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. We attempt here to more clearly identify the need for autonomous systems, the role of middleware, and the future of such systems.

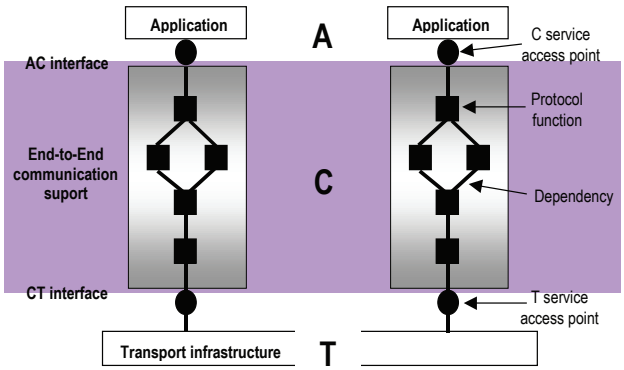
AUTONOMIC COMPUTING

The Internet is comprised of close to a billion daily users, each of which 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 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 (Tanter, Vernaillen, & Piquer, 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 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 (Saber & 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, Mc Kevitt, & Curran, 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 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 (Chen, Ge, Kurose, & Towsley, 2005).

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

Figure 1. 3 layer model



T is a generic service corresponding to layer 2, 3, or 4 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 realize 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. 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 previous 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 (Bradshaw, Kurose, Shenoy, & Towsley, 2005).

Many such dynamically reconfigurable conventional middleware systems exist (Becker, Schiele, Gubbels, & Rothermel, 2003; Blair, Coulson, & Andersen, 2001; Curran & Parr, 2004; Gill et al., 2004), and 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 (e.g., OMG, 2002; Schmidt, 2004), 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, Kon, & Campbell, 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 (Fry & West, 2004). 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.

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 (Murch, 2004):

- 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. It should be able to cope with software and or hardware failures whether they are due to 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 incorporate new components seamlessly (Tanter et al., 2002).
- Modern systems may contain large amounts of different variables/options/parameters, which a user can 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.
- 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 problems 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 (Murch, 2004).

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

The IBM autonomic computing toolkit[1] enables developers to add self-configuring and other autonomic capabilities to their software. 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 behavior 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.

FUTURE TRENDS

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. Therefore, 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 (i.e., it is about making systems self-managing). Future trends in network design, which will 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 QoS needs. Another trend is introspection, which provides run-time system information allowing applications to examine their environment and act accordingly. Autonomic computing systems can provide an infrastructure for building adaptive applications that can deal with drastic environment changes. The Internet with its various standards and interconnection of components such as decoders, middleware, and databases deserve more than a plug, try, and play mentality. The introduction of mobility will also increase the complexity due to the proliferation in possible actions. A key goal for next generation networks 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

CONCLUSION

Modern networks offer end-to-end connectivity, however, the increasing amount of traditional offered services may still not fulfill 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. 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. Autonomous systems are capable of performing activities by taking into account the local environment and adapting to it. 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, self-organisation, self-configuration, self-adaptation, self-healing, and self-protection). Autonomic computing refers to the ability of a system to self-diagnose without the need for operator intervention. Traditionally, systems management has focused on monitoring and measurement, with an emphasis on end-to-end management but autonomic computing focuses on the self-managing capabilities of the infrastructure itself.

REFERENCES

Becker C., Schiele, G., Gubbels, H., & Rothermel, K. (2003, July). BASE—A micro-broker-based middleware for pervasive computing. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communication (PerCom)*, Fort Worth.

Blair, G. S., Coulson, G., & Andersen, A. (2001). The design and implementation of OpenORB version 2. *IEEE Distributed Systems Online Journal*, *2*(6), 45-52.
Bradshaw, M., Kurose, J., Shenoy, P., & Towsley, D. (2005, June). Online scheduling in modular multimedia systems with stream reuse. In *Proceedings of NOSSDAV*, Skamania, Washington.

Chen, W., Ge, Z., Kurose, J. , & Towsley, D. (2005). Optimizing event distribution in publish/subscribe systems in the presence of policy-constraints and composite events. *IEEE ICNP 2005*.

Curran, K., & Parr, G. (2004, October 20-22). Introducing IP domain flexible middleware stacks for multicast multimedia distribution in heterogeneous environments. In *MATA 2004—International Workshop on Mobility Aware Technologies and Applications*, Florianopolis, Brazil (LNCS, pp. 313). Springer-Verlag Heidelberg.

Fry, G., & West, R. (2004, May 25-28). Adaptive routing of QoS constrained media streams over scalable overlay topologies. In *Tenth IEEE Real-Time and Embedded Technology and Applications Symposium*, Le Royal Meridien, Toronto, Canada.

Gill, C. M., Gossett, J., Corman, D. P., Loyall, J. E., Schantz, R., Atighetchi, M., et al. (2004, May 25-28). Integrated adaptive qos management in middleware: An empirical case study. Submitted to the *Tenth Real-Time Technology and Application Symposium*, Le Royal Meridien, Toronto, Canada.

Murch, R. (2004). *Autonomic computing.* IBM Press, Prentice Hall PTR.

Object Management Group. (2002, July). The common object request broker: Architecture and specification, Revision 3.0.

Roman, M., Kon, F., & Campbell, R. (2001, July). Reflective middleware: From your desk to your hand. *IEEE Distributed Systems Online Journal*. Special issue on Reflective Middleware.

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

Solon, A., Mc Kevitt, P., & Curran, K. (2005, February). TeleMorph: Bandwidth determined mobile multimodal presentation. *Information Technology and Tourism, 7*(1), 33-47.

Tanter, E., Vernaillen, M., & Piquer, J. (2002). Towards transparent adaptation of migration policies. Position paper submitted to *EWMOS 2002*, Chile (pp. 34-39).

KEY TERMS

Autonomic: Relating to, or controlled by the autonomic nervous system.

Closed Control Loop: This technique stems from process control theory. A closed control loop in a self-managing system monitors some resource and autonomously tries to keep its parameters within a desired range.

Distributed Computing: A system where tasks are divided among multiple computers rather than having

all processes originating from one main central computer. Client/server systems are one type of distributed computing. It can also be described as a system in which services are provided by teams of computers collaborating over a network.

Grid Computing: A computing model that provides the ability to perform higher throughput computing by taking advantage of many networked computers to model a virtual computer architecture that is able to distribute process execution across a parallel infrastructure. GRID Computing is basically taking a number of inexpensive personal computers and connecting them via a network to build a supercomputer, which can utilize the idle processing time on each machine to carry out tasks that would have previously required an expensive mainframe. One comparison that is often used to describe a computational GRID is that of the electrical GRIDs responsible for providing electricity.

Pervasive Computing: This is the trend toward increasingly ubiquitous connected computing devices in the environment and particularly, wireless technologies and the Internet. Pervasive computing devices are not broadly speaking personal computers as we tend to think of them, but rather small (often micro like)—electronic mobile embedded devices in almost any type of real world object, including cars, tools, household appliances, clothes, and so forth—all communicating through increasingly interconnected networks.

Self Healing: Having the power or property of healing one's self or itself. Autonomic computing refers to the ability of systems to self-diagnose and self-heal without the need for operator intervention.

Self-Management: The process by which computer systems manage their own operation without human intervention.

ENDNOTES

[1] http://www-03.ibm.com/autonomic/