HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **36**; December 1995
ISSN 0783–5396
ISBN 951-22-2917-X

A HIGH-LEVEL NET MODEL OF A VIDEO ON DEMAND SYSTEM

Tino Pyssysalo and Leo Ojala

Digital Systems Laboratory
Department of Computer Science
Helsinki University of Technology
Otaniemi, Finland

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports — ISSN 0783–5396
No. **36**; December 1995 — ISBN 951-22-2917-X

A High-Level Net Model of a Video on Demand System

Tino Pyssysalo and Leo Ojala

Abstract:

In this report we give a high-level net model for a specific "video on demand" system. This system has some very interesting modeling features: FIFO queues, the *bunching property*, the parallel machine scheduling and a complicated resource allocation mechanism. The bunching property stands for feasibility of several requests being served together by one resource. The resource allocation mechanism for this video on demand system has a special *refusal by choice* feature which means that the resource is not necessarily allocated even though it is available. The predicate/transition net formalism has been used in the modelling and reachability analysis in studying the properties of the model. The causal model developed here allows a natural extension for the use of time or stochastic net theoretical methods for performance analysis.

Keywords. video on demand, high-level net models, FIFO with the bunching property, parallel scheduling algorithms, reachability analysis.

Printing: TKK Monistamo; Otaniemi 1996

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

Phone: $\frac{90}{+358\text{–}0}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–0–465 077
E-mail: lab@saturn.hut.fi

Contents

1 Introduction

Video on demand (VOD) is a system for distributing movies to the homes of users. It consists of a video server and a number of user sites connected to the server via a high bandwidth communication media. Several servers can be connected together with a high capacity wide area networks.

When a user wants to watch a movie, he sends a movie request to the server, which accepts or rejects it according to its movie scheduling policy. Immediately after the request (*immediate notification*) or some time later (*delayed notification*) the server informs the user, whether his request has been accepted or rejected, by sending a *notification* [1, 2].

The efficiency of VOD depends on how the scheduling algorithm can exploit the bottleneck resources, i.e., the number of broadcast channels available on the transmission cable. There are several alternative algorithms to control the usage of the scarce resources in an efficient way. *On-line* algorithms have to make decisions, whether to accept or reject a movie request without any knowledge about the future requests, while *off-line* algorithms have a full knowledge about all the requests. It is clear that off-line algorithms can collect more information about the requests and in that way do better decisions in terms of efficiency.

In this report we shall model a VOD scheduling algorithm that has been studied and introduced by Aggarwal et.al. [1, 2]. The algorithm is an *adaptive on-line harmonic prepartition algorithm* with delayed notification and *refusal* *by choice*. We shall explain the name in the sequel. In [1, 2] the on-line algorithms are evaluated in terms of their competitiveness. An algorithm is called competitive, if it performs, up to a constant factor, as well as its off-line, clairvoyant counterparts for the same problem. In [2] it is shown that the class of harmonic prepartition algorithms exhibits an asymptotically optimal behavior (the behavior of off-line algorithms), when the number of channels and the notification time grow indefinitely. This is one reason why this class of algorithms is of our interest.

Our introductory work concerning the causal modeling of VOD algorithms is presented in [8]. In this report we shall use the Pr/T-net formalism to model the VOD system more precisely. In subsequent papers we shall study the performance of the system using time or stochastic net formalisms.

The structure of the basic video on demand system is given in Fig. 1 [1]. VOD systems can be partitioned into a number of categories: near, adaptive and full VOD. In the near video on demand a fixed set of movies is played regularly, at fixed time intervals. Many users watch the same movie, which they cannot for example rewind or stop. In order to maximize the number of accepted user requests, the set of movies should contain only the most popular movies like "The Golden Eye". In the adaptive VOD users make requests for a movie they want to watch. The movie requests are accepted, if there are enough requests for a movie, or rejected, if there are too few requests. As in the near VOD many users watch the same movie, the run of which they cannot affect. In the full VOD user's request is served immediately, whenever it arrives. Requests are accepted, if there is any channel capacity available and rejected otherwise. In the full VOD every user watches his own movie on his own channel.

