QoS Management: A Model-Based Approach

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

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

Abstract

Quality of Service (QoS) management is an important issue in today's high-speed distributed systems supporting multimedia applications. Most existing QoS management schemes usually just cope with technical issues of resource reservations and QoS guarantees, often completely neglecting revenue issues which are especially important for service providers in order to maximize their profit. The revenue to be expected does not only depend on the stream itself, but also very much on stochastic events such as network failures or QoS violations. A QoS management system taking revenue issues and the possibly stochastic behavior of the environment into account thus 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 models using a tool environment in order to obtain strategies for the QoS management system.

Keywords: *Controlled Stochastic Petri Nets, Multimedia, QoS Management, Performability, Optimization*

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 utilization of networking bandwidth and processing power in the end systems. In an environment where users pay for a certain level of quality, the traditional best-effort approach is often no longer suitable, especially when applications run in a shared environment. Rather, resources have to be allocated and managed during the application's lifetime in order to guarantee the requested and negotiated quality levels.

Most existing QoS management schemes (for an overview see [2]) just cope with technical issues of resource reservations and QoS guarantees, often completely neglecting *revenue issues* which are especially important for service providers in order to maximize their profit. The revenue to be expected does not only depend on the stream itself, but also very much on stochastic events such as network failures which 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). But also the arrival of admittance requests of other sources can be characterized as a stochastic process. Consider the case that a source has just been admitted and then, a new request arrives from a source which promises a much higher reward than the first one due to the different nature of the stream (color video instead of black&white, for instance). Unfortunately, it may happen that there are not enough resources to run the new source. Therefore, it could have been advantageous for the service provider not to admit the low-revenue source but to wait for the arrival of one offering high revenue.

For this reason, a QoS management system taking stochastic events, the probability of future behavior and revenue issues into account will usually perform better than 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 (COSTPNs)* [5] and the numerical analysis of the underlying model of Extended Markov Reward Models (EMRMs).

The paper is organized as follows: in Sections 2 and 3, we give brief introductions into QoS management and COSTPNs/EMRMs, respectively. Section 4 then shows how typical QoS management functions can be modeled by COSTPNs. Section 5 analyzes the examples in order to show how strategies for controller programs can be derived from the COSTPN. The analysis is done using a tool environment. Finally, Section 6 gives an outlook on future activities.

2 QoS management

QoS architectures and their respective management systems as they exist today [2] are providing the user with end-to-end service quality guarantees. They usually offer a number of *QoS management functions* to cope with the different situations occurring during the lifetime of a system. Examples of such QoS functions are QoS specification, mapping, negotiation, renegotiation, monitoring, and adaptation. Furthermore, management functions make use of lower-level *QoS mechanisms*, examples of which are resource reservation and admission control.

The basic QoS function available in any QoS management system is QoS specification. It defines which QoS parameters are available and determines their syntax and semantics.

Since most QoS architectures consist of several levels of abstraction — for instance QoS on the user, the transport, the network and the operating system level — a mapping function has to be provided which translates QoS requirements expressed at one level to those of another level. As an example, consider a user requirement to receive a movie in high-quality, big size and color. This has to be translated into transport requirements which are formulated in terms of throughput, delay and jitter.

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 already participating users. This function usually employs several QoS mechanisms to fulfill its task: admission control is used to determine whether a new user can be served, while resource reservation has to be called as soon as the user is admitted, in order to guarantee the requested service quality.

A renegotiation may be initiated by the user or the service provider. 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 renegotiation is crucial for applications where a constant QoS cannot be specified for the whole duration of the service. Examples are medical applications, tele-conferencing systems, and computer supported cooperative work 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 (such a violation is 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 upon 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. More generally, the role of QoS adaptation is to keep providing the service, eventually lowering the service quality in case of resource shortage. 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.

3 COSTPN 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. Another feature of EMRMs is provided by the so-called *branching* states. 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 or equal than the performability measure under any other strategy $\mathbf{S}(t)$.

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 two examples which show how COSTPNs can be used to model QoS management problems. The examples deal with one QoS mechanism, namely admission control of audio and video sources to a multimedia transmission system (a router, for instance), and one QoS function, namely QoS adaptation.

4.1 Admission Control

In our example, audio and video sources arrive in a waiting room and ask for admission to the system. This means that 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 throughout the source's presence in the system (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 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 this system is depicted graphically in Figure 1, and Table 1 gives transition priorities and firing rates for this COSTPN.

The waiting rooms modeled by places p_1 and p_3 have limited capacity, and therefore the maximum number of tokens in each of these places is bounded. This fact is modeled by inhibitor edges ending with a circle instead of an arrow head. The edge's label determines the maximum number of tokens in the place.

