Journal of Circuits, Systems, and Computers
Vol. 15, No. 4 (2006) 571–606

World Scientific
www.worldscientific.com

PERFORMANCE MODELS FOR STUDYING AND ANALYZING REAL-TIME MULTIMEDIA SYSTEMS

SAMIR M. KORIEM*

Systems and Computers Engineering Department,
Faculty of Engineering, Al-Azhar University,
Naser City, Cairo, Egypt
samirkoriem@yahoo.com

Revised 2 March 2006

As real-time systems continue to grow, performance evaluation plays a critical role in the design of these systems since the computation time, the service time, and the responsive actions must satisfy the time constraints. One of these systems is the real-time distributed multimedia-on-demand (MOD) service system. The MOD system usually fails when it misses a task deadline. The main units of the MOD system usually communicate with each other and work concurrently under timing constraints. The MOD system is designed to store, retrieve, schedule, synchronize and communicate objects comprising of mixed data types including images, text, video and audio, in real-time. In the MOD system, such data types represent the main concept of movie files. Modeling of such concurrency, communication, timing, and multimedia service (e.g., store, retrieve) is essential for evaluating the real-time MOD system. To illustrate how to model and analyze the important multimedia aspects of the MOD system, we use the Real-net (R-net) modeling technique. We choose R-net as an extension of Time Petri Net due to its ability to specify hard real-time process interaction, represent the synchronization of multimedia entities, describe concurrent multimedia activities, and illustrate the interprocess timing relationships as required for multimedia presentation. Based on modular techniques, we build three R-net performance models for describing the dynamic behavior of the MOD service system. The first model adopts the Earliest Deadline First (EDF) disk scheduling algorithm. The other models adopt the Scan-EDF algorithm. These algorithms help us to illustrate how the real-time user requests can be satisfied within the specified deadline times. Since R-nets are amenable to analysis including Markov process modeling, the interesting performance measures of the MOD service system such as the quality of service, the request response time, the disk scheduling algorithm time, and the actual retrieval time can be easily computed. In the performance analysis of the MOD models, we use our R-NET package.

Keywords: Distributed multimedia systems; disk scheduling algorithms; time Petri nets; performance models; Markov chain; performance analysis.

*Current address: Chairman of Hardware Section, Computer Technology Department, College of Technology at Abha, P. O. Box 1604, Abha, Saudi Arabia.

1. Introduction

1.1. *Background*

Multimedia information systems are useful in areas such as education, medicine, entertainment, and space research.[1,2] One of the most visible applications of multimedia information systems is the distributed multimedia-on-demand (MOD) service system. This distributed MOD system describes how to organize, store hundreds or thousands of movie files (i.e., text, images, audio and video data) over multiple disks and allow tens of thousands of viewers (i.e., users) to concurrently access (e.g., retrieve) these movie files. The distributed MOD service system gives the user greater flexibility in choosing which movie to watch from a varied daily menu. Users can also decide when they want to see the movie they have chosen. Although these services provide flexibility, speed and a high level of efficiency, they require new support technology and considerable effort needs to be made to improve the quality of service provided by the existing systems. The storage of movies in the distributed MOD service system imposes three requirements: storage, management and concurrent access requirements. The storage requirement is defined as the disk array architecture that needs to be occupied by hundreds or thousands of movies in digital format. The management requirement is defined as the large amount of data movie files to be managed, tens or hundreds of disks are available online to the user. The concurrent access requirement is defined as the large number of concurrent accesses that must be achieved to serve thousands of users at the same time.

Most recent research works have focused on the investigation of the following interesting multimedia subjects: (i) The design of several configurations to obtain an efficient and reliable multimedia storage system.[3,4] (ii) The study of various approaches that are required to manage the traffic of real-time user requests to the storage disks.[5] (iii) The analysis of various techniques that are proposed to efficiently store and retrieve large multimedia data on/from the storage devices.[1,6–12] These research works have concentrated on constructing simulation and theoretical models for the architectures of storage part of multimedia systems to study and analyze the various problems of the available multimedia systems. Such problems are the disk load balancing problem,[13] the buffer management problem,[14] the reliability problem of large disk arrays,[3,4] and the problem of disk scheduling algorithms.[6,9,10,14–16] Also, the augmented transition network technique[17] has been used to model the semantic aspects of a multimedia presentation, browsing, and database searching.

1.2. *Related work*

Petri nets (PNs) are a graphical and mathematical modeling tool applicable to many systems. As a graphical tool, PN provides a natural way to represent complex logical interactions among parts or activities in information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel,

nondeterministic, real-time, and/or stochastic. As a mathematical tool, PN provides a mechanism to set up state equations, algebraic equations, and other mathematical models governing the behavior of systems. Both practitioners and theoreticians use PNs. Thus, PNs provide a powerful medium of communication between them: practitioners can learn from theoreticians how to make their models more methodical, and theoreticians can learn from practitioners how to make their models realistic.[18]

To provide a good means for describing temporal aspects, various timed extensions of PNs have been proposed by attaching timing constraints to transitions, places, and/or arcs. For performance modeling, stochastic Petri nets (SPNs),[19] generalized SPNs (GSPNs),[20] and stochastic well-formed nets[21] associate exponentially distributed firing delays to the transitions of the original PN model. These stochastic nets can be used for modeling the parallel and distributed systems. In contrast to stochastic nets, Time Petri nets (TPNs)[22] extend the original PNs by assigning constant delays to their transitions. In Ref. 22, TPNs have been used for analyzing the recoverability of communication protocols. In Ref. 23, Timed PNs have been used to model multimedia processing synchronization scenarios. Time constraint PNs (TCPNs)[24] extend the original PNs by assigning static time intervals with constraints to transitions or places of the net. TCPNs are suitable for systems with conflict structures.

In Ref. 25, Hierarchical Time Stream Petri nets (HTSPNs) have been used for the specification of temporal (i.e., multimedia) and logical (i.e., hypertext) synchronization within hypermedia distributed and weakly synchronous systems. In HTSPNs, the temporal nondeterminism is expressed using timed arcs. In Ref. 26, Extended Object Composition PN (XOCPN) combines the logic of temporal intervals and TPNs. XOCPN is a network model and a good data structure for controlling the synchronization of multimedia presentation. XOCPN can serve as a visualization structure for users to understand the presentation sequence of media streams. XOCPN faces two problems when replaying objects in distributed multimedia systems[27]: (1) It lacks method to describe the details of synchronization across distributed platforms. (2) It does not deal with the schedule change caused by user interactions in interactive multimedia systems.

In Ref. 28, we have developed a new methodology to facilitate the analysis of real-time systems using Real-nets (R-nets). R-nets extend TPNs[22] by distributing *probability density functions* (pdfs) with *uniform distributions* over the *time intervals* associated with the *transitions* of the net. These pdfs are used to determine (dynamically) the state transition probability of branching from the current state to the possible next states. The structure of the R-net contains multiple tokens, multiple arcs, inhibitor arcs and two types of transitions: real-time transitions and zero-time transitions. R-net is a powerful technique for studying and analyzing the specification, verification and temporal behavior of real-time distributed systems.[28] By R-net is a powerful technique we mean that the ability of the R-net formalism to represent classes of problems as well as the practical ability to represent a given behavior in a natural way. Recently, the delay TPNs (DTPNs) have been developed

in Ref. 29. DTPNs extend TPNs by assigning static delay intervals to the transitions and arcs of the net. Also, it uses reduction rules to analysis the real-time systems. The DTPN technique is close to our R-net technique.

1.3. *Contributions of this research work*

This paper addresses the following important problems:

(1) How to develop specification MOD models for describing the storage and retrieval processes of time-dependent multimedia data.
(2) How to model the parallel (or the sequential) retrieval processes of multimedia data from multiple disks (or a single disk).
(3) How to model the real-time communication processes between the users and MOD architecture.
(4) How to adopt the disk scheduling algorithms in the R-net MOD models for describing the real-time scheduling processes of the concurrent user requests.
(5) How to build an accurate model able to describe the dynamic behavior of the MOD architecture in a natural way to derive the important multimedia performance measures such as the quality of service, the request response time, the disk scheduling time, and the actual retrieval time.
(6) How to study and analyze the performance results of the R-net MOD models. Such studies can be used in designing an efficient multimedia system.

1.4. *Organization of the remainder of the paper*

In Sec. 2, we illustrate how the R-net modeling technique can be used to obtain a realistic and accurate description for real-time distributed systems. In Sec. 3, we describe the basic structure of the distributed MOD system. In Sec. 4, we explain how the R-net technique can be used to construct Petri net models for the MOD system. We illustrate this explanation throughout two simple R-net MOD models. The first model adopts the EDF algorithm. The second model adopts the Scan-EDF algorithm. These models describe accurately the dynamic behavior of the MOD system when it receives concurrent user requests. In Sec. 5, we explain the construction process of building a realistic R-net MOD model. Performance measures of the R-net models of the MOD system are calculated and evaluated in Sec. 6. Section 7 summarizes the conclusions of our work.

2. R-net Modeling Technique

The R-net modeling technique is one of the most suitable methodologies for studying the performance evaluation of real-time concurrent systems and distributed systems.[28] To perform this study, the R-net technique distributes the probability density functions (pdfs) of uniform distribution over the firing intervals associated with the concurrent transitions and/or competitive transitions of the net. These

pdfs are used to determine (dynamically based on the overlapped intervals shown in the current state) the probabilities of branching to possible next state(s). In this way, the reachability graph obtained from the developed model can be interpreted as a Markov process.

Definition 1. A PN is a tuple (P, T, A, M_0, W, H), where $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places, $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, $A \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs between places and transitions, W: $A \to \{1, 2, 3, \ldots\}$ is the arc weight function, $H \subset (P \times T)$, $A \cap H = \emptyset$ is an inhibition function, and M_0: $P \to N$ is the initial marking function, where N is the set of natural numbers.

In this definition, a place means the state of an event. A transition corresponds to the rule of firing or nonfiring. All events concurring at a transition must be complete and ready before firing. Input functions and output functions change the state of events. Tokens are used to simulate the dynamic and concurrent activities of the desired systems.

Definition 2. An R-net is a tuple (PN, β, $\Re$), where PN is a marked PN, called the corresponding untimed net; β: $T \to (Q^+ \cup 0)(Q^+ \cup \infty)$ is the interval function that defines the permitted transition firing times, where Q^+ is the set of positive rational numbers, and

$$\beta_i = \{[\tau_{i\,\min}, \tau_{i\,\max}] \,|\, \forall\, t_i \in T\} \quad \text{with } 0 \le \tau_{i\,\min} \le \tau_{i\,\max};$$

$\Re : T \to F^+$ is the pdf that can be distributed over the firing interval of each transition, where F^+ is the set of non-negative real numbers: $F^+(t_i) = 1$ if t_i is only an enabled (firable) transition; $0 < F^+(t_i) < 1$ if t_i is an element of concurrent (parallel activities) or conflicting (conflicting events) transitions.

From Definition 2, we should remark the following points.

- $\tau_{i\,\min}$ denotes the minimal time that must elapse from the time that all the input conditions of transition $t_i \in T$ are enabled. $\tau_{i\,\max}$ denotes the maximum time that the input conditions can be enabled and the transition $t_i \in T$ must fire. Thus, in such nets, the firing of transition $t_i \in T$ takes no time, but a transition $t_i \in T$ must be enabled for a minimum time $\tau_{i\,\min}$ and possibly up to a maximum time $\tau_{i\,\max}$.
- R-net adopts two types of transitions: real-time transitions (RTT) and zero-time transitions (ZTT).

Definition 3. (*Execution rule of* RTT) Each $t_i \in$ RTT has a time interval $[\tau_{i\,\min}, \tau_{i\,\max}]$. The transition $t_i \in$ RTT is enabled as soon as its input places are marked, but firing of transition $t_i \in$ RTT could only take place within a specified time interval β. This interval is measured from the time epoch that a transition $t_i \in$ RTT is enabled. When the input conditions of transition $t_i \in$ RTT hold for a period equal to $\tau_{i\,\min}$ ($\tau_{i\,\max}$), the transition $t_i \in$ RTT can (must) fire.

Definition 4. (*Execution rule of* ZTT) Each $t_i \in$ ZTT has a time interval [0, 0]. When the transition $t_i \in$ ZTT is enabled in a particular state, it fires immediately with probability one and cannot remain enabled for any duration of time. This concept means that the transition $t_i \in$ ZTT takes zero time to fire. Therefore, $t_i \in$ ZTT has priority over $t_j \in$ RTT. Thus, when several RTT and one $t_i \in$ ZTT are enabled in a particular state, the only a transition $t_i \in$ ZTT fires with probability one.

From the various interval relations shown in Fig. 1, we noticed that the intervals of equal length have equal probability of containing an observed β value. Also, from Definition 3, we noticed that the enabled RTT might fire at any time during an allowed bounded firing interval $[\tau_{\min}, \tau_{\max}]$. Based on these reasons, we model the time β as a *continuous random* variable with a *uniform distribution* over the interval $[\tau_{\min}, \tau_{\max}]$. Using the "probability density function" and the "cumulative distribution function" of this distribution,[28] we continue our analysis as follows.

