Hybrid performance modeling of open-loop video distribution with VCR functionality

Armin Heindl[a*], Ernst Biersack[b]

[a]Technische Informatik, Fak. IV, TU Berlin, Einsteinufer 17, 10587 Berlin, Germany

[b]Institut Eurécom, BP 193, 06904 Sophia Antipolis, France

In open-loop video broadcasting systems, the video is commonly partitioned into segments, which are transmitted periodically and simultaneously on different channels. Recently, a bandwidth-efficient rate-increasing scheme has been proposed, which provides probabilistic support of VCR functionality and at the same time retains the excellent scalability of open-loop video distribution. We present a hybrid (i.e., discrete-continuous) stochastic Petri net model of the Video-on-Demand client-server system, which captures all relevant system aspects in a concise way and is assessed by simulation. The hybrid stochastic Petri net excels in its flexibility – both in defining various performance measures and in its extensibility to include, for instance, full VCR functionality and arbitrary user behavior. Unlike in many simulation studies, a compact graphical representation of the Petri net model unambiguously reveals all implemented details.

1. Introduction

Providing a scalable VoD (Video-on-Demand) service where a large number of users receives the same video simultaneously excludes the use of closed-loop video distribution systems that serve each client individually by a separate stream. Instead, the server must adopt an open-loop approach, in which the cost for the server is *independent* of the number of clients. Thus, open-loop video distribution schemes are especially suited for (very) popular videos. Typically, these distribution systems do not support interactive VCR functions such as Fast Forward or Jump. Instead, the user is restricted to view the video from the beginning until the end. In particular, Fast Forward may cause video reception to fall behind the video consumption leading to undesired discontinuous playout. In [1], it is shown how to support VCR functions either deterministically or probabilistically (zero or low probability for discontinuous playout). This scheme, in contrast to related work like staggered broadcast [5], tolerates arbitrary VCR functionality and thus provides full VoD services.

In this paper, we study the transmission scheme of [1] for probabilistic support of VCR functionality with a twofold purpose: First, we develop a hybrid performance model that conveniently allows to consider more complex video viewing behaviors and allows to investigate a wider range of performance measures in the trade-off between the additional bandwidth requirements and client satisfaction (expressed in measures related to (dis)continuous playout). Second, modeling the VoD client-server process as a *fluid stochastic Petri net* (FSPN, [8]) reveals all details of the performance model, which are

*This work was supported by DFG under grant Ho1257/17.

usually hidden in source code and typically not outlined in simulation studies. Using the publicly available tool SPNP [3] for evaluation helps to make the results reproducible.

FSPNs, as an extension of discrete-state stochastic Petri nets [9], are well suited to represent concurrency and synchronization in hybrid systems, i.e., in systems having both discrete and continuous components that evolve over time. Different variants of FSPNs have been shown to be practicable in performance and dependability evaluation – both in analytical and simulative studies (e.g., [8,7,4]). The model presented in this paper refers to the FSPN definition in [4], which provides non-exponential timing in addition to continuous model parts. This allows to build a realistic model, which still admits a stable and efficient simulation, e.g., by means of the tool SPNP.

The paper is organized as follows. In Section 2, we present the so-called open-loop tailored transmission scheme and discuss how to adapt it to support VCR interaction. After a description of the developed FSPN model in Section 3, numerical experiments based on this model are presented in Section 4. Section 5 concludes the paper.

2. Tailored Transmission Schemes for Open-loop Video-on-Demand

In [2], Birk and Mondri propose *tailored transmission* schemes, which generalize many other previously published open-loop VoD schemes. In this paper, we focus on the base version of the tailored transmission scheme, in which the video is partitioned into N equal-length segments of size D (in bits)[2]. Each segment is transmitted periodically and indefinitely often on its own channel with transmission rate r_i (in bits/sec) for segment i, where r_i decreases with increasing segment number (i.e., $r_i \geq r_{i+1}$ for $i = 1, \ldots, N-1$). A client who wants to view the video listens to all N channels simultaneously and records the segments. Once a segment is fully received, this part of the video can be consumed, say with the video consumption rate b (bits/sec) in PLAY mode.

Throughout the paper, we assume the following:

- The client starts recording all segments at the same instant. He is not required to begin at the starting point of a segment. Independent of this instant, the reception of full segment i will always be completed after $\frac{D}{r_i}$ seconds.
- The client has enough disk storage to buffer the contents of the whole video and sufficient capabilities with respect to network access and disk I/O bandwidth to record all segments simultaneously.