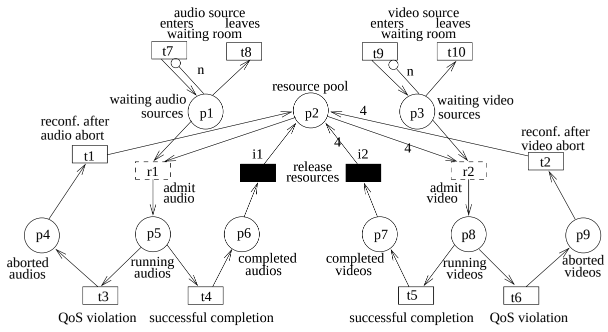

Figure 1. COSTPN for an admission controller

transition	prio.	firing rate/prob.
t_1, t_2	0	$\#p_4 * \tau_1, \#p_9 * \tau_2$
t_3, t_6	0	$(max(p_2) - \#p_2 + 1)^2 * \tau_{3/6} * \gamma$
t_4, t_5	0	$\#p_5 * \tau_4, \#p_8 * \tau_5$
t_7, t_9	0	τ_7, τ_9
t_8, t_{10}	0	$\#p_1 * \tau_8, \#p_3 * \tau_{10}$
i_1, i_2	1	1
r_1, r_2	0	—

Table 1. Transition descriptions

The possibilities of *taking decisions* are modeled by the reconfiguration transitions r_1 and r_2[1], 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 is 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 source).

Each timed 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: $\#p_5 * \tau_4$. Transitions t_3 and t_6 model severe QoS degradation leading to transmission interruption. Transitions t_7 and t_9 put newly arriving sources into the waiting room. If they are not served within a certain period of time determined by the firing rates of t_8 and t_{10}, these sources leave the system unserved.

Furthermore, the COSTPN contains two immediate transitions, i_1 and i_2. They have been introduced to assign *pulse* rewards to successfully transmitted streams and to model the instant release of occupied resources. Thus as soon as a token appears in either p_6 or p_7, i_1 or i_2, respectively, are immediately executed.

Rewards may be assigned throughout the lifetime of a stream or when it finishes successfully. Therefore, we use the following reward function:

$$rew(M) = \begin{cases} \#p_5 * rew_a + \#p_8 * rew_v & : \text{M tangible} \\ \#p_6 * xrew_a + \#p_7 * xrew_v & : \text{otherwise} \end{cases}$$

4.2 QoS adaptation

Our second example deals with QoS adaptation. A QoS manager able to switch between two quality levels (both marked as acceptable by the application) for running streams is modeled by the COSTPN shown in Figure 2; transition descriptions are given in Table 2.

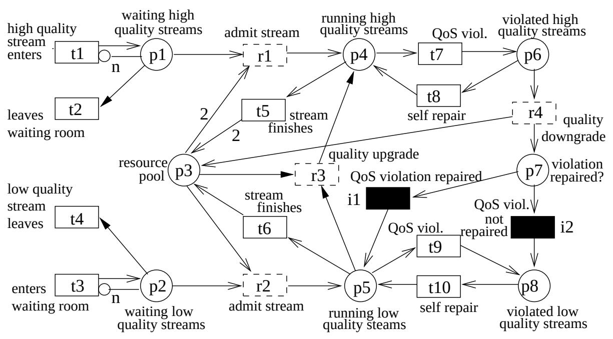

Figure 2. COSTPN for QoS adaptation

transition	prio.	firing rate/prob.
t_1, t_3	0	τ_1, τ_3
t_2, t_4	0	$\#p_1 * \tau_2, \#p_2 * \tau_4$
t_5, t_6	0	$\#p_4 * \tau_5, \#p_5 * \tau_6$
t_7, t_9	0	$\#p_4 * \tau_7, \#p_5 * \tau_9$
t_8, t_{10}	0	$\#p_6 * \tau_8, \#p_8 * \tau_{10}$
i_1, i_2	1	$prob(i_1), 1 - prob(i_1)$
r_1, r_2, r_3, r_6	0	–

Table 2. Transition descriptions

Similar to the example in Section 4.1, requests for both quality levels (high and low) may arrive (t_1, t_3) in waiting rooms (p_1, p_2), and they may leave the system if they remain unserved for a certain period of time (t_2, t_4). We assume that a high quality stream needs two resource units and a low quality stream one unit. Any waiting stream can

[1] In COSTPNs, reconfiguration transitions are graphically indicated by dashed transition symbols, while timed transitions are represented by regular and immediate transitions by black boxes.

be admitted by firing reconfiguration transitions r_1 or r_2, respectively, as long as enough resources are available. As in the previous example, the waiting room capacity is limited.

