Adobe’s PostScript Language Reference Manual, 2nd Edition, section H.2.4
says your EPS file is not valid, as it calls setpagedevice

Decision Support in Cooperative QoS Management

Stefan Fischer
University of Mannheim
Praktische Informatik IV
D-68131 Mannheim, Germany
stefis@pi4.informatik.uni-mannheim.de

Hermann de Meer[a]
University of Hamburg, CS Department
Vogt-Koelln-Str. 30
22527 Hamburg, Germany
demeer@informatik.uni-hamburg.de

a. Please direct all communication to Hermann de Meer, Center for Telecommunications Research, Columbia University, New York, NY 10027, USA, hdm@ctr.columbia.edu.

Abstract

Cooperative QoS management is a new quality of service management scheme which is based on QoS agents distributed within a system and cooperating with each other to provide the QoS negotiated with users, thereby ameliorating the overall system's resource usage and decreasing the communication costs. During their operations, agents have to take decisions in order to react on QoS violations, initiate QoS renegotiation processes or react on renegotiation requests from other QoS agents. In this paper, we present two tools which support cooperating QoS agents in their decision processes: a model called Quality of Operation, based on a mathematical formula, and an approach based on a new variant of Stochastic Petri Nets, so-called Controlled Stochastic Petri Nets.

Keywords: QoS, Intelligent Agents, Resource Management

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 utilisation 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.

In general, best-effort approaches are not suitable for distributed multimedia systems, 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 [2]), which include also functions for performance monitoring, resource allocation and QoS management.

In multimedia applications including multicasting to many users, such as teleconferencing or educational applications, this global QoS management approach is not workable any more, because the number of users involved is too large. For instance, negotiation of QoS parameters between the sender and every single receiver becomes impossible, since (1) the system would quickly become overloaded and (2) it would have to take into account (and possibly provide) many different qualities requested by users. Instead, a more decentralized approach seems suitable, where QoS management functions such as QoS negotiation, adaptation or renegotiation are distributed over the network. We developed such an approach called *Cooperative QoS Management* [9], where so-called *QoS agents* are installed on the routers and end systems participating in an application. These agents cooperate with each other in order to provide the QoS levels requested by the application. An interesting new feature is the possibility of communication between users resp. their local QoS agents, allowing for a cooperative selection of desired qualities. If users cooperate and decide to request a service in the same quality, less resources have to be reserved, which in turn leads to lower communication costs and higher resource availability for other applications.

We adapted resp. developed two different models to support the decision process of QoS agents: the first one is called the Quality of Operation and is based on a mathematical formula, taking into account the value of used and free resources and the cost of QoS violations, and the second one is based on a new variant of Stochastic Petri Nets, called Controlled Stochastic Petri Nets (COSTPN) [4]. Both approaches are special in that they take revenue issues into account, i.e., decisions are not only based on resource availability, but also on the possible benefit for the user, the information provider or the communication provider, depending on who runs the QoS management system.

The rest of this paper is organized as follows: in Section 2, we give a brief introduction into the architecture and new possibilities of Cooperative QoS Management. Sections 3 and 4 build the main part and discuss the application of the two models in our QoS management scheme. Section 5 concludes the paper.

2 Principles of cooperative QoS management

Cooperative QoS management has been developed with multimedia applications in mind, in which many users participate at the same time, such as teleeducation systems or life video transmissions of major sports events. We assume that single data streams are multicast to many users and that senders offer the same media stream in several qualities, for example a high, a medium and a low quality video stream. There are no individual QoS negotiations between senders and receivers; rather, receivers have to select among the qualities offered by the senders.

The principle idea of the new QoS management scheme is to distribute the QoS functions by installing *QoS agents* in the network. These agents act on behalf of the data source in that they negotiate and renegotiate qualities of data streams with the final receivers. In addition, they are able, due to their knowledge about data stream distribution in the network area they are responsible for, to support more technical QoS functions such as QoS adaptation .

The architecture of this system is depicted in Figure 1. Media streams are distributed in different qualities via multicast. Agents communicate only with neighboring agents in order to avoid communication overload situations in regions near to a source. They may also get information from and send information to local system components such as QoS monitoring functions, reservation and routing protocols.

