Controlled Stochastic Petri Nets for Multimedia QoS Management

Hermann de Meer
University of Hamburg, Department of Computer Science
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
Email: demeer@informatik.uni-hamburg.de

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

Abstract

Most QoS management systems for multimedia applications just cope with technical issues of QoS guarantees. A QoS management system taking revenue issues and the possibly stochastic behavior of the environment into account seems to be superior to the existing ones. In this paper, we show how controller programs for such enhanced QoS management systems can be developed based on a new kind of Petri Nets, so-called Controlled Stochastic Petri Nets. We show how to numerically analyze such networks using a tool environment in order to obtain strategies for the QoS mangement system.

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.

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

*This work was done while the author was at the University of Montreal, Canada. It was partially supported by a grant from the Canadian Institute for Telecommunication Research (CITR), under the Networks of Centres of Excellence Program of the Canadian Government.

provided by the network. Various global QoS architectures have been developed (for a recent overview see [2]), which include also functions for performance monitoring, resource allocation, admission control and QoS management.

Most existing QoS management systems meet their decisisons based only on the current resource situation. For instance, a data source asking for admittance to the system usually is accepted if enough resources are available. There are at least two important factors in real-life systems which are not taken into account by this approach: first, stochastic behavior of the underlying communication and computing environment, and second, revenue issues. Both points are related to each other. In the first case, stochastic events such as network failure can lead to QoS guarantee violations for admitted streams, in turn leading to a revenue decrease for the service provider (who usually runs a QoS management system). In the second case, possible admittance requests of other sources could arrive shortly after the one just admitted. If admitted, the former could generate more revenue than the latter, but unfortunately, now there are not enough resources to run the new sources.

For this reason, a QoS management system taking stochastic events and revenue issues into account seems to be superior to the existing one. In this paper, we show how controller programs for such enhanced QoS management systems can be developed based on a new kind of Petri Nets, so-called Controlled Stochastic Petri Nets (COSTPNs) [5]. Mapping COSTPNs to Extended Markov Reward Models (EMRMs) [4] allows a numerical analysis of the managed system and the generation of controller programs which meet their decisions based on the stochastic assumptions and possible revenues in this system. A tool environment exists for assistance in the numerical analysis.

The paper is organized as follows: in Sections 2 and 3, we give a short introduction into QoS management resp. COSTPNs and EMRMs. Section 4 then presents an example for a simple but typical QoS management system and how it can be modeled by COSTPNs. Section 5 analyzes the example with respect to possible decisions of a controller program. The analysis is done using the already mentioned tool environment. Finally, Section 6 gives an outlook on future activities.

2 QoS management in multimedia systems

To ensure that the requirements of the users are satisfied, QoS management is essential. Examples of QoS management functions are QoS negotiation, QoS renegotiation, resource reservation, admission control, QoS monitoring, or QoS adaptation.

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. A QoS negotiation protocol is executed, every time a user joins a session, to verify whether the system has enough resources to accommodate the new user without jeopardizing the guarantees given to the existing users.

A renegotiation may be initiated by the user or the 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. This type of renegoti-

ation is crucial for applications where one cannot predict accurately the scenarios to be executed during the active phase of the session. On the other hand, the system initiated renegotiation usually occurs, when the system can no longer support the negotiated QoS (detected by QoS monitoring). In such a case, the user is asked to accept a lower quality.

However, before initiating a renegotiation due to QoS violation, the system should perform a QoS adaptation. The role of QoS adaptation is to maintain, as far as possible, the QoS agreed during the negotiation phase. When the maintenance of the agreed QoS is not possible, the QoS adaptation activity must be able to exhibit graceful degradation, reacting adaptively to changes in the environment. Indeed, it may be more desirable to degrade the quality of the affected service rather than to abort it. 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 or abortion of the service quality has to take place.

Admission control permits to avoid an over-usage of resources by not accepting new streams for transmission in certain situations. Resource reservation for streams during their transmission, finally, is a basic means to provide service guarantees.