Commercially available digital video recorders already meet the latter assumptions.

For now – until recalled –, let us suppose that right after the reception of the first segment the client remains in PLAY mode with consumption rate b. For continuous playout following the current segment, the client must have received the next segment entirely, before he finishes viewing the current one and desires to go to the next. We are interested in the minimal transmission rates $r_i^{\min}$ that assure the continuous playout of the whole video. In the light of our assumptions, it is easy to derive (see [1]) that

$$r_i^{\min} = \frac{b}{i} \qquad \text{for } i = 2, \ldots, N \qquad \text{(when } r_1 \text{ is set to } b \text{ rather arbitrarily).} \tag{1}$$

Figure 1 illustrates the tailored transmission scheme for minimal transmission rates and with $r_1 = b, N = 4$. For a client who starts recording at time t_0, the hatched areas cover exactly the content of each segment as received by the client. Segment i is entirely received at time $t_i = t_0 + i \cdot \frac{D}{b}$. Thus, the startup latency of the scheme corresponds to $t_1 - t_0 = \frac{D}{b}$. The total server transmission bandwidth is $R_t^{\min} = b \sum_{i=1}^{N} \frac{1}{i}$.

[2]For simplicity, we assume throughout the paper that the video is constant bit rate.

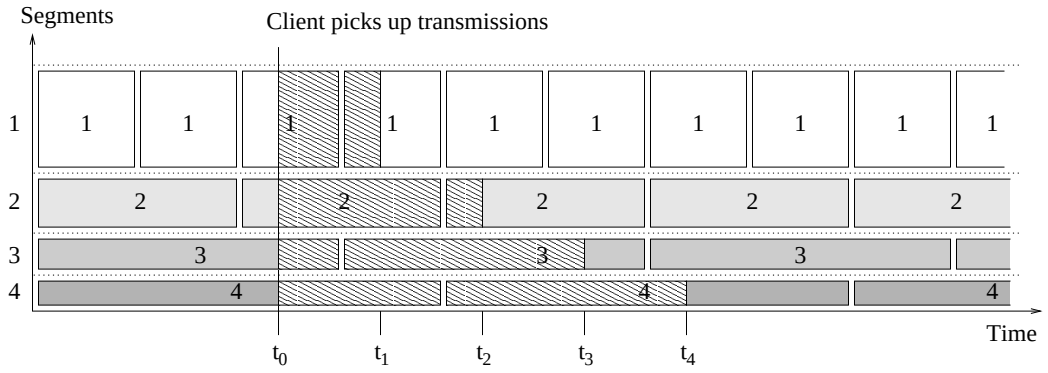


Figure 1. The base scheme of tailored transmission with minimal transmission rates. Hatched areas indicate the data of each segment necessary to fully receive a segment.

During the consumption of the video, a client may decide to make use of the various VCR functions. Besides PLAY mode, we consider

PAUSE: interrupt the playout of the video for some period,

FF (Fast Forward): playback the video at a consumption rate $X_F \cdot b$ for some period,

FB (Fast Backward): playback the video at a consumption rate $-X_F \cdot b$ for some period,

where $X_F > 1$. Additionally, SF (Slow Forward) and SB (Slow Backward) can be defined in analogy to FF and FB with a respective rate factor $X_S < 1$. These functions pose no specific difficulties compared to FF and FB and could be easily covered by our model in Section 3.

Given the tailored transmission scheme with minimal transmission rates, the VCR functions PAUSE and FB (as well as SF and SB) will never account for a failure in the sense that the client attempts to consume a segment, before it has been fully received. These user interactions are simply enabled by sufficient buffer at the client. Only the action FF, which accelerates the consumption of the video with respect to the PLAY mode, may lead to the described failure situation. How should the segment transmission rates be adapted (increased) in order to avoid that the video data required for playout of a segment are not yet available? To ensure that any possible FF command can be successfully executed, the worst case scenario, where the client remains in FF mode throughout the complete video, may be analyzed as in [1]. Playout and VCR actions begin only after the first segment has been entirely received. The resulting maximal transmission rates

$$r_i^{\max} = \frac{bX_F}{X_F + i - 1} \qquad \text{for } i = 1, 2, \ldots, N \tag{2}$$

give a deterministic guarantee that any VCR operation is supported.

