Autonomic Behaviour in QoS Management

Rachid Anane
Department of Computer and Network Systems, Coventry University, UK
r.anane@coventry.ac.uk

Abstract

In QoS provisioning adaptation defines an area of overlap between QoS management and autonomic computing. This convergence is highlighted by the adherence of QoS systems to fundamental principles such as transparency, and the ability of QoS mechanisms to adapt seamlessly to prevailing environmental conditions. This characteristic is also underpinned by a combination of monitoring and management mechanisms, features that are inherent to autonomic behaviour. This paper is concerned with an investigation into the autonomic behaviour of QoS management systems. Relevant QoS mechanisms are identified and their functionality mapped onto autonomic behaviour. In a wider context, the components of a specific QoS architecture, ITSUMO, are considered from the perspective of autonomic control. Furthermore, as part of an ongoing research project, an evaluation mechanism is presented as a fundamental component in the architectural support for autonomic behaviour.

1. Introduction

The highly dynamic and unpredictable nature of distributed systems can lead to a highly volatile environment where QoS provision needs to adapt seamlessly to changing resource levels. QoS management in wireless networks presents specific challenges because of the inherent mobility of hosts and the uneven resource availability. The trend in recent QoS system design is to shield both applications and users from the complexity of the environment, and to maintain a level of QoS compatible with a declarative statement of requirements [1]. One focal point of research in this endeavour is the need to introduce mechanisms that facilitate autonomous adaptation. In QoS provisioning adaptation identifies an area of overlap between QoS management and autonomic computing. This convergence is sustained by the adherence of QoS systems, in particular, to fundamental principles such as transparency. The ability of QoS mechanisms to adapt seamlessly to prevailing environmental conditions is underpinned by a combination of monitoring and adaptation mechanisms, features that are inherent to autonomic behaviour.

This paper is concerned with an investigation into the autonomic behaviour of QoS management systems. Relevant QoS mechanisms are identified and their behaviour mapped onto autonomic properties. The components of a specific QoS architecture, ITSUMO are considered from the perspective of the autonomic control loop. Furthermore, an evaluation mechanism is presented as a fundamental component in the architectural support for autonomic behaviour. The aim of the mechanism is to provide support for adaptation and for accurate, transparent and efficient assessment and evaluation of QoS requests.

The remainder of the paper is as follows. Section 2 introduces autonomic computing and presents its characteristics. Section 3 describes the main functions involved in QoS management. Section 4 highlights the autonomic behaviour in QoS activities. Section 5 identifies the nature of the link between QoS architecture behaviour and autonomic control. It also details the structure of an evaluation mechanism. Some conclusions are drawn in Section 6.

2. Autonomic Computing

Autonomic computing is being proposed as a way of addressing the issues raised by the increased complexity of information and communication systems [2, 3, 4, 5]. Identifying the salient features of autonomic computing has focused on defining the notion of autonomic element and its structural and behavioural properties. A system exhibits autonomic behaviour when it can manage itself and displays self-awareness. Self-management is expressed in terms of four properties: self-configuration, self-optimisation, self-healing and self-protection, often referred to as self-* properties. These properties are underpinned by two fundamental activities, monitoring and policing:

- Self-configuration: ability to adapt dynamically to environmental changes in conformance with high-level policies, expressed declaratively.
- Self-optimisation: ability to seek proactively to optimise the use of resources in order to improve functionality and operations.
- Self-healing: ability to detect, diagnose problems and initiate corrective action.
- Self-protection: ability to establish trust, anticipate and protect against internal and external threats and recover from attacks.

From an architectural point of view, an autonomic system consists of a set of interactive autonomic elements. An autonomic element is a unit of composition, which includes a single autonomic manager that controls and represents one or more managed elements. A managed element is equivalent to a traditional non-autonomic element such as a CPU [2].

3. QoS Management

In QoS provisioning, QoS management can be performed along a specific path or at node level. The Internet Engineering Task Force (IETF) has proposed two main approaches to quality of service, *InServ* and *DiffServ*. The Integrated Services *(IntServ)* framework offers explicit reservations end-to-end and tight guarantees. The *IntServ* model uses RSVP, a signalling protocol, to request specific resources along the data path. The Differentiated Services *(DiffServ)* offers hop-by-hop differentiated treatment of packets. It does not require any signalling protocol. *DiffServ* is realised by mapping the DSCP byte contained in the IP packet header to a particular treatment or per-hop behaviour (PHB), at each node along the path of the packet. Resource allocation is being defined by means of Service Level Agreement (SLA). *DiffServ* performs aggregate classifications of packets in contrast to *IntServ*, which provides a per-flow classification [6]. In this context QoS management involves a set of activities, which can be grouped into two types of functions, static and dynamic. Static functions include QoS specification, negotiation, admission control, QoS mapping and resource reservation, whereas dynamic functions are concerned with monitoring, QoS policing, control, adaptation and renegotiation [7].

