QoS Control and Adaptation in Distributed Multimedia Systems

Farid Naït-Abdesselam[1,2], Nazim Agoulmine[2]

[1] Advanced Communication Engineering Centre
Department of Electrical and Computer Engineering
The University *of* Western Ontario, London Ontario N6A 5B9, Canada.
Email: fnait@engga.uwo.ca.

[2] PR*i*SM Laboratory, University of Versailles
45 Avenue des Etats-Unis 78 000, Versailles. France.
Email: {naf, naz}@prism.uvsq.fr

Abstract. Presently, many distributed multimedia systems adapt to their changing environments and Quality of Service (QoS) requirements by exchanging control and feedback data between servers and clients. In order to realize a more flexible adaptation to QoS in distributed multimedia systems, this paper seeks to present a new approach, based on distributed software agents located in both network nodes and end-systems. By having a good knowledge of local resources at each component, and with their capabilities to communicate in order to share their knowledge, the distributed software agents can alleviate the major fluctuations in QoS, by providing load balancing and resource sharing between the competing connections. In order to show the feasibility of our active adaptation approach, simulations have been conducted to adapt a delay sensitive flow such as distributed interactive virtual environment. We have performed our evaluations for short and long range (i.e. self-similar) traffic patterns. Preliminary results show a viable system, which exhibits a smooth and noticeable improvement in perceptual QoS during a heavy loaded network. In addition, our results indicate that the network and the application have more to benefit from the algorithm, when the traffic exhibits long range dependence behavior.

1 Introduction

Recent years have seen great advances in computing and networking technology. At the network level, new high-speed technologies such as *Asynchronous Transfer Mode (ATM)* are actively being deployed in local, metropolitan and wide area network environments. These networks not only have the capability of transmitting information at high speed, but also have the potential to offer a wide range of *Quality of Service (QoS)* properties including bounds in delay, guarantees and isochronous communications.

Multimedia workstation technology for generating, processing and displaying streams of digital video and audio is also available in the marketplace with very high capacity storage systems and real-time video and audio codecs.

These technological developments are complemented by new user perspectives. New classes of distributed applications have been developed, such as Distance Learning, Desktop Video-Conferencing, Virtual Workshop and Video On Demand. These applications are characterized by their highly interactive nature and their significant use of multimedia information transfer. In these applications, communication requirements are extremely diverse and demand varying levels of service in terms of parameters such as latency, bandwidth, jitter, and loss rate [1]. These various applications share the need for *Quality of Service control and management* to ensure that the requirements of the users are satisfied [2]. However, even with service guarantees, fluctuations in the Quality of Service may occur because of shortage of resources, e.g., network congestion or switch failure.

In this work, we propose an adaptation approach that allows automatic recovery, from violations in Quality of Service, by deploying a new allocation policy to distribute the level of QoS, that

should be supported by all the components involved in meeting the end-to-end requirements of a multimedia connection. This approach, based on the mutual help between all the components involved in the support of the requested service, allows to reach a higher resource utilization in the underlying system, increase the probability of admitting a new connection in the network, and reduce the probability of violations in the negotiated Quality of Service.
The approach undertaken is based on distributed software agents, located on all the components involved in providing QoS guarantees [3]. We associate for each component of the Distributed Multimedia System an Agent called *QoS-Agent (QoSA)*. Hence, the Distributed Multimedia System can be seen as a multi-agent system, able to communicate and evaluate, , in the duration of the connection, the values of the used QoS parameters applicable to the resource utilization. Close cooperation between the agents is required to improve the Quality of Service of a given application, or react to violations that may occur. In our designs, we have taken into consideration the traffic volumes produced by the agents, and we have devised algorithmic approaches that reduce these traffic volumes. In order to assess the capabilities and behavior of the proposed scheme, we have applied it to a delay sensitive application such as video or virtual reality application, operating in a distributed environment. Our results have indicated that the proposed approach reduces the frequency of occurrence of violations significantly, it enables the system to recover very fast when a violation occurs, and improves the utilization of the underlying resources.
The remainder of this paper is organized as follows. Section two gives a short related work on QoS adaptation. Section three describes the agent-based approach for QoS adaptation, the inter-agents communication protocol and the exchanged. In section four, we describe our simulation results and analysis. Finally, in section five we present our conclusions.

