Quality Information Modeling for QDD Support

Laila Fetjah, Abderrahim Sekkaki

Department of Mathematics and Computer Science
University Hassan II, Ain Chock, Faculty of Sciences
P.O Box 5366, Mâarif – Casablanca, Morocco.

Summary

Nowadays QoS (Quality of Service) architectures have been required to support most of distributed applications, particularly in wide environments like the Internet. In this context, it aims to be more interesting to offer services according to user requirements and expectations. We will be able to talk about Quality Driven Delivery (QDD) when we consider the users expectations in terms of non-functional requirements like cost. In this paper, we propose a framework to support QDD in distributed multimedia systems that enable QoS specification, QoS mapping and more generally QoS decisions in order to provide resources allocation and adaptation. We focus on the quality information management to support QDD while proposing an UML diagram for QoS information model.

Key words:

QDD, QoS and Distributed Multimedia System

1. Introduction

During these last years, the world of business was revolutionized by the appearance of new applications such as electronic commerce, News-On-Demand or Video-On-Demand. These applications allow the companies to reach more customers by offering them an outfit of services, and so increase their productivity and their flexibility. These applications use generally distributed multimedia systems, which have the peculiarity to integrate several media such as sound, images and video sequences. Thus, it would be necessary to supply to these applications some system management mechanisms that must offer QoS support, application adaptation and system scalability [3]. For example, a VoD application needs a certain bandwidth so that the passed on images are correct. System scalability presents the system capacity to evolve according to the met loads. Application adaptation concerns the capacity of the system to change its behaviour according to the variations in the processing environment.

QoS management is a general concept, which represents all the techniques allowing to guarantee that a distributed system offers the QoS level required by users or applications. Generally, a user specifies his or her requirements, which concern system performance, and the system has to deliver the specified level by transforming the user specifications, in constraints aiming at the transport layer [12]. Most of QoS-related research works are interested in resources allocation, few of them are interested in content adaptation or the quality perceived by the user. New approaches have to consider the user in the first rank by taking into account user-perceived QoS characteristics which must be then transformed into constraints targeted the actors of the distributed multimedia system.

We are talking about Quality-Driven-Delivery (QDD)[8], where the objective is to supply services by considering the quality level specified by the user. The QDD allows offering and support levels of service adapted to users requirements by offering them the specification of non-functional requirements. The QDD resumes some QoS activities like specification, monitoring and mapping [8]. Quality information is going to act in these activities and is generally going to come from different sources what make them heterogeneous in their definition, representation and manipulation.

Therefore, the need of homogenization of this information is indispensable. More particularly, an approach based on modeling and model management is imperative for the extension and the adaptation of quality information models. This modelling is going to allow separating QoS management from QoS information management in order to make them independent from the systems that implement them. In the majority of existing architectures (i.e. QoSME, QuO, TAO and OMEGA) [5] [6] [13] [14] [17] [20], information models are defined by an ad-hoc way and remain very dependent on their architectures what makes their interoperability quasi impossible.

In this paper, we will focus on quality information management to support QDD. More particularly we shall propose architecture for QDD management in a distributed context and we shall present an UML model for QDD information management.

The rest of the paper will be organized as follows: Section 2 explains the difference between QoS and QDD by emphasizing on the necessity of directing towards this new notion. Section 3 gives a brief review of some

Manuscript received September 5, 2007
Manuscript revised September 20, 2007

existing QoS architectures by putting the light on quality information management within these architectures. Section 4 describes our proposition of QDD support in distributed multimedia systems. Section 5 presents then our UML model of quality information management for QDD support. Section 6 explains the QoS Mapping activity performed on the information models. Section 7 illustrates some experimental results for a video delivery application used for making QoS decisions. Section 8 suggests our developed prototype for QoS Monitoring and mention a pratical illustration. Finally, section 9 concludes and presents some future works.

2. QoS vs QDD

QoS concept was first introduced in the field of telecommunication networks. Several definitions are associated with it according to the context of use. For example, the QoS of a network represents the assurance of a debit specified beforehand on a data link. While the QoS of a distributed multimedia system represents all the quantitative and qualitative characteristics of this system, necessary to support features required by an application [20].

