Discrete-State and Fluid Stochastic Petri Net Models for Open-loop Video-on-Demand Systems: A Comparative Case Study

Armin Heindl
Institut für Technische Informatik und Mikroelektronik
Fakultät IV, TU Berlin, 10587 Berlin, Germany
heindl@cs.tu-berlin.de

Ernst W. Biersack
Institut Eurécom
BP 193, 06904 Sophia Antipolis, France
Ernst.Biersack@eurecom.fr

Abstract

We compare discrete-state stochastic Petri nets (SPNs) and fluid stochastic Petri nets (FSPNs) with a mixed discrete-continuous state space in their application to open-loop video broadcasting systems. For both classes, we present models of the Video-on-Demand client-server system, which generally capture all relevant system aspects in a concise and graphical way. Differences in model design as well as solution techniques for various variants of these models are discussed. By simulation, we obtain measures related to blocked video playout, which allow us to evaluate customer satisfaction against bandwidth requirements. While the FSPN model has advantages in model specification and evaluation efficiency, all presented models excel in their flexibility – both in defining various performance measures and in their extensibility to include, for instance, full VCR functionality and arbitrary user behavior.

1. Introduction

Scalable VoD (Video-on-Demand) services (see [12] for a complete description of service control aspects of VoD), where a large number of users receives the same video simultaneously, cannot be provided by closed-loop video distribution systems that serve each client individually by a separate stream. The server must adopt an open-loop approach: The video is partitioned into smaller blocks called segments, which are transmitted periodically and perpetually on separate channels. A client who wishes to view the video may link up with the channels at any time and simultaneously receive the different segments. Since the first parts of the video are needed sooner in the playback, the first segments are either shorter than later segments or transmitted at higher rates. With open-loop video distribution, 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, interactive VCR functions such as Fast Forward or Jump are not supported. Instead, the user is restricted to view the video from the beginning until the end (e.g., see [13] for a study assuming constant video playout duration). If admitted, Fast Forward, for instance, may cause video reception to fall behind video consumption leading to undesired discontinuous playout. By adjusting the rates at which the segments are transmitted with a uniform *rate increase factor,* VCR functions may be supported probabilistically (i.e., with a low probability for discontinuous playout, [1]). This scheme, in contrast to related work like staggered broadcast [7], tolerates arbitrary VCR functionality and thus provides full VoD services.

Discrete-state and fluid SPN models are presented to study the transmission scheme of [1] for probabilistic support of VCR functionality. As common merits of both model classes, they conveniently allow to consider complex video viewing behaviors and allow to investigate a wide range of performance measures in the trade-off between the additional bandwidth requirements and client satisfaction (expressed in measures related to (dis)continuous playout). At the same time, the stochastic Petri net models reveal all the details of the VoD system, which are usually hidden in source code and typically not outlined in simulation studies. Using the publicly available tools TimeNET [18] and SPNP [4] for their respective evaluation helps to make the results reproducible.

This paper, however, focuses on the differences of the discrete-state and hybrid model with respect to modeling and evaluation issues in the context of the given scenario. As this application might more likely be interpreted as a system with discrete and continuous components that evolve over time, this paper scrutinizes how well such a behavior can be captured by a purely discrete model.

Discrete-state SPNs and FSPNs have been shown to be practicable in performance and dependability modeling especially for systems with concurrency and synchronization. Various analytical algorithms and simulative procedures are available for the (automated) evaluation of different variants of (F)SPNs (e.g., see [3, 8] for SPNs and [11, 17, 9] for FSPNs). In extending discrete-state SPNs, the hybrid model in this contribution draws on the definition in [5]. For realistic modeling, we require non-exponential timing and, in case of FSPNs, basic fluid elements (e.g., constant fluid change rates will be sufficient).

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. Sections 3 and 4 introduce the developed discrete-state and fluid SPNs, respectively. Numerical results obtained by simulation based on these models are compared in Section 5. Note that the FSPN model has already been validated with results from another publication in [10]. Concluding remarks are given in Section 6.

