In: Proc. of the 4th IEEE Int. Conf. on Multimedia Computing and Systems (ICMCS'97), Ottawa, Canada, pages 303 - 310, IEEE Computer Society Press, June 1997.

Cooperative QoS Management for Multimedia Applications

Stefan Fischer[1], Abdelhakim Hafid[1], Gregor v. Bochmann[1] and Hermann de Meer[2]

[1]University of Montréal, DIRO, C.P. 6128, succ. Centre-Ville,Montréal (PQ) H3C 3J7, CANADA
Email: {fischer,hafid,bochmann}@iro.umontreal.ca

[2]University of Hamburg, CS Department, Vogt-Koelln-Str. 30, 22527 Hamburg, GERMANY
Email: demeer@informatik.uni-hamburg.de

Abstract

Quality of Service (QoS) management becomes more and more important, especially in networks where many applications are competing for a limited number of resources. As these applications become more complex (consider e.g. multiparty multimedia applications), the number of options for QoS management increases, leading to more complex decision processes. In this paper, we propose an approach for cooperative QoS management, where application-oriented QoS agents are distributed throughout the network and the end systems, communicating with each other. This distributed management system tries to guarantee the QoS level negotiated with the users, at the same time optimizing resource usage. The advantages of distributing the management process are (i) an easier and more precise localization of the cause of QoS problems, (ii) better knowledge of local situations, (iii) a lower complexity for a single QoS agent and (iv) an increase in possible actions. We describe management procedures for QoS negotiation, adaptation and renegotiation.

1. Introduction

The design of distributed multimedia applications, such as systems for access to remote multimedia databases or teleconferencing, requires careful consideration of quality of service (QoS) issues, because the presentation quality of live media, especially video, requires relatively high utililisation of networking bandwidth and processing power in the end systems. For applications running in a shared environment, the allocation and management of these resources is an important question, although most existing systems are based on a best-effort approach.

Best-effort approaches are not suitable for distributed multimedia systems, in general, because some users may be ready to pay some higher price for obtaining a maximum quality, while others may prefer low-cost presentations with lower quality. In addition, for a teleconferencing application involving many users, a single quality of service level may not be appropriate for all participating users, since some users may participate with a very limited local workstation which cannot provide for the quality which is adopted by the majority of the conference participants. We therefore adopt the premise that different levels of quality, often corresponding to different levels of cost, must be provided in the context of distributed multimedia applications.

Much work on QoS has been done in the context of high-speed networks in order to provide for some guarantee of quality for the provided communication service, which is characterized by the bandwidth of the media stream and the delay, jitter and loss rate provided by the network. More recently, QoS have been considered in a more global context, including also the end systems, such as the user's workstations and database servers. Various global QoS architectures have been developed (for a recent overview see [1]), which include also functions for performance monitoring, resource allocation and QoS management. For instance, in previous work [8], we have developed a framework for QoS management of distributed multimedia applications which stresses two points: (a) the user should define (through a suitable user interface for QoS negotiation) the criteria which are used by the system to select the "best" system configuration for the application at hand, and (b) the selection of an appropriate system configuration is the first step of the QoS management process, followed by resource reservation and commitment, which is performed during the initialization of the multimedia application and each time a QoS renegotiation is required.

A prototype system has been developed which implemented the above ideas for the application of remote access to multimedia databases [7]. In this context, it was assumed that, for a given monomedia component of a multimedia document, such as a video clip, there may be several variants with identical "content", but with different QoS characteristics, possibly stored in different continuous-media file servers. During the initial access to the document, the QoS manager would select the most suitable variant to be presented to the user, and in case of network congestion, for instance, with the video server in

question, another variant may be selected which resides in a different server. The negotiation process which leads to the selection of a variant involves three parties: (1) the database server, which contains the meta-information of the documents including all existing variants, (2) the network and (3) the user workstation, which knows the user's preferences and may also impose certain QoS restrictions.

In this paper, we consider multimedia applications including multicasting to many users, such as teleconferencing or educational applications. In this context, the global QoS management approach which involves a few system components, as described above for remote database access, is not workable any more, because the number of users involved is too large for a global management approach. Instead, it is necessary to distribute part of the QoS management process and allow each user process to make certain QoS decisions based on its local context. As discussed above for access to multimedia documents, we assume that different variants (of video streams, for instance) are available to the users' workstations throughout the network by means of multi-casting. In the case of a tele-teaching application, for instance, the video showing the teacher may be available in several variants, differing in frame rate, color quality and/or resolution. Each student may then select the video stream which is most appropriate given his/her preferences and the capacities of the local workstation.