Although some QoS architectures may not conform to a clear separation of concern, the functionality of the QoS manager and of the resource manager can be delineated and clearly identified. The QoS manager is responsible for maintaining the QoS levels agreed with the applications, whereas the resource manager is responsible for controlling the environment resource usage.

3.1 Static Functions

These functions are mainly concerned with initial setup, but they share some activities with the run-time functions. Admission control relies on QoS evaluation in particular, an activity, which is central to adaptation [8].

QoS specification. The ability to specify QoS requirement in a declarative manner is highly desirable. This facility reinforces the transparency of QoS management and shields both users and applications from low level details. For example, it is more convenient for users to refer to frame rate rather than megabytes per seconds. The specification can be translated into more technical terms for various levels of abstractions.

QoS negotiation and QoS admission control. The negotiation process between an application and a QoS manager may result in a Service Level Agreement (SLA), which is translated into a Service Level Specification (SLS) or QoS profile, and from which QoS parameters are extracted. The SLS is the basis for renegotiation and adaptation. The admission control activity is concerned with determining whether a new QoS request can be supported by the system without affecting the SLS of the existing applications. The mathematical models used at these tests involve system behaviour as well as scheduling and network traffic models.

QoS mapping and reservation. The QoS manager negotiates with the system resource manager the reservation of resources for a specific QoS requirement. The resource manager is responsible for low-level activities such as maintaining throughput, delay, or jitter at agreed levels. The mapping of QoS to resources is achieved by a negotiation between QoS manager and resource manager. It involves the translation of QoS parameters into specific units of resources such as buffers, CPU and bandwidth. For example allocation of bandwidth can be in terms of time slots required for a connection. In the *IntServ* model it involves conditioning network traffic through all the nodes along a particular path.

3.2 Dynamic Functions

These functions define the scope for self-management in QoS provision and the related activities that give arise to autonomic behaviour:

QoS monitoring and QoS control. QoS monitoring is an activity that is closely related to QoS policing. It underpins the functions in QoS management, in particular QoS renegotiation and adaptation. Monitoring enables the QoS manager to react to changes in QoS provision and to proactively initiate the renegotiation of an SLA. This function implies that procedures and methods for QoS measurement need to be determined. QoS policing, on the other hand, is aimed at monitoring user traffic in order to ensure conformance to the SLA, to prevent QoS violations and to shield system resources from user misbehaviour. On detection of non-conformance to SLA (QoS violation), actions that may be taken by the policing activity include notifying users, increasing cost or shaping QoS traffic to ensure conformance. In the case of QoS provision in wireless networks, the transition between administrative domains requires consultation with old and new Authentication, Authorisation and Accounting (AAA) servers.

QoS adaptation and QoS negotiation. Adaptation in QoS is a core activity that relies on other functions such as negotiation, monitoring and policing. Adaptation can

be performed at application, middleware or network level. Hafid *et al* [8] describes a mechanism for adaptation at application level. The reconfiguration of the application follows renegotiation. In Nahrstedt *et al* [9] a QoS-aware middleware system was introduced to support QoS adaptation of applications at run-time. Under this scheme, a QoS profile is generated and then acted upon by the middleware during the phases of QoS provision and, especially, adaptation of QoS. It is dynamically tuned to fit available resources. QoS violation can be handled at different levels. In some architectures, the QoS manager and the resource manager negotiate a repairing action without the involvement of the application [6]. This approach may limit the level of adaptation of the application to new environmental conditions.

QoS adaptation is particularly relevant to wireless networks and may be triggered by a QoS violation, which may arise during handover between access points. Change in IP address and contention for resources between users may lead to the situation where the QoS provided to a mobile host may not be assured by an adjacent cell. QoS violation can also occur because of long delays, packet loss or even denial of service. This highly dynamic environment is the cause of frequent renegotiations and adaptations. A new QoS may have to be negotiated along a new route as part of a handover. QoS adaptation may also be caused by proactively seeking to optimise the use of resources. A service provider may advertise unused resource if it is underutilised. On the other hand, a service provider can request and negotiate with users to lower their service level if the demand for QoS provision is too high [1]. Adaptation takes place within a specific context of which the QoS manager and the applications need to be aware. Context awareness can be provided explicitly by a context manager or by reflective techniques.