As soon as a router becomes part of the multicast tree for a given source, an agent is sent to this router and all other routers on the branch between the new user end system and the first router already part of the multicast tree. Also, one agent will be sent to the end system and will then contact its upstream agent in order to receive information about the available products and their qualities on this source. This agent, in turn, contacts its own upstream agent to ask for this information. This process continues until the first agent already part of the tree is contacted which then forwards the information about available qualities back downstream. On each link, the agents have to find out whether the link can support all available qualities by contacting the communication system. If a given quality cannot be supported on that link, it is removed from the list of available qualities and not offered to downstream agents. This is also the reason why agents do not already have the quality information when they are sent to their router. Finally, the end system agent will only receive information about those qualities supportable on the whole branch.

Figure 1. Architecture of the Cooperative QoS Management Scheme

Based on the quality information available from the agent, a negotiation process takes place that includes the user, the end-system and router agents as well as the communication system. The negotiation process leads to one quality being selected by the user, and to resources being reserved in the network in order to guarantee this quality. However, the reservation process does only have to cover that part of the tree where the stream in the desired quality is not yet available. On the remaining path to the source, resources are already reserved for this stream.

As soon as agents are set up on a router, they start to perform QoS management functions. Usually, they *monitor* the current QoS. Whenever they detect a QoS problem, they may become active by informing their neighboring agents. In [9,5] we describe how this kind of agent cooperation can be used to support technical QoS function like QoS adaptation. [11] gives details about inter-agent protocols. In addition, agents may become active on their own in order to optimize system performance and resource usage.

It should be noted that our techniques are independent of underlying protocols and mechanisms and work for different coding and routing techniques, such as hierarchical video coding [15], QoS-based multicast routing techniques as for instance described in [12], MBone routing techniques [8] or video selection using group management protocols [13]. Our initial considerations were based on Core-Based Tree routing [3] and RSVP [16] as resource reservation protocol.

Especially for the negotiation and renegotiation process, the QoS agents need some kind of decision support that tells them when to start which action and how to react on arriving proposals and requests. In the following two sections, we discuss our two approaches to this problem. The COSTPN model addresses a particular problem of negotiation, namely admission control, while the Quality of Operation approach is applied to the renegotiation problem.

3 QoS negotiation based on Controlled Stochastic Petri Nets

3.1 Controlled Stochastic Petri Nets

Stochastic Petri Nets (SPNs) have been used for the analysis of stochastic systems for several years now. Compared to traditional Petri Nets, SPNs have so-called timed transitions which are labelled with firing rates. These rates are exponentially distributed random variables, taking stochastic uncertainties into account. By mapping SPNs onto Markov Reward Models (MRMs), they can be analyzed by a number of numerical algorithms. As a result, one gets predictions about the modeled system's performance.

For a *dynamic optimization* of performability measures [1], a new feature is introduced to SPN. It comprises a control structure that allows one to specify a controlled switching between markings of a SPN. Such a controlled switching is interpreted as a *reconfiguration* in the modeled system and represents a decision of the system's controller program to switch to another state. A reconfiguration is modeled by the firing of a new type of transition, called a *reconfiguring transition*. The introduction of reconfiguring transitions leads to a new modeling tool, called *Controlled Stochastic Petri Nets (COSTPN)*, and provides a way to combine the classical performability modeling of SPNs with the option to dynamically optimize measures [4].

When a COSTPN has reached a marking where one or more reconfiguration transitions are enabled, the controller program has to select among several options. One option is to instantaneously reconfigure to the marking which is reached through the firing of one of the enabled reconfiguring transition; no timed transition can fire in the current marking in this case. Another option is to stay in the current marking and*not* to fire a reconfiguring transition, so that the enabled timed transitions can fire in the current marking in their usual manner. The decision, which option and which reconfiguring transition to select, is based on the comparison of optimization criteria. The optimization criterion is computed for all options and the one with the highest expected reward is selected. In order to apply numerical algorithms on the optimization criteria, COSTPNs first have to be mapped onto Extended Markov Reward Models (EMRMs) [7], which are an extension of standard MRMs with respect to reconfiguration edges.