The ITU [10] defined the directed QoS user "as the collective effect of services performances which determine user satisfaction degree of the service» and the oriented QoS providers " as the capacity of a network or an element of network to assure the functions bound to communications between users ". QoS management consists of five distincts activities [4] :

- Specification: allows defining the QoS level required by users or applications;
- Mapping: consists of mapping the user's requirements onto QoS parameters such as resolution, frame rate or throughput;
- Monitoring: allows supplying a visibility on the system performance.

Before measuring this performance, it is necessary to define performance objectives on the basis of business needs, system workload, and available resources. These objectives would include an acceptable response time, an average debit, and system availability. They are usually formalized in a "service-level-agreement" (SLA) between users and service provider;

- Negotiation: during negotiation, a contract is established between the various layers and components of the distributed multimedia system in order to satisfy the user's requirements;
- Adaptation: refers to the ability of the system to change its behavior according to the variations occurring in the processing environment.

QoS concept has presented some incapacities in distributed multimedia applications which involve several heterogeneous constituents namely: servers, networks, databases, software, media, etc. It would be necessary, besides QoS support, to consider scalability, adaptation and the mobility of equipments in the new QoS management approaches. The QoS information is represented by system offers and user specifications which are generally different. Indeed, user specifications are qualitative whereas system offers are quantitative. A mapping is then necessary to convert qualitative and subjective quality levels specified by the user into quantitative and measurable quality levels, as well as to convert these quantitative quality levels to constraints corresponding to resource requirements for the object delivery.

Having said that, QoS management must be considered in a dynamic way allowing the change of specification or the declaration at any time. Indeed, some researches were led in this sense and allowed to propose new QoS approaches wider than those defined previously [3].

However, the majority of these approaches focus on the performance requirements. More particularly, the performance of networks supporting these distributed multimedia applications for example the debit or the time-answer.

Our work is placed in a more general context called Quality Driven Delivery (QDD)[8] which puts the user at the center of the architecture by allowing him the specification of the performance information but also other informations such as availability, cost or more generally data quality. We are talking about QDD that refers to the capacity of services to deliver objects, while considering the users expectations in terms of non-functional requirements [8]. For example, if the QoS expressed by the user concerns the application response time, the traditional QoS approach will proceed to resources allocation by increasing for example the line debit. Whereas with the QDD, alternately, we can use resources allocation but also data compression techniques or change the data server. We can assume that QDD can be viewed as a generalization of QoS management.

The user delivery service in a distributed environment bases itself on the interactions between the actors of the system namely: user, application, and resources. The quality information is specified at first by the user in a subjective and non formal way according to his perception of the service for example «the sound must be clear and neat ". This qualitative specification must be then transformed into a specification to the application quality as for example Pulses Code Modulation, which defines a coding of audio of 64kbs, or Adaptive Delta Modulation 6, which defines a coding of audio of 56kbps.

What allows defining the resources quality as for example the debit of 64 Kbps.
Quality information associated to QDD acts in several activities: specification, mapping and monitoring. Moreover, this quality information result from several sources what makes them heterogeneous at the level of their definition, size and manipulation. Therefore, there is a need of integration and homogenization of this quality information. More particularly, an approach based on modeling and model management is imperative for the extension and adaptation of the quality information models. This approach will have to provide description, integration and translation of quality information resulting from various sources.

3. Related Works

The first QoS architectures for distributed systems were introduced at the beginning of the 90s [4]. But with the evolution of Object Oriented, other architectures have appeared benefiting from the new tendency of distributed systems: middlewares. Middlewares represents an intermediary between applications and resources.
We shall present in this section four QoS architectures among the most wide spread: QoSME, QuO, TAO and OMEGA. We shall concentrate on QoS Information management within these architectures.
QoSME (Quality of service management Environment), proposed by the University of Columbia [5][6], provides a generic architecture for application QoS demanding and management. It consists of applications that require quality levels in transport layer and in operating systems.
It uses QuAL (Quality of service assurance Language) for the specification of QoS information of QoS. It offers an automatic monitoring mechanism of QoS information based on SNMP agents. Consequently, all QoS activities are based on MIBs managed by SNMP protocol. This makes the base dependent on the SNMP protocol and consequently not stretchable.
QuO (Quality of service Object)[20] is an object oriented QoS architecture. It uses a model implemented in the CORBA layer so exploiting the QoS of the communication layer and allowing applications to adapt itself dynamically to their environment.
QuO defines a set of QoS specification languages such as : QDL (Quality Description Language) [16] based on the CORBA IDL, CDL (Contract Description Language), RDL (Resource Description Language) [20] and SDL (Structure Description Language) [20]. QDLs supports only the numeric and measurable dimensions. QuO divides QoS information into two regions:

- The negotiated region representing the QoS requirement of the user or the system;
- The real region defining the available QoS.