In this paper, we present our solution for a distributed or cooperative QoS management. In Section 2, we first introduce a sample application and use it to show that current QoS management schemes have some shortcomings. Then, in Sections 3 and 4, we describe our management scheme in terms of architecture and communication protocols between the distributed parts. Section 5 describes some strategies that could be followed by QoS management, i.e. it discusses in which situation to take which action. The mechansisms provided by the protocols in the given architecture may be used according to the chosen strategy. Section 6 concludes the paper.

2. Multimedia applications and QoS management

2.1 An example application: remote teaching

The *remote-teaching application* we use as a base for our investigations supports the delivery of a lecture from a given site to students located in several remote locations. The delivery consists of video and audio from the lecturer. In addition, the lecturer may present multimedia documents stored locally or in some other locations. Students have the possibility to ask questions, but they first have to get permission to do so. In this prototype, we allow only one student to talk at a time. The lecturer is always allowed to talk. The overall structure of this application is visualized in Figure 1.

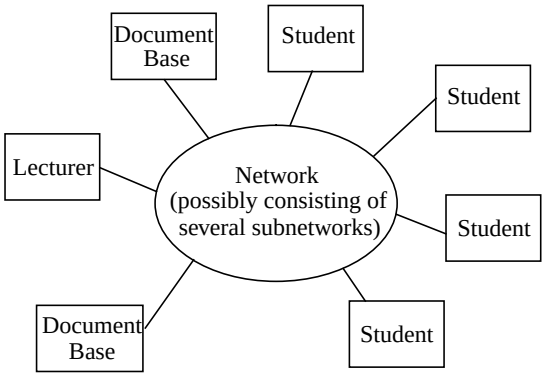

Fig. 1. Structure of the teaching application.

What makes this kind of application especially interesting is its point-to-multipoint communication. The streams sent out by a given source (e.g. the lecturer) will be received by several other users (here: the students). This communication structure is best supported by multicast transfer, and in the following sections, we will show how multicast influences the QoS characteristics of an application and how this situation can be handled by QoS management.

2.2 QoS management scenarios

To ensure that the requirements of the users are satisfied, QoS management is essential. Examples of QoS management functions are QoS negotiation, QoS renegotiation, QoS mapping, resource reservation, QoS monitoring, and QoS adaptation. In this paper we restrict our attention to negotiation, renegotiation and adaptation functions in the context of distributed MM conversational applications. A more detailed description of some QoS management functions in the context of distributed MM presentational applications, e.g, news-on-demand, can be found in [8].

In the following, these QoS management functions are described using a concrete example. Consider the situation described in Figure 2, which is a snapshot of our teaching application. For this example, we assume that video streams may be sent in different qualities simultaneously, and that the multicast routing algorithm is based on core-based tree routing (CBT) [2], i.e., there is only one multicast tree per multicast address. Resources are reserved using RSVP [15], and receivers may set filters to decide which streams they wish to receive (i.e. use the reserved resources for). However, we do only use these assumptions as an example. Since our techniques are independent of underlying protocols and mechanisms, they also work for other coding and routing techniques, such as hierarchical video encoding [13], multicast routing already including resource reservations as discussed e.g. in [10], MBone routing techniques [4] or video selection using group management protocols [11]. The sample system consists of two senders and five

receivers. Source s1 (the lecturer) is sending out two video streams simultaneously, representing two different qualities. The thin regular arrows indicate the low, the thick ones the high quality. Source s2 (one student) is sending an audio stream (dashed arrows) in one quality (In reality, both senders are also receivers, but for the sake of clarity, we omit these additional arrows). The other boxes represent routers, and the grey box represents the core router, i.e. the root of the multicast tree. The goal of QoS management is to satisfy the QoS requirements of the service users while optimizing resource usage. We now briefly introduce each of the three managememt functions discussed here and show how they might work in this scenario.