3.2 Example: admission control

We now present a simple example which shows how COSTPNs can be used to help single QoS agents in their decision processes. The example deals with admission control of two classes of streams, say audio and video sources, to a multimedia transmission system (a router, for example). Admission control is a very relevant QoS function in both unicast and multicast when QoS guarantees are to be provided. On each single transmission system, a QoS agent is installed which autonomously decides whether to admit a requesting source. An agent not accepting a source means that either another path including other transmission systems has to be found or the source cannot be admitted at all to the overall system.

In our example an application like tele-cooperation with fixed run-time T is investigated, where participants may dynamically apply for audio or video transmissions of random duration to accomplish their work. We are able to take non deterministic or random multimedia stream duration in the planning into account, as this often naturally occurs in conversational applications.

High level admission control strategies are needed at every instant of the system's run-time for revenue maximization in an error prone environment. Audio and video sources in a waiting room ask for admission to the system, i.e., these sources ask for using the system's resources for transmission of audio resp. video data. In turn, they pay for the resource usage and the service provided by the system. Once admitted, they expect a certain level of QoS. If this level cannot be kept (QoS violation), they will pay less for the provided service.

The resource manager (agent) has to decide whether to admit a certain source or not. Its goal will be revenue maximization. Acceptance of sources will result in different rewards, depending on the type of stream (audio or video), their resource requirements, the transmission length, the risk of QoS violations during transmission (and possible abort of the stream) or additional rewards for successful transmission completion. Once a source is active, it will finish successfully or suffer from QoS violation. In this example, the latter case leads to a transmission abort, resulting in a longer reconfiguration period where resources are reorganized and freed for re-usage. The former case entitles the system to an extra revenue.

The COSTPN modeling of this system is depicted graphically in Figure 2. The meaning of transitions and places are indicated directly in the figure, while the firing rates of timed transitions are given in Table 1.

.

Figure 2. COSTPN for QoS negotiation

The possibilities of making decisions are modeled by the reconfiguration transitions *r1* and *r2*, by which the QoS manager may decide to admit one or more audio streams or videos to the system. A reconfiguration can only be executed if the necessary number of tokens are available. To admit a video, for instance, *p2* has to contain at least 4 (since videos need four resource units in this example) and *p3* at least one token (the waiting connection).

Each transition t_i has a certain firing rate τ_i attached to it. Duration of an audio transmission is given by an exponentially distributed random variable with parameter τ_4, so that $e^{-\tau_4 t}$ is the probability that audio transmission will last longer than *t* units of time. $1/\tau_4$ is the mean audio transmission time. The firing rate of t_4 is proportional to the number of audio sources being actively transmitted. Transition t_3 models severe QoS degradation leading to interruption of transmission. The mean time between interruptions is a function of the load imposed on the system, of the particular media type and of the current network state. Transitions t_1 and t_2 model the necessary reconfigurations after a stream has been interrupted. All random variable are assumed to be exponentially distributed with the respective parameter.

transition	firing rate	transition	firing rate
t_1	$\lvert p_4 \rvert \cdot \tau_1$	t_4	$\lvert p_5 \rvert \cdot \tau_4$
t_2	$\lvert p_9 \rvert \cdot \tau_2$	t_5	$\lvert p_8 \rvert \cdot \tau_5$
t_3	$(max(p_2) - \lvert p_2 \rvert + 1)^2 \cdot \tau_3 \cdot \gamma$	t_6	$(max(p_2) - \lvert p_2 \rvert + 1)^2 \cdot \tau_6 \cdot \gamma$

Table 1: Transition firing rates

Furthermore, the COSTPN contains two so-called *immediate transitions i1* and *i2*, which fire as soon as they are enabled regardless of other transitions being enabled. They have been introduced to assign *pulse* rewards to successfully transmitted streams and to model the instant release of occupied resources. For the purposes of this example and an easy analysis, the model is kept simple in that a constant number of audio and video sources are present. Additional transitions could be inserted to model the stochastic arrival and departure of waiting sources.