Definition 5. The *probability density function* $f_I(\beta_I)$ of a uniformly distributed random variable β_I on the time interval $[\tau_{\min}, \tau_{\max}]$ of a transition $t_I \in$ RTT can be formulated as follows.

$$f_I(\beta_I) = 1/[(\tau_I)_{\max} - (\tau_I)_{\min}] \quad \text{if } (\tau_I)_{\min} \le \beta_I \le (\tau_I)_{\max}\,.$$

Definition 6. The *cumulative distribution function* $F_I(\beta_I)$ of the uniformly distributed random variable β_I can be formulated as follows:

$$\begin{aligned}\text{Prob}\,[(\tau_I)_{\min} \le \beta_I \le x] &= \text{Prob}(\beta_I \le x) = F_I(x) = \int_{\beta_I} f_I(\beta_I)d\beta_I \\ &= (x - (\tau_I)_{\min})/[(\tau_I)_{\max} - (\tau_I)_{\min}]\,.\end{aligned}$$

After appending the uniform pdfs of Definitions 5 and 6 to the time intervals associated with the transitions of the modeled system, we need to understand how these pdfs can be used to calculate the firing probabilities of these transitions. We continue our analysis by considering the current state of the modeled system is the state S_i.

Definition 7. When a particular state S_i contains one enabled transition $t_I \in$ RTTs associating with the time interval β_I, this transition can fire at any time between $\tau_{\min}$ and $\tau_{\max}$ with probability one as follows:

$$\text{Prob}(t_I) = \text{Prob}(\tau_{\min} \le \beta_I \le \tau_{\max}) = \int_{\beta_I} f_I(\beta_I)d\beta_I = 1\,.$$

Theorem 1. (Firing transition probability method) *When a particular state S_i contains two enabled transitions (t_I and t_J) and their time intervals ($a \le \beta_I \le b$ and $c \le \beta_J \le d$) are overlapped, then either one can fire with the following probability,*

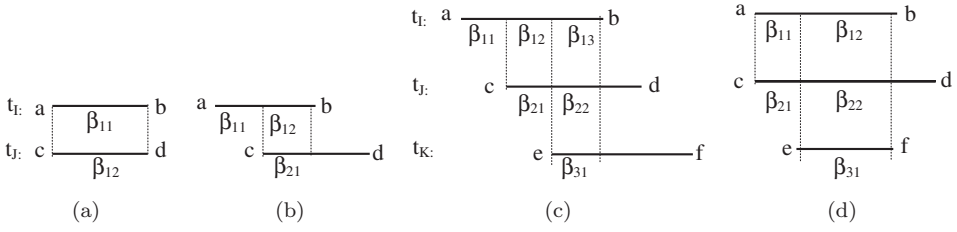


Fig. 1. Distinct ways for interval relations (t_I, t_J, t_K: denote the enabled transitions in the current state, and β denotes the time interval).

(*see* Figs. 1(a) and 1(b)).

$$\text{Prob}(t_I) = \int_{\beta_I} f_I(\beta_I)[1 - F_J(x)]dx\,,$$

$$\text{Prob}(t_J) = \int_{\beta_J} f_J(\beta_J)[1 - F_I(x)]dx\,.$$

The proof of this theorem is found in Ref. 28.

Definition 8. Let us consider that the overlapped interval set $\beta_n((\tau_{\min})_n \le \beta_n \le (\tau_{\max})_n)$ shown in a state S_i is associated with the enabled transition set α_n. Then, the *sojourn time SJ* for n enabled transitions can be defined as follows:

$$SJ \text{ of } \alpha_n \text{ in the state } S_i = [(\tau_{\min})_{\min} \text{ of } \alpha_n, (\tau_{\max})_{\min} \text{ of } \alpha_n]\,.$$

Definition 9. The *time interval* θ of the fireable transition $t_I \in \alpha_n$ $(I = 1, 2, \ldots, n)$ from a state S_i can be defined as follows:

$$\tau_{\min} \text{ of } t_I \le \theta \le (\tau_{\max})_{\min} \text{ of } \alpha_n\,.$$

Definition 10. By using Theorem 1 and Definitions 8 and 9, we can easily build the *algorithm* of the proposed *firing* (*transition*) *probability method*. This method achieves an accurate characterization for the firing behavior of concurrent and conflicting RTTs.

Step 1. (a) Use Definition 8 to determine the smallest of $\tau_{\min}$ and $\tau_{\max}$ for all the n enabled transitions shown in the desired state S_i.

(b) Sort the time intervals of β_I according to the ascending order of their $\tau_{\min}$ in the state S_i. Consider the least $\tau_{\min}$ in the state S_i is associated with a transition t_I.

(c) Use Definition 9 to determine the firing time of each enabled transition in the state S_i. Then, divide the time interval β_I that is associated with t_I into k sub-intervals as follows [see Figs. 1(c) and 1(d)].

$$\begin{aligned}\beta_{I1}, \beta_{I2}, \ldots, \beta_{I(k-1)}, \beta_{Ik} &= [(\tau_{I1})_{\min}, (\tau_{I2})_{\min}], [(\tau_{I2})_{\min}, (\tau_{I3})_{\min}], \ldots, \\ &\quad [(\tau_{I(k-1)})_{\min}, (\tau_{Ik})_{\min}], [(\tau_{Ik})_{\min}, (\tau_{\max})_{\min} \text{ of } \alpha_n]\,.\end{aligned}$$

Step 2. Use the idea of Step 1(c) to divide each time interval $\beta_I \in \beta_n$ $(I = 1, 2, \ldots, n)$ associated with its enabled transition $t_I \in \alpha_n$ $(I = 1, 2, \ldots, n)$ into k sub-intervals [see Figs. 1(c) and 1(d)].

For $I = 1$ to n

For $u = 1$ to k

If $u < k$, then $\beta_{Iu} = [(\tau_{Iu})_{\min}, (\tau_{I(u+1)})_{\min}]$

If $u = k$, then $\beta_{Iu} = [(\tau_{Iu})_{\min}, (\tau_{\max})_{\min}$ of $\alpha_n]$.

Step 3. Based on Theorem 1, we have developed the following general formal expression for calculating the firing probability of each firable transition indicated in the state S_i.

For $I = 1$ to n

$$\text{Prob}(t_I) = \left[\sum_{1 \le u \le k} \int_{\beta_{Iu}} f_I(x) (\Pi_{1 \le v \le n, v \ne I} [1 - F_v(x)]) dx \right]_{\textbf{cond.}}$$

cond. : If the time interval of t_I is overlapped with the time interval of t_v

Then $F_v(x) = [x - (\tau_v)_{\min}] / [(\tau_v)_{\max} - (\tau_v)_{\min}]$

$f_I(x) = 1 / [(\tau_I)_{\max} - (\tau_I)_{\min}]$

Else $F_v(x) =$ zero

$f_I(x) = 1 / [(\tau_I)_{\max} - (\tau_I)_{\min}]$

where n is the number of concurrent or conflicting overlapped transitions. k is the number of sub-intervals in the time interval of the transition τ_I that overlapped with the time intervals of the other transitions. In the next section, we use some examples to illustrate the mechanism of handling the overlapped intervals.

Definition 11. A state S of the R-net can be defined as a tuple (M, NI, RI, FS, SJ), where M is the marking function, NI is the new firing interval function, RI is the remaining firing interval function, FS is the actual firing transition function, and SJ is the sojourn time function.

Definition 12. By reaching a state S_j from a state S_i, the RI_j function computes the remaining time intervals (say, β_{n-1}) for all other transitions (say, α_{n-1}) that continue their firing in the state S_j, as follows:

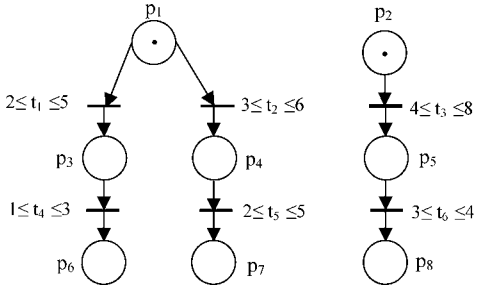
$$\beta_{n-1} \text{ of } \alpha_{n-1} = [\max(0, ((\tau_{\min})_{n-1} - \theta_{\max}), ((\tau_{\max})_{n-1} - \theta_{\min}))] .$$

2.1. *R-nets examples*

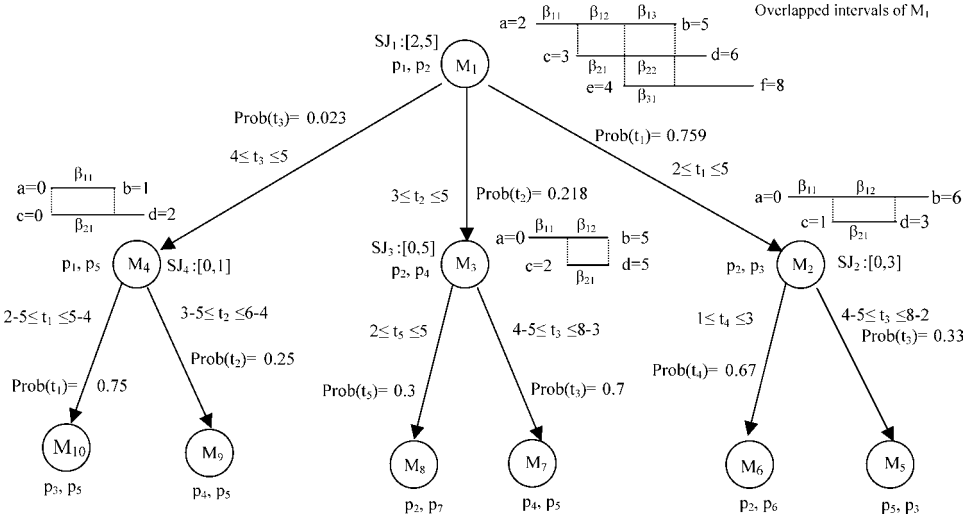
In this section, we develop two R-net models to illustrate the solution steps and the verification processes of the R-net technique.

2.1.1. *Example* 1

In this example, we illustrate the modeling mechanism of the R-net technique as shown in Fig. 2(a). Then, we illustrate the calculation processes of the Markov chain derived from the developed model under the R-net firing rules as shown in Fig. 2(b). In Fig. 2, we should remark the following interesting points.



(a) An example of R-net model.



(b) Markov processes of the R-net model.

From M_2 to M_5: Prob[t_3]= (d+c-2a)/ 2(b-a) = 0.33

From M_2 to M_6: Prob[t_4]= (2b -d-c)/ 2(b-a) = 0.67

From M_4 to M_9: Prob[t_2]= (b-a)/ 2(d-c) = 0.25

From M_4 to M_{10}: Prob[t_1]= (2d-b-a)/ 2(d-a) = 0.75

From M_3 to M_7: Prob[t_3]= $(2bd-2ad+2ac-b^2-c^2) / 2(b-a)(d-c) = 0.7$

From M_3 to M_8: Prob[t_5]= $(b-c)^2 / 2(b-a)(d-c) = 0.3$

(c) Calculation of the transition state probabilities of the Markov processes shown in Fig. (b).

Fig. 2. An illustration of solution steps of the R-net modeling technique.

- The state configuration of the R-net sub-model of Fig. 2(a) is shown in Fig. 2(b) as follows:

S_1:	S_2:	S_3:	S_4:
M_1: p_1, p_2	M_2: p_2, p_3	M_3: p_2, p_4	M_4: p_1, p_5
NI_1: $2 \le \beta_1 \le 5$	NI_2: $1 \le \beta_4 \le 3$	NI_3: $2 \le \beta_5 \le 5$	NI_4: $3 \le \beta_6 \le 4$
$: 3 \le \beta_2 \le 6$			
$: 4 \le \beta_3 \le 8$			
RI_1: 0	RI_2: $0 \le \beta_3 \le 6$	RI_3: $0 \le \beta_3 \le 5$	RI_4: $0 \le \beta_1 \le 1$
			$: 0 \le \beta_2 \le 2$
FS_1: $(t_1, \times t_2)//t_3$	FS_2: $t_3//t_4$	FS_3: $t_3//t_5$	FS_4: $t_1, \times t_2$
SJ_1: $[2, 5]$	SJ_2: $[0, 3]$	SJ_3: $[0, 5]$	SJ_4: $[0, 1]$

- In the function FS_i, the mark // denotes concurrency behavior and the mark $\times$ denotes conflicting behavior.
- In Fig. 2(b) the transition probabilities from a state S_1 to states S_2, S_3 and S_4 are calculated based on both Theorem 1 and the overlapped intervals associated with a state S_1 as follows:

$$\begin{aligned}\text{Prob}\,(t_1) &= \sum_{1\le u\le 3} \int_{\beta_{1u}} f_1(x)(\Pi_{1\le v\le 3, v\ne I}[1 - F_v(x)])dx \\ &= \int_{\beta_{11}} f_1(x)dx + \int_{\beta_{12}} f_1(x)[1 - F_2(x)]dx \\ &\quad + \int_{\beta_{13}} f_1(x)[1 - F_2(x)][1 - F_3(x)]dx\,,\end{aligned}$$

where $a \le \beta_{11} \le c, c \le \beta_{12} \le e, e \le \beta_{13} \le b$

$$\begin{aligned}\text{Prob}(t_2) &= \sum_{1\le u\le 2} \int_{\beta_{2u}} f_2(x)(\Pi_{1\le v\le 3, v\ne I}[1 - F_v(x)])dx \\ &= \int_{\beta_{21}} f_2(x)[1 - F_1(x)]dx \\ &\quad + \int_{\beta_{22}} f_2(x)[1 - F_1(x)][1 - F_3(x)]dx\,,\end{aligned}$$

where $c \le \beta_{21} \le e$, $e \le \beta_{22} \le b$

$$\begin{aligned}\text{Prob}(t_3) &= \sum_{1\le u\le 1} \int_{\beta_{3u}} f_3(x)(\Pi_{1\le v\le 3, v\ne I}[1 - F_v(x)])dx \\ &= \int_{\beta_{31}} f_3(x)[1 - F_1(x)][1 - F_2(x)]dx\,,\end{aligned}$$

where $e \le \beta_{31} \le b$.