3 Controlled Stochastic Petri Nets and EMRM

Performability modeling [1] makes extensive use of Markov reward models (MRMs). Let $Z = \{Z(t), t \geq 0\}$ denote a continuous time Markov chain with finite state space Ω. To each state $s \in \Omega$ a real-valued *reward rate* $r(s)$, $r : \Omega \rightarrow \mathbb{R}$, is assigned, such that if the Markov chain is in state $Z(t) \in \Omega$ at time t, then the *instantaneous reward rate* of the Markov chain at time t is defined as $X(t) = r_{Z(t)}$. In the time horizon $[0, ...t)$ the *total reward* $Y(t) = \int_0^t X(\tau)d\tau$ is accumulated. Note that $X(t)$ and $Y(t)$ depend on $Z(t)$ and on an initial state. The probability distribution function $\Psi(y,t) = P(Y(t) \leq y)$ is called the *performability*. For ergodic models the instantaneous reward rate and the time averaged total reward converge in the limit to the same overall reward rate $E[X] = \lim_{t\rightarrow\infty} E[X(t)] = \lim_{t\rightarrow\infty} \frac{1}{t}E[Y(t)]$. The introduction of reward functions provides a framework for a formal definition of a "yield measure" or a "loss measure" being imposed on the model under investigation.

EMRMs provide a framework for the combined evaluation and optimization of reconfigurable systems by introducing some new features for MRMs [4]. EMRMs are the result of a marriage between Markov decision processes and performability techniques. A *reconfiguration* arc, which can originate from any Markov state of a model, specifies an optional, instantaneous state transition that can be controlled for an optimization. The resulting strategy is commonly time-dependent. The so called *branching* states provide another feature of EMRMs. No time is spent in such states, but a pulse reward may be associated with them. The introduction of branching states has motivation similar to the introduction of immediate transitions to stochastic Petri nets [7], so that branching states also are called *vanishing* states.

Reconfiguration arcs denote options to reconfigure from one state to another. At every point of time a different decision is possible for each reconfiguration arc. A strategy $\mathbf{S}(t)$ comprises a tuple of decisions for all options in the model at a particular point of time t,

$0 \leq t \leq T$. Strategies can be time dependent, $\mathbf{S}(t)$, or time independent, $\mathbf{S} = \mathbf{S}(t)$. A strategy $\hat{\mathbf{S}}(t)$ is considered *optimal* if the performability measure under strategy $\hat{\mathbf{S}}(t)$ is greater equal than the performability measure under any other strategy $\mathbf{S}(t)$. With $X^{\mathbf{S}}$, $Y_i^{\mathbf{S}}(\infty)$, and $Y_i^{\mathbf{S}(t)}(t)$ denoting the overall reward rate, the conditional accumulated reward and the accumulated reward until absorption gained *under strategy* $\mathbf{S}$ *or* $\mathbf{S}(t)$, respectively, a strategy $\hat{\mathbf{S}}$ or $\hat{\mathbf{S}}(t)$ is optimal, iff

$$\begin{cases} E[Y_i^{\hat{\mathbf{S}}(t)}(t)] \geq E[Y_i^{\mathbf{S}(t)}(t)] \, \forall \mathbf{S}(t) \, \forall i & \text{trans. opt.} \\ E[X^{\hat{\mathbf{S}}}] \geq E[X^{\mathbf{S}}] \, \forall \mathbf{S} & \text{stat. opt. (ergodic)} \\ E[Y_i^{\hat{\mathbf{S}}}(\infty)] \geq E[Y_i^{\mathbf{S}}(\infty)] \, \forall \mathbf{S} \, \forall i & \text{stat. opt. (non-erg.)} \end{cases}$$

For a dynamic optimization of performability measures, 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. 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 COSTPN, and provides a way to combine the classical performability modeling of SPNs with the option to dynamically optimize measures [5]. In the following we discuss the enabling and the firing rule of reconfiguring transitions for COSTPN first. The enabling rule of reconfiguring transitions is applied in the *model generation phase*, in which an EMRM is constructed from the COSTPN. In the *model evaluation* phase, in which the constructed EMRM is computationally analyzed, the firing rule of reconfiguring transitions is applied in order to optimize a given performability measure.

