A TIMED AUTOMATA APPROACH TO QOS RESOLUTION

BEHZAD BORDBAR[1], RACHID ANANE[2] AND KOZO OKANO[3]

[1]*School of Computer Science, University of Birmingham, UK.*
B.Bordbar@cs.bham.ac.uk
[2]*Department of Computer and Network Systems, Coventry University, UK.*
R.Anane@coventry.ac.uk
[3]*Graduate School of Information Science and Technology, Osaka University, Japan*
Okano@ist.osaka-u.ac.jp

Abstract: Concern over the accurate evaluation of QoS requirements has been one of driving forces in the development of QoS management architectures. This paper presents an architectural approach to QoS evaluation and admission control, based on the modelling of both system behaviour and QoS requirements. Two aspects are considered. The first refers to QoS management, and to a component-based architecture for QoS evaluation. The second illustrates the approach with the help of a case study based on a Personal Area Network. The proposed approach is model-based and makes use of models representing both behaviour and QoS aspects of the system via Timed Automata. The compatibility of the mechanism with architectures, which promote QoS management in its own right, such as ITSUMO, is also highlighted.

1. INTRODUCTION

The flexibility afforded by IP mobility in distributed systems is often at odds with the challenge of ensuring continuity of service and maintaining an agreed level of QoS (Burness, Hepworth et al. 2001). The highly dynamic nature of distributed systems can be mediated by a negotiation phase between clients and QoS managers to reflect prevailing conditions. The outcome can be expressed in terms of a service level agreement (SLA), which is often translated into a service level specification (SLS) and from which QoS parameters are extracted. In addition to the inherent channel errors, user mobility and contention between users for scarce resources may lead to situations where a service level may not be honoured by nodes, especially when handover takes place (Cavanaugh, Welch et al. 2000). Handover is symptomatic of the complexity of QoS management because of its implications for QoS provision. It may lead to the re-negotiation of service levels and to the re-allocation of resources, a disruption that may increase network latency (Lu, Lee et al. 1997). Hence, there is increasing interest in QoS management architectures. The aims of the design of QoS management architectures include support for adaptivity and for accurate, transparent and efficient evaluation of QoS requests. These aims can be achieved by an architecture that should allow for the gathering and storage of global QoS information, and also for the accurate evaluation of QoS requests.

The remainder of the paper is organised as follows. Section 2 gives an introduction to table-based QoS management. Section 3 describes the architecture of a QoS evaluation mechanism. Section 4 illustrates the proposed approach with a case study. Section 5 discusses issues raised in the paper. Section 6 presents related work and Section 7 concludes the paper.

2. TABLE-BASED QOS MANAGEMENT

Support for seamless mobility and adaptive computing in QoS provision are important requirements of QoS management. The transition period generated by a handover needs to be managed by the transfer of the SLS of mobile stations between adjacent nodes. Transfer can follow a reactive approach and be performed on demand such as in the architecture proposed in (Stattenberger and Braun 2001). An architecture such as ITSUMO (Chen, McAuley et al. 2000; Chen, McAuley et al. 2002), which will be used in this paper for reference, on the other hand, promotes a proactive approach; an SLS, once determined, is broadcast to all nodes in the same domain in order to ensure a seamless handover. ITSUMO is a reference architecture that adopts a principled approach to QoS management.

In many QoS architectures tables are the focal point of activity especially in admission control, or when re-negotiation is mandated (Cardoso and Kon 2004). Sugawara et al (Sugawara and Tatsukawa 1999) present an example of a table-based implementation, where information about QoS levels is maintained. The QoS table holds the resources required by all the scheduled tasks in the system, and its purpose is to facilitate the

resource allocation to tasks and the determination of system resource requirement. Both QoS control and QoS transport (shaping etc.) issues are dealt with by one component. This particular implementation is one point in a wider spectrum. In the architecture proposed by Pau *et al* (Pau, Maniezzo et al. 2003), where the Wireless Quality Enhancer (WQE), a QoS manager and policy maker, makes use of a table for interacting with the access points (AP), which are policy implementers. The tables are also directly relevant to the modelling and validation of QoS requests. One advantage of tables in QoS schemes is that policy-based QoS management can be enforced (Nanda and Simmonds 2003). ITSUMO (Chen, McAuley et al. 2000; Chen, McAuley et al. 2002) presents a more sophisticated approach to the use of tables. The GQS keeps various items of information including service levels agreements and their derivatives, patterns of mobility and domain resource availability. In conjunction with this centralised information, each QLN holds a subset of information in a local table that is used for run-time purposes and updated frequently by the GQS. The different types of table in the two components reflect the nature and the scope of their functions. The GQS, endowed with more intelligence, is concerned with QoS global decisions whereas the QLN is responsible for their implementation, at local level. In both these architectures the QoS manager, in the discharge of its functions relies mainly on the information stored in the table. This approach also puts the onus on the client application itself to specify unambiguously its QoS requirements, since the tables hold partial information. It may also be prescriptive. Reaching an agreement may be a lengthy and complicated process.