Probabilistic VCR support: the rate increase factor

Usually, a client will alternate between different VCR modes including PLAY and FB thus taking the strain off the transmission requirements. Furthermore, it might be tolerable to support VCR interactions with a high probability only (instead of a 100% guarantee), when – as a trade-off – segments can be transmitted at an aggregate rate lower than $R_t^{\max} = \sum_{i=1}^{N} r_i^{\max}$. In this context, we recall the proposal in [1] for a probabilistic support of VCR functionality based on a *rate increase factor* A: While segment 1

is still transmitted at rate $r_1 = b$, the server delivers the segments $i = 2, \ldots, N$ at rates $r_i^I = A \cdot r_i^{\min}$, where $1 \leq A \leq X_F$ and $r_i^{\min}$ is computed in (1).

In the next section, we provide a hybrid stochastic Petri net model which allows us to evaluate the performance of the probabilistic VCR support (e.g., in terms of blocking probabilities) depending on the rate increase factor. Obviously, different user profiles impose different rate requirements (specified by A) to achieve the same performance.

3. An FSPN model for probabilistic VCR support

We develop an FSPN model of the client-server process in which a video that is broadcasted via the tailored transmission scheme is viewed by a user with a specific, but random behavior pattern. The model will be specified in an SPNL-like notation, which we prefer due to its well-organized mixed textual and graphical representation. Since SPNL (*stochastic Petri net language*, [6]) currently does not envision fluid components of stochastic Petri nets, straightforward and common extensions for their specification are added to the model in Figure 2 and will be explained following the informal introduction of the discrete formalism. Thereafter, we describe the behavior of the presented model along with specified measures.

SPNL and fluid extensions

In SPNL, common SPN elements, like (discrete) *places* (represented as plain circles), input, output and inhibitor *arcs* (the latter with a small circle instead of an arrowhead), indistinguishable *tokens* (dots or parameters in places), timed and immediate *transitions* (empty rectangles or black bars, respectively) are used as in ordinary discrete-state SPNs. Although firing times may be chosen to be quite general, we will employ only exponential (exponential transitions) and deterministic distributions (deterministic transitions) in this study. We also use (marking-dependent) expressions for *arc multiplicities* and *guards*. Marking-dependent expressions, also called *rate rewards*, are formulated with respect to the number of tokens in places (`#P` denotes the number of tokens in discrete place `P` and – as we will see – may also be used for the non-integral token number in a fluid place). In SPNL, arc multiplicities as well as all identifiers are located in the proximity of the corresponding objects, while all other expressions, like transition attributes (e.g., firing time distributions, priorities, weights, guards), are listed below the graphical representation of the net. Among those transitions,

whose input places have at least as many tokens as the respective arc multiplicities, and

whose firing is not prevented a priori by an activated inhibitor arc, and

whose (possibly) associated guard – a rate reward that can be interpreted as a Boolean expression – evaluates to TRUE,

only these with the highest priority are actually enabled. Timed transitions have (the lowest) priority zero by default. Conflicts between simultaneously enabled immediate transitions are probabilistically resolved according to their weights.

In coexistence with the discrete elements introduced so far, an FSPN additionally provides continuous components[3], namely

fluid places (depicted as two concentric circles, e.g., `FPAvailVideo` and `FPforFB` in Figure 2), which may contain non-integral tokens or a fluid quantity, like the initial parameter `sd` in fluid place `FPAvailVideo`,

[3]For easier distinction, we choose identifiers starting with an F for fluid elements.

fluid transitions[4] (depicted as rectangles with fill pattern, e.g., `FTconsume` and `FTrewind`), for which no firing time distributions are defined, but which are formally enabled by the discrete marking in connected non-fluid places or by guards, and

fluid arcs, which connect fluid places and fluid transitions and carry a continuous flow (depicted by thicker arrows, e.g., between `FBforFB` and `FTrewind`).

At the same time conventional arcs, which connect fluid places and non-fluid transitions, are extended to real-valued arc-multiplicities corresponding to fluid impulses: The firing of these transitions instantly adds or removes a fluid quantity to/from the fluid place. In contrast, fluid transitions – if enabled – cause the fluid to flow continuously into or out of fluid places along the linked fluid arcs (with the respective change rates). Otherwise, guards and other marking-dependencies may be quite generally defined with mutual impact between the discrete and continuous parts of the stochastic Petri net.