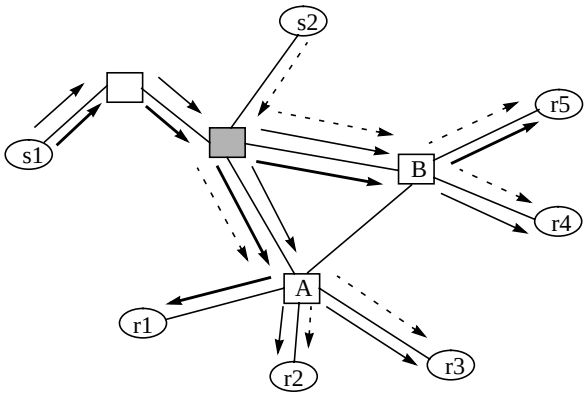

Fig. 2. A typical situation within a network supporting a multimedia application.

QoS negotiation. The role of *QoS negotiation* is to find an agreement on the required values of QoS parameters between the system and the users, e.g. participants in a teleconference. In many applications, including presentational MM applications, this process includes three parties: the user, the communication subsystem and the information provider. QoS negotiation in this scenario, however, is different from negotiation in such typical unicast connections. Consider an application with several hundred receivers, as it is e.g. typical for MBone sessions. Every receiver would have to negotiate with the sender, and the sender would have to keep the state of every receiver. Obviously, this approach scales very poorly. A popular solution to this problem consists of doing no negotiation at all between sender and receiver, instead let the sender broadcast streams in several qualities (e.g. low resolution black&white and high-resolution color) among which the receiver may select (see e.g. [5]). A characteristics of the existing QoS management schemes for this approach is that neighboring receivers have no means of coordinating their QoS requests. If a new user knew about the currently supported qualities in his region, he could select one of them instead of one not received in his region so far. This would result in saving resources and would make the communication service cheaper. It should be noted that the network provider himself certainly has such information and could convey it to the user, but he may not always be interested in doing so, since in times of low network load, that could decrease his profit. As we will see later, our cooperative approach provides an application-oriented solution for this.

QoS adaptation. The role of QoS adaptation is to keep providing the negotiated quality of service, eventually lowering it in case of resource shortages. The user usually specifies a degradation path along which the quality can be lowered, and he also specifies a minimum acceptable quality which defines the point where renegotiation of the quality or abortion of the service has to take place.

Considering multicast as our focus, adaptation may consist of reconfiguring the multicast tree, if we assume that one possible problem could be that parts of the tree are no longer capable of providing the negotiated service. Reconfiguration means destroying the old multicast tree and building a new one using the multicast routing and resource reservation mechanisms. Building complete new multicast trees is algorithmically expensive; depending on the goals (e.g. minimum overall cost), it could even be NP-complete [9]. In addition, it seems to make more sense to do the reconfiguration only in that part of the tree that causes the QoS problem. This, however, is difficult when application-oriented QoS management functions are only located in the end system, since then, it may be difficult to detect the problem area.

If, in our example, receiver r1 detects a violation of his negotiated QoS level, he may try to solve this problem himself, e.g. by switching to the lower-quality video. But this definitely leads to a lower quality, while the problem perhaps could have been solved by a reconfiguration without lowering the quality, if is were located somewhere deep in the net. On the other hand, would he ask instead the net to provide a solution (e.g. by multicast tree reconfiguration), then such a reconfiguration could be without any effect, if the problem is located on his own link or in the workstation. He could, as a third option, also send a message to the sender, asking him for a solution. The sender, however, has the same problem of lack of information. He could either adjust the quality of the stream he sends (as it is done in IVS [14]) or ask the net to solve the problem. Thus, a partial reconfiguration would still be impossible since the location of the problem is not known. Apparently, due to the lack of information, QoS management here consists mainly of guessing a solution and examining the results. Our approach will provide a solution for this problem by localizing the problem source.

QoS renegotiation. A *renegotiation* may be initiated by the user or the system, e.g., the communication system. The user-initiated renegotiation allows a user to request a better quality, e.g., a user asks for color quality while the currently delivered quality is black&white, or to reduce his/her requirements from the service provider in order to reduce the cost of the current session. On the other hand, the system initiated renegotiation usually occurs, when the system can no longer support the negotiated QoS and

the quality drops below the acceptable limit. In such a case, the user is asked to accept a lower quality.