We propose an approach to QoS management and evaluation that takes into account system behaviour, and is designed to offer a more accurate evaluation of the requests, minimise negotiation and allow for extensibility. In reaching a decision, a QoS manager puts more emphasis on the dynamic behaviour of the system instead of confining its processing to the manipulation of the information in the table. An additional aim of the design is to maintain compatibility with the ITSUMO architectures and to supports its goals for scalability, adaptability and accuracy.

3. A QOS EVALUATION MECHANISM

The scope of the proposed architecture is determined by the desire to enhance admission control in QoS management. The architecture is presented in Figure 1. On receipt of a QoS request the QoS Manager calls upon the QoS Evaluation Module (QEM) to determine whether a QoS request can be satisfied.

3.1 Component description

The evaluation process involves a number of components as follows.

Table of Commitments (TOC): holds information on the current state of the system. This includes information about system nodes, resources and QoS allocated to them. TOC can be used to create a model representing the behaviour of the system and a model representing the QoS. The information in TOC is logged in a repository for optimisation purposes.

Repository: holds previously instantiated models and their requested QoS level. This allows the system to look up a QoS when a previous situation arises again. In the case of a new scenario the evaluation is delegated to a component called QEM.

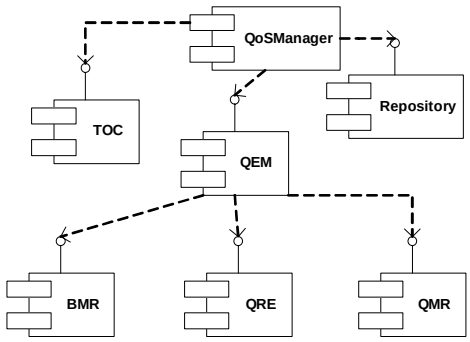

Figure 1: An architecture for QoS evaluation

QoS Evaluation Module (QEM): The evaluation process requires three elements. First, a model of the behaviour of the system, which includes the behaviour of the existing system components, and also the behaviour of the new user. Second, to verify a QoS request, a formal representation of such expression is generated. Third, the QoS requirements must be checked against the behavioural model. This is achieved by the use of QoS Resolution Engine, explained further below.

Behavioural Model Repository (BMR): BMR is a repository that contains various templates, which represent the models of behaviour of components such as communication protocols and channels. The templates are the building blocks from which the overall behaviour of the system can be composed. QEM uses the templates in BMR to instantiate different parts of a model, and creates a behavioural model for the overall system.

QoS Model Repository (QMR): Similarly, QMR consist of a set of templates that can be used to provide models for a QoS statement. QEM uses the templates in the QMR to instantiate formal representations of QoS aspects of the system.

QoS Resolution Engine (QRE): QRE operates on the behavioural and QoS models generated from BMR and QMR and TOC. A QRE is the intelligent component that receives a model of the behaviour and a model of the QoS request and checks the validity of the QoS statement against the behaviour of the model. The QoS request may be an aggregation of the QoS request stored in the table of commitments.

3.2 Implementation of architecture

In an earlier paper (Bordbar and Anane 2005), an implementation of the architecture, with the focus on the BMR, QMR and QRE, was outlined in terms of Timed Automata. BMR includes various templates Timed Automata (Clarke, Grumberg et al. 1999) for *source*, *sink*, different types of *buffer*, *decoder* (Bordbar and Okano 2003) as well as *communication protocol* (Bengtsson, Griffioen et al. 2002). These are the underlying building blocks for the creation of the behavioural models. The template Timed Automata represent the behaviour of the sub-components and include parameters for the variables of the model. The behavioural models of the system are networks of Timed Automata (Larsen, Pettersson et al. 1997), aggregated from the instantiation of templates in BMR, by assigning values to parameters in each template. QMR, on the other hand, is a repository of template Timed Automata corresponding to various Timeliness QoS properties such as jitter, latency and throughput (Chalmers and Sloman; Bordbar and Okano 2003**).** These are to be used as Test Timed Automata (Ageto, Bouyer et al. 2003). A Test Timed Automaton instantiated from the templates can be used to verify the corresponding QoS statement against the behaviour of the system, as modelled by instantiations from the BMR (Bordbar and Anane 2005). The final component, QRE is based on the model checker UPPAAL (Larsen, Pettersson et al. 1997; UPPAAL 2005**),** which can perform the verification of networks of Timed Automata. To ensure compatibility with the UPPAAL files, which are stored as XML files, the concrete representation of the Timed Automata is in XML. Despite this bias towards XML as a specific representation, the main implication is that XML is also a suitable form for holding information in the Repository and the Table of Commitment (TOC).