TAO (The ACE ORB) is a QoS middleware architecture proposed by the Distributed Object Computing Group of the University of Washington [17].
It's an Object oriented architecture based on CORBA, more particularly TAO allows the definition of an CORBA infrastructure supporting certain quality levels. TAO offers pre-defined interfaces IDLs allowing the specification of QoS requirements. The available QoS characteristics are strictly related to transport layer and operating system. This information is expressed by integers or real numbers.
OMEGA of the University of Pennsylvania [13] [14] defines a working frame between application QoS requirements and available global or local resources. It concentrates on resource reservation and management by defining two new protocols RTNP (Real Time Network Protocol) at network layer and RTAP (Real Time Application Protocol) at application layer. A QoS broker is also proposed to allow QoS mapping of user interface and to share available resources on the various applications in the ATM network. The QoS broker also allows the negotiation of the applications requirements with the real system offer in order to establish an end to end connection. QoS Information used by OMEGA is classified in three groups: application, system and network. They are stored in a treelike structure. However, there are no relations between QoS dimensions of the same group.
We can classify QoS architectures in two categories: (1) Layers architectures such as QoSME and OMEGA and (2) middleware architectures such as QuO and TAO.
From the point of view of QoS information modeling, we can deduct that:
- The architecture QoSME depends on the SNMP MIB. Furthermore, it does not consider some dimensions relative to the types of multimedia data such as synchronization, size of the sample or the transmission rate;
- The architecture QuO and TAO based on CORBA, that restricts the environment of development to this particular middleware;
- The architecture OMEGA did not consider explicitly QoS information management. Indeed, QoS information is stored in profiles but there is no description of the data structure containing them. Furthermore, resources management are limited to the transport layer. Information models are defined by an ad-hoc way in all the proposed architectures what makes them very dependent on the architectures so raising problems of

extensibility. These architectures did not treat the heterogeneous aspect of distributed systems.
In front of a complex environment such as distributed multimedia systems and by considering the quantities of QoS information, a standard and homogeneous base of QoS information is imperative. This base will have to allow collection, analysis, storage and mapping of QoS information. We thus need to define a generic QoS information models which are going to allow the specification of QoS user requirements. This modeling will have to allow a more effective management of QoS information.

4. QDD Architecture for QoS decisions

Our QDD approach aims to consider QoS user-oriented characteristics and to transform them into constraints at the actor's level of the distributed multimedia system. The figure 1 presents QDD architecture consisted of a QDDManager, a global QDDIB (Quality Driven Delivery Information Base), a local QDDIB, a decision engine and an adaptation & delivery engine.

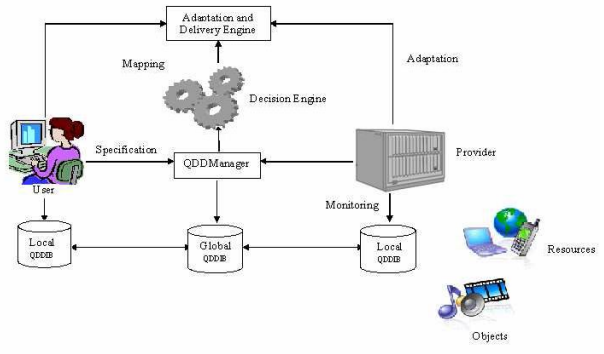


Fig. 1 QDD architecture for QoS decision.