2. Tailored transmission schemes for open-loop Video-on-Demand

Birk and Mondri [2] 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; for sake of simplicity, we assume throughout the paper that the video is constant bit rate.). 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 [15] already meet the latter two 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

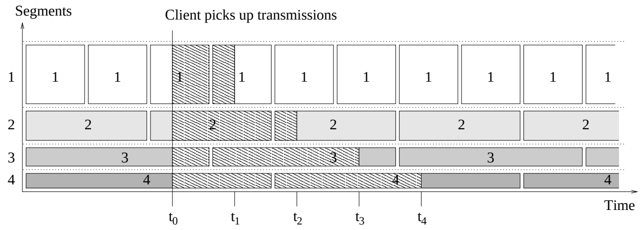


Figure 1. The base scheme of tailored transmission with minimal transmission rates. The hatched areas indicate the data of each segment necessary to completely receive a segment.

$$r_i^{\min} = \frac{b}{i} \qquad \text{for } i = 2, \ldots, N \,, \tag{1}$$

when r_1 is set to b rather arbitrarily. Naturally, smaller transmission rates for the first segment do not harm the continuous playout, but in the above setting it does not seem reasonable that r_1 falls below $r_2^{\min} = \frac{b}{2}$. Then, the reception of segment 1 would finish after that of segment 2.

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 $R_t^{\min} = b \sum_{i=1}^{N} \frac{1}{i} (\approx b \cdot \ln(N)$ for large $N)$.

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 models in Sections 3 and 4.

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 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

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

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 two sections, we provide the discrete-state and fluid stochastic Petri net model, which both allow 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. A discrete-state SPN model for probabilistic VCR support

First, we develop an SPNL 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. In SPNL, common SPN elements, like *places* (represented as 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 SPNs. Although firing times may be chosen to be quite general in SPNL, we will employ only exponential (exponential transitions) and deterministic distributions (deterministic transitions) in this study. Transitions, which are never preempted, are referred to as *persistent*. For non-exponential non-persistent transitions, a firing policy has to be defined. In this paper, we generally adopt the policy *preemptive repeat different* (prd) [14], which means that, when such a transition is disabled without having fired, its already performed work is lost. We also use (marking-dependent) *arc multiplicities*. 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 place `P` and – as we will see – may also be used for the non-integer 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), are listed below the graphical representation of the net. Among those transitions, whose input places are covered by at least as many tokens as the respective current arc multiplicities and whose firing is not prevented a priori by an activated inhibitor arc, 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.

Ignoring the possible hierarchies in SPNL (similar to the module concept of programming languages like Ada), we arrange the complete SPN model in a single SPNL module `VoD_as_SPN` encompassing the *process* transmit_consume (see Figure 2). Following the list of parameters used below to define the initial marking, arc multiplicities, rates and delays, two (stationary) performance measures are declared in the public part of the process (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 place `P` and E{`#T`} the mean number of firings per unit time (throughput) of transition `T`.

3.1. Description of the model dynamcis

The tailored transmission scheme, the video consumption and the user behavior profile constitute the VoD client-server process. The model of Figure 2 implements the dynamics of the VoD client-server process 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` re-

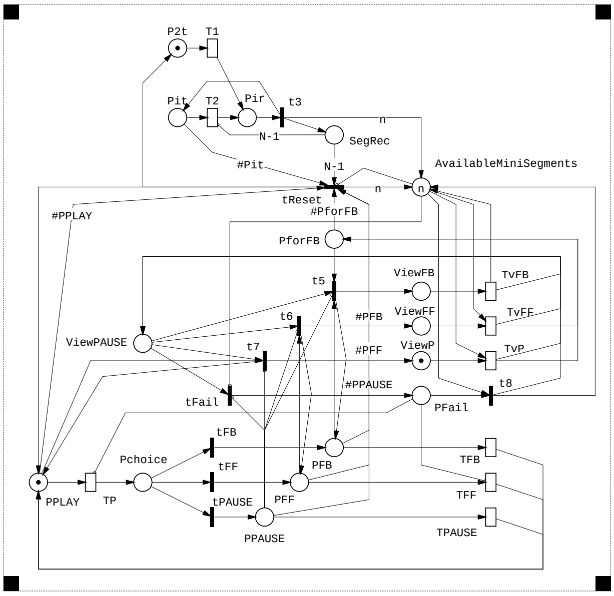
```
module VoD_as_SPN;
 parameter N=36;   (* number of equal-length segments, in which video is transmitted
                      on different channels *)
           n=1000; (* number of minisegments within a single segment describing the
                      units, in which the video is consumed  -> a pure model paramter *)

           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;
 private
```




```
  smeasure   blocking_time =  E{#PFail};          blocking_prob =  E{#tFail}/E{#t3};
  transatt   T1:    dist=det(sd*(2-A)/A);         T2:      dist=det(sd/A);
             TvP:   dist=det(sd/n);
             TvFF:  dist=det(sd/(XF*n));          TvFB:    dist=det(sd/(XF*n));
             TP:    dist=exp(lam_P);              TPAUSE:  dist=exp(lam_PAUSE);
             TFF:   dist=exp(lam_FF);             TFB:     dist=exp(lam_FB);

             tReset:    prio=3;
             t5,t6,t7:  prio=1;
             t8:        prio=4;
             tFail:     prio=2;

             tFB,tPAUSE:  weight=0.1;
             tFF:         weight=0.8;
end transmit_consume;
end VoD_as_SPN.
```