Rewards are associated only with currently served streams and are awarded per unit of time. A stream transmitted in high quality will allow the service provider a higher reward compared to a stream transmitted in low quality. We use the following reward function, with rew_{q_h} and rew_{q_l} denoting the reward per unit of time for a high and low quality stream, respectively:

$$rew(M) = \begin{cases} rew_{q_h} * \#p_4 + rew_{q_l} * \#p_5 & : \text{M tangible} \\ 0 & : \text{otherwise} \end{cases}$$

Since all qualities have been negotiated as acceptable for each stream, the switching of qualities for a given stream can be done without renegotiation between service provider and user. The QoS management may switch a stream to a higher quality (r_3) in order to improve the revenue gained from the stream transmission. Such an action requires the availability of further resource units. The manager may also downgrade the quality (r_4) in order to reduce the system load to be able to react to QoS violations. Such a downgrade implies two effects: First, the likelihood of a quick state improvement is increased, and second, resources are freed to allow for further access of waiting streams.

QoS violations may occur from time to time (t_7, t_9). QoS violations decrease the revenue a service provider receives in that the violated stream does not contribute as long as the violation condition exists. The violation condition may disappear after a while (t_8, t_{10}) due to a change of conditions somewhere in the environment, or the manager may react on the violation. A reaction consists in a quality downgrade for one or more streams (r_4). However, such a downgrade need not necessarily lead to an improvement of the situation. By the immediate transition i_1, we model successful downgrades, while i_2 denotes a violation condition which lasts despite the downgrade.

5 Computing Strategies for QoS managers

To allow for a numerical analysis of COSTPNs resp. their corresponding EMRMs, the tool PENELOPE has been developed at the University of Hamburg [3]. It allows the application of several algorithms from Markov decision theory for transient or stationary optimization of performability measures. In practice, short term behavior is of high importance for an efficient control. Transient control strategies should be applied whenever steady-state conditions cannot be guaranteed, which is mostly the case in practical settings. Therefore, we concentrate on transient optimization in the examples presented here. Until recently transient control has been neglected merely due to its inherent mathematical complexity.

In this section, we show for both examples from the previous section how a numerical analysis can be employed to compute transient strategies for QoS management controller programs. We use PENELOPE's capabilities to graphically display resulting strategies in diagrams (Figures 3 and 4). Each of these diagrams is valid in exactly one marking of the analyzed COSTPN model. It visualizes the optimal decision in this state as a function of the system runtime (on the y-axis) and some other parameter (on the x-axis). Time is represented in a reverse pattern, namely as the *remaining time* $t' = T - t$ where T is the time horizon and t the elapsed time. The decision space is divided into *strategy control regions*. In order to find the decision to take, one has to locate the system in the decision space according to the current parameter setting and the remaining time and then use the strategy recommended by the strategy control region applicable in this situation.

5.1 Admission control

For the analysis, the COSTPN from Figure 1 first has to be translated into an EMRM, which is done automatically by the tool. In this example, the initial marking of the admission control COSTPN is 8 tokens (resource units) in place $p2$ and no further token in any other place. Due to the size of the resulting EMRM, the advantage a modeller gains when using high-level COSTPNs instead of EMRMs directly is obvious: the COSTPN has 9 places, 10 timed, 2 immediate and 2 reconfiguring transitions, while the corresponding EMRM has 952 markov and 788 vanishing states. It also has 4268 timed, 788 probability and 394 reconfiguring edges.

In order to conduct experiments, values have to be selected for the parameters described in Table 1. The following values were selected for the experiment described below: $\gamma = 0.001$, $\tau_1 = 1, \tau_2 = \frac{1}{2}, \tau_3 = 5, \tau_4 = \frac{1}{10}, \tau_5 = \frac{1}{60}, \tau_6 = \frac{1}{10}, rew_a = rev_v = 0$ and $xrew_a = 1$.

As time unit, we assume one minute. Thus, a transition with firing rate of $\frac{1}{2}$ implies that, on the average, the transition fires every 2 minutes. We assume an average audio duration of 10 minutes ($1/\tau_4$) and an average video duration of 60 ($1/\tau_5$) minutes. Firing of error transitions (t_3 and t_6) depends on the network reliability parameter γ, where $1/\gamma$ indicates the mean time between connection disruptions, and on the susceptibility to QoS violations of the different media types, expressed by the values of τ_3 and τ_6.