3.3 Analysis

For a numerical analysis of this example, we use the tool environment PENELOPE developed at the University of Hamburg [6]. The current version of the tool accepts EMRMs as input and allows the application of several algorithms from Markov decision theory for transient or stationary optimization of performability measures.

Therefore, the COSTPN has to be translated into an EMRM first. In our study, the COSTPN/EMRM is evaluated in an initial setting of two waiting audio streams (two marks in *p1*), two waiting video streams (two in *p3*), and a pool of eight resource units (eight in *p2*). The EMRM model corresponding to the COSTPN of Figure 2 with this initial marking comprises 21 Markov states, 17 vanishing states, 18 reconfiguration edges and more than 40 transitions, and is therefore not shown here. The EMRM states, which correspond to COSTPN markings, are denoted by a shorthand notation, such that the first three digits are truncated. *M000000*, for instance, refers to initial marking $M_0 = (2, 8, 2, 0, 0, 0, 0, 0, 0)$.

With the notation *(i,j)* we refer to *i* audio and *j* video streams being admitted. Then, the following combinations of streams are possibly admitted to the system: *(0,0)*, *(1,0)*, *(2,0)*, *(0,1), (0,2)*, *(1,1)*, and *(2,1).* Due to the limitation of eight resources, other combinations are excluded.

For the series of experiments, the parameter settings are as follows: As time unit, we assume one minute. Thus, a transition with firing rate of 0.5 implies that, on the average, the transition fires every 2 minutes. We further assume an average audio duration of 10 minutes ($\tau_4 = \frac{1}{10}|p5|$) and a video duration of 60 minutes. Firing of error transitions *t3* and *t6* depends first of all on a network reliability parameter γ, where $1/\gamma$ indicates the mean time between connection disruptions. Since this parameter strongly depends on the network state, we allow possible variations of γ between 10^{-6} and 10^{-2}, which is equivalent to a mean-time-to-failure ranging from more than 16 years to 1h 40 min. Note that we are not primarily interested in modeling the failure causes in a detailed manner, but rather aggregate the effects into a single parameter γ. This hierarchical approach is admissible due to differences in time scales in the order of magnitudes for the interesting cases. Furthermore, interruption likelihood is assumed to depend on the type of stream and its susceptibility for QoS violations and, inversely, on the number of still available resources providing redundancy for possible error recoveries. Thus, we set $\tau_3 = 5 \cdot \gamma \cdot (max(p2) - |p2| + 1)^2$ and $\tau_6 = \frac{1}{10} \cdot \gamma \cdot (max(p2) - |p2| + 1)^2$, making audio streams much more sensitive for QoS violations (which reflects the reality). The strongly rising value of free resources under high load is expressed by the square functions in those firing rates. Part of the results of a transient optimization process with an assumed application run-time T of 1000 minutes can be seen in Figure 3.

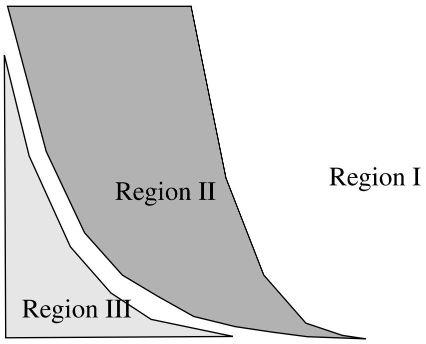

Figure 3. Strategy control regions for 12 (left) and 14 (right) video reward units

Curves on such result graphics separate regions where different strategies apply as depicted in Figure 3. The first symbol in the name of the curve determines the current marking. Only if the system is in the corresponding state, the strategy applies. For interpretation, one has to relate the current situation of the system to the parameter space represented in the diagram, according to time and network reliability parameter γ. If the system's state is classified to be in the region above the applicable curve, the system should be reconfigured to the state corresponding to the second marking being indicated after the arrow in the symbolic names of the legend. The resulting reconfiguration will lead to a higher expected reward. If the current system state is classified to be in some region below the corresponding curve, the system manager should better switch to an alternative reconfiguration.