Figure 2. The SPNL model of NVoD

sets 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 place `AvailableMiniSegments`. The token circulating through the bottom places `PPLAY`, `Pchoice`, `PFB`, `PFF`, `PPAUSE` emulates the user behavior as suggested in the last paragraph of the previous section. 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 FB mode are 0.1 each, and the probability to enter FF mode is 0.8 correspondingly (see weights for immediate transitions `tPAUSE`, `tFB` 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. The marking-dependent arcs to transition `tReset` empty all places in the subnet when `tReset` fires, which reputs a single token into place `PPLAY`.

For our performance evaluation, we may abstract from the fact that the N video segments are transmitted on different channels. Instead, it suffices to know when the 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. (Actually, it is assumed that $A < 2$ so that segment 2 is fully received only after segment 1. This guarantees a positive delay of transition `T1` as $\frac{D}{r_2^f} - \frac{D}{r_1} = D(\frac{2}{A \cdot b} - \frac{1}{b}) = \frac{\texttt{sd}}{A}(2 - A) > 0$.) Since the cycle starts right after the reception of the first segment, the inhibitor arc from place `SegRec` to `T2` with multiplicity $N - 1$ guarantees that only $N - 1$ more segments are delivered within a cycle period. Until then, transition `t3` keeps re-enabling transition `T2` and at the same time puts n tokens in place `AvailableMiniSegments`. The n minisegments represent a model artifact in form of a discretized workload: It is assumed that each segment of size D is consumed in units of size $\frac{D}{n}$. The consumption of these units is modeled by the middle subnet governed prevalently by the user behavior subnet. As long as the user remains in PLAY mode (token in `PPLAY`), the token in place `ViewP` will loop along `TvP`, `ViewPAUSE`, `t7`, `ViewP`,... removing one token from place `AvailableMiniSegments` each time it passes transition `TvP`. With the user entering a different VCR mode (i.e., token in place `PPAUSE`, `PFF` or `PFB`), the token in the middle part will either be stopped in place `ViewPAUSE` due to the inhibitor arcs from place `PPAUSE` to the four immediate transitions `t5`, `t6`, `t7`, `tFail` or move along the alternative branches for FB and FF (note arcs between `PFB`/`PFF` and `t5`/`t6`, respectively). In the former case, no further minisegments are consumed in PAUSE mode. In the latter case, the deterministic transitions `TvFB` and `TvFF` with delays divided by the playout factor X_F (compared to transition `TvP`, see keyword **det** in **transatt** section) accelerate the consumption of a minisegment accordingly. Whereas `TvFF` subtracts a token from `AvailableMiniSegments`, `TvFB` adds one to this place, i.e., this *fast-backwarded* minisegment has become available again for later playout. A nonempty place `PforFB`, into which transitions `TvFF` and `TvP` deposit tokens, indicates to transition `t5` that it is still possible to rewind the video for another minisegment.

In probabilistic support of VCR functionality, it may occur that the consumption of the video gets ahead of its reception. In our model, this corresponds to the situation that the consumption token enters place `ViewPAUSE` with no tokens in `AvailableMiniSegments`. Due to the higher priority compared with `t5`, `t6`, `t7` (which is usually counteracted by the inihibitor arc from `AvailableMiniSegments`), transition `tFail` passes this token to place `PFail`, where it resides until the next video segment is received, i.e., `t3` refills `AvailableMiniSegments`. The time a token spends in place `PFail` can thus be interpreted as 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`; such a failure cannot arise when the user is in FB or PAUSE mode).

3.2. Performance measures and solution techniques

The measure *blocking_time* yields the (stationary) 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 in VoD studies, another measure is considered, which is defined as the mean (proportional) number of video segments the user attempts to access before their reception is entirely completed: In our model, this measure – referred to as *blocking_prob* – is simply given by the ratio of two mean firing frequencies E{#tFail} and E{#t3}. Basically, the quotient relates all out-of-time video segments to all transmitted segments.