We use architecture at three levels:
- The first level is the user level: it includes the user desire to use a distributed multimedia application but under certain constraints. The user specifies his quality requirements with the QDDManager. He uses a local QDDIB which allows to store the user quality information and describes his requirements and preferences;
- The second level is made up of QDDManager, decision engine and adaptation and delivery engine. The QDDManager contains most of the intelligence and functions of our architecture. It communicates with the users of first level, treats all the system tasks of management and administers the distribution of these tasks on the whole system. It uses for that a data structure called global QDDIB "QDD Information Base", it contains user quality information, media quality information and resources quality information. The global QDDIB allows collection, storage, integration and access to the system quality information. This information will be then used by the decision engine which will have, first, to transform the qualitative specifications of the users into quantitative specifications, then, to make decisions based on specified constraints and to see if it is necessary to make an adaptation of the application content, a resources allocation or adaptation. The Adaptation and Delivery Engine will be responsible for executing the plan produced by the decision engine. It will have to interact with the various components of the system (network, media, data server, etc...) so as to deliver objects respecting user constraints;
- The third level is represented by the service provider that represents at the same time the media and the used resources. This level uses a data structure called local QDDIB. It contains the media and resources quality information. This information is regularly collected further to the provider monitoring process.
Supposing that the user response time increases during a video delivery, causing a violation of the quality contract of service pre-established between the user and the service provider, the decision engine can make several decisions at several levels:
- At the level of resources allocation: increasing the bandwidth;
- At the level of the contents optimization: changing the method of encoding which could use for example data compression;
- At the level of resources adaptation: changing the video server.
The main purpose of contents adaptation is to maintain the service level agreement (SLA) built at the negotiation phase between the user and the service provider. This adaptation shoud be done online and transparently.

5. QDD information Modeling

Our approach is based on a modeling using the UML (Unified Modeling Language) as well as the concept of quality dimensions. The quality dimensions describe qualitative or quantitative information related to the quality levels of the delivery service actors or the specified users one. The qualitative dimensions concern the level of quality from the user perspective whereas the quantitative dimensions are about the measurable quality levels representing the real capacities of the service provider. Our architecture is based on the quality information model described in [8] and which is composed of the user quality model and the actor quality model.

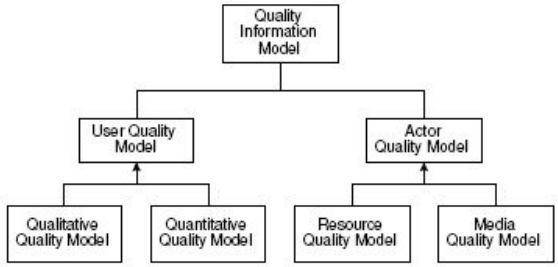


Fig. 2 QDD Information Model.

The user quality model describes the dimensions used to specify the desired quality level. The architecture distinguishes the qualitative quality model, which contains the qualitative dimensions, from the quantitative quality model that groups together the quantitative quality dimensions. The Actor Quality Model groups the quantitative quality dimension along which is described a quality level. The architecture distinguishes the Media Quality Model, which contains the dimensions that represents the quality level of an object to be delivered (image, video sequences, audio, binary data…), and a Resource Quality Model, which describes the quality level offered by a system component (communication network, Database, operating system, storage equipement, video server etc.).

The information models used to build the QDDIB, described previously, has to contain all the quality information resulting from the user specifications, the system resources and the media objects. To generate QDD information and to treat the information models, we define operations such as :

- Derivation: allows generating new models from the existing ones;
- Mapping: allows converting a model to another one;
- Instantiation: allows obtaining information from a model.

In this paper, we propose QDD information modeling based on UML for two reasons: the re-use and the interoperability. Furthermore, the UML modeling language offers several advantages presented by the various approaches, namely abstraction, encapsulation, inheritance and stability.