In these equations, we should remark the following.

$$f_1(t) = \frac{1}{(b-a)}, \quad F_1(t) = \frac{(t-a)}{(b-a)}, \quad f_2(t) = \frac{1}{(d-c)},$$

$$F_2(t) = \frac{(t-c)}{(d-c)}, \quad f_3(t) = \frac{1}{(f-e)}, \quad \text{and} \quad F_3(t) = \frac{(t-e)}{(f-e)}.$$

- In Fig. 2(c), we illustrate the calculation of the transition probabilities of the other states that are shown in Fig. 2(b).

2.1.2. *Example* 2

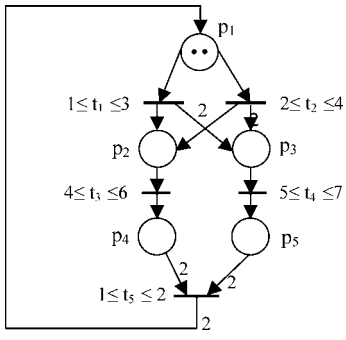
In Fig. 3(a), we describe a complete R-net model with its time intervals. The structure of this model incorporates the parallel, sequential, conflict and synchronous events that can occur in the real-time computer systems. The R-net technique is able to model such events as follows. Transitions t_1 and t_2 model the conflict (decision, choice) event. Transitions t_3 and t_4 model the concurrent (parallel) event. Transition t_5 models the synchronization event. In Fig. 3(b), we describe the complete Markov chain of the model shown in Fig. 3(a). This description is like that of Figs. 2(b) and 2(c). To verify from the correctness of the developed R-net model, the following conditions must be satisfied in its Markov processes.

(1) An R-net model is said to be k-bounded if the number of tokens in each place does not exceed a finite number k for any marking M_i reachable from the initial marking M_0.
(2) An R-net model is said to be live (guarantees deadlock-free operation) if every transition in the model is fireable.
(3) An R-net model is said to be reversible if, for each marking M_i in the Markov process of this model, M_0 is reachable from M_i. Thus, in a reversible net one can always get back to the initial state M_0.
(4) An R-net model is said to be persistent if, for any two enabled transitions (t_i, t_j) the firing of one transition (t_i) will not disable the other (t_j). A transition in a persistent net, once it is enabled, will stay enabled until it fires. Persistency is closely related to conflict-free nets.

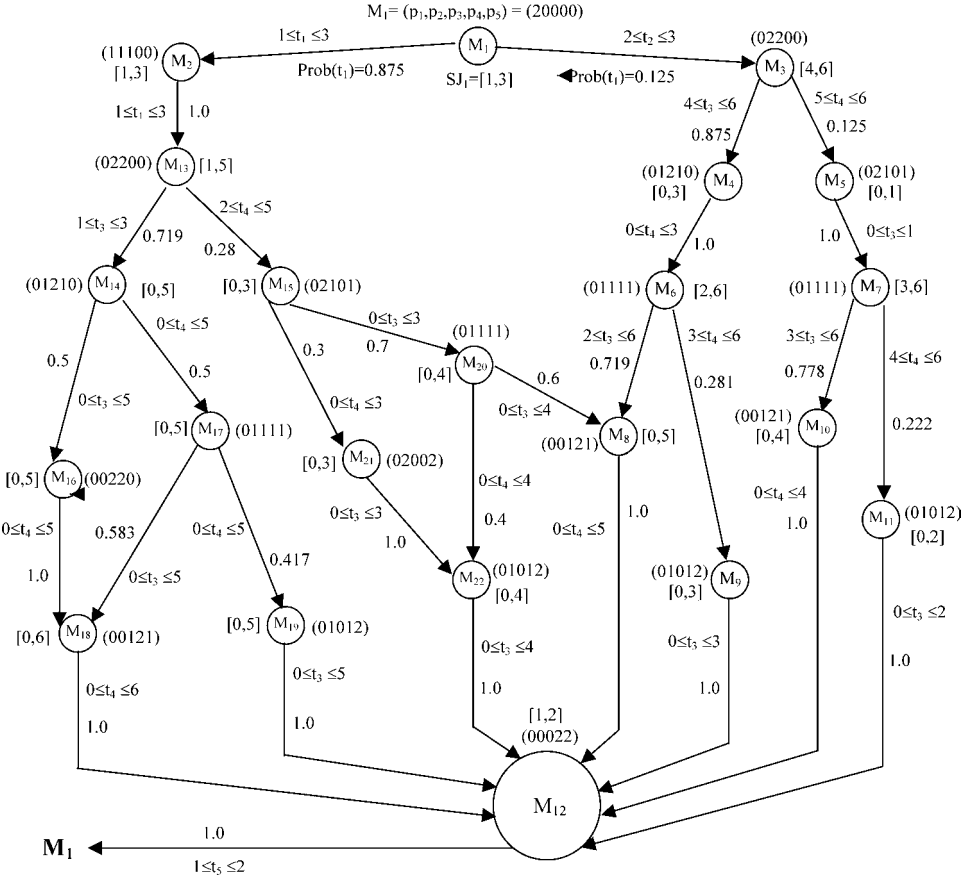
In Fig. 3(b), all these conditions are satisfied. It is interesting to note that the solving steps, firing rules, and verification processes of the R-net technique have been automated through our R-NET package.[28]

2.2. *Solution technique for R-nets*

Since, the R-net comprises the ZTT and RTT transitions, the states of this net can be classified into the flash and exact states, respectively. The behavior of the reachability set of R-net model is recognized as Embedded Markov Chain (EMC).[28] Thus, the EMC comprises both flash and exact states. Each flash state (i.e., time interval [0, 0]) in the EMC can be eliminated (because it represents the logical



(a) An example of a complete R-net model.



(b) Markov processes of the R-net model shown in Fig. (a).

Fig. 3. An illustration of solution steps and verification processes of the R-net model shown in (a).

behavior of the modeled system) as follows. Since the probability of firing the transition $t_i \in \mathrm{ZTT}$ from the flash state is one, the state-transition probability from the flash state to its next state is also one. Thus, we can simply remove the flash state by directly connecting its preceding state to its next state. After removing the flash state from the EMC, we call it a *reduced* EMC (REMC). A REMC comprises only "exact" states.

The stationary embedded probabilities $X(S)$ of the REMC can be obtained from the state-transition probabilities $Q(S_i, S_j)$ by solving a system of simultaneous linear equations.

$$\left.\begin{aligned} \sum_{1\le j\le N} Q(S_i, S_j)X(S_j) = X(S_i), \\ \sum_{1\le i\le N} X(S_i) = 1, \end{aligned}\right\} \quad i = 1, 2, \ldots, N$$

where N denotes the number of "exact" states in the REMC.

Let ASJ_i be the average sojourn time in a state S_i for α_n transitions. ASJ_i can be calculated as follows:

$$\mathrm{ASJ}_i = \frac{[(\tau_{\min})_{\min} + (\tau_{\max})_{\min}] \text{ of } \alpha_n}{2}.$$

The steady state probabilities of the REMC $(\pi_1, \pi_2, \ldots, \pi_N)$ are calculated as follows:

$$\pi_i = \pi(S_i) = \frac{(X(S_i) \times \mathrm{ASJ}(S_i))}{\sum_{1\le j\le N} X(S_j)}.$$

From these steady probabilities, one can compute several generic performance measures.[28] The calculation processes of the steady state probabilities of the derived Markov chain from the developed R-net model have been automated through our R-NET package.[28]

3. MOD Architecture

Several architectures and configurations for efficient storage, reliable storage, and delivery of multimedia data have been proposed and utilized such as MPEG-1, MPEG-2, MPEG-4, H.261, Berkeley, INDEO, Oracle media system, Microsoft tiger multimedia file system, Hong Kong telecom interactive multimedia service system.[1,2,5,8–10,14,30] The main objective of our work is to develop a modeling framework has the capability of describing, analyzing, and studying such architectures in an easy and accurate way. Therefore, we have adopted the basic multimedia aspects of such architectures in the distributed MOD service system shown in Fig. 4. The distributed MOD system consists of: (i) a central multimedia server, (ii) a wide area network, (iii) a group of Local Multimedia Servers (LMS) (LMS: $\mathrm{LMS}_1, \mathrm{LMS}_2, \ldots, \mathrm{LMS}_n$) and (v) a group of User Display Equipment (UDE) (UDE:

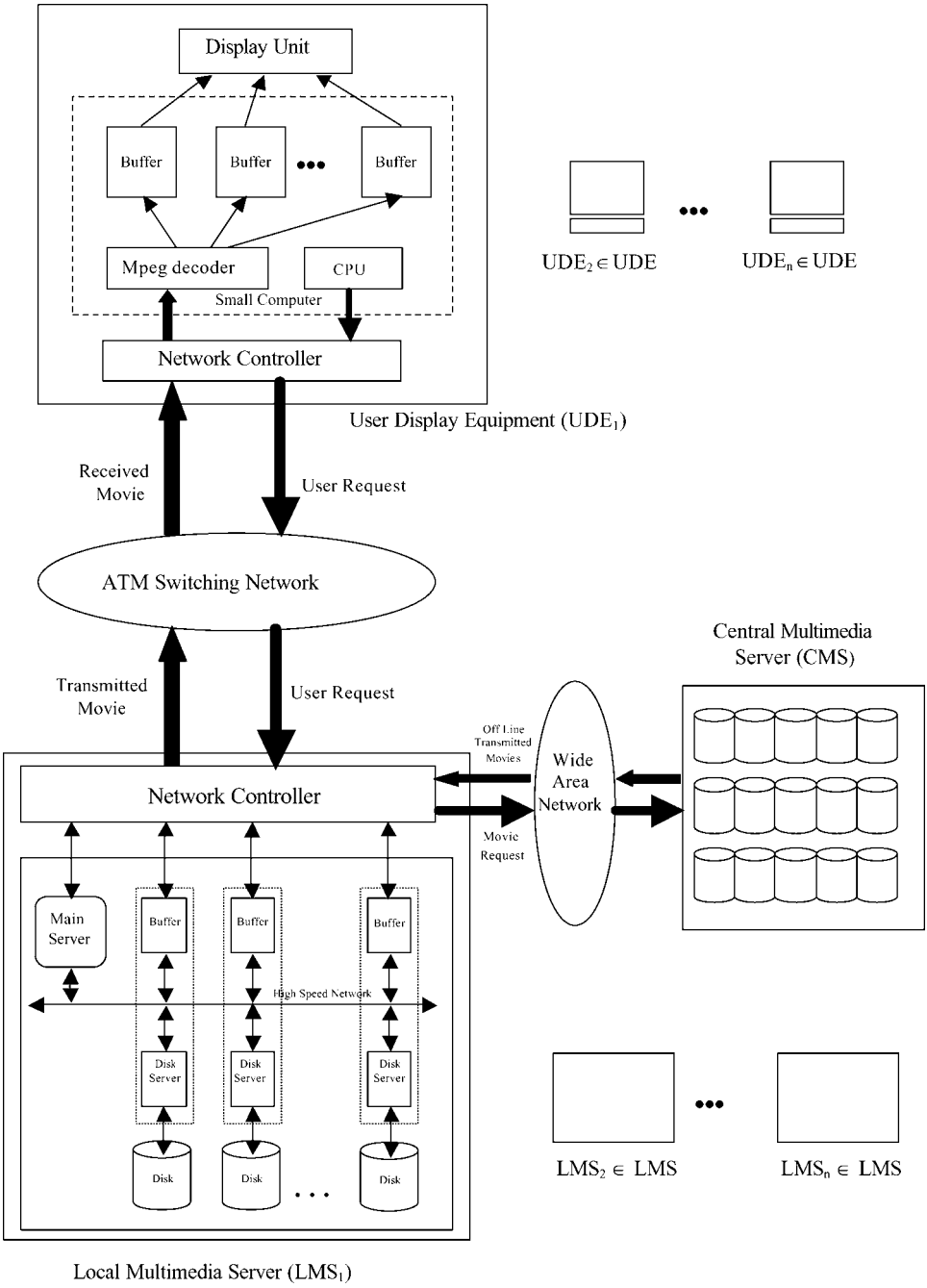


Fig. 4. MOD architecture.

UDE_1, $UDE_2, \ldots, UDE_m$). Each $LMS_i \in LMS$ is connected through an ATM interconnection network to its own group of UDE. Thus, the MOD system illustrates the main features of the available multimedia systems.[2,8,10,12,30] A brief description of the MOD components is given later.

3.1. *User Display Equipment*

The UDE is a group of display units. Each $UDE_j \in UDE$ can be represented as a computer has the capability of sending/receiving multimedia user request(s) to/from the $LMS_i \in LMS$.

3.2. *Central Multimedia Server*

The CMS consists of a main server, a network controller, and hundreds of gigabyte disks. These disks are used as an archive containing all the original movie files that can be requested by the users. The CMS uses a high-speed network (e.g., ATM switching technology[5,12,30]) to send copies of these movie files to each $LMS_i \in LMS$.

3.3. *Local Multimedia Servers*

Each $LMS_i \in LMS$ is a system which comprises a main server, a network controller, a high-speed network, a group of disk servers (DS) (DS: $DS_1, DS_2, \ldots, DS_n$), a group of buffers (BF) (BF: $BF_1, BF_2, \ldots, BF_n$), and a group of storage disk arrays.[6,8,10–13] Each storage disk array (SDA) (SDA: $SDA_1, SDA_2, \ldots, SDA_k$) has the capability of storing a subset of the total set of movie files. Each disk server $DS_x \in DS$ has a local storage disk $SDA_k \in SDA$ and a local buffer $BF_i \in BF$. The network controller of $LMS_i \in LMS$ is responsible for receiving the incoming multimedia user requests from the $UDE_j \in UDE$ and sending them to its main server. The main server has the capability of controlling and accessing any disk server $DS_x \in DS$ through a high-speed network. In this way, the various storage disks of LMS are managed separately, from both the logical and physical point of view.