The experiments (Figure 3) have been made with 1 reward unit for a completed audio and 12 resp. 14 units for a completed video. No rewards are assigned for audios or videos during run-time, so only a successfully completed transmission will be advantageous for the controller.

Consider the curve $M010010 \rightarrow M010010$ in Figure 3 (left side) as an example. When the system is in state $M010010$ (it has one active audio and one active video stream), while the current value of γ is 0.001 and the remaining run-time $t' = T - t$ (where t is the elapsed time and T the total run-time) of the application is 100 minutes, the system is classified above the curve in Region I and it is recommended to *remain* in state *M*010010. Assume now that 75 minutes later, the system being again in the same state. Now, the remaining run-time is only 25 minutes, which means, the system's running condition is classified below the curve in Region II. The QoS manager should now trigger the alternative option, which is executing the indicated reconfiguration. Under the given circumstances this amounts to a switching to state *M*020010, adding another audio source in the current situation and system state.

The curve $M000000 \rightarrow M000020$, which is identical with the x-axis, tells the QoS agent to always admit two video sources (*M*000020), when the system is in its idle state *M*000000. This, however, does not mean, that the system will always only run video streams. Audio streams may be admitted, for instance, when the system is in a video error state, depending on the corresponding curve.

Comparing the strategy curves of the left and right side of Figure 3, in the case of 14 video reward units, it totally becomes less rewarding to add audio sources when there is still a lot of application run-time left. Even in the case of two active sources and a current $\gamma \approx 5.5 \cdot 10^{-4}$ (Figure 3, right side), an audio source would be added no earlier than about 15 minutes before the end. The more reward a video completion gets compared with an audio completion, the more advantageous will it be to run videos instead of audios, even if the remaining system run-time and thus the probability of video completion becomes very small.

For some problems, as for example the one described above, the COSTPN structure is static and doesn't evolve during system run-time. Therefore, the analysis can be carried out in advance and the decision support can be hardwired into the agent. In any system state, the agent then simply has to acquire (by measurements, for instance) the relevant parameter values and use them as indices to look up the optimal strategy. For other problems, however, the COSTPN topology may be dynamic and thus require a frequent recomputation of COSTPNs, EMRMs and strategy curves. As a result, algorithms for automatic topology adjustment, generation of EMRMs and strategy computation have to be included in an agent's code.

4 QoS renegotiation based on the Quality of Operation

4.1 The Quality of Operation

In order to initiate renegotiation in the sense described in Section 2, a QoS agent has to carefully evaluate the current situation of its resource domain. Several parameters have to be taken into account and included in an overall measure. We borrow the name for this overall measure from [14] 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 was not considered in [14]).

The QoO is a measure for the quality of the current system state. 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, giving both of them a motivation to switch the system state. Note that renegotiation is only initiated if a potential increase in revenue exceeds a certain threshold.

The *quality of operation* is defined as follows:

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

The QoO is defined as a cumulative measure of the reward gained by accommodating a set of media streams in the resource domain of a certain QoS agent. For each stream *j* resp. each stream type *t* the following measurement parameters are defined (Stream types are certain classes of streams such as high-quality color video or low-quality audio):

- 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 accommo-

date 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. Therefore, we can avoid the unwarranted situation of "upgrading to bad quality".

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.

4.2 An Example

In what follows, the QoO will be evaluated for the scenario depicted in Figure 4, where we have two senders, five receivers and three routers. Sender s1 send s a high-quality (thick arrow) and a low-quality (dashed arrow) video, while sender s2 sends an audio stream (dotted arrow). All receivers receive the audio, while receivers r1 and r5 receive the high-quality and the other receivers the low-quality video. CBT is used as multicast routing algorithm, and the core router is depicted by the grey box. All streams are first unicast to the core and from there multicast to the receivers.