2 Dynamic QoS Adaptation

In the literature there are many proposals to deal with QoS adaptation. Most of them highlight the need for QoS degradation or re-negotiation, in case of violations in the initially agreed Quality of Service [5][4][8]. For instance, [5] exploits the existing hierarchical coding techniques used by video coders to make a range of adaptation strategies, by selectively adding or removing coding layers in case of fluctuations in the network bandwidth. In [4][8], the approach undertaken is to adapt the application to the service provided by the network, by using information reports sent by receivers to detect the congestion state of the network, and then to react accordingly by regulating the sending rate. The approach considered in our work follows another avenue since it has the capabilities to keep, when it is possible, the agreed QoS provided to the users without any degradation in the service. The approach provides dynamic QoS adaptation, using dedicated active software agents, to balance resources between the components (nodes and end-systems) involved in the connections experiencing violations. A similar research work is presented in [7], which also uses the agent paradigm to deal with QoS adaptation in distributed multimedia systems. The major differences between [7] and our work are the following; the distributed architecture proposed in [7] uses a ring approach. When an agent detects congestion, it sends a violation message, along with the violation degree to the neighbouring network component. The agent of this component, forwards the violation message to the next in the ring, after attaching the maximum amount of resources, the component can provide. The process repeats, until the message returns to the node of the initiating agent. After that, the latter sends along the ring a message that includes the allocations. This approach requires the use of variable size messages. The recovery interval is twice the amount of the time that takes to go around the ring. Also, in case the messages are lost due to some overflow or network failure, no action will be taken. In addition, the recovering time of the approach is unbounded, and depends on the characteristics of the network topology (e.g. number of nodes in the ring, speed of links etc.) and the network load. This makes the method

unable to take into consideration the actual QoS requirements of the applications. Our method is not based on the use of a ring topology. Violation messages are forwarded in both directions (forwards and backwards) and receiving agents reply immediately to the request. This increases the reliability and fault tolerance of the method (since loss of several reply messages will not halt the process). In addition to that, in our case, the agents have to respond within a limited amount of time and the message is only forwarded up to a fixed amount of hops depending on the load of the network. This maintains good relevance between derived solution and the present state of the connection or the network. Finally, our method uses timers, in order to limit the time required for transmission and recovery and imposes an upper bound on the waiting time to a certain upper bound. This guarantees the fast provision of effective solutions for recovery of violations.

The major motivation for dynamic QoS adaptation, using dedicated software agents in both network nodes and end-systems, is to explore the dynamic behavior of multimedia connections in order to improve resource utilization and increase the number of connections satisfied by the network. Dynamic QoS adaptation differs from the conventional management functions due to its real time characteristics. For instance, a connection which is experiencing degradation on QoS in a specific node would have to be recovered as soon as possible. For connections with Variable Bit Rate (VBR) pattern, QoS fluctuations can be so frequent that they should be handled in way that is transparent to the users; therefore, this action should be left to the QoS management system [6].

3 An Agent-based Approach for QoS Adaptation

This section presents our proposed approach for QoS adaptation based on software agents. The approach is based on distributed *QoS-Agents (QoSA)* located on all the components involved in providing QoS guarantees (Fig.1). Thus, the distributed multimedia system can be seen as a multi-agents system able to communicate and evaluate, at each instant, the values of the used QoS parameters applicable to the resources utilization.

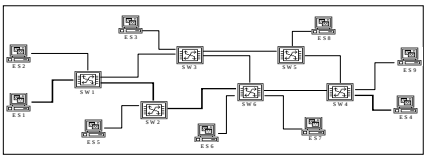


Fig. 1. Distributed Virtual Environment System Configuration.

The use of an agent-based infrastructure for Quality of Service adaptation seems to be a good choice for implementing several adaptation strategies, in which available resources need to be re-distributed in order to maintain satisfaction of the end user requirements [7][10]. Such operations, in general, imply an intense (and sometimes complex) exchange of messages among the parties involved. The distributed software agents can therefore adequately support such kind of functions, and permit more flexibility in performing them.

Particularly interesting features, in performing this kind of operations, is the exploitation of agents' cooperation for optimization purposes. Each agent (QoSA) is able to collect information about nearest agents, and then complement the partial knowledge concerning the state of its own resources. As a result, each agent (QoSA), although autonomous, is influenced by, and can influence, the behavior of other agents.