The complete FSPN model is arranged in the SPNL module VoD (see Figure 2) – similar to the module concept of programming languages like Ada. Following the list of parameters used below to define the initial marking, arc multiplicities, rates and delays, two (stationary) performance measures (and two rate rewards) are declared in the public part of the *process* transmit_consume (between the boldfaced keywords **process** and **private**). Underneath the graphical area (keyword **smeasure**), these measures related to a blocking situation are expressed in terms of rate and impulse rewards. E{`#P`} gives the expected number of tokens in (discrete) place `P` and E{`#T`} the mean number of firings per unit time (throughput) of non-fluid transition `T`.

Description of the model dynamcis

The tailored transmission scheme, the video consumption and the user behavior profile make up the VoD client-server process. The model of Figure 2 describes the dynamics of this system starting at the instant at which the user has just received the first of N segments until the last segment is completely consumed. At the end of such a cycle, immediate transition `tReset` resets the tailored transmission scheme to the initial cycle state. The tailored transmission scheme is modeled in the upper part of the graphical area above transition `tReset` and fluid places `FPAvailVideo` and `FPforFB`. The token circulating through the bottom places `PPLAY`, `Pchoice`, `PFB`, `PFF`, `PPAUSE` emulates the considered user behavior. We assume that in each PLAY phase the user may activate a VCR function before returning to another PLAY phase (place `PPLAY`). The probabilities for choosing Pause or Fast Backward are 0.1 each, and the probability to enter Fast Forward is 0.8 correspondingly (see weights for immediate transitions `tFB`, `tPAUSE` and `tFF`). Unless influenced by the actual video consumption process (shown in the middle part), the sojourn time in each VCR mode is exponentially distributed (keyword **exp** with rate parameter in brackets in **transatt** section). The arcs connecting `tReset` with the user behavior subnet ensure that video consumption starts in PLAY mode again at the beginning of a cycle.

For our performance evaluation, it suffices to know when the N video segments have been fully received. The firing of immediate transition `t3` models the occurrence of such an event. Since the rate increase factor A only affects the transmission rates of segments $2, \ldots, N$, the interarrival time between the first and second segment will differ from the other interarrival times, which are identical due to the specific properties of the tailored transmission base scheme (see Section 2). Deterministic transitions `T1` and `T2` account for this fact. The initial fluid level `sd` in fluid place `FPAvailVideo` (which corresponds to the

[4] This term is not discerned from timed transition in [4], but helps our intuition for the considered example and corresponds to the inf-transition in SPNP.