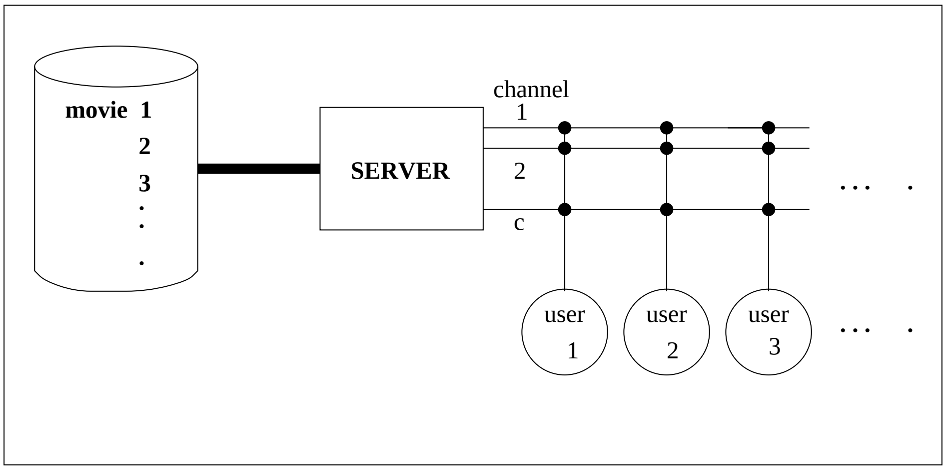

Figure 1: The basic video on demand system.

In this report we shall study an algorithm belonging to the adaptive VOD category. In the adaptive VOD approach the algorithm can decide to reject user requests, although there are available channels, if there are not enough requests for the movie (*refusal by choice*). The scheduling algorithm determines the responses (accept or reject notifications of user requests) at any moment, based on the requests up to that moment. The so called harmonic prepartition algorithm $\mathcal{H}$ divides the d movie channels into μ sets $(S_1, S_2, ..., S_i, ..., S_\mu)$. The set S_i contains $\frac{d}{i\mathcal{H}_\mu}$ (rounded to the nearest integer) channels, where $\mathcal{H}_\mu = \sum_{r=1}^{\mu} \frac{1}{r}$. So any transmission made on the channel in the set S_i must serve at least i requests.

Every movie name m_j has a distinct FIFO queue Q_j for the requests of this movie. When a user issues a request, it is immediately inserted into the corresponding Q_j, which is initially empty. Always the earliest request is chosen for the decision to be accepted or rejected and the user will be notified about the decision. The scheduling algorithm checks if there are free channels in any set S_i, the index of which must be smaller than or equal to the number of requests in the queue Q_j. If there is no free channel available, the first request is rejected and deleted from Q_j. Otherwise all the requests in the queue are served, immediately after the waiting time of the first request (the bunching property). In our model, however, all the requests are inserted in the order they have arrived into one special common FIFO queue with *the bunching property* as described in 2.2.

2 Modeling the VOD System with Pr/T-Nets

2.1 The Simplified Net Model

For an introduction a simplified Pr/T-net model with initial marking for the VOD system is given in Fig. 2. We describe the behaviour of the Pr/T-net by explaining the system, which is modelled in Fig. 2. In this system users make requests for movies and for the simplicity, the requests are decided one by one; so the bunching property is not demonstrated. Requests are saved in the FIFO, until they are served. The service means the acceptance or the rejection of a request.

Movie archive contains the names of all the movies that are available in the server; movies that are active are considered unavailable. We assume that there is only one channel set, which contains c_0 channels initially. The users, whose requests are rejected, are notified of the rejections. A request is rejected, if there are no free channels or the movie is not available. In Fig. 2 tokens, indicating active movies, are saved into the movie archive complement place. The transition `Reject` is enabled, if the place contains a token, indicating the movie, the user has requested, although there were free channels. A request is accepted, when there is a free channel that can be occupied for the movie and the name of the requested movie is in the movie archive. A channel and a movie are allocated for the user, whose request has been accepted. The information about allocated channels and movies and the users, whose requests have been accepted, is saved into another FIFO. This FIFO is filled in the order the requests are accepted and resources for the accepted requests allocated. Those users, whose requests have been accepted, are notified of the acceptance. When the movie ends, all allocated resources become free and the users return back to the idle state. So our model is *causal* in the sense that the movies are shown in the *order* they are requested (if not rejected); so there are no timing considerations.