The newly introduced control mechanism can only be applied in tangible markings, but reconfiguration itself is assumed to be instantaneously performed. These properties are reflected by the *enabling rule* of reconfiguring transitions. Reconfiguring transitions are only enabled in tangible markings. The conditions for the enabling of reconfiguring transitions are the same as the conditions for the enabling of timed transitions. In particular, immediate transitions have higher priority than reconfiguring transitions. The *firing rule* of reconfiguring transitions is based on the aim to optimize a performability measure, which is defined through a reward structure. Reconfiguring transitions are enabled together with timed transitions and the conflict between enabled reconfiguring transitions and enabled timed transitions is solved in order to optimize a performability measure. Whenever the modeled system resides in a tangible marking, in which a reconfiguring transition is enabled, the following options are given. One option is to instantaneously reconfigure to the marking which is reached through the firing 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 the enabled reconfiguring transition, so that the enabled timed transitions can fire in the current marking in their usual manner. The decision, which option to select, i.e. the optimal one, is based on the comparison of optimization criteria as described earlier. The optimization criterion is computed for all options and the one with the highest expected reward is selected.

For transient optimization, the expected accumulated reward $E[Y_i(t)]$ is computed. For stationary optimization, either the time averaged mean total reward $E[X]$ is computed, if the model is ergodic, or the accumulated reward until absorption $E[Y_i(\infty)]$ is computed, if the model is non-ergodic.

4 The COSTPN model in QoS management

In this Section, we present a simple example which shows how COSTPNs can be used to solve QoS management problems. The example deals with admission control of audio and video sources to a multimedia transmission system (e.g. a router). Due to space restrictions, we do not present further examples, but COSTPNs are equally applicable to, for instance, QoS mapping, negotiation, adaptation, etc.

In our example, 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 QoS manager 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 recource 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 this system is depicted graphically in Fig. 1. The meaning of transitions and places are described in Tables 1 and 2.

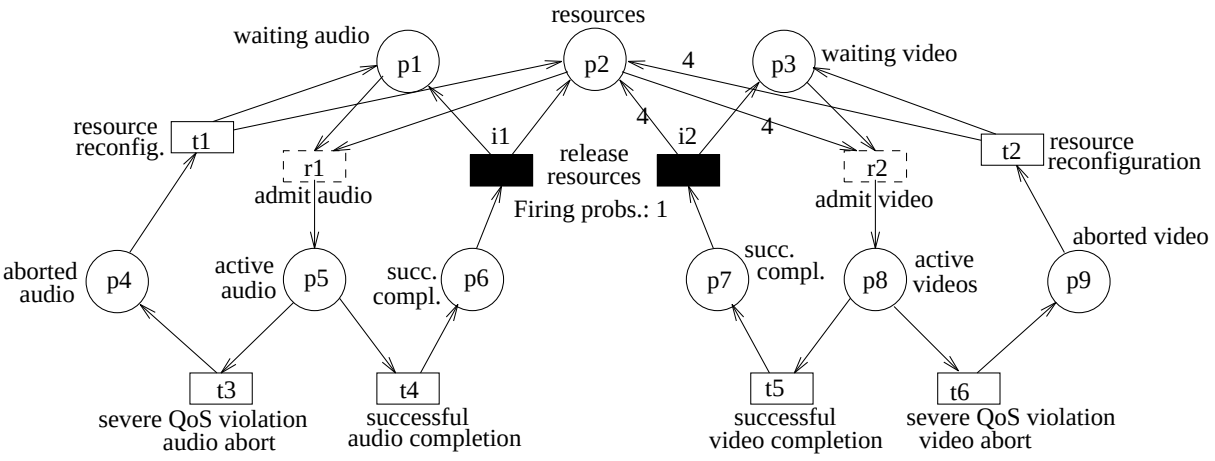

Figure 1: COSTPN for a simple QoS management example

transition	description	prio.	firing rate/prob.
t_1	reconfiguration after audio abort	0	$\#p_4 * \tau_1$
t_2	reconfiguration after video abort	0	$\#p_9 * \tau_2$
t_3	QoS violation for audio stream	0	$(max(p_2) - \#p_2 + 1)^2 * \tau_3 * \gamma$
t_4	successful audio completion	0	$\#p_5 * \tau_4$
t_5	successful video completion	0	$\#p_8 * \tau_5$
t_6	QoS violation for video stream	0	$(max(p_2) - \#p_2 + 1)^2 * \tau_6 * \gamma$
i_1	free resources of successful audio	1	1
i_2	free resources of successful video	1	1
r_1	admit an audio stream to the system	0	–
r_2	admit a video stream to the system	0	–