Another interesting option would be to do renegotiation in order to optimize resource usage, i.e. without any QoS violation triggering it. Consider again our example. Receiver r1 is receiving the high-quality video, while r2 and r3 are receiving the lower quality. This leads to the next upstream link having to support both qualities. It would be helpful if the receivers could communicate and coordinate their resource requirements. Thus, r2 and r3 could switch to a higher quality without paying much more, or r1 could switch to a lower quality, thus saving a lot of money since a huge amount of resources is no longer used. Even more, it could be interesting if receivers of different applications could coordinate their QoS requests, leading e.g. to temporal shifting of QoS requirements of one application to facilitate high quality for another during a certain period of time. In the existing schemes, such a communication is impossible, since receivers do not know each other. Again, such a coordination could be done via the sender, but senders do not know receivers either, and if they did, we would quickly have a scaling problem.

We believe that the described scenarios often occur in distributed multimedia applications and that they are not handled adequatly by existing QoS management schemes with respect to optimized resource usage. Therefore, we have developed a new *cooperative* QoS management scheme which provides more information about the state of the network and the QoS requirements of receivers. Based on this information, better decision may be met to fulfill such requirements and optimize resource usage.

3. An Architecture for Cooperative QoS Management

The basic idea of our new scheme is to install an application-oriented QoS agent on each router of the underlying network and on every end system participating in an application. These QoS agents are able to communicate with their neighboring agents, informing them e.g. about current QoS values supported in their local area or about possible QoS problems. This knowledge is basically *application-oriented*, i.e. the agents know about QoS requirements and negotiated values for users. This constitutes a main difference of this approach compared to existing QoS management functions on network nodes which deal with lower-layer QoS, such as ATM cell loss priority etc., and which do not have any information about relationships between streams and applications.

In our approach, however, not every agent may contact any other agent. Rather, communication depends on the existing multicast trees, leading to a hierarchical communication structure. For each multicast tree in which a given router is involved, the QoS agent knows its upstream and all downstream neighbors. If the neighboring node is an end system, the agent knows all receivers on this end system. A receiver's QoS agent knows only its upstream QoS agent, and a sender's agent its downstream neighbors. The information about neighbors may be easily set up during the establishment of the multicast tree, resp. when a member leaves or a new member joins.

We are now giving an overview of how this new management scheme works for the three QoS management functions described above. A discussion of the protocols used between neighboring agents follows in Section 4.

3.1 QoS negotiation

QoS negotiation occurs when a new receiver enters an existing multicast tree. The following steps are executed during this join process:

1. The user resp. its local QoS agent gets the address information for an application to join from a session directory. Since the available streams and their resp. costs depend on the current application situation resp. the actual network load and stream distribution, the local manager first gets this information from the agents in the net. This means contacting an agent for each multicast stream. In case of CBT, e.g., the local agent only has to contact one agent for each multicast group. A QoS agent which is already part of the respective application may be found using e.g. parts of the multicast join algorithm. The contacted agents send back all information (including cost) about available streams and already supported streams in this part of the tree. This action may include several other QoS agents not yet on the tree, but on the route between the tree and the new receiver.
2. The end-system QoS agent assembles a list of all streams which can be supported by network and workstation and offers it to the user. The user then selects the desired streams and returns his selection to the local agent.
3. The agent has the information of how to deliver the stream to the user. This information is passed to the respective underlying protocols, e.g. RSVP or the MBone group management protocol. Then, the join procedure is executed to become part of the application's multicast trees. Resources are reserved along the selected new path, in order to guarantee the desired QoS. If a quality was requested that so far has not been supported on the branch of the tree where the new member is attached, QoS reservations have to be increased accordingly on that branch. If a resource reservation is not possible due to a lack of resources, the receiver's QoS agent may initiate a new negotiation process with the user (i.e., go back to step 2), or he may ask the multicast routing algorithm to find a new route.

3.2 QoS adaptation

Adaptation occurs when the negotiated quality can no longer be supported, but also when new members join the receiver group or current members leave. The second and third case offer possibilities for resource usage optimization, which could be done e.g. by a multicast tree reconfiguration. More interesting in the context of this paper is the first case.

Receivers usually monitor the QoS they receive from the network. By comparing these values to the negotiated QoS, it is possible to discover QoS problems. If a receiver realizes such a problem, it sends a QoS violation message to its upstream neighbor. Upon receipt of such a message, this agent waits a certain amount of time to see if it receives more such messages from other downstream agents. If this is not the case, the problem is likely to be located between the receiver and its upstream neighbor. Therefore, the upstream agent sends a message back to the receiver telling him this. A possible solution of the problem could then be to select a lower quality, since the link to the receiver or the workstation might not be able to support the quality currently selected.