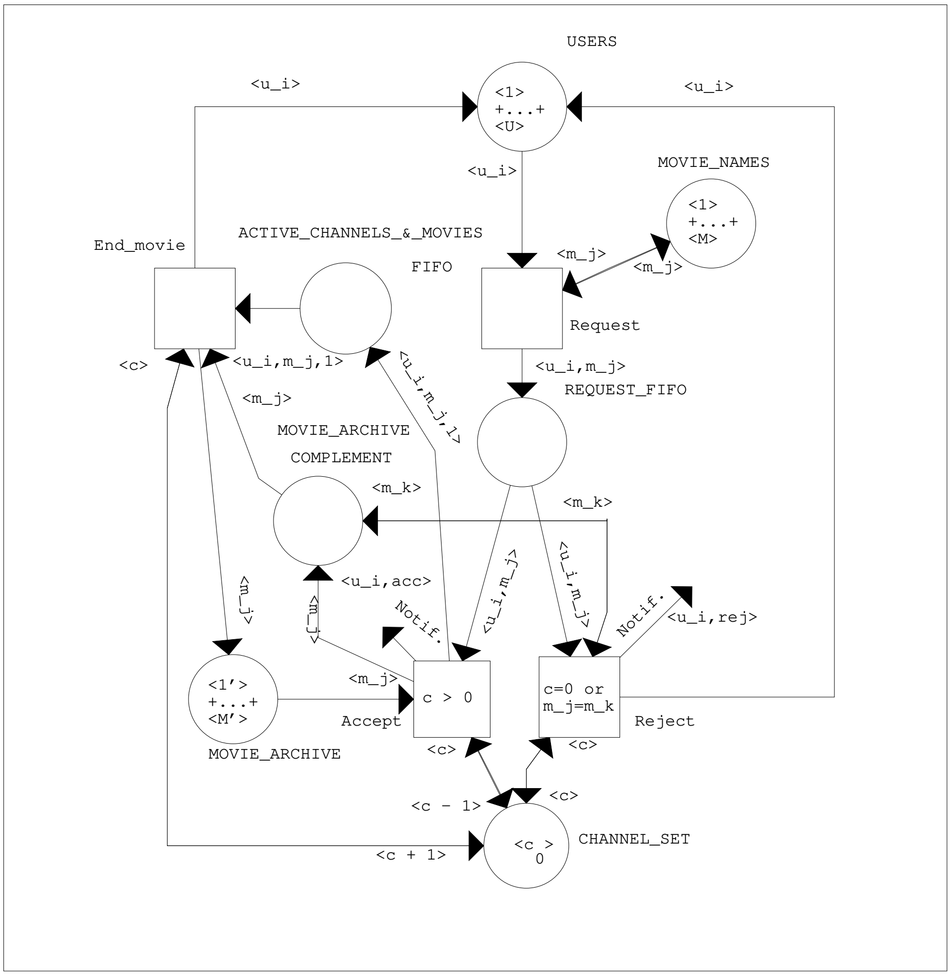

Figure 2: A simplified net model for VOD without the bunching property.

It can easily be seen that the use of the reject mechanism eliminates the deadlocks resulting of the finiteness of the channel resources. The proper operation of the net can be verified using the well-known methods of invariant analysis [4].

2.2 The FIFO Queue with the Bunching Property

FIFO Queue In Fig. 3 a simple Pr/T-net model of the FIFO queue is given based on the well-known complementary place construction. When a place p is free, the complement place p' contains a `no data` (`ND`) token. Shift transitions move tokens from the input of the FIFO towards the output as long as there are free positions. In the initial marking of Fig. 3 the FIFO contains data items `d(1)`, `d(2)`, $\cdots$, `d(p-2)` and `d(p-1)`. However, an efficient model of FIFO queue is of primary importance when the reachability analysis is used to study the system properties. In Fig. 3 the shifting of the contents of the FIFO has to be performed in several steps thus generating several unnecessary nodes into the reachability graph.

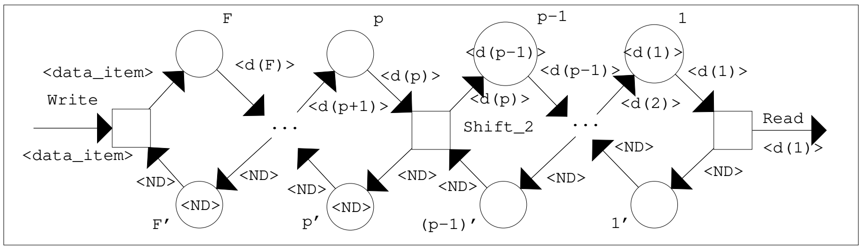

Figure 3: A straightforward Pr/T-net model of the FIFO queue.

So we have chosen a more efficient Pr/T-net FIFO model given in Fig. 4 [6, 7]. The idea of this FIFO model is that we rather write messages directly into the correct positions and also reorder the FIFO simultaneously while sending a message from the FIFO. This net model reduces the state space explosion very effectively as given in a critical comparison between different high-level FIFO net models in [6, 7]. We have used a letter `p` as an abbreviation of the first free position and `ND` as an abbreviation of no data in Fig. 4.

From now on we shall consider the FIFO as a *macro* operation. So we have in use a data structure, to which we can insert and from which we can remove data items in such a way that the items remaining in the structure do never violate the FIFO principle.

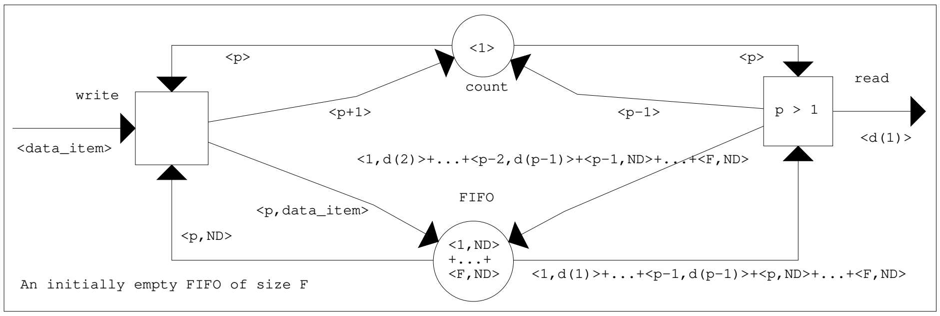

Figure 4: An efficient way to model a FIFO using Pr/T-nets.