The flexibility of the SPNL formalism makes it easy to specify other performance measures of interest such as the blocking time on the condition that a failure situation occurred in PLAY mode, $\mathrm{E}\{\#\mathtt{PFail}|\#\mathtt{PPLAY} = 1\}$ or the true video consumption rate $b\,\mathrm{E}\{\#\mathtt{TvP}\} + X_F b\,\mathrm{E}\{\#\mathtt{TvFF}\} - X_F b\,\mathrm{E}\{\#\mathtt{TvFB}\}$. Slight modifications of the model in Figure 2 allow to obtain the failure probability: Eliminating immediate transition `t8` transforms place `PFail` into a trap; e.g., an additional inhibitor arc from `PFail` to `t3` renders some global states absorbing. The desired failure probability (i.e., the probability that the video is not consumed without unwanted interruption) can be obtained, say, from a transient simulation of the measure *blocking_prob* in the altered model.

As long as the SPNL model contains concurrently enabled non-exponential activities (e.g., deterministic transitions `TvP` and `T2`), there currently is little hope for an automated numerical analysis. The common approaches for Markov regenerative stochastic Petri nets [3] usually require that in each marking at most one general transition is enabled. We can satisfy this condition in our model by replacing deterministic transitions `TvFB`, `TvFF` and `TvP` with exponential ones with the same mean delays. This approximation appears justified, if n is large: Then, the consecutive runs of the consumption token through either of the three transitions mimics a near-deterministic behavior (just as an Erlang-m distribution may serve as an approximation to a constant interval for large m). Generally, the mentioned replacement tends to overestimate the true blocking time and probability. On the other hand, more general user behavior patterns, e.g., with a deterministic interval for the PAUSE period (deterministic transition `TPAUSE`), may quickly destroy the modeling requirements imposed by analytical solution techniques, which leaves simulation as the only available method. In case of generally distributed firing times, the inhibitor arcs to transitions `TP` and `TFF` suggest to attribute the *preemptive resume* (prs) policy (instead of prd) to these transitions. Thus, already performed work is preserved during preemption.

3.3. Discussion of model parameter n

We now comment on how to choose the arbitrary model parameter n, the number of minisegments of which a single video segment is composed. Obviously, the larger n, the more precise results may be expected – however at the expense of increased state spaces and prolonged processing times. In contrast, too small values of n pronounce a modeling approximation intrinsic in the presented net thus leading to noticeable errors. For example, when a user leaves the PLAY mode (transition `TP` fires), the currently viewed minisegment will still be displayed until its end (until `TvP` fires) – and analogously for FB and FF. By comparing the measures E{#PFB} and E{#ViewFB} and/or E{#PPAUSE} and E{#ViewPAUSE}, one may check whether n has been selected sufficiently large for an appropriate model of the video consumption process. This drawback of a discretized workload (#`AvailableMiniSegments`) is inherent to a discrete-state model and can only be remedied by incorporating a fluid place in the stochastic Petri net.

4. A fluid SPN model for probabilistic VCR support

From the discussion in the previous section, the drawbacks of a discrete-state model have become apparent: the artificially introduced minisegments – though they might even be closer to a fine-grained reality – complicate the model design and blow up the state space. The increased number of discrete events to be processed slows down a Monte-Carlo simulation. On the contrary, a hybrid system with a continuous part for the video consumption process

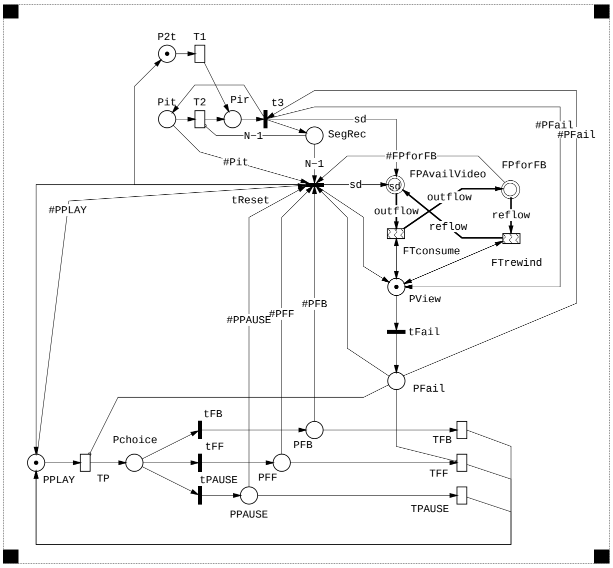
```
module VoD_as_FSPN;
 parameter N=36;   (* number of equal-length segments, in which video is transmitted
                      on different channels *)

           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{#tReset})/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);

            FPAvailVideo, FPforFB :  upperbound =L;
end transmit_consume;
end VoD_as_FSPN.
```

Figure 3. The FSPN model of VoD (in pseudo SPNL notation)

would reflect the VoD client-server system in a more intuitive way – with potential benefits to the intelligibility of the model and durations of simulation runs.