In case that the upstream agent receives several violation messages, it sends a violation message to its own upstream router. By the combined usage of solve and violation messages, the part of the multicast tree causing the problem could be easily located. The solution may be a partial reconfiguration of the tree. Remember that partial reconfigurations are much less costly than a complete reconfiguration of the tree. Figure 3 shows this situation.

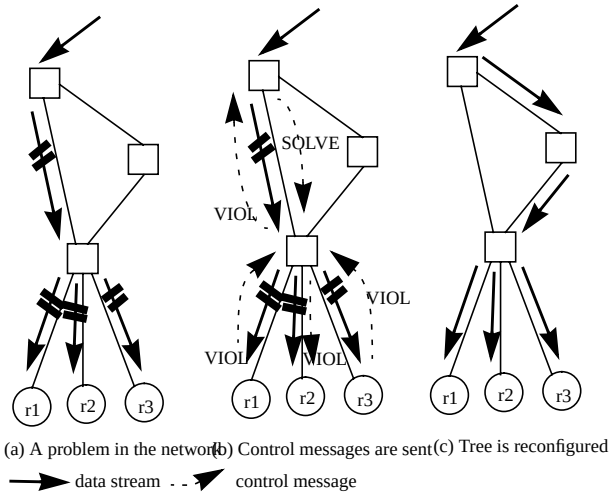

Fig. 3. A problem somewhere in the network

A problem occurs on one of the network links (Figure 3(a)). All three receivers are affected and send a violation message. The upstream router collects them and decides to send a message to its own upstream router. Since the latter does not receive any violation message from any other branch, it tells the former to solve the problem (Figure 3(b)). A partial tree reconfiguration is then initiated (Figure 3(c)).

It may also be found that the problem is at the source, e.g., if the first link cannot support the quality of the data stream sent. In this case, a solution would be to change the qualities offered by the sender. This could be done by completely stopping one stream and using the additional bandwidth for the other streams, or by scaling the existing media streams e.g. by switching to another MPEG frame pattern.

This approach of localizing and solving possible QoS problems works well when the branches of the multicast trees are real trees, i.e., a QoS agent always has more than one downstream agent. However, it is not unlikely to encounter situations where agents only have one downstream agent.

When such an agent receives a violation message from its only downstream router, it cannot decide whether the problem is likely to be upstream or downstream since there is no way of getting more information by waiting for messages from other downstream routers. In such a case, the agent has to contact its own QoS monitor which is constantly checking the QoS situation on the router. Since it knows the stream the quality of which is violated, it may ask the monitor for current statistics of that stream. If these statistics do not indicate any violation, the problem must be located downstream. Otherwise, the problem may be located upstream or in the own router. If the monitor indicates a violation, the agent sends a message to its own upstream router. If it gets back the answer to solve the problem himself, this problem must be located in the own router, otherwise, the problem is located upstream and will be solved there.

3.3 QoS renegotiation

In a context where QoS guarantees are provided, constraints on admission are usually imposed to account for the limitation of resources. Admission control is usually implemented on the premise that admission is granted as long as sufficient resources are available. But this approach could lead to inefficiencies if certain forms of group communication, like multicast, are used. Multicast is based on the principle of resource sharing and takes advantage of group members' “common interests”. In principle, the group members share the same application and receive the same media streams. But media scaling due to user preferences and adaptation due to system dynamics could lead to a situation where the resulting multicast tree appears to be rather degenerated, providing many users with specifically selected qualities. As a consequence, too many heterogeneous requirements would have to be supported so that reserved resources would rather be dedicated to particular users than be shared by many ones.

As an example, consider the multicast tree in Figure 2. Only receivers r1 and r5 prefer high quality video while receivers two, three and four are content with low quality

video. Since all intermediate links, in particular the upstream routers, accordingly have to provide resource reservations, a rather inefficient resource usage results. This is particularly true if other applications compete as well for limited resources of the involved components. In such a situation we suggest to potentially apply a renegotiation procedure based on the strategies discussed in Section 5. The general idea of the renegotiation procedure is that a given agent sends PERSUADE messages to some downstream agents, as soon as it detects a possibility of resource usage optimization. A PERSUADE message contains an offer to switch, for a given stream, to another quality. Such offers may be generated by any of the QoS agents. Each agent receiving such a message has to decide which action to take. Its reaction depends on its own strategy.