4. Autonomic Behaviour in QoS Provision

An autonomic approach to QoS management has the merit of delineating clearly the nature and the scope of its functions. Although wireless systems have different requirements from fixed networks, many QoS self-management functions can be used in both domains. In the context of this investigation, the autonomic behaviour in QoS management is associated with the ability of the QoS manager to adapt QoS provisions seamlessly in response to changes in the environment. The activities performed by the QoS system, in general, display many of the charactteristics that identify autonomic behaviour. All these activities and their properties are related to dynamic functions and adaptation, in particular.

Self-configuration arises from the need to adjust the level of QoS provided to an application. This self-configuration may affect a path under *IntServ* provisions, and therefore node resources and traffic bandwidth. The initiative for self-configuration may be taken either by the application or the service provider and can the result of a negotiation process.

Self-optimisation assumes a careful monitoring of resource usage and underlines a proactive approach to QoS provisioning. Chen *et al* propose a protocol for supporting negotiation in order to optimise the use of resources in the system [1]. In their framework the Dynamic Service Negotiation Protocol (DSNP) is used to request clients to adjust their QoS requirements to suit the current network conditions. Similarly, the DSNP can advertise the availability of unused resources and offer them at a reduced price. This proactive approach strikes a compromise between maximising the use of resources and maintaining an acceptable level of service [4]. The need for optimisation also arises in situation where, for example, real-time applications are being considered. It is possible for the QoS manager to negotiate with the resource manager to divert resources from a low-priority application to one with a higher priority**.**

Self-healing is concerned either with outright QoS violation or QoS degradation. The tasks of the QoS manager include monitoring path health, diagnosing the causes of poor health, and requesting computation and communication resources to maintain and restore health adaptively. An adaptive resource manager performs a number of steps to bring the system back into an acceptable state, and perform system diagnosis to determine a set of possible actions to restore QoS to required levels. Possible actions include moving a program, replicating a program, removing a program, and re-routing a communication sub-path. QoS management in wireless networks deals with many situations where the self-healing process unfolds, following a QoS violation. In wireless systems the DSNP, for example, can be used to renegotiate a QoS or to terminate the SLS of a client as part of the self-healing process.

Self-protection is closely linked to policing and monitoring, and is the manifestation of a constant and dynamic function. It also assumes the mediation of AAA servers. In autonomic systems, the management of identity and trust acquires special significance [10].

Self-awareness is also an important feature of autonomic behaviour. Since this paper is concerned with QoS management it is assumed that self-awareness is closely related to QoS awareness. QoS-aware in this context means that application and middleware can perform adaptation depending on the changing environment and on the SLS. QoS-awareness can be enhanced by making the application, the middleware or the network QoS-aware [10]. At the application level this may involve the grafting of a QoS handler onto the application [11], or a more loosely coupled QoS component that can become part of an application after a

search and discovery process [12]. The augmentation of an application with a QoS–aware component contributes to the creation of entities endowed with some embryonic self-awareness, and therefore autonomic behaviour. This application-centric approach allows applications to specify QoS requirements, monitor the provided QoS level and adapt to QoS changes. This approach is also characterised by its distributed nature and its application-oriented management.

Reflective architectures at the middleware level have also been investigated as an effective means of offering flexibility and support for application configuration. Reflection is particularly attractive because of its inherent property of self-awareness. Reflection allows for the adaptations of core software and hardware without explicit intervention by application or end user, an important characteristic of autonomic behaviour [13]. Some of the benefits that accrue from the adoption of reflective middleware techniques for QoS-enabled component-based techniques relate to the enhanced adaptation at various levels. More specifically, optimal communications mechanisms can be selected and run-time components can be configured dynamically. Blair *et al* [14] developed a reflective architecture where configuration is achieved by introducing or removing management components on demand. At a higher level, reflection promotes also a policy-driven approach to QoS management. A policy for management can be inspected and adapted dynamically, and meta-managers and meta-policies can be introduced to manage the management structure itself. Policies expressed declaratively at the business level are mapped onto intermediate layers to the node configuration level [15, 16].