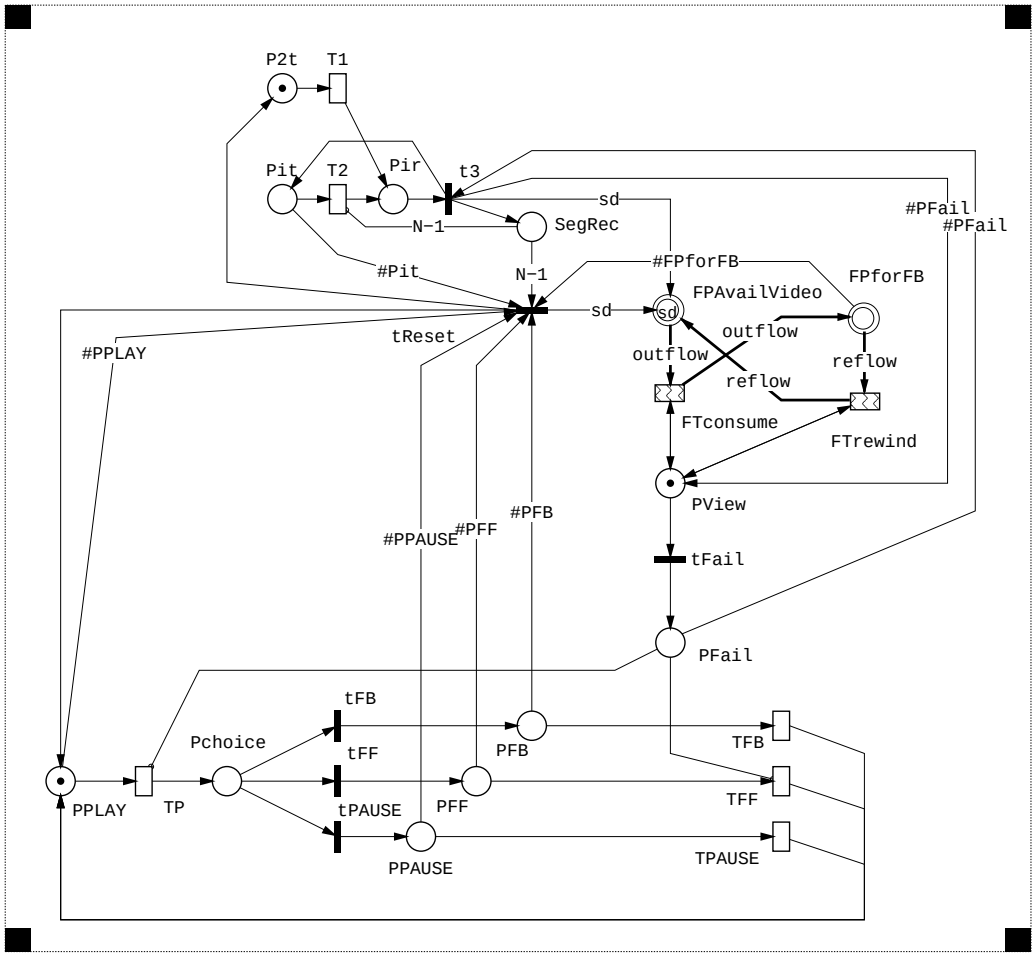
```
module VoD;
 parameter N=36;   (* number of equal-length segments, in which video is transmitted
                     on different channels *)

           L=7200; (* video length in sec *)
           sd=L/N; (* segment duration in sec, equals D/b *)

           XF=3;   (* FF playout factor *)
           A=1.4;  (* rate increase factor for transmission,
                     applies only to segments 2,3,...,N *)

           lam_P =1/45;   (* rate for exponential PLAY period *)
           lam_FF=1/9;    (* rate for exponential FF period *)
           lam_FB=1/9;    (* rate for exponential FB period *)
           lam_PAUSE=1/9; (* rate for exponential PAUSE period *)

process transmit_consume;
    smeasure blocking_time, blocking_prob;
    rreward  outflow, reflow;
 private
```




```
 smeasure   blocking_time = E{#PFail};       blocking_prob =   E{#tFail} / E{#t3};
 rreward    outflow = #PPLAY + XF*(#PFF);    reflow = XF*(#PFB);
 transatt   T1:   dist=det(sd*(2-A)/A);      T2:      dist=det(sd/A);
            TP:   dist=exp(lam_P);           TPAUSE:  dist=exp(lam_PAUSE);
            TFF:  dist=exp(lam_FF);          TFB:     dist=exp(lam_FB);

            tFB,tPAUSE:  weight=0.1;
            tFF:         weight=0.8;

            (* pseudo SPNL code for fluid part of model: *)
            FTrewind:  guard=(#FPforFB>0);
             tFail:    guard=(#FPAvailVideo=0);
             tReset:   guard=(#FPAvailVideo=0), prio=2;

end transmit_consume;
end VoD.
```

Figure 2. The SPNL model of NVoD

segment duration in PLAY mode), indicates that the cycle starts right after the reception of the first segment. Thus, the inhibitor arc from place `SegRec` to `T2` with multiplicity $N-1$ warrants that only $N-1$ more segments are delivered within a cycle period.

Until then, immediate transition `t3` keeps re-enabling transition `T2` (when a segment has been entirely received) and at the same time deposits the fluid impulse `sd` into fluid place `FPAvailVideo`. The consumption of these video sequences is modeled by the middle subnet, which in turn is influenced by the user behavior subnet. Among other discrete elements, the middle subnet contains two fluid places (`FP...`) and two fluid transitions (`FT...`) with pertinent fluid and fluid-impulse arcs. Much of the behavioral complexity of the video consumption process is encoded in guards (see keyword **guard** for `FTrewind` and `tFail`) and marking-dependent flows (see keyword **rreward** for declaration/definition of rate rewards `outflow` and `reflow`). Video is only consumed (including FB and PAUSE), if place `PView` is marked: its initial token remains there, as long as video sequences are available.

In our model, disrupted video consumption corresponds to the situation that fluid place `FPAvailVideo` becomes empty before the last segment has been received (`#SegRec`$< N-1$). These conditions[5] trigger immediate transition `tFail` (see guard `#FPAvailVideo`=0), which moves the token from place `PView` to place `PFail`. Eventually – with the arrival of the next segment – transition `t3`, which fires independently of the marking of `PFail` (due to the marking-dependent arc multiplicities), shifts the token – if present – back to place `PView`. The time a token spends in `PFail` thus corresponds to an unvoluntary interruption in the video consumption process, during which the phases PLAY and FF are suspended (see inhibitor arcs from `PFail` to `TP`/`TFF`)[6].