3.3 Component Interaction

An illustration of the dynamic behaviour and interaction of the three main components of the architecture, namely, QoS Manager, TOC and the Repository is given below, at a higher level of abstraction. Suppose that the QoS Manager receives a request from a new client. This request provides details of the pattern of interaction and the required QoS of the client. This is denoted as Requested Model (*RM*: *Model*) and Requested QoS (*RQ*: *QoS*). Once it receives the request the task of the QoS Manager is to consider the requested pattern of interaction of the new client and the resources allocated to the existing clients, and resolve the newly requested QoS *RQ*, i.e. to determine whether the new request can be satisfied. In order to achieve this, the QoS Manager obtains copies of the existing model of the system from the TOC, denoted by messages *getCurrentModel()*, which is returned as *CurrentModel*. The next step for the QoS Manager is to assemble the current model along with the requested model and QoS, i.e. *RM* and *RQ*, and forward them to the Repository. As stated earlier, the main function of the Repository is to keep a record of previous models of the system, in order to optimise system performance. For example, if a configuration consists of two applications running on a PC and another application running on a laptop, and their respective QoS requirements and behaviour can be supported by the system, then this fact is recorded in the repository. Future requests can be resolved by a process similar to a table look up. Work is currently being carried on the enhancement of the Repository so as to allow inference of new information from stored configurations.

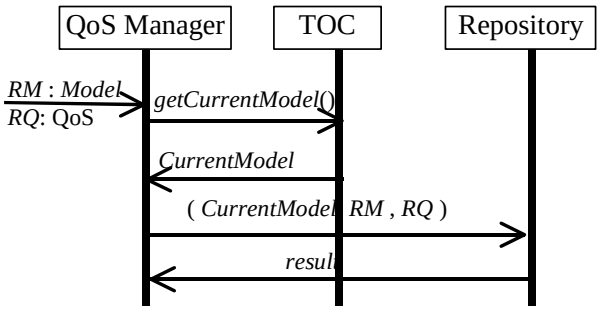

Figure 2: Interaction between TOC, Repository and QoS Manager

A response to a request by the QoS Manager to the Repository is returned by *result*, as a Boolean value. If *result* is true, the requested combination (*CurrentModel*, *RM*, *RQ*) can be supported by the system. If, on the other

hand, *result* is false, the combination *(CurrentModel, RM, RQ)* is passed to QEM for further resolution as described earlier. If the QEM resolves that the system can support both the committed QoS, encapsulated in *CurrentModel*, and the newly requested QoS, then the information captured in *(CurrentModel, RM, RQ)* is added to the repository for future reference.

4. CASE STUDY: A PERSONAL AREA NETWORK

In this section, a case study is introduced in order to illustrate the behavioural and QoS modelling process, and the evaluation mechanism.

4.1 Scenario

Let us consider a Personal Area Network (PAN) that consists of a Wireless router connected to the Internet. Figure 3 depicts a number of users (Stations), namely two PCs (PC1, PC2) and a Laptop (L1), which access the Internet via the router (Access Point). L2 is not part of the initial configuration. The stations are competing with each other to acquire bandwidth and to achieve a better QoS.

Now, consider a laptop L2, which wants to join the PAN as depicted in Figure 3. The task of the QoS manager is to determine the effect of a provision of service to L2 on the existing components. From the information it holds and the description of the behaviour of L2 and its QoS, the QoS Manager can create a model of the overall system. Let us refer to the model that includes the description of the behaviour of L1, L2, PC1 and PC2 as *m*. Under the behaviour specified in *m*, the system must not only satisfy the new QoS request from L2, but also the committed QoS requests for L1, PC1 and PC2. Such QoS requirements (including the request from L2) will be referred to as *q*.