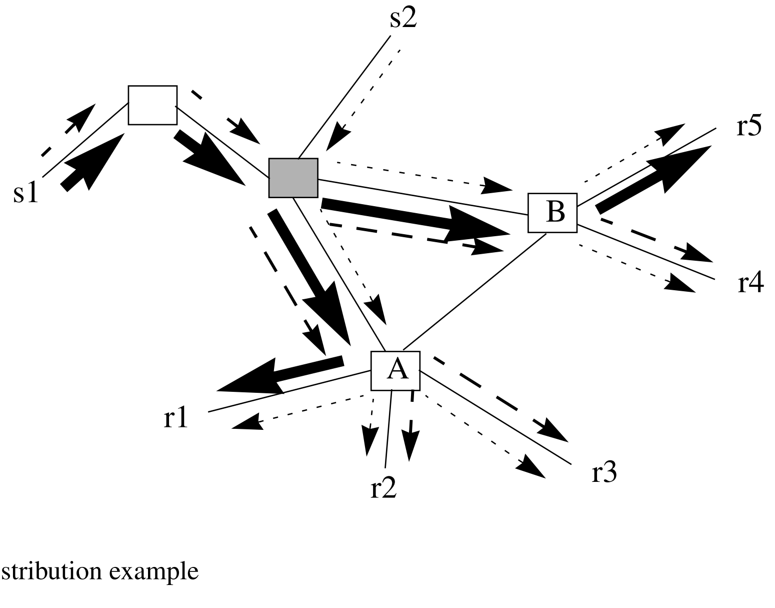

Figure 4. Stream Distribution example

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 $A_2 = 3$ (the revenue for the delivery 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 potential revenue, we set $\beta_3 = 0.5$[1].

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}:

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

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:

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

$$QoO_{B1} = (2 \cdot 1 + 0.5 \cdot 1) + (2 \cdot 3 + 0.5 \cdot 3) + (0.5 \cdot 5) = 12.5$$

$$QoO_{B2} = (2 \cdot 1 + 0.5 \cdot 1) + (0.5 \cdot 3) + (2 \cdot (5 - 1.2)) = 11.6$$

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.

4.3 Coordination between QoS Agents

So far we have only discussed how an agent can locally improve the quality of operation. This approach can already improve efficiency considerably, as has been demonstrated in the example. More global optimization can be achieved if coordination between agents is also taken into account.

Consider again the example from Figure 4 and imagine that receiver r1 accepts a video quality downgrade while receiver r4 negotiates on a video quality upgrade. As a result of such a scenario, router *A* and other related intermediate links do not have to support high quality video any more and can free the corresponding bandwidth. Similarly router *B* is freed from supporting low quality video. But if the QoS agents corresponding to routers *A* and *B* would be able to plan and act collectively, more savings could result. The diversity of supported QoS levels would be reduced to a minimum throughout the whole network. To this end we adopt the general framework of [10] and apply it to our QoS management scheme. This framework provides mechanisms for group *decision making* processes, for *negotiation* among *competing* proposals, handling *resource conflicts* and reaching *consensus*. As a prerequisite for the less centralized QoS management, the agents are able to form mutual “beliefs” about each other's intentions, which complement the

---

[1] Note that due to the possible multiplication of outgoing streams the actual revenue could finally be a multiple of B_t rather than a fraction. This notion should emphasize the importance of keeping free resources.

partial knowledge of each agent concerning the state of its own resources. As a result, each QoS agent has responsibilities with respect to the actions of other agents. The QoS agents are committed to their joint activity of overall efficient resource management under the constraint of user satisfaction. Note that, since the agents and users are still autonomous in their negotiation procedure, our approach is different from a centralized management concept.

In particular, the situation of bottleneck routers are preferably taken into account if local negotiations are being performed between other QoS agents and users. The values of β_t could be increased if it were desirable to have more resources available for accommodating media streams of type *t*. Referring to our example in the previous section, such a situation is given for router *B*. If the β_t values were increased across the network, then this would lead to a situation were both the QoS agents for *B* and *A* would clearly argue for lower video qualities.