Bunching Property The queuing system of the VOD becomes very complex partly because of a FIFO queue and partly because of the bunching property. It is easy to reserve a channel for a large number of users (for example one bit field, indicating whether the channel is free or occupied), but to empty the FIFO after the movie has ended is more problematic (an arbitrary, but finite number of data items must be changed). The FIFO must be emptied in such a way that the requests of other movies remain unchanged and the FIFO discipline is not violated. To demonstrate how *a FIFO with the bunching property* can be emptied a possible four-staged net model is given in Fig. 5.

An accepted request token and all other request tokens for the same movie are removed from the FIFO and placed into the place `ACTIVE_MOVIES` by firing the transition `Read` and all those transitions Et_i (Empty transition), whose input place i contains a token `<d(1)>`. The transitions Et_i add `no data` tokens to those places i', whose complementary places i are emptied. The contents of those FIFO places i, which contain other tokens than `<d(1)>` or are empty, remain unchanged. In this case the transitions Et_i' empty the places E_i, whose usage help modelling the bunching property. The transitions Et_i' are enabled only then when the contents of the places E_i and i differ. This is expressed with the firing conditions G_i below the net.

In Fig. 6 it is shown in greater detail, how the transition $\mathtt{Et}_i$ removes a token from the FIFO, if the token `<d(i)>` has the same value as the token indicating an accepted request `<d(j)>`. Otherwise the transition $\mathtt{Et}_i$' is fired, which only removes the token `<d(j)>` from the place $\mathtt{E}_i$.

The actual FIFO with the bunching property realizationed in this work is given in Chapter 3.

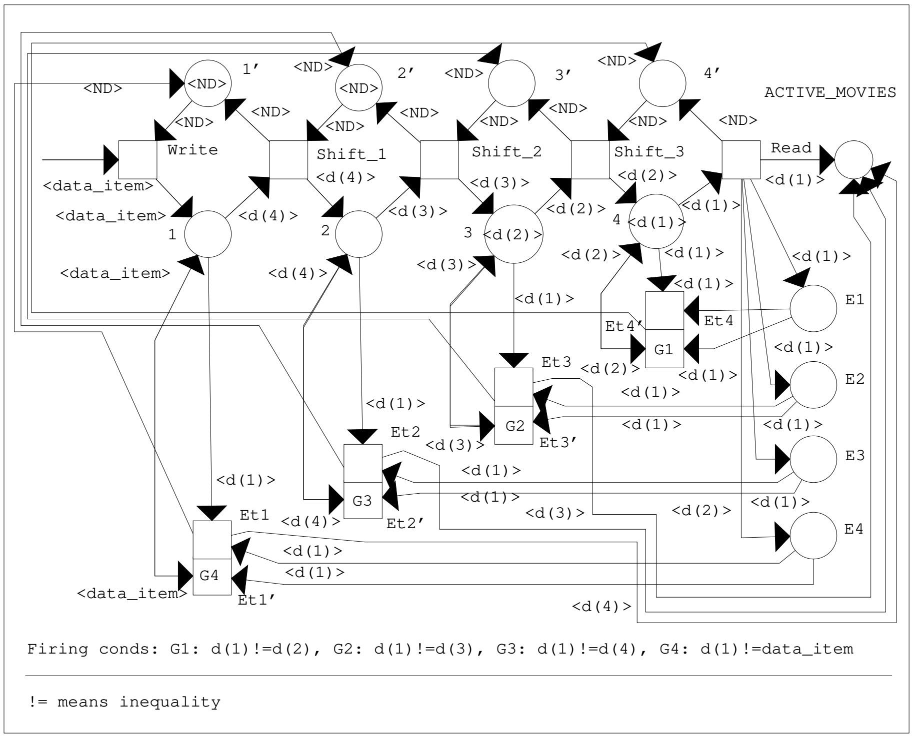

Figure 5: A Pr/T-net model of the FIFO with the bunching property.