The reward values we use for the experiments assign rewards only for successfully completed streams ($rew_a = rew_b = 0; xrew_a, xrew_v \neq 0$), i.e., when a token appears in p_6 or p_7. Thus, there is, unlike in the QoS adaptation model as described in Section 4, no reward if a source suffers from QoS violation and is interrupted.

In the experiment, we varied the number of video rewards between 4 and 11, keeping the audio reward constantly at a value of 1. The resulting strategy control regions for state M(1,8,1,0,0,0,0,0,0) (one audio and video waiting, no active streams) are displayed in Figure 3. Here we see that in the area where less than about 9 reward units are paid, it becomes more and more attractive to only admit the audio source, at least within a certain interval of the remaining time. Still, there is a region where it is worthwhile admitting both waiting sources (note the interesting shape of this region), but for higher rewards and much runtime left, only the video will be admitted. It can be expected that with lower video rewards (between 0 and 2), the "audio only" region will be completely dominant such that only audios will be admitted during the whole system lifetime.

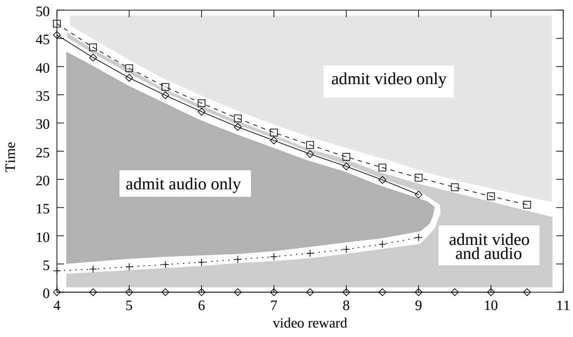

Figure 3. Admission control results.

5.2 QoS adaptation

For the adaptation example, the following settings have been selected: $\tau_{1/3} = \frac{1}{2}$, $\tau_{2/4} = 1$, $\tau_5 = \frac{1}{20}$, $\tau_6 = \frac{1}{10}$, $\tau_7 = \frac{1}{40}$, $\tau_8 = \frac{1}{2}$, $\tau_9 = \frac{1}{20}$, $\tau_{10} = \frac{1}{10}$, $rew_{qh} = 5$ and $rew_{ql} = 2$.

In this example, we are mainly interested in the question whether a violated high quality stream in place p_6 should be downgraded (r_4). We do not discuss the question whether newly arrived sources should be admitted (r_1, r_2).

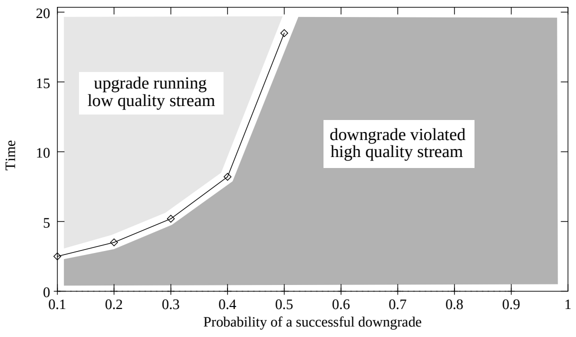

Figure 4. QoS Adaptation results.

Figure 4 displays the strategy regions for the marking M(0,0,2,0,1,1,0,0) (one violated high quality stream and one running low quality stream), where the downgrade success probability has been varied. There is a region with high success probability where a downgrade will always be performed ($prob(i_1) \geq 0.6$), even though the high quality reward rate is 5 units and thus muchhigher than the low quality rate. This is simply due to the fact that a downgraded stream can easily be upgraded again by switching r_3. With lower success probabilities, downgrades will only be performed with a few minutes remaining time, since an unsuccessful downgrade (which is quite likely in this region) will lead to a long waiting period in p_8 (around 10 minutes: $1/\tau_8$). With more time left, the alternative should be selected which in this case is not to do nothing as in the previous diagram, but to upgrade the active low quality stream.

6 Conclusions and Outlook

We are currently extending our coverage and analysis of QoS management functions to, among others, QoS negotiation and renegotiation. COSTPNs of such systems are already being developed. We are also in the process of applying this new modeling method to a new kind of QoS management recently developed jointly at the University of Montreal and the University of 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.

[3] H. de Meer and H. Sevcikova. PENELOPE: dePENdability EvaLuation and the Optimization of PErformability. In *Proc. 9th Int. Conf. on Modelling Tools and Techniques, St. Malo, France, LNCS 1245*, pages 19–31. Springer, June 1997.

[4] H. de Meer, K. S. Trivedi, and M. D. Cin. Guarded Repair of Dependable Systems. *Theoretical Computer Science, Special Issue on Dependable Parallel Computing 128*, 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*, pages 18–25, Oct. 1997.

[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.