Furthermore, the quality of operation of an upstream router has significant impact on the negotiation process performed by downstream QoS agents. This is realized by the way in which the weights α_j are set. If an upstream QoS agent realizes, for example, that multiple video qualities are only supported at one outgoing link while video quality is being uniformly supported in low level by all other links, it may modify the weights with the intention to unify the traffic further. The value of the corresponding α_j would be lowered, indicating the reduced reward of spending the resources A_j for this connection. Again referring to our example in the previous section, the QoS agent attached to the core router would suggest to increase α_2 and decrease α_3 to its downstream agents if the renegotiation with receivers r4 and r5 was successful with respect to lowering video quality. As a result, it would also be argued with receiver r1 to reduce the corresponding video quality. More details on the approach can be found in [5].

5 Conclusion and Outlook

In this paper, we described two approaches we developed to support autonomous decision making in distributed QoS agents of our Cooperative Qos Management scheme. The COSTPN model has been applied to QoS negotiation and the resource reservation and admission control decisions necessary during this process, while the Quality of Operation approach dealt with QoS renegotiation. It is easy to apply both models to other QoS problems such as QoS adaptation, QoS mapping etc.

Both models have been theoretically developed and some first numerical evaluations have been executed. However, the project is still in progress and much work remains to be done. Our final goal is to implement the approaches as part of the management scheme. For this purpose, we are currently developing a simulation environment based on the formal language SDL which will allow us to simulate the behavior of both models within configurable environments consisting of several senders, receivers and routers. Both models will be integrated as external functions. We plan to compare the behavior of both models in a number of different situations, with varying system parameters such as number and cost of available resources, degree of uncertainty, network failures, network topology etc. We assume that the simulation results will give us strong hints, whether one of the models is superior to the other or which model should be applied to which situation and how they complement each other. It could be expected, for instance, that the COSTPN model yields better results in a highly stochastic environment, since such situations are only rudimentarily addressed by the Quality of Operation approach. On the other hand, QoO seems to be more suitable for the cooperative approach, since cooperation between single agents is complex to be modeled by COSTPNs.

References

[1] *Performance Evaluation*. Special Issue on Performability, Feb. 1992.

[2] C. Aurrecoechea, A. Campbell, and L. Hauw. A Survey of QOS Architectures. *Multimedia Systems Journal, Special Issue on QoS Architectures*, 1998. To appear.

[3] 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.

[4] H. de Meer and O.-R. Düsterhöft. Controlled Stochastic Petri Nets. In *16th IEEE Symposium on Reliable Distributed Systems (SRDS'97), Durham NC, USA*, Oct. 1997.

[5] H. de Meer, A. Puliafito, and O. Tomarchio. Software Agents for QoS Management. *Cybernetics and Systems: An International Journal*, 27(5), 1998. To appear.

[6] H. de Meer and H. Sevcikova. PENELOPE: dePENdability EvaLuation and the Optimization of PErformability. In *9th Int. Conf. on Computer Performance Evaluation - Modelling Techniques and Tools, St. Malo, France*, Lecture Notes in Computer Science 1245, pages 19–32. Springer, June 1997.

[7] H. de Meer, K. S. Trivedi, and M. Dal Cin. Guarded Repair of Dependable Systems. *Theoretical Computer Science, Special Issue on Dependable Parallel Computing 129*, pages 179–210, 1994.

[8] S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.

[9] S. Fischer, A. Hafid, G. v. Bochmann, and H. de Meer. Cooperative QoS Management in Multimedia Applications. In *4th IEEE Int. Conf. on Multimedia Computing and Systems (ICMCS'97), Ottawa, Canada*, pages 303–310. IEEE Computer Society Press, June 1997.

[10] 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.

[11] A. Hafid and S. Fischer. A multi-agent architecture for cooperative qos management. In *Management of Multimedia Networks and Services (Proc. of MMNS'97)*, pages 41–54. Chapman & Hall, 1998.

[12] 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.

[13] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *ACM SIGCOMM'96*, Stanford, CA, Aug. 1996.

[14] 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.

[15] N. Shacham. Multipoint communication by hierarchically encoded data. In *IEEE Infocom'92*, pages 2107–2114, 1992.

[16] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network*, 7(5), Sept. 1993.

clear

cleardictstack