Table 1: Transition descriptions

place	description
p_1	inactive and waiting audio sources
p_2	currently free resources
p_3	inactive and waiting video sources
p_4	aborted audio streams
p_5	active audio streams
p_6	successfully terminated audio streams
p_7	successfully terminated video streams
p_8	active video streams
p_9	aborted video streams

Table 2: Place descriptions

The possibilities of making decisions are modeled by the reconfiguration transitions r_1 and r_2, 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, p_2 has to contain at least 4 (since videos need four resource units in this example) and p_3 at least one token (the waiting connection).

Each transition has a certain firing rate 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: $(\#p_5 * \tau_4)$. 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. All random variable are assumed to be exponentially distributed with the respective parameter.

Furthermore, the COSTPN contains two immediate transitions. They have been in-

troduced to assign *pulse* rewards to successfully transmitted streams and to model the instant release of occupied resources.

5 The Results

For a numerical analysis of the example described in the previous section (see Fig. 1), we use the tool environment XPenelope developed at the University of Hamburg [3]. 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 first has to be translated into a EMRM first. Fig. 2 shows the EMRM with initial marking $M_0 = (2, 4, 1, 0, 0, 0, 0, 0, 0)$, i.e., two waiting audio and one video stream and four resource units. The EMRM states, which correspond to COSTPN markings, are denoted by a shorthand notation, such that the first three digits are truncated. $M000000$, for example, refers to initial state M_0.

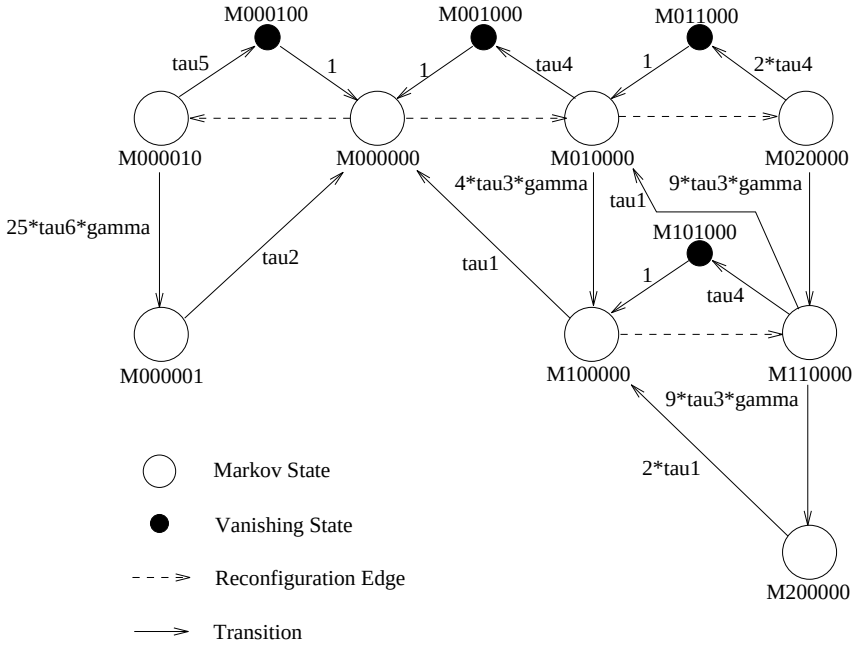

Figure 2: EMRM corresponding to the COSTPN of Fig. 1

In our study, the EMRM is evaluated in an initial setting of two audio streams, two video streams, and a pool of eight resource units. The model comprises 21 Markov states, 17 vanishing states, 18 reconfiguration edges and more than 40 transitions. For simplicity's sake, we do not present this EMRM here, but use it for our analysis.

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

For the series of experiments, the parameter settings, i.e., transition firing and probability values, are used as given in Table 3.

τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	γ
1	$\frac{1}{2}$	5	$\frac{1}{10}$	$\frac{1}{60}$	$\frac{1}{10}$	$10^{-6} - 10^{-2}$

Table 3: Parameter settings for the experiment

As time unit, we assume one minute. Thus, a transition with firing rate of 0.5 implies that, on the avarage, the transition fires every 2 minutes. We further assume an average audio duration of 10 minutes and a video duration of 60 minutes. Firing of error transitions (t_3 and t_6) depends first of all on a network reliability parameter γ, where $1/\gamma$ indicates the mean time between connection disruptions. Since this parameter stronly depends on the network state, we assume to vary γ between 10^{-6} and 10^{-2}. Furthermore, interruption likelihood is assumed to depend on the type of stream and its susceptability for QoS violations, and, inversely, on the number of still available resources providing redundancy for possible error recoveries.

In the first two experiments, rewards are only given for successfully completed streams, i.e., when a token appears in p_6 or p_7. Figures 3 and 4 show some of the results of a transient optimization process executed by the XPenelope tool for two cases: in both cases, pulse reward of one unit is assigned to a successfully completed audio connection. Video pulse rewards are 12 units in the first and 14 units in the second case.

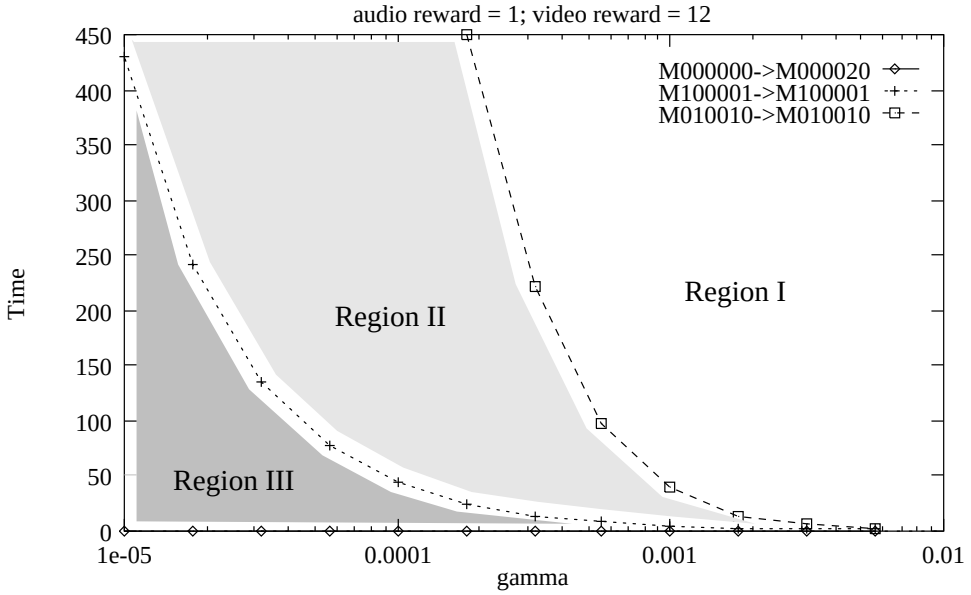

Figure 3: Strategy control regions for 12 video reward units

Curves on such result graphics generally have to be interpreted as follows: the first symbol in the name of the curve determines the marking for which this curve is valid. Only if the system is in the corresponding state, the curve applies. Now, one has to relate