When providing end-to-end QoS, each component involved in a specific multimedia connection has to contribute with certain guarantees on its own resources. If a component violates such guarantees, the associated (QoSA) will detect this and cooperation among involved agents will start in order to reassign resource reservations to different components. Agents whose resources are not

fully utilized may reserve additional resources, in order to compensate for the violations of other agents.

3.1 Agents-based Distribution Scheme

We introduce here the principle of the Agents-based Distribution Scheme (ADS), which is proposed in this paper. The ADS is an adaptation scheme that helps to maintain the initially agreed Quality of Service for connections, even when one or more components involved in the configuration of interest failed to meet their commitment. It does this by re-allocating resources in a dynamic fashion and, in such a way that the end-to-end QoS requirements still unchanged. While the ADS principle can be used for any type of configurations, we focus in this work only on configurations involving a single stream of information.

When a QoS violation occurs in a particular component, the associated QoSA tries to find a local solution to the problem (e.g. modify locally the parameters and re-assign resources between all the connections) [11]. If it is not possible, this QoSA initiates a dialogue with other agents, involved in the same connection, and requests additional resources that can compensate for the current violation. If the overall system does not have enough resources to recover from the violation, the end-application is notified to make a self-adaptation [12], or to re-negotiate new Quality of Service parameters [9].

3.2 Agents Communication Protocol and Operations

In this section we give a short description of the operations that a QoSA performs, the inter-agents communication protocol, and the supported messages that are exchanged for the purpose of QoS adaptation and recovery.

Structure of the Messages

We have defined three services to support the interactions between agents: *QoS_Request* message, *Available_QoS* message, and *Allocate_QoS* message.

QoS_Request (Sender_Id, Connection_Id, Violation_Type, Requested_QoS, Hops)

This message is issued by the agent, which is experiencing violation in a specific connection. The message specifies the identification of the agent (Sender_Id), the link on which the violation occurred (Connection_Id), the type of violation (Violation_Type), as well as the extra QoS needed for QoS adaptation (Requested_QoS). Note that in this work, the extra QoS requested is in terms of delay and is expressed in ms. The Connection_Id and the Violation_Type uniquely identify each request for extra QoS, and it is local for each QoSA.

In order to inform the other neighboring agents involved in the connection, that segment of the connection going through the specific network element experiences violation, the agent sends the message in forward and backward directions to ask for extra QoS (See Fig.2). At this point we have to comment on the term neighboring. Every agent must solve and recover from the violation in a certain period of time. If this time period expires and the adaptation is not achieved, the service might be severely degraded. Therefore, the agent decides how far will send the QoS_Request message, and the decision is based on the application's requirements, and the characteristics of the connection. The Hops parameter in the message is used for that purpose, and it is set at the beginning equal to the maximum number of agents (components) that an agent, experiencing violation, should contact and ask for recovery. The value of this parameter is decremented by 1 every time another agent receives the message. The message will be either forwarded if the field is decremented to a non-zero value, or discarded if it is decremented to zero or has reached an end-system.

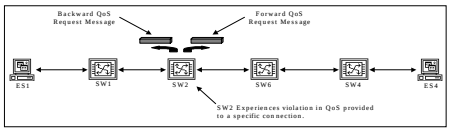


Fig. 2. QoS Request messages sent by the agent located in *SW2* to the participating agents in the connection.

Available_QoS (Sender_Id, Receiver_Id, Connection_Id, Violation_Type, Available_QoS)
The agents that have received a QoS_Request message from the agent experiencing violation issue this message, stating the amount of QoS that each of them can give in order to help the connection to recover from the violation (See Fig.3). In the parameters of this message, the amount of QoS that can be provided is included (Available_QoS). This helps the sender of the QoS_Request message to determine the exact amount of QoS that is needed, and it does so by applying the algorithm described and developed in [10].

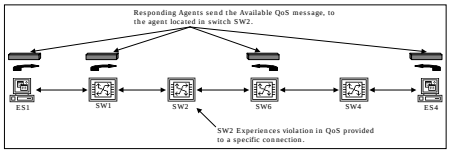


Fig. 3. Available QoS messages sent by all the agents willing to help recover the violation in the switch *SW2*.