3 A Detailed Pr/T-Net Model

3.1 Places of the Net

The detailed causal Pr/T-net model of the VOD system given in Fig. 7 has seven places. The initial markings of the places and interpretation of the tuple tokens in the places are listed in **TABLE 1**. The field variables of the tuples in the arc expressions are shown in parenthesis in **TABLE 1**. The place `USERS` is an input place of the writing transition of the request FIFO as well as the places `COUNT_1`, `REQUEST_FIFO` with the bunching property, `MOVIE_INFO` and `ALREADY_REQUESTED`. The net model of the request FIFO contains the places `COUNT_1` and `REQUEST_FIFO`, the write transition `Request` and two read transitions `Reject` and `Accept` that remove tokens from the request FIFO place. The transition `Accept` puts accepted request tokens to the place `ACTIVE_MOVIES`. The transition `Empty` removes the accepted request tokens from the place `ACTIVE_MOVIES` and returns the user tokens to the place `USERS`. The tokens of the places `MOVIE_INFO`, `CHANNELS_SETS` and `ALREADY_REQUESTED` contain information, according to which the resources of the system (channel sets) are scheduled between the requesting users. In our

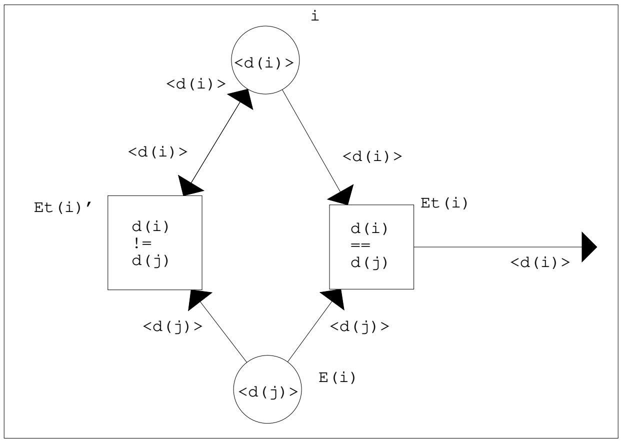

Figure 6: Removal of a token from the FIFO.

model channel sets are represented by a pair of numbers. The first number indicates the minimum number of requests that is needed to reserve a channel from that channel set. The second number indicates the number of free channels in that channel set.

TABLE 1. The initial markings of the places.

Place	Initial marking	Interpretation
USERS	<1>+..+<U>	U 1-tuple tokens indicating the users (`u`)
COUNT_1	<1>	one 1-tuple token indicating the first free position in the request FIFO (`p`)
REQUEST_ FIFO	<0,0,1> +..+ <0,0,U>	U 3-tuple tokens indicating the request FIFO items: <movie name (`m`), user name (`u`), position in the request FIFO (`p`)>
ACTIVE_ MOVIES	∅	2-tuple tokens indicating the accepted FIFO items: <movie name (`m`), user name (`u`)>
CHANNEL_ SETS	<1,c1> +..+ <s,cs>	s 2-tuple tokens indicating the s channel sets and the number of channels in each set (`s`, `n`)
ALREADY_ REQUESTED	<0>	the 1-tuple token indicating the users, whose request is rejected during an active movie (`ar`)
MOVIE_ INFO	<1,0,0,0> +..+ <M,0,0,0>	M 4-tuple tokens indicating the movie parameters: <movie name, number of requests, active/inactive bit, the channel reserved for the movie> (`m` or `n`, `r`, `a`, `c`)

Figure 7: A Pr/T-net model of a VOD system.

3.2 Transitions of the Net

The arc expressions of the Pr/T-net in Fig. 7 are listed in **TABLE 2**. Arc expressions are named in such a way that the first upper case and the following lower case letters denote the transition name, the upper case letters in the middle of the expression name denote the name of the place, which is connected to the transition and the last letter (I or O) denotes, if the expression belongs to the input or output arc of the transition. The macros used in **TABLE 2** are listed in **TABLE 3**. We have used an expression $\texttt{(}\mathit{expr}\texttt{)<}\mathit{tuple}\texttt{>}$ as a conditional expression. When the expression *expr* is evaluated to true, the arc expression shall contain the tuple token *tuple*. The symbolic sum of tuple tokens from i to j is denoted by $\sum_{k=i}^{j}\texttt{<}\mathit{tuple}_k\texttt{>}$.

TABLE 2. Arc expressions corresponding the macros used in Fig. 7.

Arc Expressions:	
ReqCO	$\texttt{(a==0)<p1+1>+(a==1)<p1>}$
ReqRFO	$\texttt{(a==0)<m,u,p1>+(a==1)<NM,NU,p1>}$
RejCSI	$\sum_{k=1}^{S}\texttt{<}\mathtt{s}_k,\ \mathtt{n}_k\texttt{>}$
RejCSO	RejCSI
RejRFI	$\sum_{j=1}^{U}\texttt{<}\mathtt{m}_j,\ \mathtt{u}_j,\ j\texttt{>}$
RejRFO	$\sum_{j=2}^{U}\texttt{<}\mathtt{m}_j,\ \mathtt{u}_j,\ j-1\texttt{>+<0,0,}U\texttt{>}$
AccRFI	$\sum_{j=1}^{U}\texttt{<}\mathtt{m}_j,\ \mathtt{u}_j,\ j\texttt{>}$
AccRFO	$\sum_{j=1}^{U}\texttt{<}\mathbf{UnServed}(\mathrm{m},0,j),\ \mathbf{UnServed}(\mathrm{u},0,j),\ j\texttt{>}$
AccAMO	$\sum_{j=1}^{U}\texttt{<}\mathbf{Served}(\mathrm{m},0,j),\ \mathbf{Served}(\mathrm{u},0,j)\texttt{>}$
EUO	$\sum_{j=1}^{U}\texttt{(}\mathtt{m}_j\texttt{==}\mathtt{m}_1\texttt{)<}\mathtt{u}_j\texttt{>}$
EAMI	$\sum_{j=1}^{U}\texttt{<}\mathtt{m}_j,\ \mathtt{u}_j\texttt{>}$
EAMO	$\sum_{j=1}^{U}\texttt{<}\mathbf{UnServed}(\mathrm{m},0,j),\ \mathbf{UnServed}(\mathrm{u},0,j)\texttt{>}$