In the realm of stochastic Petri nets, such a hybrid system may be modeled as hybrid Petri net [6] or a Fluid Stochastic Petri Net (FSPN, [16]). In fact, in this paper, we refer to the FSPN definition given in [5], or more precisely to a less general subclass with constant fluid (though marking-dependent) change rates and concurrent general transitions, which are persistent.

For consistency, we also present the FSPN model in SPNL notation. However, we emphasize that the software package SPNP [4] is used for simulating the FSPN (with results given in Section 5). Since SPNL currently does not support fluid components of stochastic Petri nets, straightforward and common extensions for their specification are added to the SPNL model in Figure 3. In coexistence with the elements introduced in the previous section, an FSPN additionally provides

- fluid places (depicted as two concentric circles, e.g., `FPAvailVideo` and `FPforFB` in Figure 3), which may contain non-integer tokens or a fluid quantity, like the initial parameter `sd` in fluid place `FPAvailVideo`,
- fluid transitions[1] (depicted as rectangles with a 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, marking-dependencies may be quite generally defined with mutual impact between the discrete and continuous parts of the stochastic Petri net. In particular, we will use guards, i.e., rate rewards that can be interpreted as Boolean expressions, to restrict the enabling of a transition.

4.1. Description of the FSPN model

Let us now describe the FSPN model in Figure 3 in more detail. Its SPNL notation – when compared with Figure 2 – at first sight reveals a reduced modeling complexity for the video consumption process (middle part), while the subnets for the segment transmission (top part) and the user profile (bottom part) remain unchanged. Elements of the latter two subnets are found in the same location as in Figure 2. The middle subnet now contains two fluid places (`FP`...), two fluid transitions (`FT`...), four fluid arcs and three fluid-impulse arcs besides discrete places `PView` and `Pfail` and immediate transition `tFail` (and pertinent arcs). Thus, the places `ViewFF`, `ViewP`, `ViewFB` and `ViewPAUSE` of Figure 2 have collapsed into place `PView` with much of the behavioral complexity now encoded in guards (see keyword **guard** for `FTrewind` and `tFail`) and marking-dependent flows (see keyword **rreward** for declaration/definition of

[1]This term is not discerned from timed transition in [5], but helps our intuition for the considered example and corresponds to the inf-transition in SPNP.

rate rewards outflow and reflow). The fluid part allows us to dispense with the minisegments of the discrete-state SPN, where fluid places FPAvailVideo and FPforFB take over the roles of places AvailableMiniSegments and PforFB, respectively.

Instead of n minisegments, immediate transition t3 now deposits the segment duration sd (in PLAY mode) into FPAvailVideo when the segment has been entirely received. Again, the first segment is available initially. Together with the above-mentioned guards and marking-dependent arc multiplicities for the fluid arcs, the video consumption is governed by the token in place PView: it remains there, as long as video sequences are available; only when video consumption outruns its reception (i.e., fluid place FPAvailVideo becomes empty), this token moves to PFail via tFail (see guard #FPAvailVideo=0). As before, a token in PFail indicates that the video consumption is blocked unvoluntarily (see measure *blocking_time*). 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.

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). In case the video consumption is paused (token in PPAUSE), the rate rewards outflow and reflow account for zero rates along all fluid arcs, even when place PView is nonempty.

Finally, when all video segments have been provided ($N - 1$ tokens in place SegRec) and consumed (tFail fires one more time after last segment), tReset restores the intial marking. This last firing of tFail within a regeneration period, which actually does not correspond to a blocking situation, is due to peculiarities in SPNP and requires a slightly different definition of the measure *blocking_prob*. For the sake of reproducibility of the simulation results, the SPNL model closely abides by the employed SPNP specifications. However, minor modifications are required for proper input to SPNP and are outlined in the Appendix.

4.2. Performance measures and solution techniques

We use the FSPN model to obtain the same performance measures as for the discrete-state SPN. In the previous subsection, we already addressed the specification of the familiar blocking measures (slightly modified for *blocking_prob*). The failure probability is obtained by eliminating the arcs between place PFail and immediate transition t3 together with a transient simulation.