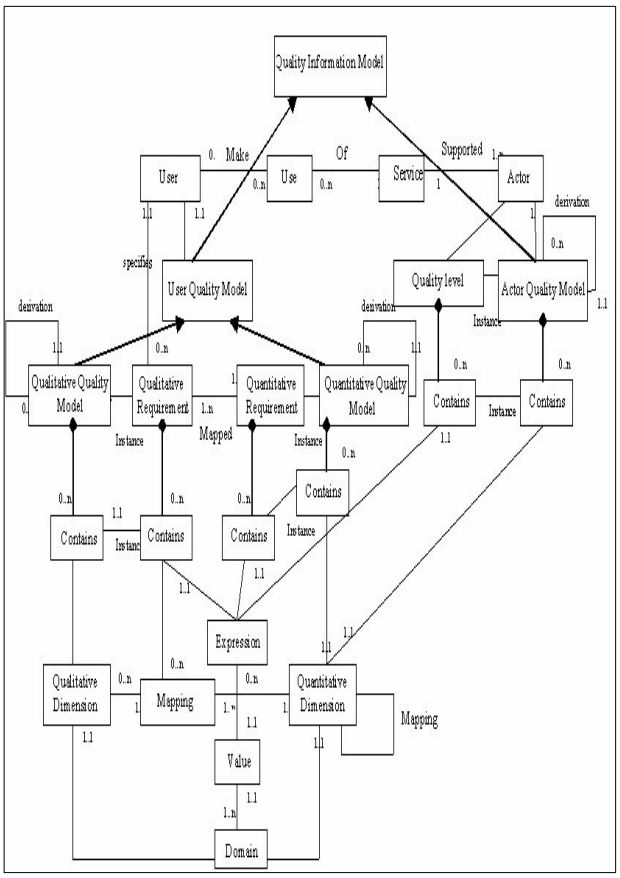


Fig 3. An UML diagram for QDD information model.

6. QoS Mapping

Most of current distributed systems are based on layering architectures such as user, service, system or resource layer, each layer has a specific data. QoS mapping enable translation of data between these layers. Mapping activity is essentiel for making QoS decisions. In [10], the authors identify two types of mapping, vertical and horizontal, used respectively for transforming information between layers and exchanging information between services of the same layer.

In our architecture, these two categories have to be implemented by the mapping between quality information models. Figure 4 presents the Quality Information Model for the Video Delivery Service, it also illustrates the vertical and the horizontal mapping in this application.

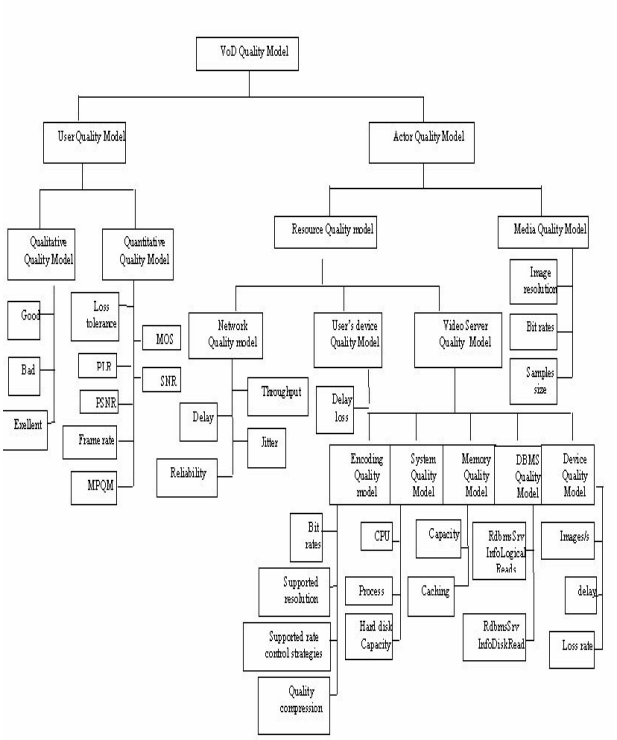


Fig.4 Video on Demand Quality Information Model.

Mapping activity used mapping rules. We classify mapping rules in two categories:

- Mathematics rules: based on mathematics formula between QoS dimensions of the differents quality models.
- Experimentation rules: built by the QoS Manager or the user using experimental tests.

QoS dimensions like packet rate, inter-arrival jitter and end-to-end-delay can be expressed in terms of QoS dimensions of the Network quality model like delay, jitter and packet loss using mathematics rules provided by [9]. Unfortenatly, we do not have a sufficient number of mathematics rules allowing us to make all the mappings in a system. This, led us to use experimentations rules as an alternative method.

7. QoS decisions

We have conducted some expirements for a video delivery application in order to make some QoS decisions. Our objective was to find experiments rules and construct a table that will be used for delivering QoS decisions to our system.

As a first expirement, we have focused on two QoS dimensions: delay and packet loss. When a QoS violation occurs, we have made QoS decisions based on changing network configuration such as bandwidth.

Delay: During a decreasing of a video transmission rate caused by an increasing of delay, we could make these QoS decisions:

- Change codec algorithm, it requires more information about all the codecs algorihms availables in the client and sever side.
- Increase encoding buffer. The amount of delay introduced is related to the size of the encoding buffer.
- Increase server image transmission rate,
- Allocate more memory for this application in the server side,
- Change video server, it requires more information about all the video servers delivering the same service. This decision could be taken if the problem came from the curent server.
- Allocate more bandwith.