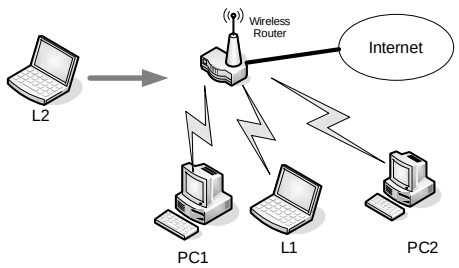

Figure 3: Wireless Router used in a PAN

Given the dynamic nature of wireless networks, it is possible that the above scenario, modelled in terms of *m* and *q*, may have occurred before. In this case, the QoS Manager may find such information in the Repository. As discussed in previous section, the QoS Manager can retrieve the models and use them to decide if QoS *q* is achievable within the behavioural model *m*. If, on the other hand, the models are not in the Repository, the evaluation process goes through a number of steps, detailed as follows. QEM receives a request to check if *q* is valid for the system. The request includes the parameters representing the QoS statement and a model of the behaviour of the system. This can be generated with the help of the information in TOC. QEM then instructs BMR and QMR to instantiate the behavioural model and the QoS statement, which are transferred to the QRE. The QRE carries the *check* and returns the *result* to QEM.

4.2 Components and Behaviour

The modelling process requires an explicit identification of the components of the network, and their interactions. To this end, it was decided to focus on the case where the applications on the station are *just* downloading packets from the Internet, i.e. there is negligible or no traffic from any station towards the router. As depicted in Figure 4, the Internet is the provider of the *packets*. The Wireless Router sends the packets to the Stations. Each station is assumed to contain an Input Module, which receives the packets from the Wireless Router and passes it to the Application Layer. The Application Layer represents a group of applications, which are viewed as "consumers of packets".

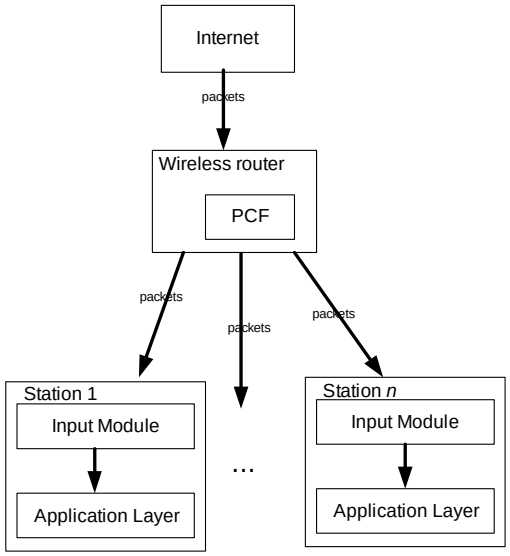

Figure 4: Flow of packets

For the sake of clarity caching and various other protocols involved in the transfer of the packets were not

included. In order to communicate with the stations the Wireless Router needs to access the medium by means of a protocol. The wireless local area network (802.11) (IEEE 1999) defines three basic access mechanisms. Firstly, there is a mechanism method based on Carrier Sense Multiple Access and Collision Avoidance (CSMA/CA). The second type of access method aims to address the hidden station problem; 802.11 enhances the first method by using two signal Request To Send and Clear To Send (RTS/CTS). The above two methods are refereed to as Distributed Coordinate Function (DCF). This example adopts the Point Coordinate Function (PCF) as the access mechanism, described Figure 5, which shows the timing of one AP and two stations in a Contention Free Period. First the AP downstream some DATA and a frame called CF-poll to ask Station 1 (STA1) to upload the data. After a fix period of time SIFS (Simple Interframe Space), STA1 upstreams its data and an acknowledgement (CF-ACK). The same process repeats for STA2. To end the Contention Free Period the AP sends a frame CF-END. After a SIFS, a new Contention Free Period can start.

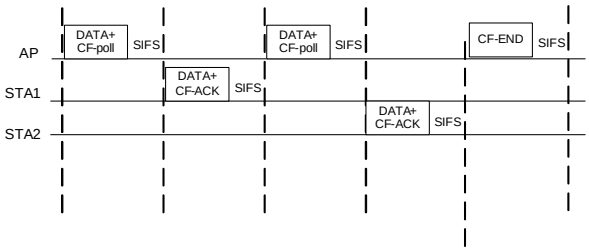

Figure 5: Contention Free Period and Polling

For further information on the WLAN and PCF, the reader is referred to (Schiller 2003).