The model of Figure 3 with constant fluid change rates belongs to an FSPN subclass (as identified in [5]), for which stable and efficient simulation algorithms exist and have been implemented in the software tool SPNP. In fact, SPNP provides four techniques for simulating FSPNs (batched means, independent replications, restart, splitting) – with obvious benefits compared to the discrete-state model, since fewer discrete events have to be processed. For FSPN models containing more than one fluid place or timed transitions with generally distributed firing times (e.g., deterministic transitions T1 and T2), no automated numerical analysis is available. The known analytical methods for so-called first-order FSPNs usually require only a single fluid place and solely exponential and immediate transitions [11, 17, 9]. For a special case, we can satisfy these conditions in our model:

- When we ignore the VCR function FB, fluid place FPforFB and fluid transition FTrewind can be eliminated together with connected arcs (and along with the corresponding branch in the user profile subnet). Such Near-VoD systems are considered in [1].
- Assuming fluctuating transmission rates, deterministic transitions T1 and T2 might be replaced by exponential ones with the same mean delays. Of course, this assumption would have to be validated carefully.

Considering the original models, the FSPN does not as easily allow an (approximate) analytical approach as the discrete-state SPN. We also point out that the structure of the model in Figure 2 is only seemingly more complex. By exploiting guards and marking-dependent (deterministic) firing times in a single transition for minisegment consumption (i.e., collapsing ViewPause, ViewP and ViewFF into a single place), the discrete-state model will resemble

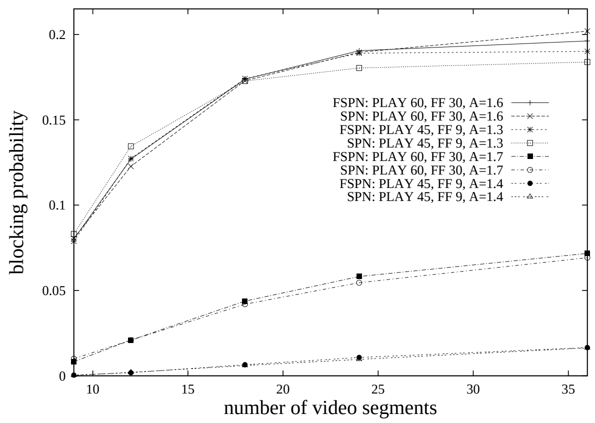


Figure 4. Blocking probabilities in different VoD scenarios

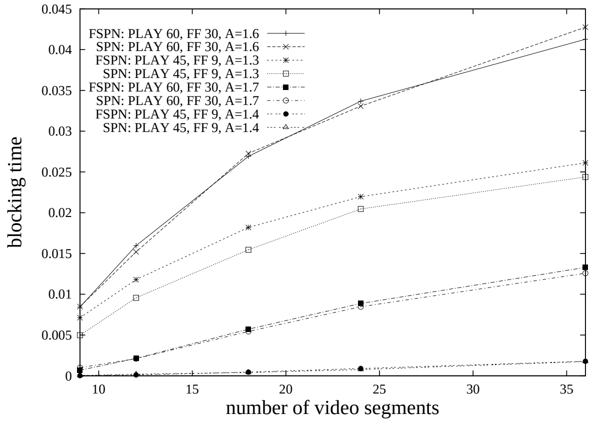


Figure 5. Blocking times in VoD scenarios

more closely the FSPN model. Since losing the already consumed video bits so that they would need to be re-viewed is unrealistic, this would implicate a prs firing policy for this transition. However, the model in Figure 2 assumes that the initial PLAY, FF or FB decision for a minisegment is not revoked during its consumption (no preemption). The smaller the minisegment, the more reasonable is this assumption.

5. Numerical results

The results of this section were computed by means of the SPNL simulation component of TimeNET [18] for the discrete-state model in Section 3 and with the software tool SPNP [4] for the FSPN in Section 4. In either case, a confidence level of 95% and a maximum relative error margin of 5% were chosen. (The negligible confidence intervals are omitted in tables and figures.) To match the simulation technique in TimeNET, we employed batched means (of length 750,000) in SPNP. Simulations with TimeNET were conducted on a Unix workstation, those with SPNP on a laptop (1GHz, 256MB, Windows2000). For an identical experiment performed on the laptop with TimeNET, the simulation would last around eight hours, while SPNP provided the solution in two hours. With SPNP, sufficiently accurate estimates can already be obtained in seconds or minutes by fixing the number of batches (input option of SPNP).

If not stated otherwise, the parameters of the model are set as in Figures 2 and 3, e.g., the playout factor for FF is $X_F = 3$.

5.1. Play and Fast Forward only

The numerical results of this subsection are obtained from slightly reduced versions of the VoD models in Figures 2 and 3. We first consider a user behavior, which alternates between PLAY and FF only (a scenario also considered in [1]). Therefore, the branches in the user profile that correspond to FB and PAUSE modes, are erased. Additionally, in the discrete-state model the corresponding FB branch ceases to apply (i.e., transitions `t5`, `TvFB` and places `ViewFB`, `PforFB` are deleted), while, equivalently in the FSPN model, we cancel the fluid place `FPforFB` and fluid transition `FTrewind` (along with connected arcs in all cases).