3.3.1. *Storing process of the movie files*

The transfer of movie files from the CMS to each disk array in the $LMS_i \in LMS$ is performed based on off-line approach[31] as follows. The network controller of each $LMS_i \in LMS$ receives some of popular movie files from the CMS. The network controller, then, forwards these movie files to its main server. The main server splits and distributes the received popular movie files among all storage disks in its disk array. In each $LMS_i \in LMS$, the main server splits each data file to be stored into macro blocks of a pre-established size, called streams. In turn, streams are split into blocks called segments, which are distributed over the various storage disks of $LMS_i \in LMS$. In this way, the main server has the capability of reading these segments concurrently from each storage disk $SDA_k \in SDA$ in the disk array of $LMS_i \in LMS$. This parallel retrieve-ability aspect represents the important advantage of storing each movie file as segments over the disk array. According to this

storing process, the main server of $\text{LMS}_i \in \text{LMS}$ serves the user request by retrieving the segments of the desired movie file and reassembling the data segments into contiguous blocks or streams to satisfy the multimedia request.

3.3.2. *Serving process of the user request*

To know the allocation places of storing the segments of each movie file over the various storage disks of the desired array, the main server of each $\text{LMS}_i \in \text{LMS}$ constructs an *allocation map* for each stored movie file into its disk array (see the part of LMS_1 in Fig. 4). Once the main server of $\text{LMS}_i \in \text{LMS}$ receives the multimedia requests from its network controller, it starts to serve these requests as follows. It downloads the desired allocation map of each target movie file to all its disk servers. Then, the main server uses the allocation map to schedule the *deadline times of the requested movie files* (TRM). Subsequently, the main server executes the accepted user requests. This schedule process can be done by using one of the available disk scheduling algorithms.[9,11,14–16] For example, the Earliest Deadline First (EDF) algorithm and the Scan-EDF algorithm. Such algorithms avoid starvation of service to requests. In fact, the main aim of disk scheduling algorithms is to manage multimedia traffic and allow performance improvements (i.e., maximize throughput) by meeting all the deadline times of the user requests.

The following equation illustrates the different parameters that effect on the *deadline times of the requested movie file* j (TRM_j).

$$\text{TRM}_j = \text{TRS} + \text{TSB} + \text{TAS} + \text{TSS}$$

where

TRM_j: is the deadline time of the requested movie file j. The user has to receive his requested movie file j before the TRM_j time.

TRS: is the deadline time of retrieving the desired segments. It denotes the times estimated by the disk servers (DS) of $\text{LMS}_i \in \text{LMS}$ to retrieve all the segments of the desired movie file from their disk array (SDA). By using parallel retrieving approach, $\text{TRS} = \max\{\text{TRS}_1, \text{TRS}_2, \ldots, \text{TRS}_n\}$. The parameter n represents the number of the desired segments. From this equation, the disk server $\text{DS}_j \in \text{DS}$ of $\text{LMS}_i \in \text{LMS}$ spends $\text{TRS}_n \in \text{TRS}$ time to retrieve a segment-n from its storage disk $\text{SDA}_k \in \text{SDA}$.

TSB: is the deadline time of storing the desired segments into buffers. It denotes the time estimated by the DS of $\text{LMS}_i \in \text{LMS}$ to store the retrieved n-segments of the desired movie file into their local buffers. By using parallel storing approach, $\text{TSB} = \max\{\text{TSB}_1, \text{TSB}_2, \ldots, \text{TSB}_n\}$. The parameter n represents the number of the desired segments. From this equation, the disk server $\text{DS}_j \in \text{DS}$ of $\text{LMS}_i \in \text{LMS}$ spends $\text{TSB}_n \in \text{TSB}$ time to store a retrieved segment-n into its local buffer.

TAS: is the deadline time of reassembling the desired segments. It denotes the time estimated by the main server of $\text{LMS}_i \in \text{LMS}$ to reassemble all the retrieved segments of the desired user request into composed

streams. By using parallel reassembling (RSS) approach, TAS = max{RSS-stream$_{-1}$, RSS-stream$_{-2}$, ..., RSS-stream$_{-m}$}. The parameter m represents the number of the desired streams. These streams are constituted from the composed segments of the requested movie file.

TSS: is the deadline time of sending the desired streams. It denotes the time estimated by the main server of LMS$_i \in$ LMS to send all streams to the requested user through its network controller. Sending processes of streams can be done by using pipeline approach as follows: TSS = max{send-stream$_{-1}$, send-stream$_{-2}$, ..., send-stream$_{-m}$}.

As a conclusion, each LMS$_i \in$ LMS starts to serve the user request by collecting the desired segments of the requested movie file. Then, it reassembles these streams. Finally, it sends these streams to the requested user within the TRM$_j$ time.

4. R-net Model for a Simple MOD Architecture

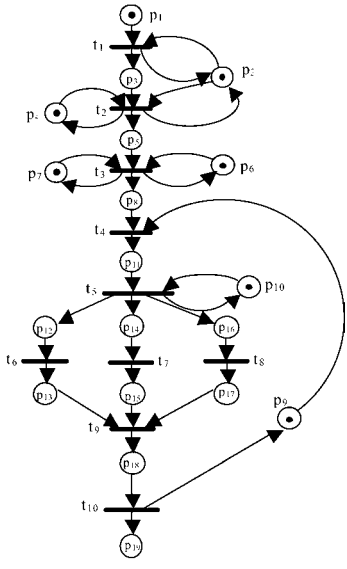
To illustrate the dynamic behavior of the MOD system shown in Fig. 4, we have used the R-net modeling technique. The R-net technique has the capability of describing the interaction between the real-time processes that are required for multimedia presentation. By incorporating such R-net aspects into the developed model, we can easily build an R-net multimedia model with the ability to capture the time dependencies of multimedia data presentation, indicate the sequential and parallel multimedia activities, and describe how to store, retrieve, schedule, and communicate multimedia data. To illustrate the mechanism of constructing an R-net model for the complicated MOD architecture shown in Fig. 4, we have developed two models. The first R-net model describes the dynamic behavior of the MOD architecture in a simple way. The second R-net model describes the dynamic behavior of an exact MOD architecture. In this section, we discuss the simple R-net MOD model. In the next section, we discuss the exact R-net MOD model.

Based on the following concept, we have developed a simple R-net model for the MOD architecture of Fig. 4. In this model, a user UDE$_j \in$ UDE sends a multimedia request to the main server of LMS$_i \in$ LMS. This LMS$_i \in$ LMS has three disk servers DS$_x$, DS$_y$, and DS$_z \in$ DS with their storage disks SDA$_{k1}$, SDA$_{k2}$ and SDA$_{k3} \in$ SDA and their local buffers BF$_1$, BF$_2$ and BF$_3 \in$ BF, respectively. The requested movie file has been stored in the storage disks SDA$_{k1}$, SDA$_{k2}$ and SDA$_{k3} \in$ SDA, as explained in Sec. 3.3.1. The disk server DS$_x \in$ DS handles the main part of the requested movie file. The disk server DS$_x \in$ DS sends control signals to the disk servers DS$_y \in$ DS and DS$_z \in$ DS to retrieve the remaining parts of the requested movie file. For better understanding, we have used the *modular approach* to develop such R-net model. Thus, the developed R-net MOD model consists of the following sub-models.

4.1. *Sub-model of the receiving mechanism of the main server*

The R-net model shown in Fig. 5 describes, how, the following processes can be implemented. The main server of LMS$_i \in$ LMS receives the user request through the

p_1: A user submission.
p_2: An $UDE_j \in UDE$.
t_1: A user uses his $UDE_j \in UDE$ to prepare his requested movie file.
p_3: A network controller of $UDE_j \in UDE$ receives the user request.
p_4: An ATM switching network.
t_2: A network controller of $UDE_j \in UDE$ transmits a user request to the desired $LMS_i \in LMS$ through the ATM switching network.
p_5: A user request is passing through the ATM switching network.
p_6: A network controller of $LMS_i \in LMS$.
p_7: An $LMS_i \in LMS$ is idle.
t_3: A network controller receives the user request.
p_8: A network controller is ready to send the user request to its main server.
p_9: The main server is idle.
t_4: The main server receives the user request.
p_{10}: The map allocation of the requested movie file.
p_{11}: The main server downloads the map allocation of the requested movie file. The main server uses this map to determine the group of storage disks that contain the required segments of the requested movie file.
t_5: The main server sends control signals to the disk servers (DS: DS_x, DS_y, and DS_z) of the desired storage disks (SDA: SDA_{k1}, SDA_{k2}, and SDA_{k3}). Then, each disk server checks the queue status of its attached storage disk.
p_{12}: The main server interrogation being received by $DS_x \in DS$.
p_{14}: The main server interrogation being received by $DS_y \in DS$.
p_{16}: The main server interrogation being received by $DS_z \in DS$.
t_6: A disk server $DS_x \in DS$ sends a reply signal to its main server.
t_7: A disk server $DS_y \in DS$ sends a reply signal to its main server.
t_8: A disk server $DS_z \in DS$ sends a reply signal to its main server.
p_{13}: The main server receives a reply signal from $DS_x \in DS$.
p_{15}: The main server receives a reply signal from $DS_y \in DS$.
p_{17}: The main server receives a reply signal from $DS_z \in DS$.
t_9: The main server performs a comparison process on the queue status of the disk servers DS_x, DS_y, and DS_z.
p_{18}: A comparison result. Based on this comparison result, the main server decides which disk server has a fewer queues. The selected disk server will be used to handle the user request(s).
t_{10}: The main server selects a disk server $DS_x \in DS$ to handle the desired user request(s). To allow the proposed model to handle more than one user request, the main server of $LMS_i \in LMS$ will proceed to an idle state (after t_{10} fires) to become ready to handle another user request.
p_{19}: The state of queued requests.



Time Intervals (ms)
$t_1 = [3,5]$, $t_2 = [4,7]$, $t_3 = [0,1]$,
$t_4 = [0,1]$, $t_5 = [0,1]$, $t_6 = [1,2]$,
$t_7 = [1,2$, $t_8 = [1,2]$, $t_9 = [2,3]$,
$t_{10} = [1,3]$

Fig. 5. R-net model of the main server of $LMS_i \in LMS$.

ATM switching network. The main server of $LMS_i \in LMS$ uses its map allocation to determine the disk servers and their storage disks that contain the required segments of the requested movie file. The main server communicates with its disk servers DS_x, DS_y and $DS_z \in DS$. The disk servers DS_x, DS_y and $DS_z \in DS$ communicate with their storage disks SDA_{k1}, SDA_{k2} and $SDA_{k3} \in SDA$, respectively. Based on these communication processes, the main server performs a comparison process on the queue status of the disk servers SDA_{k1}, SDA_{k2} and $SDA_{k3} \in SDA$. As a comparison result, the main server selects the disk server and its storage disk that has a fewer queues to handle the user requests.

4.2. *Sub-model of executing the EDF scheduling algorithm by the disk server $DS_x \in DS$*

From the user requests waiting to be served, the EDF algorithm selects the one with the earliest deadline time and serves it. If various requests have the same deadline time, they are served on FIFO policy. As pointed out, the main server of $LMS_i \in$ LMS receives the accepted user requests from its network controller. Then, the main server divides each user request into multiple segments (i.e., n-segments) and sends the segment requests (SR) (SR: $SR_1, SR_2, \ldots, SR_n$) with their deadline times (TRS: $TRS_1, TRS_2, \ldots, TRS_n$) to all its DS (DS: $DS_1, DS_2, \ldots, DS_m$) concurrently. Subsequently, each disk server $DS_x \in$ DS accesses its storage disk to retrieve the requested segment(s). To understand the schedule policy of the EDF algorithm, we use the following example.

Example. Let us assume that there is a queue of unsorted segment requests at the disk server $DS_x \in$ DS of $LMS_i \in$ LMS. Such requests are SR_1, SR_2, SR_3, SR_4 and SR_5 with their deadline times TRS: $TRS_1 = 700$, $TRS_2 = 800$, $TRS_3 = 600$, $TRS_4 = 500$ and $TRS_5 = 500$, respectively. The disk server $DS_x \in$ DS uses the EDF algorithm to perform the following steps before serving any of them.

Step 1. The disk server $DS_x \in$ DS starts to arrange the unsorted segment requests in ascending order according to their deadline times. In this example, the $DS_x \in$ DS uses the EDF policy to schedule the segment requests as follows: SR_4, SR_5, SR_3, SR_1 and SR_2.

Step 2. If some of the segment requests have equal deadline times, then the EDF algorithm uses the FIFO technique to select the one to be served before the others. In this example, the EDF uses the FIFO technique to handle the segment requests SR_4 and SR_5.

Step 3. The disk server $DS_x \in$ DS serves the segment requests according to their arrangement. In this example, the $DS_x \in$ DS uses the EDF technique to serve the segment requests SR_3, SR_1 and SR_2.

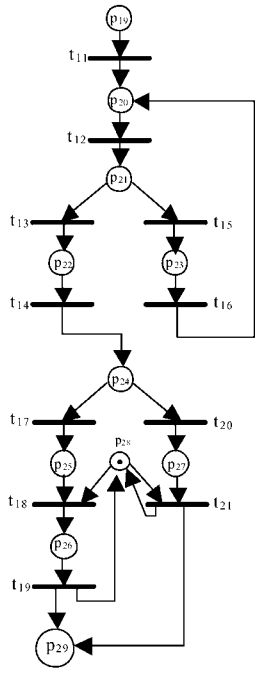
The R-net model shown in Fig. 6 describes the dynamic behavior of the disk server $DS_x \in$ DS when it applies the steps of the EDF algorithm on the unsorted segment requests.