4.3 Modelling behaviour

The modelling of a system's behaviour is an aggregation of the behavioural models of its components. This section presents a brief description of the behavioural models of the components in terms of networks of Timed Automata (Larsen, Pettersson et al. 1997) as depicted in Figure 6. The Wireless medium is modelled via the Timed Automata for the medium (TA for medium), which represents two states for the medium, *busy* and *free*. The switch between states is modelled via urgent actions, which occur as soon as they are enabled (UPPAAL 2005). The interaction with the router, which makes use of PCF, is modelled as *TA for PCF*. At the start of a contention free period, the medium gets busy, and this is shown with the signal *access?* Of *TA for PCF*. The integer value *i* ranges over the number of stations. There are *N* stations, i.e. $i = 1, \ldots, N$. Depending on the value of *i*, the downlink (*data!*) is meant to be delivered to station number *i*. The start with value of *i* is 1 and, it is incremented each time before the data is delivered to the next station.

After gaining access to the medium, the PCF sends data to the station. The data sent by the DCF must be broken into units of maximum length of MAC Service Data Unit (MDSU) (IEEE 1999; Schiller 2003). *A* denotes the amount of time required for the MDSU to reach the destination. As a result, at state *Sending_Data*, within *A* unit of time *data!* Is sent. Depending on the value of *i*, the signal *data?* Is used in the Application Layer of Station *i*. When the transmission of data finishes, an urgent acting *CF-poll* signal is sent to mark the end of data. To notify the medium, an *idle!* Signal is sent to mark the end of access. Then the PCF waits for SIFS (SIFS is 10 ms[1]). At exactly SIFS units it receives a *CF_ACK!* Signal from the Station that the data has been received. However, if $i < N$, in order to ensure that the next downstream goes to station $i+1$, the value of *i* is incremented. If $i = N$, this indicates that one contention free period is finished and a *CF-end* signal is sent. In this case, since no contention period is used, the CF-end is replaced with a simple acknowledgement signal *CF_ACK*. If the *CF_ACK* is sent a back-off period of SIFS is required.

Each station has an identifier *j* with a range of values between 1 and *N*. From the scenario described in Figure 4, the model of STA*j* is the parallel composition of two Timed Automata; (*TA for I/O*) and (*TA for App*) for consuming *data?* Created by the *TA for PCF*. Each part is shown in the diagram. In the TA for I/O, on receiving the signal CF_POLL! From the PCF, a clock starts. The station waits for SIFS unit and then sends a CF_ACK? To be used by TA for PCF. Since the scenario presented in the paper is concerned with downloading data, no upload time for sending data from the Station to the router is included. The TA for App periodically receives data? From PCF. As it is also possible to receive a frame with no data, TA for App models this via dataE? The PCF. *TA for Internet* models downstream flow from the Internet to the router. It periodically creates a signal packet!, and its period is specified by constants MP and mP. The signal packet! Is emitted during the period $[(i-1)\text{MP}+i\text{mP}, i\text{MP}]$. A constant WM specifies the size of buffer between an internet-side receiver and PCF in the wireless router. A global variable *q*, which specifies the current size of data in the buffer, is shared among TA for Internet and TA for PCF.

[1] It is 10 ms if FHSS is used and 28 ms if DSSS is used.

4.4 QoS modelling and Verification

Consider the system of the previous section, which consists of two PCs and a laptop, and with the parameters, WM = 5, WM = 5, SIFS = 40, PA = 20, BD = 20 and Ad (application delay) of 5. Suppose one of the requirements is that the throughput (Anchored throughput (Bordbar and Okano 2003)) is at least 1 frame every 127 units for laptop 1 (LP1), i.e. the occurrence of at least 6 signal VSChunk! In each period of SIFS*19, see Figure 6 *"TA for I/O"*.

Now, assume that a new laptop LP2, with a specification identical to LP1 wants to join the system. Also, assume that LP2 requests the same level of throughput (at least 1 frame per 127 unit). If the scenario involving LP1, LP2, PC1 and PC2 has not been modelled before, i.e. is not held in the TOC, the QoS Manager has to evaluate the achievability of the QoS for LP2 using the QEM. In order to do so a model of the system is created from the templates in BMR. This includes using the templates depicted in Figure 6 and the numerical parameters to create a network of Timed Automata model of the system. In order to check the QoS required by LP2, a Test Timed Automata for anchored throughput is required, as depicted in Figure 7. For further details on Test Timed Automata for the verification of QoS we refer the reader to (Bordbar and Okano 2003).