Users may instruct their QoS agents to ignore such offers in order to follow the session without interruption.

4. Protocol Descriptions

In the following, we describe some of the operations of a given QoS agent in terms of algorithms. An agent's action is usually triggered by an incoming message from other agents or when the agent detects poor resource usage. Due to space restrictions, we concentrate on a QoS agent in the network and do not describe the behavior of an agent on end systems.

A QoS agent qa on a multicast tree mct (to keep the description simple, we assume that all senders use the same multicast tree as in CBT routing) keeps the following information:

- QA_D: the set of neighbouring QoS agents down in the multicast tree associated with qa
- qa_u: the neighbouring QoS agent up in the multicast tree.
- M: the set of media available on mct
- QoS_m: the set of qualities available for the medium $m \in M$ available on mct
- QoS_m^x: the set of qualities for medium $m \in M$ currently supported on QoS agent x

For a given qa in the multicast tree in question, the following operations are performed.

4.1 Reaction on QoS negotiation

When qa receives Ask_QOS_Info(x) (x is a QoS agent not yet on mct)

- send Give_QOS_Info($qoslist$) to x where $qoslist$ is an ordered (according to the strategy) list of all $QoS_m, \forall m \in M$ (note that all available qualities are offered, but preferences are expressed by the agent)

4.2 Initiation QoS renegotiation

When qa receives a QoS_added($x, qos \in QoS_m$) or QoS_removed($x, qos \in QoS_m$) (to indicate that a new quality is supported by the downstream agent x or an existing one has been removed; it is used for join, leave and for indication of renegotiated QoS values)

1. update QoS_m^x for $x \in QA_D$
2. send persuade() messages if the chosen strategy proposes to do so:

- in case a new QoS has been added: send Persuade($m, qos \in QoS_m$) to all $x \in QA_D$ with $QoS_m^x \neq \{qos\}$ and $QoS_m^x \neq \emptyset$ (qos is not the currently selected quality of the medium and the medium is transmitted on this branch)
- in case an existing QoS has been removed: select new qos' according to strategy and send Persuade($m, qos' \in QoS_m$) to all $x \in QA_D$ with $QoS_m^x \neq \{qos'\}$ and $QoS_m^x \neq \emptyset$

4.3 Reaction on QoS renegotiation request

When qa receives Persuade($m, qos \in QoS_m$) from qa_u

- if the chosen strategy encourages the promotion of qos then
  send Persuade($m, qos \in QoS_m$) to all $x \in QA_D$ with $QoS_m^x \neq \{qos\}$ and $QoS_m^x \neq \emptyset$

4.4 QoS adaptation

When qa receives Viol($x \in QA_D, qos \in QoS_m$)

1. wait a certain time interval d
2. if the number of Viol($y \in QA_D, qos$) messages that have arrived from other $y \in QA_D$ and $qos \in QoS_m^y$ is larger than a given limit (depending on the strategy) then
   send Viol($qa, qos \in QoS_m$) to the upstream agent qa_u
   else send Solve() to all x for which Viol($x \in QA_D, qos \in QoS_m$) has been received

5. Strategies for the QoS agents

5.1 The Quality of Operation

In order to initiate renegotiation, a QoS agent has to carefully evaluate the current situation of its resource

domain. Several parameters have to be taken into account and an overall measure has to be used. We borrow the name for this overall measure from [12] and call it the *quality of operation QoO*. We also adopt their definition of QoO but modify it in a way such that properties of multicast communication can be captured (which has not been considered in [12]).

The measure is applied so that if the current QoO is relatively low, renegotiation will be initiated that would lead to higher QoO if accepted by some users. More than one modes of operation corresponding to higher QoO could alternatively be suggested to users. The difference between the current QoO and the candidate QoO is used as a measure for a potential increase in revenue if the mode of operation were changed. Part of the potential increase in revenue, 50% say, is either used as a discount if a *decrease* in *quality of service* is suggested or as additional service cost if *quality of service* is suggested to be *increased*. As an effect, the potential increase in revenue is shared among service provider and service users. Note that renegotiation is only initiated if a potential increase in revenue exceeds a certain threshold.