Allocate_QoS (Sender_Id, Receiver_Id, Connection_Id, Violation_Type, Allocated_QoS)
The agent, that has previously detected the violation, will send this message to all the agents that have responded to the QoS_Request message, asking for a specific amount of QoS from each of them in order to resolve the violation (See Fig.4). Its request is uniquely identified to distinguish between requests originating from different nodes or different QoS types. This distinction can be made using the combination of Connection_Id and Violation_Type.

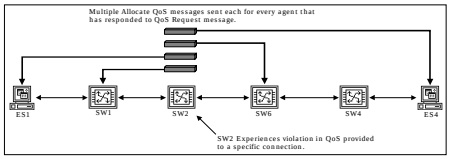


Fig. 4. Allocate QoS messages sent by the agent in the switch *SW2*, to the agents that have responded for recovery.

4 Simulation Results

In order to test our mechanisms within the multi-agents system, we ran several experiments using extensive simulations. The simulations are based on the Optimization Network Tool (OPNET) software. The simulated network configuration is similar to the one presented in the Fig.1. As controlled application, we took a model that best characterized a distributed interactive virtual application with frequent and timely sensitive traffic. The main QoS metric measured during the experiments was the end-to-end delay experienced by the traffic going between the two end-systems *ES1* and *ES*4. The traffic between these two end-systems goes through the switches *SW1*, *SW2*, *SW6*, and *SW4*, which constitutes the route of the controlled connection. The required value of this delay was set to 120 ms. As depicted in the Fig.1, there is also other traffic that is multi-plexed in order to reach a high network load and to interleave the traffic considered, so that the packets are not all consecutive.

We have conducted evaluations for two different types of background traffic. The first follows the Poisson distribution. The second traffic model belongs to the traffic models that exhibit self-similarity and long-range dependence. The models of this traffic have been developed in our lab and are based on the use of alpha-stable distributions. We have shown [13] that these models are more accurate as compared to models proposed earlier (e.g. Fractional Brownian Motion [14] or MMPP [15]). Below we present and discuss our results.

4.1 Evaluation under Poisson Traffic

Our mechanisms, introduced to control the QoS metric delay and adapt to major fluctuations, have shown their major utility and their real effectiveness. The results are shown in Figs. 5 and 6. We have noticed during our simulations that the best results are reached when the agents probe the system every 0.01 ms (see Fig. 5 and Fig. 6). The results presented in Fig.5 are obtained by loading all the switches in our configuration (presented in Fig.1) to almost 100% of their capacity. In Fig.6, we present results obtained by loading the switches *SW2* and *SW4* at 75%, and all other switches at 100% of their capacity. Those different cases of load allowed us to test the accuracy of the mechanisms. As can be seen from Figs. 5 and 6, the proposed scheme obtains a good improvement in terms of end-to-end delay experienced by the traffic considered.

From the results reported in Fig.9(a), we also notice that the proposed algorithm reduces considerably the cell losses experienced by the aggregate traffic. The losses went down from almost 30% without implementing our mechanisms, to 2.5% when the mechanisms are used at their optimum rate. We can also see in Fig. 9(a) that our mechanisms show a much better effectiveness when the network is overloaded and when violations are more frequent. The violation of transit delay has been recovered in most of the cases for the considered service, and higher resources utilization is obtained in the underlying system.

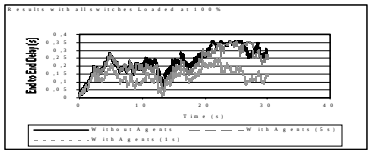
4.2 Evaluation under Self-Similar Traffic

We present in this section the results of our simulations using a self-similar traffic model [13]. Recent studies [14] have demonstrated that self-similar models that show long range dependency are more appropriate for describing the traffic behaviour in today's networks. The results we have obtained were expectable, in the sense that we had more violations occurring than in the case of Poisson traffic.

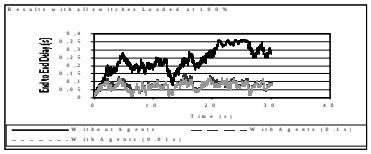
The simulations we have conducted follow the same path as the ones presented in the previous section. We have loaded first all the switches to reach 100% on average of network load (results presented in Fig.7), and then we loaded *SW2* and *SW4* at 75% and all other switches at 100% (results presented in Fig.8).