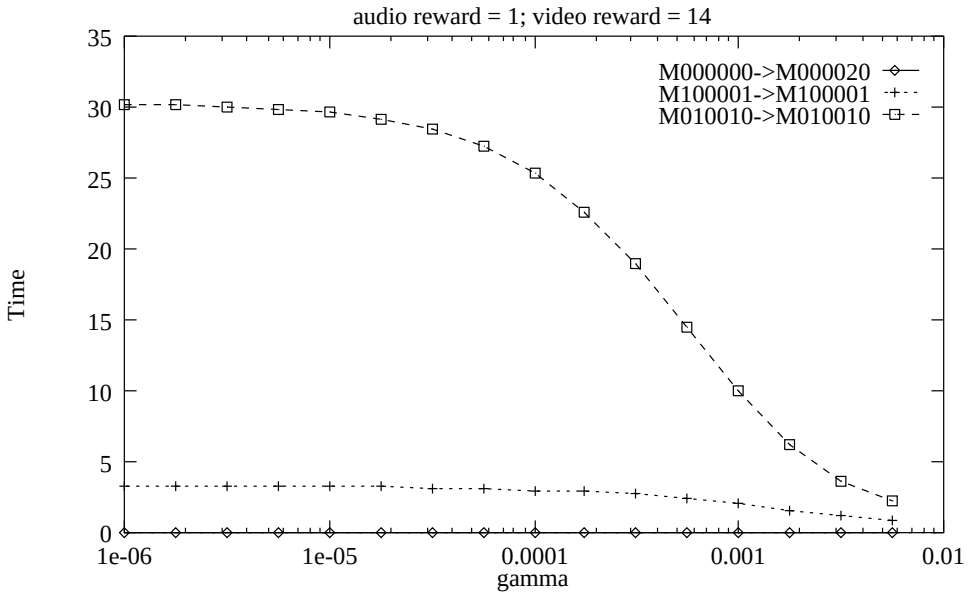

Figure 4: Strategy control regions for 14 video reward units

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 above the applicable strategy 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 below the corresponding curve, the system manager should better switch to an alternative reconfiguration[1].

Consider the curve $M010010 \rightarrow M010010$ in Fig. 3 as an example. This curve has to be considered by the QoS manager, exactly when the system is in state $M010010$, i.e., it has one active audio and one video stream. Let us further assume that the current value of γ is 0.001 and the remaining time of the application is 100 minutes. Under these circumstances, the system is classified above the curve and it is recommended *to remain* in state $M010010$, which means here: do nothing. Assume now that 75 minutes later, the system being again in the same state. Now, the remaining time is only 25 minutes, which means, the system's running condition is classified below the curve. The QoS manager should now trigger the alternative option, i.e., execute the indicated reconfiguration. Under the given circumstances this amounts to a switching to state $M020010$, i.e., to add 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 manager to always admit two video sources, when the system is in idle state. This, however, does not mean, that the system will always only run video streams. Audio streams may be admitted, e.g., when the system is in a video error state, depending on the corresponding curve.

[1]Depending on the number of reconfiguration edges originating from a given marking, the decision may not be binary.

Comparing the remaining two strategy curves of Fig. 3, it becomes obvious that the addition of further sources promises more reward, when the system contains active rather than aborted streams. Assuming a given γ of $\approx 3 * 10^{-4}$, in the case of two active sources ($M010010 \rightarrow M010010$), another source will not be added until about 100 minutes before the end of the application, while in the case of two aborted sources ($M100001 \rightarrow M100001$), the same decisison will be made only about 20 minutes before the end.

Comparing the strategies of Fig. 3 and Fig. 4, 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 runtime left. Even in the case of two active sources and a γ of 0.001, an audio source would be added no earlier than about 25 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 runtime and thus the probability of video completion becomes very small.

Fig. 5 shows results of the third experiment with an interesting variation: instead of a reward for successful completion, audio streams are rewarded during runtime. A reward rate of 0.1 is chosen per minute, which results in a total accumulated reward of 1 if completed successfully (since audio runtime is 10 minutes).