The *quality of operation* is defined as follows:

$$QuO = \sum_{j \in streams} \left(\alpha_j A_j - \sum_{i \in QoS} \delta_{ji} D_{ji} \right) + \sum_{t \in strtypes} \beta_t B_t$$

It is a cumulative measure of the reward gained by accomodating a set of media streams in the resource domain of a certain QoS agent. For each stream *j* resp. each stream type t[1] the following measurement parameters are defined:

- A_j, a measure for the value of resources (bandwidth) reserved for stream *j*;
- B_t, a measure for the value of remaining free bandwidth that could still be devoted to streams of type *t*,
- D_{ji}, a measure for the cost of a degraded quality of service parameter *i* measured for stream *j*. These parameters express the difference between actual and negotiated values. If a negotiated QoS value cannot be supplied by the provider, the user will pay less, decreasing the revenue for that stream.

α_j, β_t and δ_{ji} are control parameters that can dynamically or statically be set:

- α_j is used to characterize the revenue gained by transmitting stream *j*; α_j is chosen as proportional to the number of outgoing links of the multicast tree for stream *j*.
- β_t characterizes the importance of the current system state, i.e., the value of free resource to accommodate further streams of certain types;
- δ_{ji} characterizes the importance of a particular quality of service parameter *i* for a media stream *j*.

With these definitions the cumulative QoO measure expresses a compromise between the additional revenue of accommodating media streams, the (potential) value of free resources, and the current values of quality of service measures. Accepting a new stream of a certain type will increase the revenue, but it will also decrease the amount of available resources which in turn leads to a decrease in QoO. The importance of a higher immediate vs. a possible higher future reward (which is only possible when resources are available) can be expressed by selecting the values of α_j resp. β_t accordingly. Degraded QoS parameters of a certain media stream can have an adverse effect on QoO if such a media stream were further distributed at a router.

It should be noted that the values for single parameters have to be carefully selected. Usually, it should be avoided that one parameter dominates the QoO. A set of values for the parameters is equivalent to a strategy.

5.2 An Example

In what follows, the QoO will be evaluated for the scenario depicted in Figure 2. The QoS agent corresponding to router *A* accommodates audio stream 1 with the desired QoS parameters, which is distributed to all three immediate receivers, and therefore $\alpha_1 = 3$ and $D_{1i} = 0$, $\forall i \in QoS$.

The revenue A_1 gained for an audio stream is assumed to be one unit. In the current situation the load on router *A* is assumed to be low, so that there is no particular need to care about resources for audio streams, which have relatively low bandwidth requirements, and therefore $\beta_1 = 0$. Concerning the accommodated video streams we assume a black/white type video stream 2 distributed to receivers r2 and r3 without quality distortions, and therefore $D_{2i} = 0$, $\forall i \in QoS$, $\alpha_2 = 2$, and $A_3 = 3$ (the revenue for the delievery of a b&w video is three times higher than for the audio). Due to the low load situation there is also no need to worry about accommodating further black/white videos, and therefore $\beta_2 = 0$. Finally, there is a colored video stream 3 accommodated, which requires reservation of resources in equivalence to five units of rewards $A_3 = 5$, and $\alpha_3 = 1$. The reward for colored video suffers from additional loss $D_{31} = 1$ and from additional delay $D_{32} = 1$. Since some loss can be tolerated for video streams we let $\delta_{31} = 0.2$, but emphasize the importance of delay in conversational video applications by letting $\delta_{32} = 1$. In the current situation we can accommodate one additional colored video, and therefore let $B_3 = 5$. Since this is only a

1. Stream types are certain classes of streams such as high-quality color video or low-quality audio.

potential revenue, we set $\beta_3 = 0.5$. With these assumptions the current QoO for router *A* evaluates as follows:

$$QoO = (3 \cdot 1 + 0 + 0) + (2 \cdot 3 + 0 + 0) - (0.2 \cdot 1 + 1 \cdot 1) + (1 \cdot 5 + 0.5 \cdot 5) = 15.3$$

A first possibility to adapt the mode of operation consists in suggesting the degradation of video quality to receiver r1 which would result in more free resources to accommodate an additional colored video stream and the following QoO_{A1}:

$$QoO_{A1} = (3 \cdot 1) + (3 \cdot 3) + (0.5 \cdot 2 \cdot 5) = 17.$$

An increase of video quality for receivers two and three as a second option would result in the following QoO_{A2}:

$$QoO_{A2} = 3 + 0 - 3 \cdot (0.2 + 1) + (3 \cdot 5 + 0.5 \cdot 5) = 16.9$$

From the service-provider's point of view both adaptations would yield a similar effect with respect to revenue increase in the current situation.