In Fig.7 and Fig.8, we can see that under self-similar traffic, we reach very soon a violation in the expected end-to-end delay of our controlled connection and we keep having violations during the whole session. However, by applying the mechanisms implemented by our distributed software agents, the violations start to be reduced. By probing the resources every 0.01s, we reach always a better improvement, and a satisfaction of the end-to-end delay is obtained.

The plots shown in the Fig.9(b) give us an idea about how much we should pay in terms of cell losses, in order to keep a respectable end-to-end delay to our controlled connection. We can see that a remarkable improvement in reducing cell losses is obtained for the aggregate traffic going into the network, and at the same time we keep the quality of service of the connections at a higher level. By using the mechanisms implemented by our distributed software agents, the cell losses went down from almost 65% to 15%. This means that we pay less by using the new mechanisms to control the underlying resources.

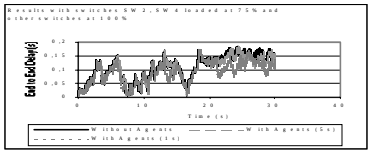


(a) Without Agents, with Agents (5s), and with Agents (1s)

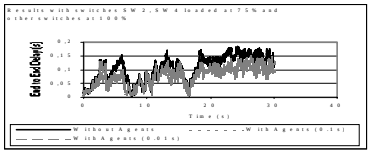


(b) Without Agents, with Agents (0.1s), and with Agents (0.01s).

Fig. 5. Measured End to End Delay between *ES1* and *ES4*, with all switches loaded at 100%.

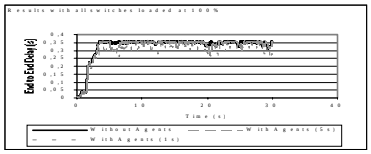


(a) Without Agents, with Agents (5s), and with Agents (1s)

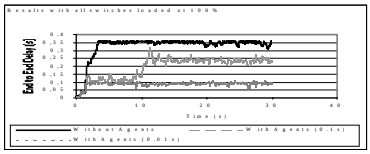


(b) Without Agents, with Agents (0.1s), and with Agents (0.01s)

Fig. 6. Measured End to End Delay between *ES1* and *ES4*, with *SW2* and *SW4* loaded at 75%.

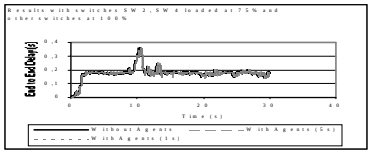


(a) Without Agents, with Agents (5s), and with Agents (1s)

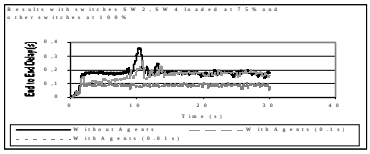


(b) Without Agents, with Agents (0.1s), and with Agents (0.01s)

Fig. 7. Measured End to End Delay between *ES1* and *ES4*, with all switches loaded at 100%.

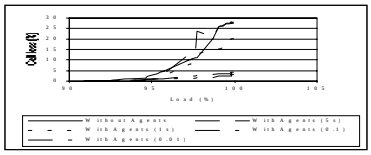


(a) Without Agents, with Agents (5s), and with Agents (1s)

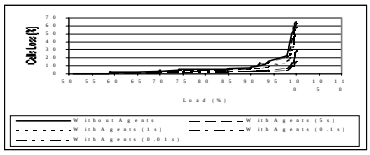


(b) Without Agents, with Agents (0.1s), and with Agents (0.01s)

Fig. 8. Measured End to End Delay between *ES1* and *ES4*, with *SW2* and *SW4* loaded at 75%.



(a). With Poisson Traffic.



(b). With Self-Similar Traffic.

Fig. 9. Overall Cost with different Network Load.

5 Conclusion

In this paper, we have presented an approach to QoS adaptation in distributed multimedia applications. The approach is based on distributed software agents able to communicate and cooperate in order to achieve a better resource utilization, and then maintain the end-to-end QoS requirement of multimedia connections. We believe that the use of an agent-based system infrastructure for QoS adaptation is a good choice for implementing several adaptation strategies in which available resources need to be re-arranged and then keep the user satisfied with the service in use.