5. QoS Architecture and Autonomic Control

In this section the ITSUMO QoS architecture is considered and its behaviour presented in terms of autonomic behaviour. As part of an effort to enhance the ability of the architecture to assess QoS requirements accurately, a QoS evaluation mechanism is also described.

5.1 ITSUMO Architecture

The ITSUMO approach to QoS provisioning is based on *DiffServ* within a wireless environment [17]. It consists of a central server, called a QoS Global Server (QGS) and of dumb ingress nodes called QoS Local Nodes (QLN). The QGS holds global information, such as availability of resources about its domain and instructs the QLNs when new QoS requests are admitted. The QGS negotiates QoS requirements with mobile nodes and generates a service level specification (SLS), which is broadcast to the QLNs. A mobile node can renegotiate its SLS, and the update sent to the QLNs. Similarly, if the resources become scarce, or released, the QGS may request the mobile host to update its SLS to suit the existing environmental conditions. The main role of the QLNs is to monitor and condition traffic according to the instructions from the QGS. The QGS polls regularly the QLNs in order to build a model of the state of resources and patterns of mobility of mobile nodes. Unlike the QGS, which deals with control, a QLN is mainly concerned with transport.

5.2 Autonomic Control Loop

Autonomic computing requires the implementation of an effective autonomic control loop. The loop involves a number of functions (Figure 1): observation (monitoring the managed element), deliberation (analysis and planning) and execution (configuration of the managed element). In addition, the proactive behaviour of the autonomic element is enhanced by negotiation as a two-way mode of interaction [18].

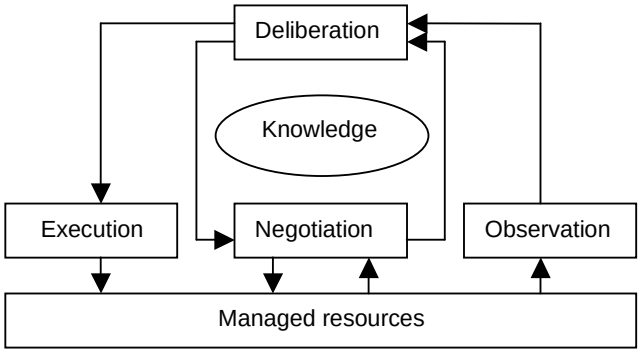

Figure 1. Autonomic control

Many of the functions performed in ITSUMO implement some form of autonomic control. An enhanced ITSUMO architecture defines an autonomic control loop as follows:

- **Observation** involves the QGS polling regularly the QLN for information on path health, new QoS requests, QoS termination and QoS violation.
- **Deliberation** by the QGS concerns the assessment of current conditions, the evaluation and re-evaluation of QoS, and the decisions to reallocate resources.
- **Execution** entails CPU, network or bandwidth reconfiguration following instructions by the QGS to a QLN in response to new environmental conditions.
- **Negotiation** facilitates agreement and review of service levels between client and service provider.

The ITSUMO architecture implicitly defines the QGS as a functional unit with many similarities to an autonomic element. The QGS controls and represents QLNs. Adaptation and negotiation are forms of self-management supported by the information held by the

QGS and the QLN. Self-configuration encompasses instructing QLNs to condition traffic in a specific manner following deliberation, while self-optimisation is achieved by renegotiating proactively new SLS with applications. As far as self-healing is concerned, this takes place when a QoS violation is detected and diagnosed or when path deterioration is reported. System self-management, at the administrative domain level, arises in response to new conditions and new requirements.

5.3 A QoS Evaluation Mechanism

One of the points of convergence of QoS architectures is QoS evaluation. QoS evaluation is an ancillary function that is common to all the phases of QoS management. The fact that admission control is often seen as a special case of adaptation underlines the importance of this function. It involves essentially estimating the current level of resource commitment and determining whether existing resources can satisfy a QoS request. This may involve the use of performance models. As stated earlier adaptation can be proactive and reactive and resource levels for applications can be varied in response to availability and application priority. Support for adaptation and therefore autonomic behaviour postulates the existence of an accurate mechanism for evaluation QoS requests and for assessing QoS requirements. Accurate and flexible assessment of QoS provision facilitates optimisation and fine-tuning.