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

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

Note that this formula ignores periods when the consumption process is blocked ($b = 0$) due to unavailable segments. The chosen values $\lambda_{\mathrm{P}} = \frac{1}{60}(\frac{1}{45})$ and $\lambda_{\mathrm{FF}} = \frac{1}{30}(\frac{1}{9})$ for the (uninterrupted) PLAY and FF phases, respectively, determine a normalized virtual consumption rate $\frac{\bar{b}}{b} = 1.67(1.33)$. 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 $A = 1.7$ and below $\frac{\bar{b}}{b}$, namely $A = 1.6$ for both the discrete-state and the fluid SPN are arranged in Table 1 and plotted in Figures 4 and 5 including results for the other user profile ($\lambda_{\mathrm{P}} = \frac{1}{45}$, $\lambda_{\mathrm{FF}} =$

Table 1. Blocking measures of VoD model given in % ($n = 2000, \lambda_{\mathrm{P}} = \frac{1}{60}, \lambda_{\mathrm{FF}} = \frac{1}{30}$)

	blocking_prob		blocking_time	
N	SPN	FSPN	SPN	FSPN
$A = 1.7$				
36	6.919	7.170	1.258	1.329
24	5.456	5.817	0.847	0.887
18	4.192	4.373	0.544	0.571
12	2.088	2.091	0.210	0.213
9	0.780	0.824	0.064	0.068
$A = 1.6$				
36	20.194	19.617	4.276	4.127
24	18.954	19.049	3.307	3.368
18	17.287	17.378	2.727	2.691
12	12.271	12.693	1.519	1.596
9	8.051	8.013	0.849	0.843

$\frac{1}{9}$: $A = 1.4$ and $A = 1.3$). With an average relative error of around 3% (where deviations do not significantly differ between the two groups of qualitatively different measures, i.e., 2.7% for *blocking_prob* and 3.4% for *blocking_time*), the deviations are surprisingly low for the considered scenario. In fact, they are smaller than the relative errors observed when validating the FSPN results with simulated data from another publication (see [10] for the comparison). The discrete-state SPN tends to underestimate the blocking measures, as it is most pronounced for the setting $\lambda_{\mathrm{P}} = \frac{1}{45}, \lambda_{\mathrm{FF}} = \frac{1}{9}, A = 1.3$ (see Figures 4 and 5). This may be attributed to the less favorable (i.e., greater) ratio of the minisegment durations to PLAY/FF mode periods.

Independently of the differences between the hybrid and the discrete-state model, it is interesting to note that scenarios of the VoD system with similar (segment) blocking probabilities can have quite different congestion blocking measures (*blocking_time*). The experiments – especially as documented in Figures 4 and 5 – 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.

5.2. Choice of model parameter n

In the discrete-state SPN, the ratio of minisegment durations to the PLAY/FF mode periods strongly depends on the model parameter n, whose impact is investigated here. By extensive experiments, we found that the number of minisegments n should be selected rather large dependent on the specific user profile to assure a satisfactory accuracy, i.e., 2000 (1000) for the first (second) user behavior. Numerical results proved quite sensitive to decreasing n: For

Table 2. Dependence of blocking measures (in %) from the discrete-state SPN on parameter n (for $A = 1.3$ and $N = 18$)

n	bl_prob	bl_time	c-mean	E{#ViewP}
2000	18.24243	1.70231	0.8283	0.8254
1000	17.26955	1.54561	0.8215	0.8200
750	16.38760	1.46549	0.8289	0.8060
500	15.41627	1.34995	0.8290	0.7860
250	13.52643	1.15342	0.8248	0.7176

example, Table 2 (where c-mean= E{#PPLAY|#PFail = 0}) highlights the corresponding decay in the blocking measures for $A = 1.3$ and $N = 18$ and corroborates the propensity of the discrete-state SPN to underestimate these measures. The auxiliary measures listed in the last two columns should be identical in the ideal case. Their difference may serve to check the accuracy of the simulation model. With respect to the confidence parameters (95%, 5%), $n = 1000$ appears to be sufficiently large for the considered setting. Unfortunately, simulation runs for $n = 1000$ and blocking times below 0.01% may easily last overnight. At the same time, the huge state space for n close to 1000 prevents the analytical approach with substituted deterministic distributions (see Section 3.2) from being an efficient alternative.