With place `PView` being marked, fluid transition `FTconsume` is formally enabled (due to the arcs between `PView` and `FTconsume`). However, the actual continuous flows along its fluid arcs depend on the state of the user profile subnet: According to the marking-dependent flow rates (see `outflow`), the video is consumed at constant rate 1, if `#PPLAY`=1 (user in PLAY mode), at constant rate `XF`, if `#PFF`=1 (user in FF mode), and at rate 0 otherwise. In the former two cases, fluid is also directed at respective rates to the second fluid place `FPforFB`, whose level records the maximum time by which the video can be rewound (i.e., the video time shown on conventional VCR displays). Consequently, this fluid level may decrease, when the user decides to rewind the video. Transition `FTrewind` with its fluid arcs models just that: A token in `PFB` (user in FB mode) sets the arc-multiplicities of these fluid arcs to a non-zero rate (namely `XF`, see rate reward `reflow`) so that `FTrewind` then refills `FPAvailVideo` at the same constant rate as it withdraws fluid from `FPforFB`, i.e., fast-backwarded video sequences become available again for later playout in PLAY and FF mode. Naturally, execution of the FB command is only possible, when the video is not at its very beginning (see guard `#FPforFB` > 0 for `FTrewind` in addition to a marked place `PView`).

Finally, when all video segments have been provided ($N-1$ tokens in place `SegRec`) and consumed (guard `#FPAvailVideo`=0), `tReset` restores the intial marking. We also note that although no upper bounds for the fluid places are specified, their fluid levels never exceed the full length of the video, i.e., 7200 seconds.

Measure definitions

The QoS of the presented system may be expressed in terms of measures related to the undesired discontinuous playout. The measure *blocking_time* yields the (stationary)

[5]Note that in case `#SegRec` $= N-1$, `tReset` (**prio**=2), which shares the same guard with transition `tFail`, has priority over `tFail` (**prio**=1 by default).

[6]Such a failure cannot arise when the user is in FB or PAUSE mode.

probability (= mean value E{`#PFail`} due to binary marking of place `PFail`) of being in the failure state, i.e., the respective proportion of video consumption time (which corresponds to the cycle time between two firings of `tReset`). Very often, another measure – defined as the mean proportional number of video segments the user attempts to access before their reception is entirely completed – is considered in VoD studies: In SPNL notation, this measure – called *blocking_prob* in our model – is simply given by a fraction of two mean firing frequencies E{`#tFail`} and E{`#t3`}.

Other performance measures of interest may easily be specified both in SPNL and in the tool SPNP. The model of Figure 2 with constant fluid change rates belongs to an FSPN subclass (as identified in [4]), for which stable and efficient simulation algorithms exist and have been implemented in the software tool SPNP. In the next section, we give simulation results for different versions of the FSPN model in Figure 2.

4. Numerical Results

All results of this section were computed by means of the software tool SPNP [3] with a confidence level of 95% and a maximum relative error margin of 5%[7]. We employed the simulation technique with batched means (of length 750,000). These strict requirements lead to rather long simulation runs of around two hours on a laptop (1GHz, 256MB) running Windows2000, but appealing estimates can already be obtained within seconds or minutes (e.g., by fixing the number of batches). If not stated otherwise, the parameters of the model are set as in Figure 2, e.g., the playout factor for FF is $X_F = 3$. The FSPN model as specified in SPNL (Figure 2) closely abides by the necessary SPNP input.

Play and Fast Forward only

The numerical results of this subsection are obtained from a reduced version of the VoD model in Figure 2 in order to be able to compare our results with the numerical values published in [1]. We first consider a user behavior, which alternates between PLAY and FF only. Therefore, the branches in the user profile that correspond to FB and PAUSE modes are erased together with fluid place `FPforFB` and fluid transition `FTrewind` (along with connected arcs).