A QoS evaluation architecture, aimed at enhancing the functionality of the QoS manager, is proposed in this section. The evaluation function can be called upon, for example, to determine the viability of increasing the QoS of a user after negotiation. The aim of the evaluation mechanism is to optimise the overall QoS level while at the same time maintaining previously agreed service levels. This requirement can be expressed in terms of a system utility function, U_s, defined as an aggregation of individual utility functions, u_i for each application in a set A [19]. Individual functions are weighted according to SLA in general and resource availability and priority in particular**:**

$$U_s = \sum_{i \in A} w_i . u_i$$

The core components of the proposed architecture are shown in Figure 2. The evaluation can be triggered by the receipt of a QoS request where the QoS Manager calls upon the QoS Evaluation Module (QEM) to determine whether a QoS request can be satisfied. The evaluation process itself involves a number of components. These are described below:

Table of Commitments (TOC): holds information on the current state of the system. This includes information about system nodes, the resources and the QoS allocated to each node. TOC can be used to create a model

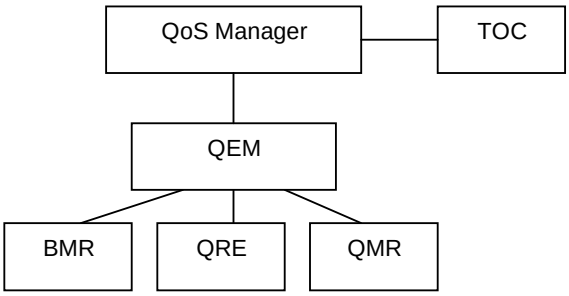

Figure 2. Architecture for QoS evaluation

representing the behaviour of the system and a model representing QoS. In order to optimise the performance of the system, the information in TOC is logged in a repository.

QoS Evaluation Module (QEM): the entry of the request for QoS q by a new user into the system may cause the QEM to initiate the verification process in order to determine whether the request can be satisfied. This verification involves three specific tasks associated with the architectural elements. First, a behavioural model of the system is generated from the specification of the behaviour of the system components, as stated in the TOC, and from the behavioural model of the new user. Second, a formal representation of the request q is produced. Third, the QoS request can then be checked against the generated behavioural model.

Behavioural Model Repository (BMR): BMR is a repository whose main function is to make available various templates, which represent the models of behaviour of components such as communication channels, sources and sinks. The overall behaviour of the system is generated by combining various templates as fundamental building blocks of the system. QEM operates on the templates in BMR by instantiating selectively different parts of a model in order to build a behavioural model for the overall system.

QoS Model Repository (QMR): QMR plays a similar role as BMR and holds a set of templates that can be used to model QoS statements. QEM uses the templates in the QMR to instantiate formal representations of the QoS features of the system.

QoS Resolution Engine (QRE): QRE is the computational component in the architecture. It takes a model of the behaviour and a model of the QoS request and verifies the compatibility of the QoS statement with the behaviour of the model. The QoS request may be single or an aggregation of the QoS requests stored in the table of commitments. This ensures that interdependent requests are taken into account.

Although the scheme presented above relies on the formal modelling of the behavioural and QoS features of wireless systems, this approach to automation is independent of the choice of the specification language. A

Timed Automata representation was used as proof of concept [20].

6. Conclusion

Irrespective of the level at which adaptation is performed, whether at network, middleware or application, the behaviour of the QoS manager exhibits some inherently autonomic properties. Its main function is to ensure conformance of resource usage to SLA, and to reconcile conflicting requirements. In the process a number of functions are invoked in order to provide adaptation both reactively and proactively. At the QoS systems level, such as in ITSUMO, architectural features can be mapped onto the components of an autonomic element, and an autonomic control loop identified.

Although a claim can be made that autonomic behaviour can be elicited in many of the functions and activities of QoS management, there is a clear need for the development of an integrated framework. Many of the examples were drawn from various architectures, different contexts and are based on different models. Reflection for instance is closely associated with the object-oriented model. Another issue for investigation is prompted by the focal point of self-awareness. Since both application and middleware collaborate in supporting adaptation, the scope of adaptation can be confined to the application, the middleware or distributed across the QoS components that make up the QoS provisioning system. A more principled approach to QoS management will undoubtedly lead to better designs, improve QoS provisioning and enhance adaptation.

7. References

[1] J.C. Chen, A. McAuley, V. Sarangan, S. Baba, and Y. Ohba, Dynamic Service Negotiation Protocol (DSNP) and Wireless DiffServ, *International Conference on Communications (ICC'02),* New York City, April 2002, pp1033-1038.