4.3. *Sub-model of executing the scan-EDF scheduling algorithm by the disk server $DS_x \in DS$*

This algorithm represents a scan type of disk scheduling algorithm. In the Scan-EDF algorithm, each disk server $DS_x \in$ DS of $LMS_i \in$ LMS can handle its queue segment requests as follows.

If the segment requests do not have the same segment deadline times,
Then the segment requests can be served according to the EDF scheduling algorithm,

p_{19}: A disk server $DS_x \in DS$ of $LMS_i \in LMS$ starts to handle the unsorted segment requests that have been stored in its local buffer $BF_1 \in BF$.
t_{11}: A disk server $DS_x \in DS$ uses the EDF algorithm.
p_{20}: Executing the sorting process on the segment requests according to their *Deadline Times of Retrieving segments* (TRS).
t_{12}: A disk server $DS_x \in DS$ checks whether the scheduling processes are finished.
p_{21}: A disk server $DS_x \in DS$ performs a decision making.
t_{13}: Yes, the scheduled segment requests are available.
p_{22}: State of scheduled segment requests.
t_{14}: A disk server $DS_x \in DS$ checks if there are segment requests which have equal deadline times (TRS).
t_{15}: No, the scheduled segment requests are not available.
p_{23}: State of unscheduled segment requests.
t_{16}: A disk server $DS_x \in DS$ updates and reschedules the segment requests whose deadline times (TRS) cannot meet.
p_{24}: A disk server $DS_x \in DS$ performs a decision making.
t_{17}: A disk server $DS_x \in DS$ starts to handle the segment requests whose deadline times (TRS) are equal.
p_{25}: A disk server $DS_x \in DS$ applies the FIFO technique on the segment requests that have equal deadline times (TRS).
t_{18}: A disk server $DS_x \in DS$ completes the arrangement processes of segment requests according to the FIFO technique.
p_{26}: The segment requests are waiting to be processed.
t_{19}: A disk server $DS_x \in DS$ proceeds with a segment request j that has *earliest* deadline time ($ETRS_j$).
t_{20}: A disk server $DS_x \in DS$ starts to handle the segment requests that have different deadline times (TRS).
p_{27}: A disk server $DS_x \in DS$ arranges the segment requests according to their deadline times (TRS).
p_{28}: Permission state.
t_{21}: A disk server $DS_x \in DS$ completes the arrangement processes of segment requests according to their deadline times (TRS).
p_{29}: A disk server $DS_x \in DS$ prepares the segment requests that are ready to be retrieved from its storage disk.



Time Intervals : (ms)
$t_{11} = [5,8]$, $t_{12} = [0,1]$,
$t_{13} = [2,4]$, $t_{14} = [3,5]$,
$t_{15} = [2,4]$, $t_{16} = [1,3]$,
$t_{17} = [0,1]$, $t_{18} = [2,6]$,
$t_{19} = [1,2]$, $t_{20} = [0,1]$,
$t_{21} = [1,5]$

Fig. 6. R-net model of the EDF disk scheduling algorithm.

Else the segment requests can be served according to their track locations on the disk. This service can be done by *modifying* the *deadline time of retrieving segments* (TRS) through the following equation.

$$\mathrm{MTRS}_j = \mathrm{TRS}_j + (N_j/N_{\max} - 1)\,,$$

where

TRS_j: Deadline time of retrieving a segment request SR_j,
N_j: Track position of the data of a segment request SR_j,

p_{19}: A disk server $DS_x \in DS$ of $LMS_i \in LMS$ starts to handle the unsorted segment requests that have been stored in its local buffer.

t_{11}: A disk server $DS_x \in DS$ uses the EDF algorithm to sort the segment requests in ascending order according to their *Deadline Times of Retrieving Segments* (TRS).

p_{20}: A disk server $DS_x \in DS$ performs a decision-making process on the sorted segment requests.

t_{16}: A disk server $DS_x \in DS$ starts to handle the segment requests whose deadline times are *unequal*.

p_{22}: A disk server $DS_x \in DS$ performs the Earliest Deadline first algorithm on the unequal deadline requests.

t_{17}: The execution process of the Earliest Deadline first algorithm is finished.

t_{12}: A disk server $DS_x \in DS$ starts to handle the segment requests whose deadline times are *equal*.

p_{21}: A disk server $DS_x \in DS$ uses the Scan-EDF algorithm to know the following parameters:

(1). The maximum track number on the disk server $DS_x \in DS$ (say N_{max}); and

(2). The track position number (say N_h) for the data segment that should be retrieved.

p_{23}: Permission state.

t_{13}: A disk server $DS_x \in DS$ uses the deadline time of retrieving a segment j (TRS_j) to calculate the *modified* TRS_j ($MTRS_j$) as follows:

$$MTRS_j = TRS_j + (N_j / N_{max} - 1)$$

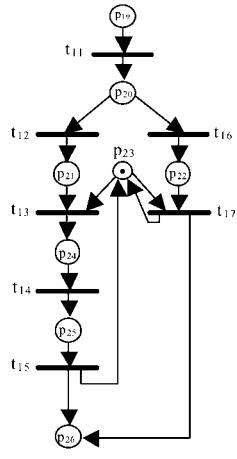
p_{24}: A disk server $DS_x \in DS$ collects the modified deadline times (MTRS) of the available segment requests.

t_{14}: A disk server $DS_x \in DS$ sorts the segment requests according to their *Earliest* MTRS (EMTRS).

p_{25}: State of sorted EMTRS.

t_{15}: A disk server $DS_x \in DS$ starts to handle a segment request j that has *earliest* deadline time ($EMTRS_j$).

p_{26}: A segment request j that has *earliest* deadline time ($EMTRS_j$) is waiting to proceed.



Time Intervals : (ms)
$t_{11} = [5,8]$, $t_{12} = [0,1]$,
$t_{13} = [2,4]$, $t_{14} = [3,5]$,
$t_{15} = [2,4]$, $t_{16} = [1,3]$,
$t_{17} = [0,1]$, $t_{18} = [2,6]$,
$t_{19} = [1,2]$, $t_{20} = [0,1]$,
$t_{21} = [1,5]$

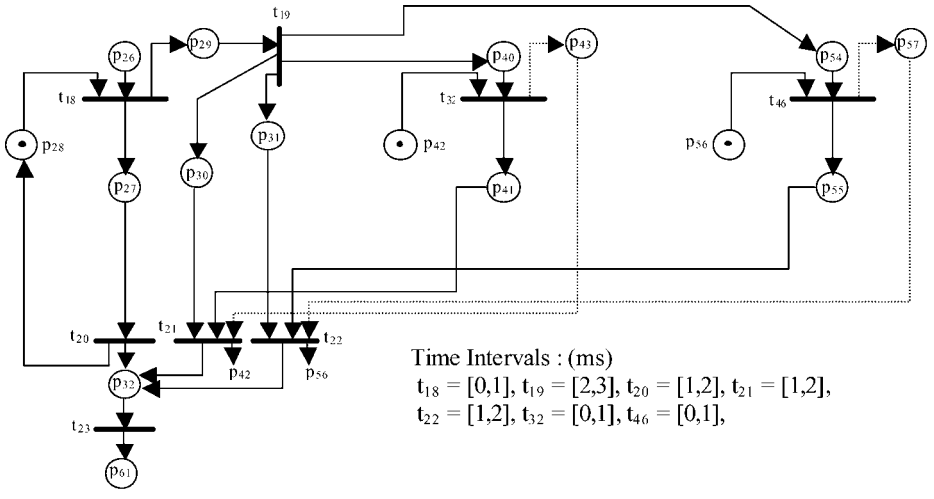
Fig. 7. R-net model representing Scan-EDF scheduling algorithm.

$N_{\max}$: Maximum track number on the disk,
MTRS_j: Modified deadline time of retrieving a segment request SR_j.

The R-net model shown in Fig. 7 describes the dynamic behavior of the disk server $DS_x \in DS$ when it performs the steps of the Scan-EDF algorithm on the unsorted segment requests.

4.4. *Sub-model of the retrieving mechanism of the disk server $DS_x \in DS$*

The R-net model shown in Fig. 8 illustrates the sequential and parallel retrieving processes of the requested movie files from the disk array architecture. Also,



p_{28}: The disk server $DS_x \in DS$ is idle.
p_{26}: The disk server $DS_x \in DS$ prepares the segment requests that are ready to be retrieved from its storage disk.
t_{18}: The disk server $DS_x \in DS$ starts to retrieve the segments that constitute the requested movie file.
p_{27}: The disk server $DS_x \in DS$ retrieves the desired segments from its storage disk $SDA_{k1} \in SDA$.
t_{20}: The disk server $DS_x \in DS$ receives into its local buffer $BF_1 \in BF$ the requested segments from its storage disk $SDA_{k1} \in SDA$.
p_{29}: The disk server $DS_x \in DS$ schedules the deadline times of the remaining segments of the requested movie file that are stored at the storage disks $SDA_{k2} \in SDA$ and $SDA_{k3} \in SDA$.
t_{19}: According to the scheduling deadline time process, the disk server $DS_x \in DS$ sends control signals to the disk servers $DS_y \in DS$ and $DS_z \in DS$ to collect the remaining segments of the requested movie file.
p_{31}: The disk server $DS_x \in DS$ controls the collection process of the remaining segments of the requested movie file from the disk server $DS_y \in DS$.
p_{30}: The disk server $DS_x \in DS$ controls the collection process of the remaining segments of the requested movie file from the disk server $DS_z \in DS$.
p_{42}: The disk server $DS_y \in DS$ is idle.
p_{40}: The disk server $DS_y \in DS$ receives a control signal from the disk server $DS_x \in DS$ to collect the remaining segments of the requested movie file.
t_{32}: The disk server $DS_y \in DS$ starts to retrieve the desired segments from its storage disk $SDA_{k2} \in SDA$. Then, the disk server transfers the retrieved segments to its local buffer $BF_2 \in BF$.
p_{41}: Retrieving process of the desired segments at the disk server $DS_y \in DS$ is in progress.
t_{21}: The disk server $DS_x \in DS$ receives into its local buffer $BF_1 \in BF$ the requested segments from the storage disk $SDA_{k2} \in SDA$ of the disk server $DS_y \in DS$.
p_{56}: The disk server $DS_z \in DS$ is idle.
p_{54}: The disk server $DS_z \in DS$ receives a control signal from the disk server $DS_x \in DS$ to collect the remaining segments of the requested movie file.
t_{46}: The disk server $DS_z \in DS$ starts to retrieve the desired segments from its storage disk $SDA_{k3} \in SDA$. Then, the disk server transfers the retrieved segments to its local buffer $BF_3 \in BF$.
p_{55}: Retrieving process of the desired segments at the disk server $DS_z \in DS$ is in progress.
t_{22}: The disk server $DS_x \in DS$ receives into its local buffer $BF_1 \in BF$ the requested segments from the storage disk $SDA_{k3} \in SDA$ of the disk server $DS_z \in DS$.
p_{32}: A local buffer $BF_1 \in BF$ of the disk server $DS_x \in DS$.
t_{23}: A disk server $DS_x \in DS$ arranges the collected segments according to the segments' annotation to make the required composed streams and sends them to the requested user.
p_{61}: The main server of $LMS_i \in LMS$ collects the composed streams of the desired movie file from its disk server $DS_x \in DS$.

Fig. 8. R-net model for describing retrieving processes of the desired segments of the requested movie file from the disk servers of $LMS_i \in LMS$: DS_x, DS_y and $DS_z \in DS$.

it illustrates how the retrieving operation of the requested movie file can be performed within the specified deadline time. In fact, this model describes how the following processes can be implemented. Let us assume that the $LMS_i \in LMS$ has three disk servers DS_x, DS_y and $DS_z \in DS$ with their storage disks SDA_{k1}, SDA_{k2} and $SDA_{k3} \in SDA$ and their local buffers BF_1, BF_2 and $BF_3 \in BF$, respectively. The disk server $DS_x \in DS$ has a fewer queues into its storage disk SDA_{k1}. Thus, the main server chooses the disk server $DS_x \in DS$ to handle the user request. Then, the main server allows the disk server $DS_x \in DS$ to retrieve the segments of the requested movie file from its storage disk $SDA_{k1} \in SDA$ and store the retrieved segments into its local buffer $BF_1 \in BF$. In parallel, the disk server $DS_x \in DS$ sends control signals to disk servers DS_y, and $DS_z \in DS$ to retrieve the remaining parts of the desired segments from their storage disks SDA_{k2}, and $SDA_{k3} \in SDA$ and to store them into their buffers BF_2, and $BF_3 \in BF$. Then, the disk server $DS_x \in DS$ collects and stores all the desired segments into its buffer $BF_1 \in BF$. Finally, the main server composes the segments of the requested movie file into streams.