Type of timeliness QoS	Verified Property	Result	CPU time (sec)
Anchored Throughput	At least 6 signal VSChunk! In each period of 1000	valid	<0.5
Anchored Throughput	At least 7 signal VSChunk! In each period of 1000	invalid	<1
Non-Anchored Throughput	At least 4 signal VSChunk! In each period of 680	valid	<3
Non-Anchored Throughput	At least 4 signal VSChunk! In each period of 640	invalid	<3
Anchored Jitter	With period 160 +/-20 sec	valid	<0.5
Non-Anchored Jitter	With period 160 +/-20 sec	valid	<0.5

Table 1: QoS parameters following joining LP2

For this case, it can be seen that anchored throughput of at least 1 frame per 127 unit is not achievable. It is possible for LP2 to negotiate with the QoS manager and request a lower level of QoS. For example, it can be checked that the system can provide at least 1 frame per 167 unit (at least 6 signal VSChunk! In each period of SIFS*25). In a similar way, QoS manager can evaluate other types of timeliness properties resulting from joining the new laptop LP2 into the system. Table 1 shows the results of experiments conducted with different types of throughput and jitter. All experiments are conducted via UPPAAL version 3.4.11 running on an Intel III 600MHz Linux machine.

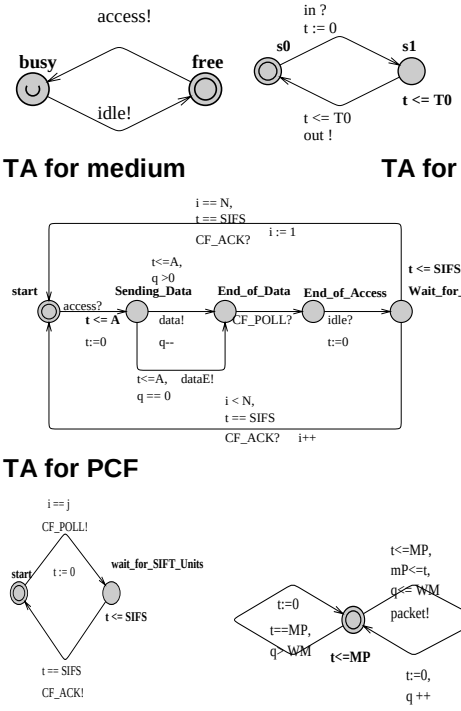

Figure 6: Behavioural Model of the system

5. DISCUSSION

The proposed architecture was motivated by two major concerns, namely the need to hold QoS information about commitments and requests, and the ability to determine accurately the viability of new QoS requests.

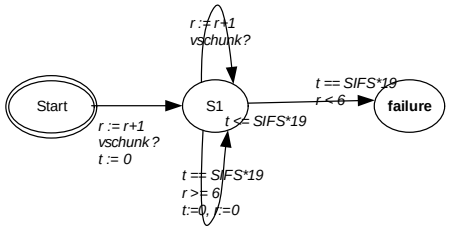

Figure 7: TA for Anchored Throughput

The first goal led to the introduction of a table of commitments for current requests, and a repository for previous ones. The evaluation and viability issue was addressed, firstly by instantiating existing templates into models of behaviour and QoS, and secondly by verifying the viability of these models. This approach simplifies the interactions between client applications and the centralised QoS server. Since the server is in possession of global information and is aware of behavioural models, QoS negotiation is minimised. A proactive approach can also be fostered by the existence of the repository. Concomitant with a transfer of complexity from application to server is a better assessment of resource management and allocation. Furthermore, the centralised approach allows for the enhancement of the level of intelligence in QoS management; a prospect that is in tune with the aims of the GQS in ITSUMO. Although a centralised decision maker can be a potential bottleneck, its role is mandated by user mobility and traffic level.

This affinity with ITSUMO is also underlined by a clear separation between QoS decisions and resource allocation. Furthermore it is the modelling approach that confers to the proposed architecture a significant role in providing support for flexibility, scalability and adaptivity in dynamic QoS management.
This feature was illustrated by the case study. The autonomy enjoyed by the QoS manager owes much to the availability of templates, the instantiation of models and their verification. These features may, however, be available at a price. There is a need to keep the template repositories up to date, and to optimise their symbiotic relationship with the table of commitments.

The concerns outlined above are pointers for further work. Most significant are the investigation of suitable structures for the table of commitments and for the repository, and the identification and selection of methods and adequate techniques for representing and manipulating models. In particular, the ability to aggregate and disaggregate instantiated models is a requirement for an efficient implementation of re-negotiation. Future work will also involve the investigation and evaluation of other formalisms for modelling behaviour.