These decisions can be reorganised after the QoS monitoring activity that will identify the problem source causing the violation or the interruption of the service.

Loss packet rate: During a decreasing of a video quality caused by an increasing of the loss packet rate, we could make these QoS decisions:

- Change the transport protocol (TCP is more reliable than UDP) ;
- Increase client buffer;
- Use a more reliable codec. In [19], the authors said that prediction and variable length coding makes the video stream very sensitive to transmission errors on the bitstream.
- Provid error correction mechanisms. Techniques like retransmission or forward error correction could be used.

8. QoS Monitoring

QoS Monitoring is one of the most important activities in QoS Management. It allows the construction of QoS information models as well as the supply of QoS information base produced by these models. It can also be used in order to identify the problem source when a QoS violation occurs. This can lead to an efficient QoS decisions. At first, we realized a prototype that allows supplying the actor quality model. The language choosen for the development of our prototype is Java. Indeed, the objectives of our architecture could not be reached without choosing the suitable language program capable of responding to all these requirements.

Our prototype has the peculiarity to be distributed. SGBDMON and its various agents are localized on distant sites.

We choose **RMI** protocol to realize the connection between SGBDMON and its various agents. The prototype uses the SNMP protocol to measure QoS parameters of the system.

The purpose of the developed prototype is to validate the architecture proposed previously. Nevertheless, to attain that, we have performed tests on a SGBD Oracle that represents the server database of a distributed multimedia application. It worth noting that as the prototype uses the SNMP protocol, then it can measure whatever equipment carrying this protocol to allow afterwards the extraction of performance information from the network and database.

When a request of performance information arrives at SGBDMON, it contacts the agent responsible for the performance information management of the component to be tested.

After that, the agent contacts the QoSSNMPAgent in order to get an answer for the SGBDMON request. The QoSSNMPAgent uses an under agent SNMP to extract the performance information from the appropriate MIB. Then it returns the asked value to the agent. This last sends back the result to SGBDMON.

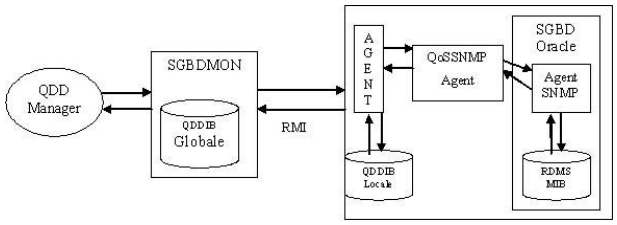
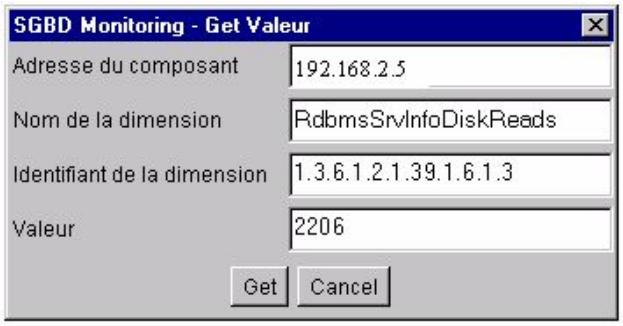


Fig 5. Distributed architecture for the prototype of measure

Example:



In this example we show how we can extract RdbmsSrvInfoDiskReads the QoS dimension of an Oracle database that could include a video sequence.

9. Conclusion and perspectives

This paper has presented our QDD architecture for the QoS decision-making which rank the user in the center of the architecture by offering him the specification of non-functional constraints and by allowing him besides resources allocation, respect of his requirements, and content adaptation in real-time. Then, we have presented an UML diagram for QDD information model independent from the implementation that enables mapping, derivation and intanciation. Afterwards, we have presented the developed prototype to supply the information base of the QDD actor.

Our objective, in the future, will focus on mapping made by the decision engine. In particular, we shall have to define inter and intra model mapping allowing the decision-making for purposes of allocation or adaptation. As the QDD concentrates on models rather than QoS information, we propose to define a mapping model, always in order to have interoperability and independence of implementation, allowing mapping representation, construction of mapping rules, use of mapping algorithms, as well as, optimization of the mapping process. Precisely, we will continue the developement of the prototype, based on the QDD information model presented in this paper, in order to build an automatic and online QoS mapping rules Builder.