5.3. Play, Fast Forward, Pause, and Fast Backward

Apart from a more intuitive model design, the FSPN model also proves superior over the discrete-state SPN in terms of simulation efficiency. When including PAUSE and FB periods in the user profile (see Figure 2), we could not obtain simulation results for the SPN in a reasonable time. Therefore, we present FSPN data only in this subsection. PAUSE and FB periods relax the strain on the bandwidth requirements. Table 3 contrasts the blocking measures of two corresponding settings (with and without FB/PAUSE) for $A = 1.4$. Obviously, with decreasing N (i.e., increas-

Table 3. Blocking measures (in %) of FSPN 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/P.	with FB/P.	no FB/P.	with FB/P.
36	1.65523	0.12701	0.178440	0.010957
24	1.07468	0.04024	0.090068	0.002633
18	0.65109	0.00953	0.045128	0.000654
12	0.18339	0.00011	0.010434	0.000039
9	0.04036	0.00003	0.002196	0.000004

ing video segments) the impact of VCR actions FB and PAUSE becomes much 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.

6. Conclusions

This paper demonstrates that the SPN formalism – both in discrete-state and hybrid domain – is well suited to model the dynamics of VoD systems in a concise form. Unlike in traditional simulation studies, all behavioral details could be described succinctly. In this particular case study, the considered system is more accurately described by an FSPN, which also has a lower execution time than its discrete-state counterpart. Minor drawbacks of the FSPN model are the increased complexity due to additional fluid components and the fact that fewer tools are currently available for their evaluation. We also saw that the results obtained from the discrete-state SPN show surprisingly good agreement with its fluid analogue.

Generally, the presented models allow to assess the rate increase technique and to optimize parameter A by trading off blocking probability and bandwidth requirements. Further research – preferably conducted with the FSPN model – 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] H. Choi, V. G. Kulkarni, and K. S. Trivedi. Markov regenerative stochastic Petri nets. *Performance Evaluation*, 20:337–357, 1994.

[4] G. Ciardo, J. Muppala, and K. S. Trivedi. SPNP: Stochastic Petri net package. In *Proc. 3rd Int. Workshop on Petri Nets and Performance Models*, pages 142–151, Kyoto, Japan, 1989.

[5] 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.

[6] R. David. Modeling of hybrid systems using continuous and hybrid petri nets. In *Proc. 7th Int. Workshop on Petri Nets and Performance Models*, pages 47–58, St. Malo, France, 1997.

[7] 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.

[8] R. German. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. John Wiley & Sons, 2000.

[9] M. Gribaudo, M. Sereno, A. Hórvath, 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.

[10] A. Heindl and E. Biersack. Hybrid performance modeling of open-loop video distribution with VCR functionality. In *Proc. 18th Int. Teletraffic Congress*, Berlin, Germany, 2003.

[11] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi. Fluid stochastic Petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research*, 105:184–201, 1998.

[12] E. T. S. Institute. *Video on Demand (VoD) networks aspects*. ETR 262, DTR/NA-052109, 1996.

[13] M. Naldi. A mixture model for the connection holding times in Video-on-Demand service. *Performance Evaluation*, 47(1):23–42, 2002.

[14] M. Telek, A. Bobbio, and A. Puliafito. Steady state solution of MRSPNs with mixed preemption policies. In *Proc. IEEE Int. Performance and Dependability Symp.*, pages 106–115, Urbana-Champaign, Illinois, USA, 1996.

[15] TiVo. What is tivo: Technical aspects, 2001.

[16] K. S. Trivedi. FSPNs: Fluid stochastic Petri nets. In *Proc. 14th Int. Conf. on Application and Theory of Petri Nets*, pages 44–51, Durham, NC, USA, 1998.

[17] K. Wolter and A. Zisowsky. On Markov reward modelling with FSPNs. *Performance Evaluation*, 44:165–186, 2001.

[18] A. Zimmermann, J. Freiheit, R. German, and G. Hommel. Petri net modelling and performability evaluation with TimeNET 3.0. In *Proc. 11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 188–202, Chicago, USA, 2000.

Appendix

The Appendix compiles minor modifications to the FSPN model stipulated by the tool SPNP with the objective to enable the reader to reproduce the presented results:

- The guard of fluid transition `FTrewind` is incorporated into the parameter `reflow` for the marking-dependent arc-multiplicities of the corresponding fluid arcs.
- Place `Pir` and transition `t3` are eliminated with obvious rearrangement of arc connections for invariant model behavior.
- The priority of immediate transition `tFail` is set to zero.
- The priorities of immediate transitions `tFB`, `tFF`, and `tPAUSE` are set to 0.5 (the default value in the SPNP-GUI for Windows2000).