6. RELATED WORK

The work presented in this paper relates to architectural models for QoS management as well as to modelling techniques for QoS. Architectures for QoS management span a wide range of approaches and can be defined by the level at which they operate and by their commitment to an explicit or implicit mode of QoS management.

6.1 QoS management

Some QoS architectures directly support both network layer and operating systems. For example, Roscoe *et al* (Roscoe and Bowen 2000) present an enhancement to the Windows NT architecture by adding a set of protocols, which can be used by an application to directly modify the network packets. Nahrstedt *et al* (Nahrstedt, Chu et al. 1999) describe an architecture that aims to mediate between the application and OS. This is achieved by a set of APIs, which allows the reservation of the CPU resources required by the applications. The middleware level is also directly relevant to QoS management because of its privileged role in mediation. Quo (Staehli, Eliassen et al. 2003; Eliassen, Staehli et al. 2004) is a middleware platform and architecture which supports the creation and composition of components by specifying the structure of the required QoS. Quo uses Reflection mechanism (Maes 1987) to identify suitable components, instantiate them and interconnect (bind) them.

At the other extreme of QoS management is the model that promotes QoS management by the application itself. Applications can be enhanced so as to make them QoS aware. For example, Enterprise Java Beans (EJB 2002), is a commercial component architecture for enterprise applications that does not include any QoS Management mechanism. The component architecture OpenORB (Coulson, Blair et al. 2002) extends the EJB by incorporating Component Architecture, a set of structures which embody policies and rules to support QoS. In OpenORB the application code is responsible for ensuring QoS. Another extension of EJB (Miguel, Ruiz et al. 2002) supports QoS by adding new container components for negotiation and adaptation. The proposed model presented in this paper offers an alternative that conforms much more to the client server model.

6.2 QoS modelling

Although the approach presented in this paper relies on the formal modelling of the behavioural and QoS aspects of wireless systems, it is important to note that this approach to automation is independent of the choice of the specification language. To allow automated analysis of configurations, Timed Automata, a variation of the Timed Automata model, was adopted. For a detailed coverage of Timed Automata, the interested reader is referred to (Clarke, Grumberg et al. 1999). The proposed architecture is, however, model agnostic, and can incorporate various modelling techniques. Timed

Automata offers accuracy potentially at the expense of speed. Most modelling techniques, reported in the literature are characterised by a bias towards the object oriented model.

Bordbar *et al* (Akehurst, Bordbar et al. 2002; Bordbar, Derrick et al. 2002a) present a framework based on ODP framework (Putman 2000) for expressing QoS. This makes use of an extension to Object Constraint Language (OCL) (Warmer 2003) and UML (UML 2003) for specifying required and provided QoS of objects. CQML (Aagedal and Ecklund 2002) is another language based on the ODP and UML for expressing QoS. CQML+ (Rottger and Zschaler 2003) extends CQML to express the demand on the resources in component-based systems and Web-based applications. For a recent review and comparison of various QoS specification methods see (Jin and Nahrstedt 2004) . These frameworks have the advantage of easing the modelling process thanks to the high level concepts they manipulate. The decomposition process, and in particular the access to, and reuse of the sub-components of a particular configuration present, however, a major challenge for the designers.

7. CONCLUSION

This work has highlighted the importance of modelling and verification in the accurate evaluation of QoS requests in wireless systems. It has shown that this can be achieved by a symbiotic relationship between table manipulation and model-based approaches. It has also pointed out that efficiency requirements for wireless systems, in highly dynamic environments, warrant a careful investigation and selection of both formalism and mechanisms in QoS management. Further work will focus on the refinement of the architecture and the investigation of other formalisms for QoS modelling.

References

Aagedal, J. and E. F. Ecklund (2002). Modelling QoS: Towards a UML Profile. 5th International Conference on Unified Modeling Language (UML 2002), Springer.

Ageto, L., P. Bouyer, et al. (2003). "The Power of Reachability Testing for Timed Automata." Theoretical Computer Science **300**(1-3): 411-475.

Akehurst, D. H., B. Bordbar, et al. (2002). Design Support for Distributed Systems: DSE4DS. Procedings of the 7th Cabernet Radicals Workshop.

Bengtsson, J., W. O. D. Griffioen, et al. (2002). "Automated Analysis of an Audio Control Protocol Using Uppaal." Journal of Logic and Algebraic Programming **52-53**: 163-181.

Bordbar, B. and R. Anane (2005). An Architecture for Automated QoS Resolution in Wireless Systems. Proceeding of the IEEE International Workshop on Web and Mobile Information Systems (WAMIS).