TABLE 3. Macros used in **TABLE 2**.

UnServed(item, init, j)	$(\exists i, \mathrm{m}_{l_1}, \cdots, \mathrm{m}_{l_{j-1}} : 1 < i \leq U \mid$ $\mathtt{m}_i \neq \mathtt{m}_1 \wedge \forall k : 1 \leq k < i \wedge k \notin [l_1, \cdots, l_n] \mid$ $\mathtt{m}_k = \mathtt{m}_1 \wedge \mathtt{m}_{l_1} \neq \mathtt{m}_1 \wedge \cdots \wedge \mathtt{m}_{l_{j-1}} \neq \mathtt{m}_1)\ \mathrm{item}_i\ :\ \mathrm{init}$
Served(item, init, j)	$(\exists i, \mathrm{m}_{l_1}, \cdots, \mathrm{m}_{l_{j-1}} : 1 < i \leq U \mid$ $\mathtt{m}_i = \mathtt{m}_1 \wedge \forall k : 1 \leq k < i \wedge k \notin [l_1, \cdots, l_n] \mid$ $\mathtt{m}_k \neq \mathtt{m}_1 \wedge \mathtt{m}_{l_1} = \mathtt{m}_1 \wedge \cdots \wedge \mathtt{m}_{l_{j-1}} = \mathtt{m}_1)\ \mathrm{item}_i\ :\ \mathrm{init}$

The effects of the firings of the transitions are described below.

Request. If a user requests a movie, which is not currently active (watched by some users), a request is placed in the request FIFO into the first unoccupied position. If the user requests an active movie, his request token is placed into a place `ACTIVE_MOVIES`. Recall that all earlier accepted requests for an active movie are kept in that place.

Reject. If the first request in the request FIFO is made for a movie, which is not requested by at least as many users as is the minimum requirement of the movie channels, the request in the FIFO is rejected, an indication is sent to the user, the request is removed from the request FIFO and all other requests are shifted one position towards the beginning of the FIFO so that the FIFO discipline is not violated.

Accept. When a request is accepted, i.e., the number of users, who have requested the movie, is at least the minimum channel capacity of an unoccupied channel, the request and all the requests for the same movie are placed into the place `ACTIVE_MOVIES` and an indication is sent to those users, whose request is accepted.

Empty. After a movie, the requests for that movie are removed from the place `ACTIVE_MOVIES`. User tokens are returned to the user place.

The transition `Request` cannot fire, if a user's request is rejected during an active movie, which has not ended. When the transition `Reject` fires, a bit vector of length U is added to the value of the token in the place `ALREADY_REQUESTED`. The u_1th bit of the vector is set to 1, indicating that the user u_1's request was rejected. The transition `Request` can fire only, if the uth bit of the bit vector in the place `ALREADY_REQUESTED` is not set, i.e., when the bitwise and of the vector and 2^u is zero. `Accept` fires only, if there is a free channel and the request FIFO contains enough requests. Otherwise the transition `Reject` is fired. These firing conditions are listed in **TABLE 4**.

TABLE 4. Firing conditions used in Fig. 7.

Gate Expressions:	
Req_fc	$((\mathtt{ar}\ \&\ 2^{\mathtt{u}})\mathtt{==0})\ \&\&$
Rej_fc	$(\forall k : \mathtt{n}_k\mathtt{==0} \parallel \mathtt{s}_k > \mathtt{r}_1\ )$
Acc_fc	$\mathtt{n} > 0\ \&\&\ \mathtt{r} \geq \mathtt{s}$

4 Reachability Studies

We had decided to use reachability analysis for studying the properties of our Pr/T-net model; therefore already in the modelling phase constructions to reduce the state space explosion were sought, especially in the FIFO models. However, it was clearly seen that increasing the number of request sequencies in the analysis should finally generate the state space explosion.