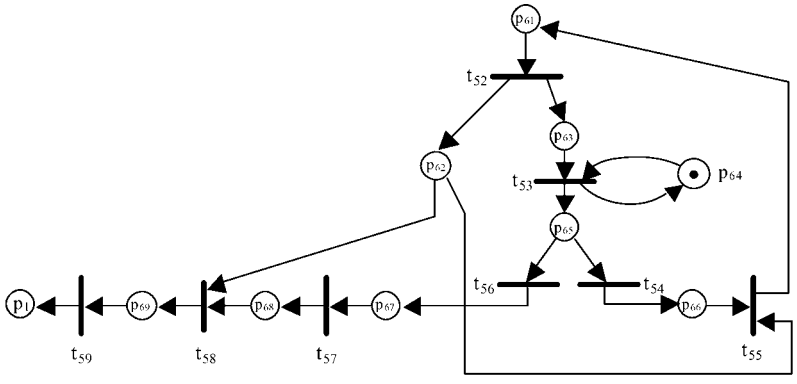
4.5. *Sub-model of the sending mechanism of the main server of $LMS_i \in LMS$*

The R-net model shown in Fig. 9 describes, how, the following processes can be implemented. The main server sends the composed streams to the requested $UDE_j \in UDE$ through the ATM switching network. The $UDE_j \in UDE$ sends +ACK message or –ACK message to the main server of $LMS_i \in LMS$. The requested user $UDE_j \in UDE$ receives and displays the composed streams of the requested movie file.

From the above-mentioned sub-models, it is interesting to note the following. Figures 5, 6, 8 and 9 describe the R-net MOD model when it executes the EDF disk scheduling algorithm. Figures 5, 7, 8 and 9 describe the R-net MOD model when it executes the Scan-EDF disk scheduling algorithm.

5. R-net Model for an Exact MOD Architecture

The R-net MOD model shown in Figs. 5, 6 (or 7), 8 and 9 illustrates the different multimedia processes that can occur when a user $UDE_j \in UDE$ accesses a disk server $DS_x \in DS$ of $LMS_i \in LMS$. We have developed this model in a simple and an easy way by using a modular technique to facilitate understanding how the exact R-net MOD model can be constructed. Thus, the R-net MOD model shown in Fig. 10 illustrates the retrieving, scheduling, synchronization, sequential, concurrent and communication multimedia processes that can occur between multiple users and multiple LMSs. In each $LMS_i \in LMS$, there are three disk servers DS_x, DS_y and $DS_z \in DS$ work concurrently to satisfy the requested movie files. In other words, these disk servers can work concurrently to retrieve the segments of three requested movie file at the same time. Based on a modular technique, we have constructed the R-net MOD model shown in Fig. 10. Throughout four sub-models incorporated



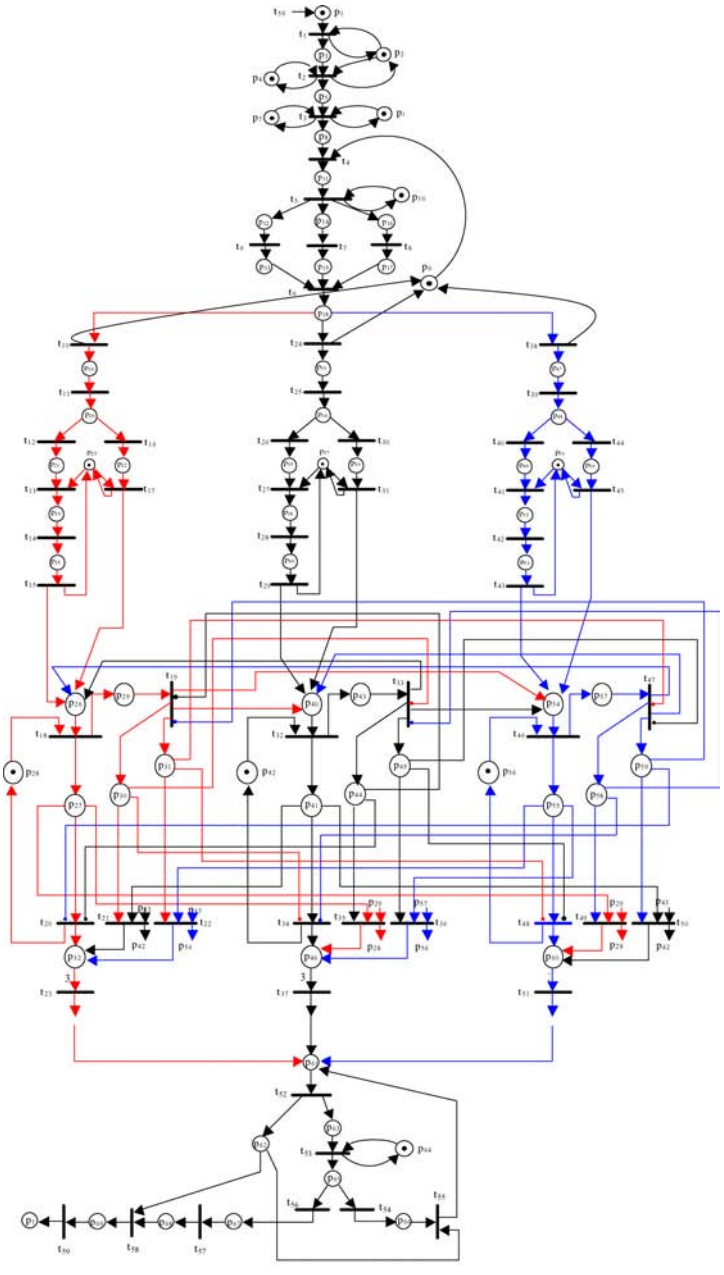
Time Intervals: (ms)
$t_{52} = [0,1]$, $t_{53} = [1,2]$, $t_{54} = [2,4]$, $t_{55} = [3,5]$,
$t_{56} = [1,2]$, $t_{57} = [3,5]$, $t_{58} = [4,7]$, $t_{59} = [2,6]$

p_{61}: The main server of $LMS_i \in$ LMS collects the composed streams of the desired movie file from its disk server $DS_x \in$ DS.
t_{52}: The main server starts to communicate with the requested $UDE_j \in$ UDE.
p_{62}: The main server keeps the composed streams in its buffer until it receives +ACK message from the requested $UDE_j \in$ UDE.
p_{63}: The communication with the requested $UDE_j \in$ UDE is in progress.
p_{64}: The requesting $UDE_j \in$ UDE is idle.
t_{53}: The requesting $UDE_j \in$ UDE receives a message from the main server of $LMS_i \in$ LMS.
p_{65}: The requesting $UDE_j \in$ UDE checks whether its buffer is empty.
t_{54}: End of checking the status of the buffer. The buffer at the requested $UDE_j \in$ UDE is full.
p_{66}: Sending process of –ACK message to the main server is in progress.
t_{55}: The main server receives –ACK message from the requested $UDE_j \in$ UDE.
t_{56}: End of checking the status of the buffer. The buffer at the requested $UDE_j \in$ UDE is empty.
p_{67}: Sending process of +ACK message to the main server is in progress.
t_{57}: The main server receives the +ACK message from the requested $UDE_j \in$ UDE.
p_{68}: The main server handles the +ACK message.
t_{58}: The main server sends the composed streams to the requested $UDE_j \in$ UDE.
p_{69}: Transmission process of the composed streams is in progress.
t_{59}: The requested $UDE_j \in$ UDE receives the composed streams of the requested mo vie file.

Fig. 9. R-net model for describing the communication processes between $LMS_i \in$ LMS and $UDE_j \in$ UDE.

in this model, we describe the following interesting multimedia processes that can occur when multiple users access the distributed MOD system of Fig. 4.

Sub-model-I describes the transmission processes of the requested movie files from multiple users to multiple LMSs through the ATM switching network. This sub-model starts from p_1 and t_1 to t_9 and p_{18}. The meanings of places and transitions of this sub-model are the same as those of Fig. 5. In this sub-model, we use the token properties of the Petri net technique to represent the access processes of



Time Intervals: (ms)
$t_1 = [3,5]$, $t_2 = [4,7]$, $t_3 = [0,1]$, $t_4 = [0,1]$, $t_5 = [0,1]$, $t_6 = [1,2]$, $t_7 = [1,2]$, $t_8 = [1,2]$, $t_9 = [2,3]$,
$t_{10} = [1,3]$, $t_{11} = [5,8]$, $t_{12} = [0,1]$, $t_{13} = [2,4]$, $t_{14} = [5,8]$, $t_{15} = [0,1]$, $t_{16} = [0,1]$, $t_{17} = [2,5]$,
$t_{18} = [0,1]$, $t_{19} = [2,3]$, $t_{20} = [1,2]$, $t_{21} = [1,2]$, $t_{22} = [1,2]$, $t_{23} = [2,5]$, $t_{24} = [1,3]$, $t_{25} = [5,8]$,
$t_{26} = [0,1]$, $t_{27} = [2,4]$, $t_{28} = [5,8]$, $t_{29} = [0,1]$, $t_{30} = [0,1]$, $t_{31} = [2,5]$, $t_{32} = [0,1]$, $t_{33} = [2,3]$,
$t_{34} = [1,2]$, $t_{35} = [1,2]$, $t_{36} = [1,2]$, $t_{37} = [2,5]$, $t_{38} = [1,3]$, $t_{39} = [5,8]$, $t_{40} = [0,1]$, $t_{41} = [2,4]$,
$t_{42} = [5,8]$, $t_{43} = [0,1]$, $t_{44} = [0,1]$, $t_{45} = [2,5]$, $t_{46} = [0,1]$, $t_{47} = [2,3]$, $t_{48} = [1,2]$, $t_{49} = [1,2]$,
$t_{50} = [1,2]$, $t_{51} = [2,5]$, $t_{52} = [0,1]$, $t_{53} = [1,2]$, $t_{54} = [2,4]$, $t_{55} = [3,5]$, $t_{56} = [1,2]$, $t_{57} = [3,5]$,
$t_{58} = [4,7]$, $t_{59} = [2,6]$

Fig. 10. An exact R-net MOD model.

multiple users to the MOD system. This description can be done by changing: (i) the number of tokens shown in place p_1 to become proportional with the number of users, and (ii) the number of tokens shown in place p_7 to become proportional with the number of LMSs.

Sub-model-II describes the sequential and parallel operations of disk servers DS_x, DS_y and $\mathrm{DS}_z \in \mathrm{DS}$ when they execute the Scan-EDF disk scheduling algorithm on the user requests. The following parts describe the dynamic behavior of this model.

- Part-IIA describes the dynamic behavior of the disk server $\mathrm{DS}_x \in \mathrm{DS}$ when it executes the Scan-EDF algorithm. This part starts from p_{19} to p_{25}. The meanings of the places and transitions of $\mathrm{DS}_x \in \mathrm{DS}$ model are the same as those of Fig. 7.
- Part-IIB describes the dynamic behavior of the disk server $\mathrm{DS}_y \in \mathrm{DS}$ when it executes the Scan-EDF algorithm. This part starts from p_{33} to p_{39}. The meanings of the places and transitions of the $\mathrm{DS}_y \in \mathrm{DS}$ model are the same as those of Fig. 7.
- Part-IIC describes the dynamic behavior of the disk server $\mathrm{DS}_z \in \mathrm{DS}$ when it executes the Scan-EDF algorithm. This part starts from p_{47} to p_{53}. The meanings of the places and transitions of $\mathrm{DS}_z \in \mathrm{DS}$ model are the same as those of Fig. 7.

Sub-model-III describes the dynamic behavior of the disk servers DS_x, DS_y, and $\mathrm{DS}_z \in \mathrm{DS}$ when they retrieve the required segments from the storage disks SDA_{k1}, SDA_{k2}, and $\mathrm{SDA}_{k3} \in \mathrm{SDA}$, respectively. Also, it illustrates the sequential and parallel processes of these disks when they communicate with each other. The following parts describe the dynamic behavior of this sub-model.

- Part-IIIA describes the dynamic behavior of the disk server $\mathrm{DS}_x \in \mathrm{DS}$, when it retrieves the main part of the requested segments from its storage disk $\mathrm{SDA}_{k1} \in \mathrm{SDA}$ and sends control signals to the disk servers DS_y, and $\mathrm{DS}_z \in \mathrm{DS}$ to retrieve the remaining parts of the requested segments from their storage disks SDA_{k2}, and $\mathrm{SDA}_{k3} \in \mathrm{SDA}$, respectively. The model of the disk server $\mathrm{DS}_x \in \mathrm{DS}$ starts from p_{26} to p_{32}. The meanings of the places and transitions of the $\mathrm{DS}_x \in \mathrm{DS}$ sub-model are the same as those of Fig. 8.
- Part-IIIB describes the dynamic behavior of the disk server $\mathrm{DS}_y \in \mathrm{DS}$ when it retrieves the main part of the requested segments from its storage disk $\mathrm{SDA}_{k2} \in \mathrm{SDA}$ and sends control signals to the disk servers DS_x, and $\mathrm{DS}_z \in \mathrm{DS}$ to retrieve the remaining parts of the requested segments from their storage disks SDA_{k1}, and $\mathrm{SDA}_{k3} \in \mathrm{SDA}$, respectively. The model of the disk server $\mathrm{DS}_x \in \mathrm{DS}$ starts from p_{40} to p_{46}. The meanings of the places and transitions of the $\mathrm{DS}_y \in \mathrm{DS}$ sub-model are the same as those of Fig. 8.
- Part-IIIC describes the dynamic behavior of the disk server $\mathrm{DS}_z \in \mathrm{DS}$ when it retrieves the main part of the requested segments from its storage disk $\mathrm{SDA}_{k3} \in \mathrm{SDA}$ and sends control signals to the disk servers DS_x, and $\mathrm{DS}_y \in \mathrm{DS}$ to retrieve the remaining parts of the requested segments from their storage

disks SDA_{k1}, and $SDA_{k2} \in SDA$, respectively. The model of the disk server $DS_x \in DS$ starts from p_{54} to p_{60}. The meanings of the places and transitions of the $DS_z \in DS$ sub-model are the same as those of Fig. 8.