When providing end-to-end QoS, each component involved in the distributed multimedia application has to contribute with certain guarantees on its own resources. If a component violates such guarantees, an agent will detect the violation and a cooperation among involved agents is started

in order to recover the violation. This can be done, for instance, by reassigning resources in such a way, the user will not notice the degradation.

The multi-agents system presented in this paper has shown his effectiveness by applying the mechanisms described to recover the delay violation of a delay sensitive traffic, such as one generated by a distributed interactive virtual environment application or video application. In our experiments we have considered two different types of background traffic. The first follows a Poisson distribution. The second traffic exhibits long-range dependence and is based on the use of alpha-stable distribution. Our results have indicated that the proposed approaches reduce the frequency of occurrence of violations significantly, it enables the system to recover very fast when a violation occurs, and improves the utilization of the network resources. However, the distributed software agents implementing the QoS-Weighted Distribution Algorithm, show their relevant utilization in the case of long burst traffic, as they try to balance as much as possible the underlying resources and therefore be able to consume the bursts.

References

1. A. Campbell, G. Coulson, F. Garcia, D. Hutchison and H. Leopold. *Integrated Quality of Service for Multimedia Communications*, Proc. IEEE INFOCOM'93, San Fransisco, USA, April 1993.
2. G.v. Bochmann, B. Kerherve, A. Hafid, P. Dini and A. Pons. *Architectural Design of Adaptive Distributed Multimedia Systems*, Proc. Of the IEEE International Workshop in Distributed Multimedia Systems Design, Berlin, Germany 1996.
3. F. Naït-Abdesselam, N. Agoulmine. *QoS Management Scheme for Distributed Multimedia Applications*, Proc. IFIP TC6 NIPS'97, Sofia , Bulgaria, Oct 1997.
4. I. Busse, Bernd Deffner, H. Schulzrinne. *Dynamic QoS Control of multimedia applications using RTP*. Computer Communications, vol19, Jan 1996.
5. Campbell, A.T. and G. Coulson. *A QoS Adaptive Multimedia Transport System: Design, Implementation and Experiences*, Distributed Systems Engineering Journal, Special Issue on Quality of Service, Vol. 4, pg. 48-58, April 1997.
6. Aurrecoechea, C., Campbell, A.T. and L. Hauw. *A Survey of QoS Architectures*, ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol. 6 No. 3, pg. 138-151, May 1998.
7. A. Hafid and G.v. Bochmann. *Quality of Service Adaptation in Distributed Multimedia Applications*. ACM Multimedia Systems Journal, volume 6, issue 5, 1997
8. J-C. Bolot et al. Scalable feedback control for multicast video distribution in the internet. ACM SIGCOMM'94, Oct 1994.
9. H. Zhang and E. Knightly. RED-VBR: A Renegotiation-Based Approach to Support Delay-Sensitive VBR Video. ACM Multimedia Systems Journal, May 1997.
10. F. Naït-Abdesselam, N. Agoulmine and A. Kasiolas. Agents-based approach for QoS Adaptation in Distributed Multimedia Applications over ATM Networks. International Conference on ATM, Colmar, France, juin 1998.
11. R. Nagarajan, J.F. Kurose, D. Towsley, *Allocation of Local Quality of Service Constraints to meet End-to-End Requirements*, Proc. of IFIP Workshop on the Performance Analysis of ATM Systems, Martinique, Jan. 1993.
12. C. Diot, C. Huitema, T. Turletti. *Multimedia Application should be Adaptive*. HPCS Workshop. Mystic (CN), August 23-25, 1995
13. J. R. Gallardo, D. Makrakis and L. Orozco-Barbosa. *On the Use of Alpha-Stable Self-Similar Stochastic Processes for Modeling Traffic in Broadband Networks*, submitted to the IEEE Trans. on Communications.
14. S.Ben-Slimane and T. Le-Ngoc. *A Doubly Stochastic Poisson Model for Self-Similar Traffic*. Proc. IEEE ICC'95, Seattle, June 1995, pp. .
15. W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. *On the self-similar Nature of Ethernet traffic* (extended version), IEEE Trans. on Networking, no 1, pp 1-14, (1994).