We analyzed our Pr/T-net model of the VOD system with our advanced predicate/transition net reachability analysis tool, `PROD` [5, 9]. It has been developed in the Digital Systems Laboratory of Helsinki University of Technology. The net description is compiled into C language, which is compiled into an executable file. The execution of the file generates the reachability graph of the model. The properties of the model can be examined with the query language, based on branching time logic CTL (Computation Tree Logic) [3].

Analysis Results The reachability analysis results of the Pr/T-net model in Fig. 7 are shown in **TABLE 5**. For simplicity we have assumed that all movies have the same duration, which means that the place `ACTIVE_MOVIES` obey the FIFO principle. This choice will decrease the size of the reachability graph considerably. No deadlocks were detected; this results of the use of the reject mechanism in the VOD system.

TABLE 5. Analysis results.

Parameters	Reachable nodes	Parameters	Reachable nodes
U = 3 M = 2 C = 2	247 (35)	U = 3 M = 5 C = 2	2326 (341)
U = 3 M = 3 C = 2	640 (38)	U = 3 M = 5 C = 3	3091 (401)
U = 3 M = 3 C = 3	775 (103)	U = 3 M = 5 C = 4	3331 (411)
U = 3 M = 3 C = 4	775 (104)	U = 4 M = 4 C = 2	20833 (705)
U = 3 M = 4 C = 2	1309 (189)	U = 4 M = 4 C = 3	28401 (813)
U = 3 M = 4 C = 3	1681 (219)	U = 4 M = 4 C = 4	31137 (833)
U = 3 M = 4 C = 4	1753 (223)	U = 4 M = 5 C = 2	45781 (1596)

From the **TABLE 5** it can be seen that the number of users affects the size of the reachability graph stronger than the number of movies or the number of channels. To be more precise, U users can be put into the FIFO in $U!$ different orders. Because every user can choose any movie from M movies, there are totally M^U different choices. Thus the requests sequences ($\langle \text{user}, \text{movie} \rangle$) can appear in $U! \cdot M^U$ ways. For comparison these values are given too in Fig. 8 which illuminates the effect of different parameters on the size of the reachability graph.

In our model we have identified the users using individual tokens; this is necessary because we have restricted the number of requests the user can make during an active movie to only one. Without this restriction only the number of users is important so we can model the users with black tokens. Then the number of user combinations possible in the request FIFO is reduced by the factor $U!$. The sizes of the reachability graphs generated using black user tokens are given in **TABLE 5** in parenthesis; see also Fig. 8. The size