In contrast, we assume that router *B* is highly loaded so that it would be of higher value to free resources; otherwise, the same assumptions apply:

$$QoO_B = (2 \cdot 1) + (1 \cdot 3) - 1.2 + (1 \cdot 5) = 8.8$$

$$QoO_{B1} = (2 + 0.5) + (2 \cdot 3 + 0.5 \cdot 3) + (0.5 \cdot 5) = 12.5$$

$$QoO_{B2} = (2 + 0.5) + (0.5 \cdot 3) + (2 \cdot (5 - 1.2)) = 11.6$$

Due to the higher load, renegotiation could improve revenue much more with respect to router *B*, regardless of whether an increase or a decrease of video quality were performed. Furthermore, video quality should be degraded for receiver r4 rather than enhanced for receiver r5.

Due to space restrictions, we can only sketch our approach to strategy selection. An important issue we do not further discuss here is the coordination between QoS agents by exchanging information about parameter settings etc. It is based on the general framework described in [6] which provides mechanisms for group *decision making* processes, for *negotiation* among *competing* proposals, handling *resource conflicts* and reaching *consensus*.

6. Conclusions and Outlook

In this paper, we have described a new distributed and cooperative QoS management scheme. The basic idea of this scheme is to install QoS agents on each node of the network and in every application program running on an end system. Agents are able to communicate with each other, thereby locating QoS problems and allowing negotiations for optimal resource usage. At the same time, single agents can be kept simple, since they only communicate with neighbors (and keep only their state thus making the scheme scale very well) and have knowledge about their local area.

Among other work, we are in the process of further formalizing the protocol and strategy descriptions, using Extended Finite State Machines resp. Stochastic Petri Nets. The former will allow for extensive simulations which will become important when the protocols become more complex, e.g. when dealing with more than one multicast tree in an application and especially with inter-application communication. With the latter, we will examine different model-based strategy selections, as described in [3].

References

1. C. Aurrecoechea, A. Campbell, and L. Hauw. A Survey of QOS Architectures. *Multimedia Systems Journal, Special Issue on QoS Architectures*, 1997. To appear.
2. A. Ballardie, J. Crowcroft, and P. Francis. Core based trees (CBT) – An Architecture for Scalable Inter-Domain Multicast Routing. In *ACM SIGCOMM '93*, pages 85–95, 1993.
3. H. de Meer. Adaptive Quality of Service Management: A Model-Based Approach. In *10th European Simulation Multiconference (ESM'96), Budapest, Hungary,* June 1996.
4. S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.
5. S. Deering. Internet multicast routing: State of the art and open research issues. In *Multimedia Integrated Conferencing for Europe (MICE). Seminar at SICS, Sweden*, 1993.
6. B. J. Grosz and S. Kraus. Collaborative Plans for Complex Group Action. Technical Report TR-20-95, Harvard University, Center for Research in Computing Technology, 1995.
7. A. Hafid and G. v. Bochmann. Quality of Service Negotiation in News-on-Demand Systems: An Implementation. In A. Azcorra, T. D. Miguel, E. Pastor, and E. Vazquez, editors, *Proc. of the 3rd Int. Workshop on Protocols for Multimedia Systems, Madrid, Spain*, pages 221–240, Oct. 1996.
8. A. Hafid and G. v. Bochmann. Some Priciples for QoS management. *Distributed System Engineering Journal*, 1996. To appear.
9. R. M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.
10. V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1(3):286–292, June 1993.
11. S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *ACM SIGCOMM'96*, Stanford, CA, Aug. 1996.
12. J. E. Neves, L. B. de Almeida, and M. J. Leitao. ATM Call Control by Neural Networks. In J. A. et al., editor, *Proc. of the 1st Intern. Workshop on Applications of Neural Networks to Telecommunication*, pages 210–217, 1993.
13. N. Shacham. Multipoint communication by hierarchically encoded data. In *IEEE Infocom'92*, pages 2107–2114, 1992.
14. T. Turletti and C. Huitema. Videoconferencing in the internet. *IEEE/ACM Transactions on Networks*, pages 340–351, June 1996.
15. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network*, 7(5), Sept. 1993.