Sub-model-IV describes the transmission processes of the composed streams of the requested movie file from the main server of $LMS_i \in LMS$ to the requested $UDE_j \in UDE$ through the ATM switching network. The meanings of the places and transitions of this sub-model are the same as those of Fig. 9.

6. Performance Evaluation

In this section, we study and analyze the important performance measures that can be calculated from the following R-net MOD models.

- From the sub-models shown in Figs. 5, 6, 8 and 9, we construct a simple R-net MOD model which adopts the EDF disk scheduling algorithm. We call it *model-A*.
- From the sub-models shown in Figs. 5, 7, 8 and 9, we construct a simple R-net MOD model which adopts the Scan-EDF disk scheduling algorithm. We call it *model-B*.
- The R-net MOD model shown in Fig. 10 is called a *model-C*. This model adopts the Scan-EDF disk scheduling algorithm.

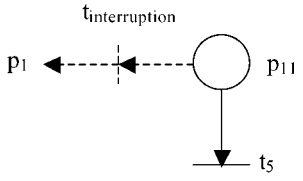
Based on the solution steps of the R-net technique (explained in Sec. 2) and the solving processes of our R-NET package,[28] the required performance measures can be calculated. To obtain realistic performance results, we have associated the R-net transitions with real-times (in ms) obtained from the benchmark data sets of actual multimedia systems.[2,6,7,16,30]

6.1. *Performance measures*

6.1.1. *Quality of Service (QoS)*

The QoS is the most important performance measure for the multimedia systems.[14,15] The QoS can be defined through the following aspects.

- *The interruption probability*: the probability of the requested movie file is not available. To incorporate such aspect in the developed R-net MOD models, we have added a transition $t_{\text{interruption}}$ to the first R-net sub-models shown in Figs. 5 and 10. Figure 11 illustrates this modification.
- *The data loss probability*: the probability of occurrence the buffer overflow problem. In our models, we assume that each disk server in the $LMS_i \in LMS$ has a finite buffer size. Thus, it is possible to overflow any buffer. To incorporate such aspect in the developed R-net MOD models, we have modified the disk server



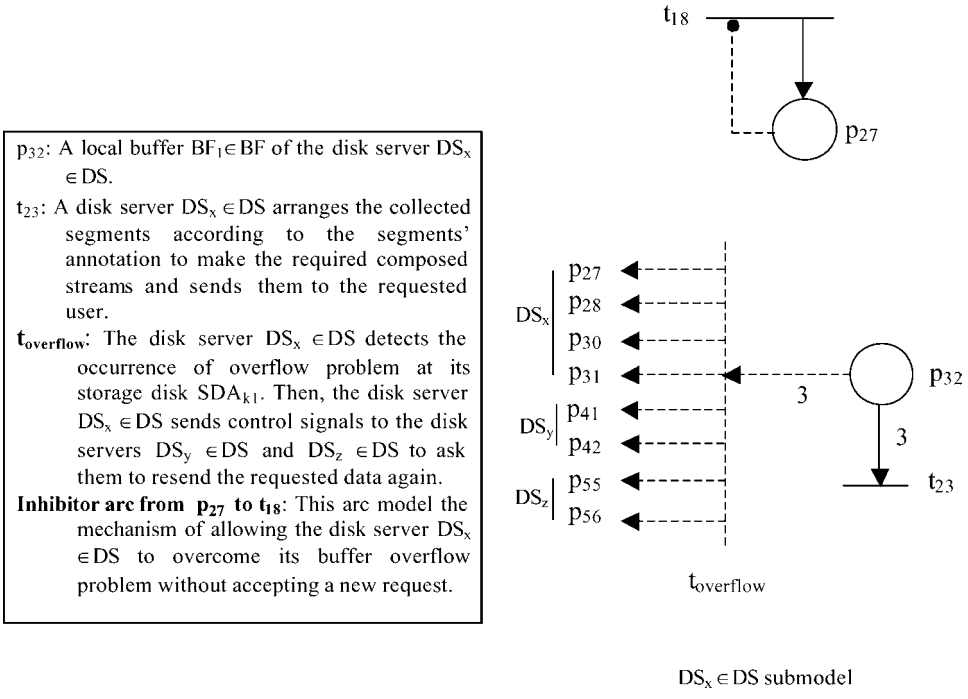
p_1 : A user submission

p_{11}: The main server of $LMS_i \in LMS$ downloads the map allocation of the requested movie file. This map contains the available movie files. The main server uses this map to determine the group of storage disks that contain the required segments of the requested movie file. As pointed out, these storage disks are SDA_{k1}, SDA_{k2}, and SDA_{k3} and their disk servers are DS_x, DS_y, and DS_z, respectively

t_5: The main server checks the status of queues at the storage disks SDA_{k1}, SDA_{k2}, and SDA_{k3} through their disk servers DS_x, DS_y, and DS_z, respectively.

$t_{interruption}$: The main server sends an *interruption signal* to the requested user. This signal means that the requested movie file is not available. Aftertime, the user can send his request.

Fig. 11. A modification for the R-net MOD models (model-A, model-B, and model-C) to handle the *interruption problem*.



p_{32}: A local buffer $BF_1 \in BF$ of the disk server $DS_x \in DS$.

t_{23}: A disk server $DS_x \in DS$ arranges the collected segments according to the segments' annotation to make the required composed streams and sends them to the requested user.

$t_{overflow}$: The disk server $DS_x \in DS$ detects the occurrence of overflow problem at its storage disk SDA_{k1}. Then, the disk server $DS_x \in DS$ sends control signals to the disk servers $DS_y \in DS$ and $DS_z \in DS$ to ask them to resend the requested data again.

Inhibitor arc from p_{27} to t_{18}: This arc model the mechanism of allowing the disk server $DS_x \in DS$ to overcome its buffer overflow problem without accepting a new request.

$DS_x \in DS$ submodel

Fig. 12. A modification for the disk server $DS_x \in$ DS sub-model shown in Fig. 8 to handle the buffer overflow problem.

sub-models shown in Figs. 8 and 10 as shown in Figs. 12 and 13, respectively. The disk server $DS_x \in$ DS sub-model shown in Fig. 8 is modified in Fig. 12. In this modification, we add a transition t_{overflow} to a place p_{32} and inhibitor arc to a transition t_{18} from a place p_{27}. Similarly, the sub-models of the disk servers

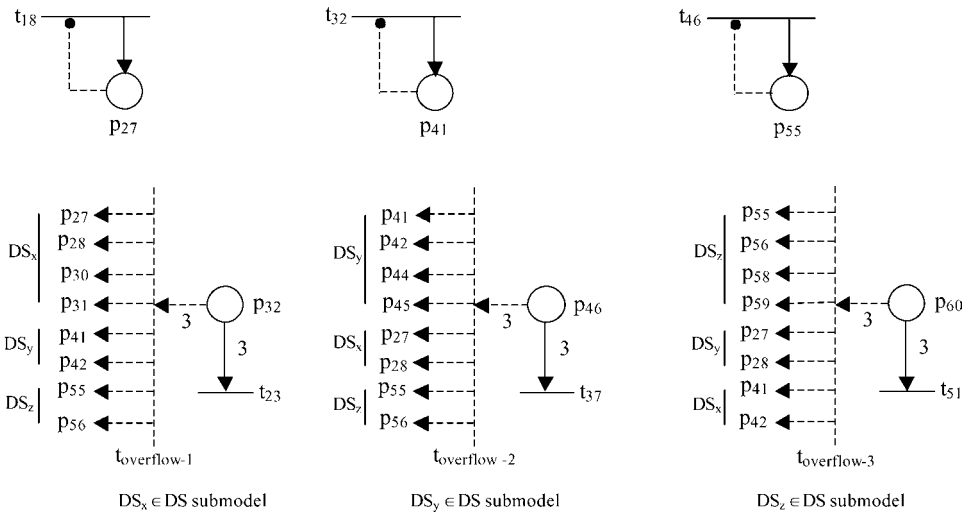


Fig. 13. A modification for the sub-models $\mathrm{DS}_x \in \mathrm{DS}$, $\mathrm{DS}_y \in \mathrm{DS}$ and $\mathrm{DS}_z \in \mathrm{DS}$ shown in Fig. 10 to handle the *buffer overflow problem*.

DS_x, DS_y and $\mathrm{DS}_z \in \mathrm{DS}$ shown in Fig. 10 are modified in Fig. 13. According to the R-net technique (explained in Sec. 2), the *interruption probability* (or the *data loss probability*) associated with the transition $t_{\mathrm{interruption}}$ (or the transition t_{overflow}) is determined dynamically in each state this transition has the ability to fire. For more detail, see Examples 1 and 2. In fact, such dynamic behavior is the main advantage of using the R-net technique.

- *The request response time* (RRT): the user has to receive his requested movie file before the deadline time of this request. The parameters of the RRT are explained mathematically through the TRM_j equation (Sec. 3.3.2). We have already incorporated these parameters in the developed R-net MOD models, as explained in Secs. 4 and 5. Also, we have attached the required deadline times with the transitions of these models.

By incorporating the interruption probability and the data loss probability in the R-net MOD models, we can easily calculate the RRT and QoS performance measures in accurate way. To calculate the RRT performance measure, we run the R-net model. Then, we derive the Markov processes and calculate the steady state probabilities of this model. Finally, the RRT can be calculated as follows:

$$\mathrm{RRT} = \frac{\mathrm{ASJ}_1}{\pi_1},$$

where ASJ_1 is the average sojourn time of the initial state S_1; π_1 is the steady state probability of the initial state S_1; S_1 is the initial state of model-A, model-B, or model-C when a place p_1 contains one (or more) token(s).

The QoS the MOD system provides can be evaluated according to the following expression.

$$\text{QoS} = \frac{N_{mr}}{\text{RRT} \times N_{Ds}},$$

where RRT is the response time of the multimedia requests. N_{mr} is the number of multimedia requests. N_{Ds} is the number of disk servers in the $\text{LMS}_i \in \text{LMS}$.

6.1.2. *Disk Scheduling Algorithm Time (DSAT)*

DSAT is defined as the time taken by the EDF algorithm or the Scan-EDF algorithm to schedule the user requests. From the calculation of DSAT, we can easily study the efficiency of the distributed MOD system under the different types of disk scheduling algorithms.

- To study the R-net EDF sub-model shown in Fig. 6, the output of place p_{29} must be connected to a transition t_{connect} (new transition added to the sub-model) and the output of transition t_{connect} to a place p_{19}. Putting one (or more) token(s) in the place p_{19} derives the Markov process of this sub-model. After deriving the Markov processes and obtaining the steady state probabilities, the DSAT can be calculated as follows:

  $$\text{DSAT of EDF algorithm} = \frac{\text{ASJ}_{19}}{\pi_{19}},$$

  where ASJ_{19} is the average sojourn time of the state S_{19}; π_{19} is the steady state probability of the state S_{19}; S_{19} is the initial state of the R-net EDF sub-model when p_{19} contains one token.
- To study the R-net Scan-EDF sub-model shown in Fig. 7, the output of place p_{26} must be connected to a transition t_{connect} (new transition added to the sub-model) and the output of transition t_{connect} to a place p_{19}. Based on the DSAT equation, the DSAT of Scan-EDF can be calculated.
- To study the R-net Scan-EDF sub-model shown in Fig. 10, we remove it from the complete model. Then, the output of places p_{26}, p_{40} and p_{54} must be connected to a transition t_{connect} and the output of transition t_{connect} to a place p_{18}. Based on the DSAT equation, the DSAT of this model can be calculated.

6.1.3. *Actual Retrieval Time (ART)*

ART is defined as the time taken by the main server of $\text{LMS}_i \in \text{LMS}$ to collect the data of the desired movie file from the storage disk array of $\text{LMS}_i \in \text{LMS}$.

- To study the R-net retrieval sub-model shown in Fig. 8, the output of transition t_{23} must be connected to a place p_{26}. After deriving the Markov processes and obtaining the steady state probabilities, the ART can be calculated as follows:

  $$\text{ART} = \frac{\text{ASJ}_{26}}{\pi_{26}},$$

where ASJ_{26} is the average sojourn time of the state S_{73}. π_{26} is the steady state probability of the state S_{73}. S_{26} is the initial state of the R-net retrieval sub-model when a place p_{26} contains one (or more) token(s).

- To study the R-net retrieval sub-model shown in the complete R-net model of Fig. 10, we remove it from this model. Then, the output of transitions t_{23}, t_{37} and t_{51} must be connected to a place p_{connect} and the output of this place to a transition t_{connect}. The output of t_{connect} must be connected to the places p_{26}, p_{40} and p_{54}. In this case, the initial token(s) of this sub-model starts from a place p_{connect}. After deriving the Markov processes and obtaining the steady state probabilities, the ART can be calculated as follows:

$$\text{ART} = \frac{\text{ASJ}_{\text{connect}}}{\pi_{\text{connect}}},$$

where $\text{ASJ}_{\text{connect}}$ is the average sojourn time of the state S_{connect}. π_{connect} is the steady state probability of the state S_{connect}. S_{connect} is the initial state of the R-net retrieval sub-model when place p_{connect} contains one (or more) token(s).

6.2. *Performance analysis*

By using the R-NET package,[28] the results of the performance measures such as QoS, RRT, DSAT and ART can be calculated. In the following, we study and analyze the results of these performance measures.

Table 1 illustrates the DSAT performance results obtained from the EDF sub-model shown in Fig. 6 and the Scan-EDF sub-model shown in Fig. 7. From these results, we observe that the Scan-EDF algorithm gives 17–23% improvement compared to the EDF algorithm. Thus, we can conclude that the Scan-EDF algorithm performs better than the EDF algorithm. This conclusion is agreed with that obtained by a simulation method shown in Ref. 16. Furthermore, the simulation results of the disk scheduling algorithms presented in Ref. 16 have been obtained in a few hundred milliseconds like our performance results. For example, in Ref. 16, the simulation result of the Scan-EDF algorithm for 6 multimedia requests is approximately 800 msec. Thus, the proposed modeling technique introduces an easy way to verify from the dynamic behavior of a new disk scheduling algorithm as well as its execution time.