In our experiments, a video of length 2 hours=7200 seconds is partitioned into N segments, with N ranging from 36 to 9. As outlined in [1], the performance of the VoD system depends on how the rate increase factor A relates to the average (virtual) consumption rate $\bar{b}$ of the video given by

$$\bar{b} = b \cdot \frac{\frac{1}{\lambda_{\mathrm{P}}} + \frac{X_F}{\lambda_{\mathrm{FF}}}}{\frac{1}{\lambda_{\mathrm{P}}} + \frac{1}{\lambda_{\mathrm{FF}}}} = b \cdot \frac{\lambda_{\mathrm{FF}} + \lambda_{\mathrm{P}} X_F}{\lambda_{\mathrm{FF}} + \lambda_{\mathrm{P}}} \quad .$$

The chosen values $\lambda_{\mathrm{P}} = \frac{1}{45}(\frac{1}{60})$ and $\lambda_{\mathrm{FF}} = \frac{1}{9}(\frac{1}{30})$ for the (uninterrupted) PLAY and FF phases, respectively, determine a normalized virtual consumption rate $\frac{\bar{b}}{b} = 1.33(1.67)$. Independent of the fact that greater values of A yield better performance, the ratio $\frac{\bar{b}}{b}$ marks a qualitative boundary: If $A > \frac{\bar{b}}{b}$, the strong law of large numbers will cause the blocking probability to approach zero for infinitely long videos (in case of a virtual consumption process as studied in [1]). If $A < \frac{\bar{b}}{b}$, the blocking probability will instead converge to 1.

Results for A above $\frac{\bar{b}}{b}$, namely $A = 1.4$ and below $\frac{\bar{b}}{b}$, namely $A = 1.3$, are arranged in Table 1 and compared with blocking probabilities given in [1]. Figure 3 shows corre-

[7] The negligible confidence intervals are omitted in tables and figures.

Table 1
Blocking measures of VoD model without FB and PAUSE ($\lambda_{\mathrm{P}} = \frac{1}{45}, \lambda_{\mathrm{FF}} = \frac{1}{9}$)

	$A = 1.4$				$A = 1.3$		
N	blocking_prob from [1]	blocking_prob	blocking_time	N	blocking_prob from [1]	blocking_prob	blocking_time
36	0.0163	0.0166	0.001784	36	0.1901	0.2002	0.026117
24	0.0102	0.0108	0.000901	24	0.1890	0.1983	0.021069
18	0.0059	0.0065	0.000451	18	0.1741	0.1796	0.018197
12	0.0015	0.0018	0.000104	12	0.1273	0.1256	0.011791
9	0.0003	0.0004	0.000022	9	0.0788	0.0863	0.007116

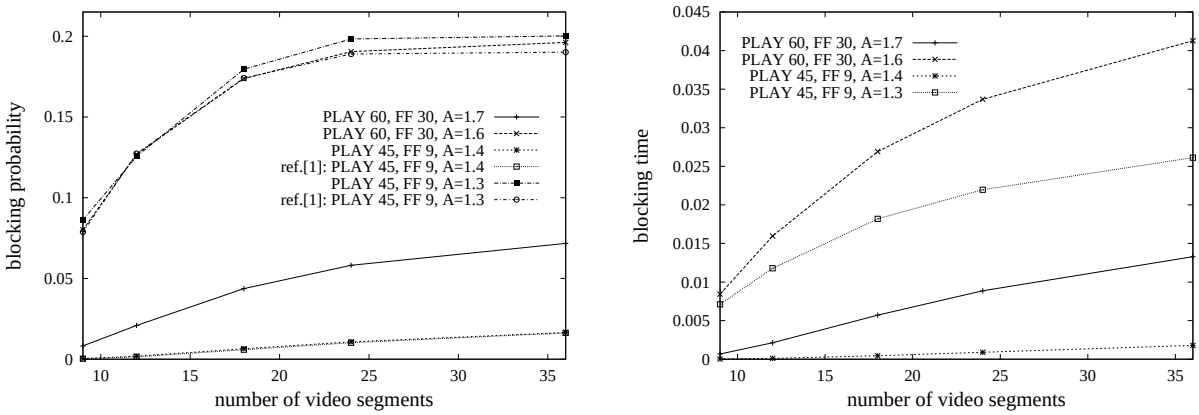


Figure 3. Blocking probabilities (left) and times (right) in different VoD scenarios

sponding plots including results for the other user profile ($\lambda_{\mathrm{P}} = \frac{1}{60}, \lambda_{\mathrm{FF}} = \frac{1}{30} : A = 1.7$ and $A = 1.6$). Not surprisingly, the numerical values for the blocking probabilities show acceptable agreement with the simulated values of the same scenarios in [1][8], where blocking probabilities derived from the FSPN model are mostly slightly higher. Furthermore, the deviations tend to become (relatively) more significant, as the segment number N decreases (making the video segments longer). Unlike in [1], we also present here numerical values for the blocking time in the two settings. The experiments – especially as documented in Figure 3 – demonstrate that reasonable performance can be achieved already for rate increase factors A only slightly greater than the normalized virtual consumption rate coupled with sufficiently large segment sizes/small numbers of segments.