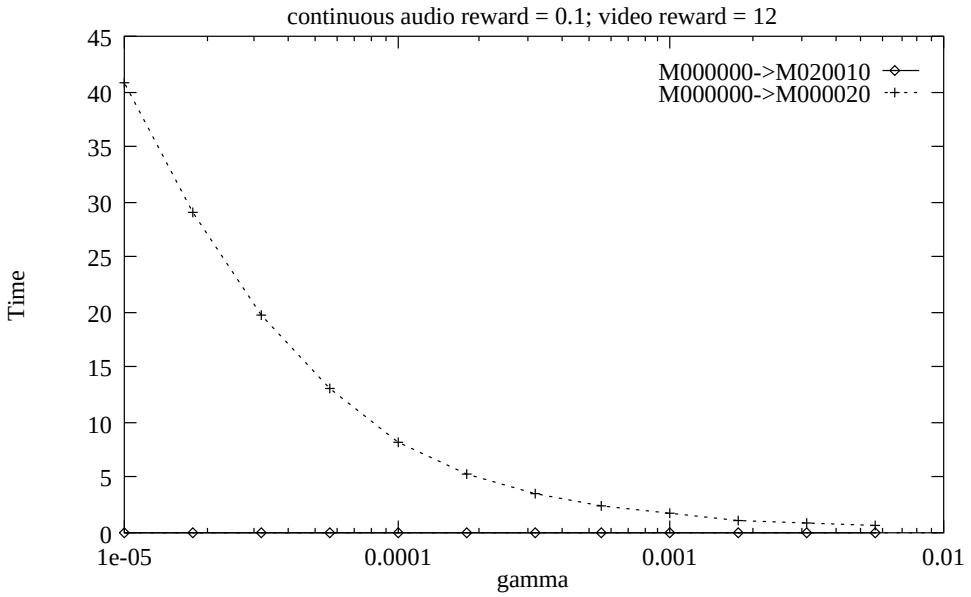

Figure 5: Strategy curves for continuous audio reward

The analysis shows that it is not always any more of advantage in state $M000000$ to run two videos. For a γ of 0.0001, for instance, two videos should only be admitted if the remaining time is more than about 8 minutes. Below that, it is better to admit 2 audio streams and one video stream. The reason for this is that it is now more likely to gain rewards for the audio connections, since such streams are already rewarded during runtime.

Finally, Fig. 6 shows the performance graph for the first experiment. The curve determines which reward can be expected for a certain γ, if the starting state is $M000000$.

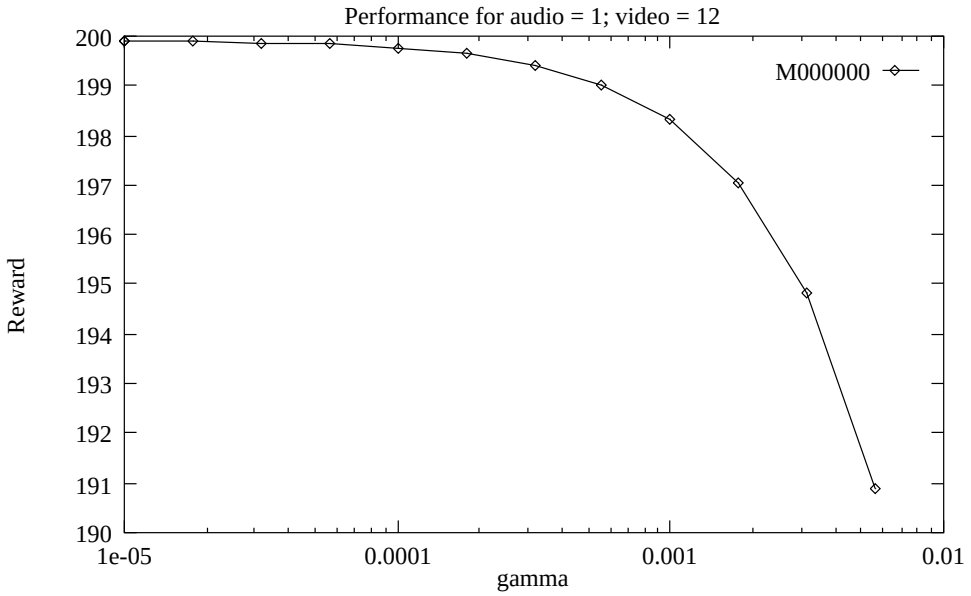

Figure 6: Performance curve

In a certain range of γ values, the reward does not depend very much on these values, whereas higher values of γ have an adverse effect on on the total expected reward.

6 Conclusions and Outlook

We are currently extending our coverage and analysis of QoS management functions to, among others, QoS adaptation, negotiation and renegotiation. COSTPNs of such systems are already being developed. We are also in the process of applying this new modelling method to a new kind of QoS management jointly developed at the Universities of Montreal and Hamburg, the Cooperative QoS Management [6].

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*, 1997. To appear.

[3] 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.

[4] 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.

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

[6] 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.

[7] M. A. Marsan, G. Conte, and G. Balbo. A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.