10. References

[1] Aagedal, J.O, Ecklund, E. F. Jr., "Modelling *QoS: Towards a UML Profile*". Springer LNCS 2460, ISBN 3-540-44254-5, pp. 275-289.

[2] Abdelzaher, T., Shin, K., Bhatti, N., "*User-Level QoSAdaptive Resource Management in Server End-Systems*", IEEE Transactions on Computers, Vol. 52, No. 5, May 2003.

[3] Bochmann G., Kerhervé B., Lutfiyya H, Salem M, Ye H. "Introducing *QoS to Electronic Commerce Applications*". In Proceedings of the 2nd International Symposium on Electronic Commerce (ISEC), Hong-Kong, 26-28 April, 2001, Proceedings published by Springer-Verlag.

[4] Campbell, A.T., "A *Quality of Service Architecture*". Ph.D thesis, Lancaster University, England, January 1996.

[5] Florissi, P. and Yemini, Y., "Management *of Application Quality of Service*", Technical Report DCC-03-94, Columbia University, 1994

[6] Florissi, P., "QuAL*: Quality Assurance Language*", Ph.D. Thesis, Columbia University, 1996

[7] Foster, I., Roy, A., Sander, V. "A *Quality of Service Architecture that Combines Resource Reservation and Application Adaptation*", 8th International Workshop on Quality of Service, 2001.

[8] Gerbé, O., Kerhervé, B. Srinivasan, U. , " *Model Operations for Quality-Driven Multimedia Delivery* ", International Conference on Conceptual Structures, ICCS 2003, Dresden, Germany, July 21-25, 2003.

[9] Huard J.-F., Lazar A. A, "*On End-to-End QoS Mapping*", In Proceedings of the International Workshop on QoS, May 1997, 303-314.

[10] ITU-T Recommendation E.800, "*Terms and Definitions Related to QoS and Network Performance Including Dependability*," 1994.

[11] Kerhervé, B., Nguyen, K., Gerbé, O., Jaumard, B., "*A Framework for Quality Driven Delivery in Distributed multimedia Systems*", ICIW'06 Conference, Guadeloupe, French Caribbean, 2/2006.

[12] Li, B., Kalter, W., Nahrstedt, K, "*A Hierarchical Quality of Service Control Architecture for Configurable Multimedia Applications*",Jour. of High Speed Networks, IOS Pres, 2001.

[13] Nahrstedt, K., "An *Architecture for End-to-End Quality of Service Provision and its Experimental Validation*". Ph.D. Thesis, University of Pennsylvania, Philadelphia, 1995.

[14] Nahrstedt, K. and Smith J. M., " *Design, implementation and experiences of the OMEGA end-point architecture* ," IEEE JSAC, Special Issue on Distributed Multimedia Systems and Technology, 14(7):1263–1279, September 1996.

[15] Nguyen K., Fetjah L., Kerhervé B., 2001. " *Quality of Service Information Base Manager for Electronic Commerce Applications* ". Poster presented at the CITR Annual Conference, Aylmer, Canada, August 2001.

[16] Pal, P.P, Loyall, JP., Schantz, RE., Zinky, JA., Shapiro, R. and Megquier, J. "*Using QDL to specify QoS aware distributed (QuO) application configuration*". In Proceedings of ISORC 2000, editor, The Third IEEE International Symposium on Object-Oriented Real-time Distributed Computing, Newport Beach, CA, March 2000.

[17] Schmidt, D., Levine, D., and Cleeland, C. "Architectures *and Patterns for High- performance, Real-time CORBA Object Request Brokers* ". In Advances in Computers, Marvin Zelkowitz, Ed., Academic Press, 1999.

[18] Vogel, A., Kerhervé, B., Bochmann, G. v., & Gecsei, J. (1995). " Quality of Service Management: a survey" . IEEE Journal of Multimedia Systems, 2(2):10-19, 1995 .

[19] Y. Wang, J. Ostermann, Y. Q. Zhang, "Video *Processing and Communications*", Prentice Hall, 2002. Chapters 9,11,13

[20] Zinky, J., Bakken, D. and Schantz, R., "Architectural *Support for Quality of Service for CORBA Objects*", Theory and Practice of Object Systems, January 1997.