Play, Fast Forward, Pause, and Fast Backward

To evaluate how the inclusion of PAUSE and FB periods in the user profile relaxes the strain on the bandwidth requirements, we also give the blocking measures for the model depicted in Figure 2. Table 2 contrasts the blocking measures of two corresponding settings (with and without FB/PAUSE) for $A = 1.4$. Obviously, with decreasing N (i.e., increasing video segments) the impact of VCR actions FB and PAUSE becomes much

[8]In [1], the measure is called failure probability.

Table 2
Blocking measures of VoD model with and without FB and PAUSE for $A = 1.4$ ($\lambda_{\mathrm{P}} = \frac{1}{45}, \lambda_{\mathrm{FF}} = \lambda_{\mathrm{FF}} = \lambda_{\mathrm{FF}} = \frac{1}{9}$)

	blocking_prob			blocking_time	
N	no FB/PAUSE	with FB/PAUSE	N	no FB/PAUSE	with FB/PAUSE
36	0.0165523	0.0012701	36	0.00178440	0.00010957
24	0.0107468	0.0004024	24	0.00090068	0.00002633
18	0.0065109	0.0000953	18	0.00045128	0.00000654
12	0.0018339	0.0000011	12	0.00010434	0.00000039
9	0.0004036	0.0000003	9	0.00002196	0.00000004

more noticeable: For the considered user profile, both blocking probabilities and blocking times are diminished by around three orders of magnitude for $N = 9$ as compared to one order of magnitude for $N = 36$. As a consequence, for such user profiles disproportionately lower values can be chosen for the rate increase factor A for smaller (fixed) N with respect to a specific QoS level. Thus, for less segmented videos relatively more bandwidth can be saved. Generally, these videos require less bandwidth already (see $R_t^{\min}$ in Section 2) at the expense of longer startup latencies.

5. Conclusions

The presented model demonstrates that the FSPN formalism is well suited to model the dynamics of VoD client-server processes in a concise form. The efficiency of the rate increase technique may be assessed and the parameter A may be optimized in a compromise between the blocking measures and the bandwidth requirements. Unlike in typical simulation studies, all behavioral details could be described succinctly. The model can be evaluated by the publicly available tool SPNP [3] using discrete-event simulation. Further research includes the study of other user behavior profiles and the potential of rate increase factors A_i, which depend on the segment number.

REFERENCES

1. E. W. Biersack, A. Jean-Marie, and P. Nain. Open-loop video distribution with support of VCR functionality. *Performance Evaluation*, 49(1-4):411–428, 2002.
2. Y. Birk and R. Mondri. Tailored transmissions for efficient near-video-on-demand service. In *Proc. of the IEEE Int. Conference on Multimedia Computing and Systems*, pages 226–231, 1999.
3. G. Ciardo, K. S. Trivedi, and J. K. Muppala. SPNP: Stochastic Petri Net Package. In *Proc. 3rd Int. Workshop on Petri Nets and Performance Models*, pages 142–151. Kyoto, Japan, 1989.
4. G. Ciardo, D. M. Nicol, and K. S. Trivedi. Discrete-event simulation of fluid stochastic Petri nets. *Trans. on Softw. Engin.*, 25(2):207–217, 1999.
5. Z. Fei, M. H. Ammar, I. Kamel, and S. Mukherjee. Providing interactive functions through active client buffer management in partitioned video broadcast. In *Proc. NGC 1999*, number 1736 in LNCS, pages 152–169. Springer Verlag, 1999.
6. R. German. SPNL: Processes as language oriented building blocks of stochastic Petri nets. In *Proc. 9th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, LNCS 1469, pages 123–134. Springer, 1997.
7. M. Gribaudo, M. Sereno, A. Horvath, and A. Bobbio. Fluid stochastic Petri nets augmented with flush-out arcs: Modelling and analysis, *Journal of Discrete Event Dynamic Systems*, 11:97–111, 2001.
8. G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi. Fluid stochastic Petri nets: Theory, applications, and solution techniques. *European J. Operational Research*, 105:184–201, 1998.
9. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.