Bordbar, B., J. Derrick, et al. (2002a). "Using UML to specify QoS constraints in ODP." Journal of Computer Network and ISDN systems, pp 279-304.

Bordbar, B. and K. Okano (2003). Verification of Timeliness QoS Properties in Multimedia Systems. Proceeding of 5th International Conference on Formal Engineering Methods.

Burness, L., E. Hepworth, et al. (2001). Architecture for Providing QoS in an IP-based Mobile Network. Proc. IST Mobile Communications Summit.

Cardoso, R. S. and F. Kon (2004). A mobile Agent Infrastructure for QoS Negotiation of Adaptive Distributed Applications. CoopIS/DOA/ODBASE.

Cavanaugh, C. D., L. R. Welch, et al. (2000). Quality of Service Negotiation for Distributed, Dynamic Real-time Systems, Parallel and Distributed Processing. International Parallel and Distributed Processing Symposium.

Chalmers, D. and M. Sloman "A Survey of Quality of Service in Mobile Computing Environments." IEEE Communications Surveys **2nd quarter**.

Chen, J. C., A. McAuley, et al. (2000). A QoS Architecture for Future Wireless IP Networks. Twelfth IASTED International Conference onParallel and Distributed Computing and Systems (PDCS).

Chen, J. C., A. McAuley, et al. (2002). Dynamic Service Negotiation Protocol (DSNP) and Wireless DiffServ. IEEE International Conference on Communication.

Clarke, E. M., O. Grumberg, et al. (1999). Model Checking. London, MIT Press.

Coulson, G., G. Blair, et al. (2002). "The design of a configurable and reconfigurable middleware platform." Distributed Computing Journal **15**(2): 109-126.

EJB (2002). Sun Microsystems, Enterprise JavaBeans ™ Specification.

Eliassen, F., R. Staehli, et al. (2004). QuA: building with reusable QoS-aware components. OOPSLA Companion**:** 154-155.

IEEE (1999). "IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard 802.11."

Jin, J. and K. Nahrstedt (2004). "QoS Specification Languages for Distributed Multimedia Applications: A Survey and Taxonomy." IEEE Multimedia Magazine **11**(33): 74-87.

Larsen, K. G., P. Pettersson, et al. (1997). "UPPAAL in a Nutshell." International Journal of Software Tools for Technology **1**(1+2).

Lu, S., K.-W. Lee, et al. (1997). Adaptive Service in Mobile Computing Environments. IFIP International Workshop on Quality of Service.

Maes, P. (1987). Concepts and Experiments in Computational Reflection. Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA'87).

Miguel, M. A., J. F. Ruiz, et al. (2002). QoS-Aware Component Frameworks. Proc. 10th International Workshop on Quality of Service (IWQoS'02).

Nahrstedt, K., H. Chu, et al. (1999). "QoS-aware resource management for distributed multimedia applications." High Speed Networks **7**(3-4): 229-257.

Nanda, P. and A. Simmonds (2003). Policy Based Architecture for QoS over Differentiated Services Network. International Conference on Internet Computing.

Pau, G., D. Maniezzo, et al. (2003). A Cross-Layer Framework for Wireless LAN QoS Support. IEEE International Conference on Information Technology Research and Education.

Putman, J. (2000). Architecting with RM-ODP, Prentice Hall.

Roscoe, T. and G. Bowen (2000). Script-driven Packet Marking for Quality of Service Support in Legacy Applications. Conference on Multimedia Computing and Networking.

Rottger, S. and S. Zschaler (2003). CQML+ : Enhancements to CQML. Proc. 1st Int'l Workshop on Quality of Service in Component-Based Software Engineering.

Schiller, J. H. (2003). Mobile Communications, 2nd Esition, Addison-Wesley.

Staehli, R., F. Eliassen, et al. (2003). Quality of Service Semantics for component based systems. proceedings for 2nd International Workshop on reflective and adaptive middleware systems.

Stattenberger, G. and T. Braun (2001). QoS Provisioning for Mobile IP Users: Applications and Services in Wireless Networks, Hermes Science.

Sugawara, T. and K. Tatsukawa (1999). Table-based QoS Control for Embedded Real-Time Systems. ACM Workshop on Languages, Compilers and Tools for Embedded Systems.

UML (2003). "UML Superstructure 2.0, Object Management Group, available at www.omg.org."

UPPAAL (2005). "www.uppaal.com."

Warmer, J. K., Anneke (2003). The Object Constraint Language: Getting Your Models Ready for MDA. Reading, Mass., Addison Wesley.