[2] Kephart J.O. and Chess D.M, The Vision of Autonomic Computing, *IEEE Computer 36(1),* January 2003, pp41-52.

[3] McCann J.A and Huebscher M., Evaluation Issues in Autonomic Computing, *3rd International Conference on Grid and Cooperative Computing,* Wuhan, China, October 2004, pp597-608.

[4] Salehie M. and Tahvildari L., Autonomic Computing: Emerging Trends and Open Problems, *First Workshop on the Design and Evolution. of Autonomic Application Software (DEAS),* , St Louis, Missouri, USA, 2005.

[5] Cylenko G., Berk V.H., Gregorio-De.Souza I and Behre C., Practical Autonomic Computing, *30th Annual International Computer Software and Applications Conference (COMPSAC'06),* Chicago, USA, 2006.

[6] Cavanaugh C.D, Welch L.R., Shirazi B.A., Huh E. and Anwar S., Quality of Service Negotiation for Distributed, Dynamic Real-time Systems, *Parallel and Distributed Processing 15 IDPDS 2000 Workshops, LNCS 1800,* Cancun, Mexico, 2000, pp757-765.

[7] Chalmers D. and Sloman M., A survey of Quality of Service in Mobile Computing Environments, *IEEE Communications Survey, Second Quarter,* 1999, pp2-10.

[8] Hafid A. and Bochman G., An approach to QoS Management in Distributed Multimedia Applications: Design and Implementation, *Multimedia Tools and Applications, Vol 9, No. 2,* 1999.

[9] Nahrstedt, Xu D., Wichadakul D. and Li B., QoS-Aware Middleware for Ubiquitous and Heterogeneous Environments, *IEEE Communications Magazine, 39(11),* November 2001, pp140-148.

[10] Dobson S., Denazis S., Fernandez A., Gaiti D., Gelenbe E., Massacci F., Nixon P., Saffre F., Schmidt N., Zambonelli F., A Survey of Autonomic Communications, *ACM Transactions on Autonomous and Adaptive Systems, Vol. 1, No. 2,* December 2006, pp223-259.

[11] Sugawara T. and Tatsukawa K., Table-based QoS Control for Embedded Real-Time Systems, *ACM Workshop on Languages, Compilers and Tools for Embedded Systems,* Atlanta, USA, 1999, pp65-72.

[12] Menasce D.A., Ruan H. and Gomaa H., A Framework for QoS-Aware Software Components, *Fourth International Workshop on Software and Performance (WOSP'04),* Redwood City, 2004, pp186-196.

[13] Wang N., Parameswaran K., Kircher M., Schmidt D.C., Applying Reflective Middleware Techniques to Optimize a QoS-enabled CORBA Component Implementation, *International Computer Software and Applications Conference (COMPSAC 2000) Conference,* Taipei, Taiwan, 2000, pp492-499.

[14] Blair G.S., Andersen A., Blair L. and Coulson G., The Role of Reflection in Supporting Dynamic QoS Management Functions, *Proceedings of the 7th IEEE International Workshop on Quality of Service (IWQoS'99),* London, June 1999, Crowcroft J., Bhatti S. & Diot C. (Eds).

[15] Chaouchi H. and Munaretto A., Adaptive QoS management for IEEE 802.11 Future Wireless ISPs, *Wireless Networks 10,* 2004, pp413-421.

[16] Bahati R., Bauer M., Vieira E. and Baek O., Using policies to drive Autonomic Management, *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06*), Bufffalo-NY, USA, 2006.

[17] Chen J.C., McAuley A., Caro A., Baba S., Ohba Y., Ramanathan R.. A QoS Architecture for Future Wireless IP Networks. In *Twelfth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2000),* Las Vegas, USA, November 2000.

[18] Koehler J., Giblin C., Gantenbien D. and Hauser R., On Autonomic Computing Architectures, *IBM Technical Report, RZ 3487,* January 2003.

[19] Vienne P. and Sourrouville J., A Middleware for Autonomic QoS Management Based on Learning, *Fifth International Workshop on Software Engineering and Middleware (SEM),* Lisbon, Portugal, 2005.

[20] Bordbar B. and Anane R., An Architecture for Automated QoS Resolution in Wireless Systems, *The IEEE AINA 2005 International Workshop on Web and Mobile Information Systems (WAMIS),* Taipei, Taiwan, March 2005, pp774-779.