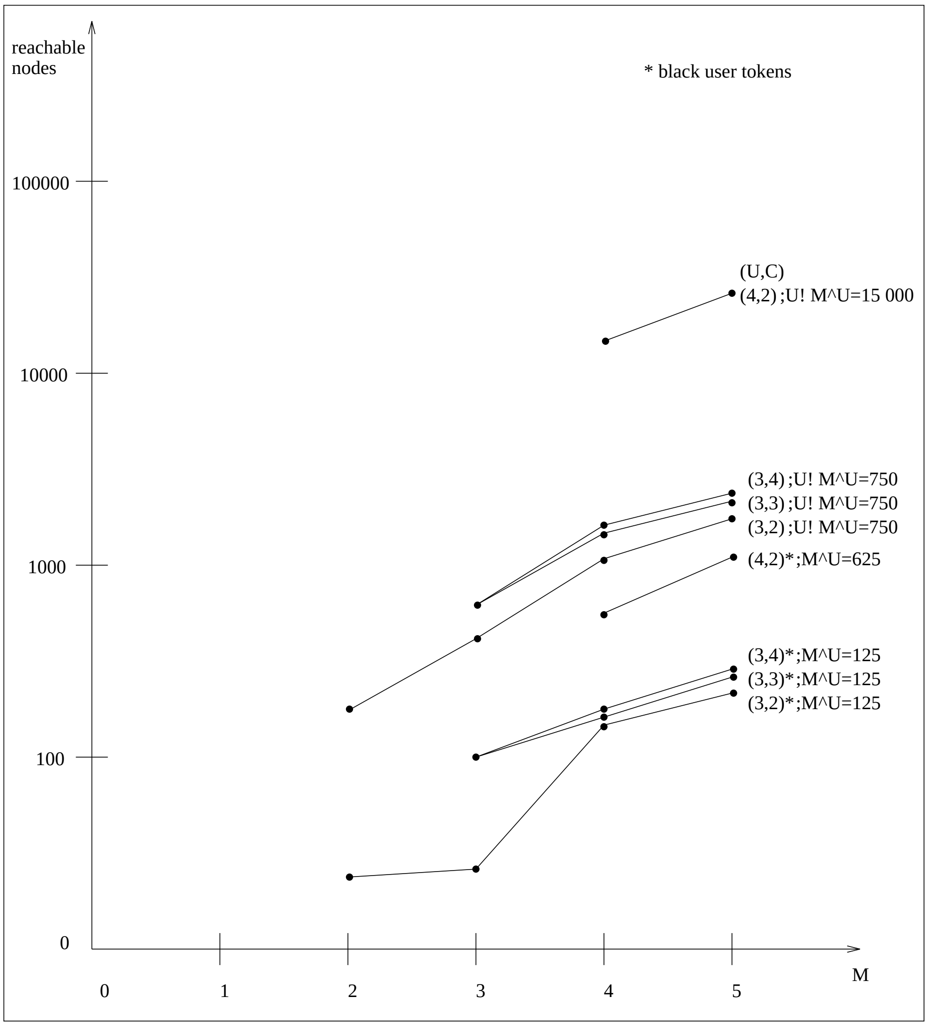

Figure 8: The size of the reachability graph with some (M, U, C) parameters.

of reachability graphs might still be reduced by using the different reduction methods realized in the PROD tool (symmetry method, stubborn set method or sleep set method) [9], but this is however not so essential in our actual research problem.

It should again be emphasized that for the reachability analysis the proper, efficient modelling of the FIFO with the bunching property is decisively important. Our previous preliminary model [8], in which the bunching property was implemented in such a way that the FIFO was not reordered after the acceptance of a set of requests, i.e, empty positions in the FIFO were not filled with unserved requests, until the movie ended, much larger FIFO was needed. For example the reachability graph of this net contained 53 292 states, when

$U = 3$, $M = 2$, $C = 2$ and the number of channel sets were two (both sets contained one channel), one requiring one request and the other two requests. The reachability graph of a net model described in this report contains only 247 states with the corresponding number of users, movies and channels.

It can be seen in Fig. 8 that our FIFO modelling technique has been quite successful; the number of request sequencies seems clearly affecting the size of the reachability graph.

5 Conclusion

We have demonstrated the use of Pr/T-net formalism in the causal modeling of a VOD system given in the literature. We have shown how the most complex parts of this system (FIFOs with the bunching property, the resource allocation and the parallel machine scheduling) can be modeled. We have studied the behaviour of the system using reachability analysis; to fight against the state space explosion in the analysis very careful modelling constructions have been needed. However, with the reachability analysis only relatively small systems can be analyzed because of the dominating role of the rapidly increasing number of request sequencies in the analysis method used.

Based on this causal model more detailed time or stochastic models will be later developed to study the performance aspects of the system, i.e., the effect of basic design parameters of the system on the behaviour of the system when the loading (the request rate) increases and thus to find the system bottlenecks. Especially interesting will then be the study of the effects of the rejection and channel set mechanisms on the total system performance.

Acknowledgements

The authors would like to thank Juan Garay for introducing this new VOD system to us and in that way giving us an idea and opportunity for this research.

References

[1] Aggarwal, S., Garay, J. and Herzberg, A.: *Adaptive Video on Demand.* Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing (PODC), 1994, Los Angeles, USA, p. 402.

[2] Aggarwal, S., Garay, J. and Herzberg, A.: *Adaptive Video on Demand.* Unpublished manuscript. 21 p.

[3] Clarke, E. M., Emerson, E. A., and Sistla, A. P.: *Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications.* ACM Transactions on Programming Languages and Systems 8 (1986) 2. pp. 244–263.

[4] Genrich, H. J.: *Predicate/Transition Nets.* Brauer, W., Reisig, W., and Rozenberg, G. (ed.). Petri Nets: Central Models and Their Properties, Advances In Petri Nets 1986, Part I. Lecture Notes in Computer Science 254. Berlin 1987, Springer-Verlag. pp. 207–247.

[5] Grönberg, P., Tiusanen, M., and Varpaaniemi, K.: *PROD—A Pr/T-Net Reachability Analysis Tool.* Otaniemi 1993, Digital Systems Laboratory, Helsinki University of Technology, Series B: Technical Reports, No. 11. 44 p.

[6] Kettunen, E., Montonen, E. and Tuuliniemi, T.: *A Comparison of Pr/T-Net Based FIFO Channel Models.* Otaniemi 1986, Digital Systems Laboratory, Helsinki University of Technology, Series B: Research Reports, No. 33. 18 p.

[7] Kettunen, E., Montonen, E. and Tuuliniemi, T.: *A Comparison of Pr/T-Net Based FIFO Channel Models.* Proceedings of IMACS 1988, 12th World Congress on Scientific Computation, 1988, France, pp. 479–482.

[8] Pyssysalo, T. and Ojala, L.: *Modelling of a "Video on Demand" System Using Pr/T-Net Formalism — a Case Study.* Proceedings of CSP'94, Humboldt-University Berlin, 1994, Berlin, 8 p.

[9] Varpaaniemi, K., Halme, J., Hiekkanen, K., and Pyssysalo, T.: *PROD Reference Manual.* Otaniemi 1995, Digital Systems Laboratory, Helsinki University of Technology, Series B: Technical Reports, No. 13. 56 p.