Table 1. The results of the DSAT performance measure (ms).

N_{mr}	EDF-sub-model (shown in Fig. 6) (sub-model-II in model-A)	Scan-EDF-sub-model (shown in Fig. 7) (sub-model-II in model-B)
1	180.608	137.908
2	329.395	266.397
3	478.584	394.509
4	628.456	519.481
5	775.064	641.564
6	914.192	767.918

Table 2. The percentage of difference between the results of the R-net models and those of the simulation method.

N_{mr}	Percentage of difference in the case of: Scan-EDF-sub-model (shown in Fig. 7) (sub-model-II in model-B) (%)
1	2.56
2	2.79
3	3.12
4	3.45
5	3.68
6	4.01

Table 2 indicates the percentage of difference (%) between the results obtained from the R-net models (in the case of using the Scan-EDF sub-model shown in Fig. 6) and those of the simulation technique. These results indicate that both results are very closed but the R-net technique is better from the viewpoint of accuracy and the flexibility of graphically modeling the synchronization activities and the concurrent computation among the multimedia data. In fact, the simulation techniques lack to describe such processes.[18,27,28] Supporting these processes is required for guaranteeing the quality, accuracy, and the responsiveness of multimedia application. Therefore, for example (Table 2), the accuracy of the R-net technique is higher than that of simulation technique by 4.01% when the system handles six multimedia requests (e.g., 1.0–797.918/800 = 4.01%). Furthermore, using the simulation technique requires building simulation program for solving each system problem but by using the R-net package, we can build wide range of models for different parallel and distributed system problems, and then run such models on the R-net package.[28]

Table 3 illustrates that the ART performance results obtained from the retrieval sub-model shown in Fig. 10 are better than those of Fig. 8 due to the following reason. The simple retrieval sub-model shown in Fig. 8 describes the dynamic behavior of the disk server $\mathrm{DS}_x \in \mathrm{DS}$ when it retrieves the requested movie file. Such disk server performs the retrieval processes of the requested movie files in a sequential manner. In contrast, the realistic retrieval sub-model shown in Fig. 10 describes the dynamic behavior of the three disk servers DS_x, DS_y and $\mathrm{DS}_z \in \mathrm{DS}$ when they

Table 3. The results of the ART performance measure (ms).

N_{mr}	Retrieval-sub-model (shown in Fig. 8)	Retrieval-sub-model (shown in Fig. 10) (sub-model-III in model-C)
1	315.675	329.187
2	609.787	341.551
3	903.037	348.162
4	1189.111	424.637
5	1468.551	501.112
6	1757.775	574.425

work together in a parallel manner to retrieve the requested movie files. The effect of these parallel retrieval processes especially appears when the number of requests is three.

Table 3 also confirms that when both sub-models of Figs. 8 and 10 receive one user request, the retrieval time obtained from a sub-model shown in Fig. 10 is slightly higher than that of Fig. 7. This difference is due to the decision time taken by the main server of $\text{LMS}_i \in \text{LMS}$ to decide which disk server will retrieve the requested movie file. From Table 3, we also remark that the retrieval time of four, five, or six requests is greater than that of three requests. This difference is due to the waiting time spent in the buffer of the main server of $\text{LMS}_i \in \text{LMS}$ until one of the three disk servers DS_x, DS_y and $\text{DS}_z \in \text{DS}$ become free. As a conclusion, the number of disk servers in each $\text{LMS}_i \in \text{LMS}$ should be a large enough to handle a large number of user requests. Furthermore, such disk servers should work together in a parallel manner to handle the user requests. This conclusion is agreed with that obtained by a simulation method.[6,7,10,12,32]

Table 4 illustrates the RRT and QoS performance results offered by the R-net MOD models for various workloads (multimedia requests). From these results, we can observe the following interesting points.

The request response time decreases as the number of disks used for retrieving the movie file is increased. The RRT performance results obtained from a model-C and a model-A (or a model-B) clarify this point. This result is due to a model-C performs its multimedia service processes in a parallel manner but a model-A (or a model-B) performs its multimedia service processes in a sequential manner. This conclusion can be very useful in making decisions for investment in a new MOD system.

The type of disk scheduling algorithm that can be used in the MOD system effects on the RRT performance results. The RRT performance results obtained from model-A and model-B clarify this point. It is interesting to note that a model-A incorporates an EDF algorithm but a model-B incorporates a Scan-EDF algorithm. Therefore, the RRT performance results obtained from a model-A are better than those of a model-B.

The request response time increases a very few msec when the number of users is increased. The RRT performance results obtained from a model-C clarify this

Table 4. The results of the RRT and QoS performance measures.

	Model-A (EDF)		Model-B (Scan-EDF)		Model-C (Scan-EDF)	
N_{mr}	RRT (s)	QoS (%)	RRT (s)	QoS (%)	RRT (s)	QoS (%)
1	1.328	75.25	1.098	91.07	1.145	29.112
2	2.422	82.57	2.121	94.29	1.188	56.116
3	3.519	85.24	3.141	95.51	1.211	82.576
4	4.621	86.55	4.135	96.69	1.477	90.213
5	5.699	87.71	5.108	97.88	1.743	95.611
6	6.722	89.25	6.114	98.12	1.998	100.000

point. Thus the delay a user experiences in viewing a movie is not affected too much by the number of users in the system.

The QoS of the MOD system increases as the number of multimedia requests increases. Furthermore, the QoS results provide insight on the max workload (multimedia requests) the MOD system is able to support. For example, the QoS of a model-C is 100% when the number of request is six. In other words, the max workload for a model-C is six requests. Thus, from the QoS performance results, we can decide the optimal number of multimedia requests that prevents the performance degradation of the MOD system.

Finally, we have compared the RRT results with those obtained by a simulation method.[7,12] For example, when six users access the MOD system (model-C) at the same time, RRT = 1.998 s. This result is close to that obtained by a simulation method, where RRT = 2.135 s. This comparison gives us greater confidence that the performance results obtained from the developed R-net MOD models are realistic. Furthermore, the R-net MOD models have been developed in an accurate way.

7. Conclusions

We have studied and analyzed the architecture of the MOD service system. To illustrate the dynamic behavior of the real MOD system, we have built three R-net performance models. The first model adopts the EDF disk scheduling algorithm. The second model adopts the Scan-EDF algorithm. These two models describe the dynamic behavior of one disk server when it performs the multimedia service processes. These models have been built in a simple way for explaining how the proposed R-net modeling technique can be used to model the dynamic behavior of real multimedia systems. In contrast, the third model describes the dynamic behavior of three disk servers when they work together in a parallel manner to perform the exact multimedia service processes. This realistic model can be extended to describe the real dynamic behavior of n disk servers.

In the developed R-net MOD models, we have concentrated our attention on studying and analyzing the following interesting multimedia problems.

- The storage problem has been studied through representing the mechanism of distributing the movie files across multiple disks to achieve data retrieval parallelism.
- The scheduling problem of the user requests has been studied through representing the EDF and Scan-EDF real-time disk scheduling algorithms. We have used this representation to analyze the effect of scheduling algorithms on the quality of service of the MOD system.
- The concurrent access problem has been studied through representing the mechanism of allowing multiple users to concurrently access the stored movie files.
- The data retrieval problem has been studied through representing the parallel mechanisms of retrieving the movie files from the desired disk array.
- The communication problem has been studied through representing the communication processes between the user and MOD service system.

To facilitate the analysis of the R-net models of the MOD service system, we have constructed them based on a modular technique. This modular technique provide an easy way to calculate the following performance measures: the quality of service, the request response time, the disk scheduling algorithm time, and the actual retrieval time.

References

1. H. Cha, J. Lee and J. Oh, Construction a video server with tertiary storage: Practice and experience, *Multimedia Systems* **8** (2002) 380–394.
2. M. T. Slingerland and A. J. Smith, Design and characterization of the Berkeley multimedia workload, *Multimedia Systems* **8** (2002) 315–327.
3. S. A. Barnett, J. Gary and A. Anido, Performability of disk-array-based video servers, *Multimedia Systems* **6** (1998) 60–74.
4. F. Panzieri and M. Roccetti, Synchronization support and group-membership services for reliable distributed multimedia applications, *Multimedia Systems* **5** (1997) 1–22.
5. A. M. Dawood and M. Ghanbari, Content-based MPEG video traffic modeling, *IEEE Trans. Multimedia* **1** (1999) 77–87.
6. I. D. D. Curico, P. Antonio, S. Riccobene and L. Vita, Design and evaluation of multimedia storage server for mixed traffic, *Multimedia Systems* **6** (1998) 367–381.
7. D. Jadav, A. Choudhary and P. B. Berra, An evaluation of design trade-off in a high-performance media-on-demand server, *Multimedia Systems* **5** (1997) 53–68.
8. A. Ghafoor, Multimedia database management systems, *ACM Comput. Surv.* **27** (1995) 593–598.
9. M. Krunz and G. Apostolopoulos, Efficient support for interactive scanning operations in MPEG-based video-on-demand systems, *Multimedia Systems* **8** (2000) 20–36.
10. T. G. Kwon, Y. Choi and S. Lee, Disk placement for arbitrary-rate playback in an interactive server, *Multimedia Systems* **5** (1997) 271–281.
11. S. W. Lau and J. C. S. Lui, Scheduling and data layout policies for a near-line multimedia storage architecture, *Multimedia Systems* **5** (1997) 310–323.
12. A. Srivastava, A. Kumar and A. Singyu, Design and analysis of a video-on-demand server, *Multimedia Systems* **5** (1997) 238–254.
13. L. W. Joel, S. Y. Philip and H. Shachnai, Disk load balancing for video-on-demand systems, *Multimedia Systems* **5** (1997) 358–370.
14. P. Huanxu, I. H. Ngoh and A. A. Lazar, A buffer-inventory-based dynamic scheduling algorithm for multimedia-on-demand servers, *Multimedia Systems* **6** (1998) 125–136.
15. A. Mauthe and G. Coulson, Scheduling and admission testing for jitter-constrained periodic threads, *Multimedia Systems* **5** (1997) 337–346.
16. A. N. Reddy and J. C. Wyllie, I/O issues in a multimedia system, *IEEE Comput.* **27** (1994) 69–74.
17. S.-C. Chen and R. L. Kashyap, A spatio-temporal semantic model for multimedia database systems and multimedia information systems, *IEEE Trans. Comput.* **13** (2001) 607–622.
18. T. Murata, Petri nets: Properties, analysis and applications, *Proc. IEE* **77** (1989) 541–580.
19. C. Hirel, K. S. Trivedi and B. Tuffin, SPNP version 6.0, Lecture Notes in Computer Science, Vol. 1786 (Springer-Verlag, New York, 2000), pp. 354–357.
20. I. Davies, W. J. Knottenbelt and P. S. Kritzinger, Symbolic methods for the state space exploration of GSPN models, *Proc. 12th Int. Conf. Modeling Techniques and Tools* (TOOLS 2002), London, UK, 14–17 April 2002, pp. 188–199.

21. P. Ballarini, S. Donatelli and G. Franceschinis, Parametric stochastic well-formed nets and compositional modeling, *Proc. 21st Int. Conf. Application and Theory of Petri Nets*, Arhus, Denmark, 26–30 June 2000, pp. 215–221.
22. P. Merlin and D. Faber, Recoverability of communication protocols — Implications of a theoretical study, *IEEE Trans. Commun.* **24** (1976) 381–404.
23. T. D. C. Little and A. Ghafoor, Spatio-temporal composition of distributed multimedia objects for value added networks, *Computer* **24** (1991) 42–50.
24. J. J. P. Tsai, S. Yang and Y. H. Chang, Timing constraint Petri nets and their application to schedulability analysis of real-time system specifications, *IEEE Trans. Software Eng.* **21** (1995) 32–49.
25. P. Senac, M. Diaz, A. Leger and P. S. Sannes, Modeling logical and temporal synchronization in hypermedia systems, *IEEE J. Select. Areas Commun.* **14** (1996) 84–103.
26. M. Woo, N. U. Qazi and A. Ghafoor, A synchronization framework for communication of pre-orchestrated multimedia information, *IEEE Network* **1** (1994) 52–61.
27. S. U. Guan, H. Y. Yu and J. S. Yang, A prioritized Petri net model and its application in distributed multimedia system, *IEEE Trans. Comput.* **47** (1998) 477–481.
28. S. M. Koriem, R-Nets for the performance evaluation of the hard real-time system, *J. Syst. Software* **46** (1999) 41–58.
29. E. Y. T. Juan, J. J. P. Tsai, T. Murata and Y. Zhou, Reduction methods for real-time systems using delay time Petri nets, *IEEE Trans. Software Eng.* **27** (2001) 422–448.
30. S. Suzuki and M. Miyake, NTT's R&D and deployment activities for the information highway Japan, *Proc. 5th Workshop on Future Trends of Distributed Computing Systems*, Cheju Island, Korea, 28–30 August 1995, pp. 216–223.
31. S. W. Lau, C. S. L. John and G. Leana, Merging video streams in a multimedia storage server: Complexity and heuristics, *Multimedia Systems* **6** (1998) 29–42.
32. S. Sengodan and V. O. K. Li, A quasi-static retrieval scheme for interactive VOD servers, *Proc. 5th Int. Conf. Computer Communications and Networks*, Rockville, Maryland, USA, 16–19 October 1996, pp. 3–8.