A Prioritized Petri Net Model and Its Application in Distributed Multimedia Systems

Sheng-Uei Guan, Hsiao-Yeh Yu, and Jen-Shun Yang

Abstract—The achievement of media synchronization has been dealt with in the Object Composition Petri Net (OCPN) model and the extended OCPN (XOCPN) model. Yet these two models are not enough for synchronization of computers in a distributed environment. This paper proposes a new Petri Net model—Prioritized Petri Net (P-net). The modeling power and properties of P-nets are analyzed. We apply the P-net model to distributed multimedia synchronization, using our version of Distributed Object Composition Petri Net (DOCPN) model. Using the DOCPN model, we can coordinate operations among distributed computer sites. The scenario is like a conductor conducting an orchestra to perform a symphony.

Index Terms—Distributed multimedia system, synchronization, OCPN, Petri Nets.

1 Introduction

NOWADAYS, many distributed multimedia applications have emerged, such as video-on-demand, teleshopping, distance education, etc. These applications already have changed our daily activities. Web surfing is one example. Yet, there are some technical problems still remain to be solved, e.g., how to store and place the bulk of multimedia data [1], how to synchronize and integrate them [2], [3].

Multimedia data when sent across networks may become out-of-sync when being displayed. For example, audio and video to be played in the same interval may start and finish at different times. This can be annoying or even unacceptable for entertainment or commercial purposes. Under the worst condition, audio should be kept at a stable output rate and video output rate should be tuned to fit with audio output rate. This is because video has a characteristic that a few frames lost in one second won't usually be perceived by human eyes.

The Petri Net model has been developed for modeling concurrent computation [4]. The model finds wide applications later in the design and simulation of operating systems, hardware, and communication protocols [6]. It has also been applied to multimedia systems with extensions. "Object Composition Petri Net" (OCPN) [2] and "Extended OCPN" (XOCPN) [3] are two graphic-based models devised to depict synchronization for multiple media. OCPN or XOCPN face two problems when replaying objects in a distributed environment:

1) OCPN and XOCPN lack methods to describe the details of synchronization across distributed platforms.
2) OCPN and XOCPN do not deal with the schedule change caused by user interactions in interactive multimedia systems.

- *S-U. Guan is with the School of Computer Science and Computer Engineering, La Trobe University, Victoria, Australia. E-mail: guan@cs.latrobe.edu.au.*
- *H.-Y. Yu is with the Institute of Computer Engineering and Science, Yuan-Ze Institute of Technology, Taiwan, Republic of China.*
- *J.-S. Yang is with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan, Republic of China. E-mail: jsyang@csie.nctu.edu.tw.*

Manuscript received 3 Jan. 1996; revised 10 July 1996.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 104712.

Asynchrony needs to be considered when media come from multiple sources or arrive at multiple destinations for presentation or other purposes. In most multimedia applications, e.g., web surfing or navigation, user interaction is a normal action that can change the display of media. Changing the display of media normally implies a reschedule of current media flow. OCPN and XOCPN lack the modeling components for these two functions. Supporting these functions is required for guaranteeing the quality and responsiveness of a multimedia application. We need a new model to guide the design of control for distributed multimedia system, handling the synchronization and interaction for the whole system.

An extension to Petri Nets is proposed in this paper. Arcs in Petri Nets can be assigned with priorities. A transition with a high priority arc ready may trigger the firing immediately. These "prioritized Petri Nets" (P-nets) can be used to solve the above problems. Before we can put P-nets to wider use, we try to answer the following questions: What is the modeling power of P-nets? Is it more powerful than Petri Nets? These questions will be addressed shortly.

The rest of the paper is organized as follows: Section 2 introduces the concepts of Petri Nets, OCPN, and XOCPN models briefly. Section 3 describes the concepts, application and properties of P-nets. Section 4 elaborates on the architecture and synchronous control of DOCPN. Section 5 discusses P-nets with multiple priorities. Section 6 gives the conclusions.

2 Related Work

The presentation of multimedia can be described using "time intervals." Little and Ghafoor described the temporal relationship between two time intervals [2]. We briefly describe their results in the following: Temporal relationships between any two time intervals can be described using: "equal," "before," "meet," "overlap," "during," "start," and "end." Symmetric relationships exist when their roles are exchanged. There are 13 temporal relationships between two time intervals in total. Synchronization among objects can be specified based on these relationships .

2.1 Petri Net

Petri Net theory has been used as a basis for asynchronous and synchronous communication [4].

DEFINITION (Petri Net). *A* Petri Net structure, C, *is a four-tuple,* $C = (P, T, I, O)$. $P = \{p_1, p_2, \ldots, p_n\}$ *is a finite set of* places, $n \geq 0$. $T = \{t_1, t_2, \ldots, t_m\}$ *is a finite set of* transitions, $m \geq 0$. *The set of places and the set of transitions are disjoint,* $P \cap T = \varnothing$. $I : T \rightarrow P^{\infty}$ *is the* input *function, a mapping from transitions to bags of places.* $O : T \rightarrow P^{\infty}$ *is the* output *function, a mapping from transitions to bags of places* [4].

A "place" in the definition 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. The advantage of Petri Net is that it describes the control flow by manner of graphs.

2.2 OCPN Model

The OCPN model proposed by Little and Ghafoor in 1990 [2] can be used to handle the synchronization of objects being played back. This model inherits the methods in Petri Net to depict the flow and synchronization control of playback, replacing "places" in Petri Nets with "objects." OCPN schedules the presentation of media objects using time intervals. The interrelationship between time intervals can be described in OCPN. For this reason, OCPN offers a mechanism not only for asynchronous but also for synchronous scheduling of events.

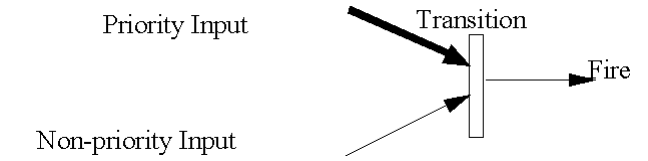

Fig. 1. A transition with a priority input.

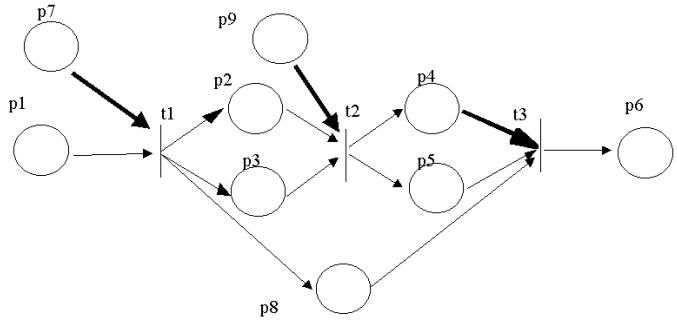

Fig. 2. A prioritized Petri Net.

2.3 XOCPN Model

XOCPN (Extended OCPN) model was proposed by Woo et al. [3]. The major issues are upgrading the ability of OCPN in distributed applications. Granulation of objects has been defined in XOCPN. A Synchronization Interval Unit (SIU) defines time characteristics of different media with different grain sizes.

XOCPN extends the expression method in OCPN. It adds some resource setup control to OCPN to ensure the exhibited conditions of objects meet the platform configuration requirement. Also, XOCPN adds resource release control to release finished objects. An exhibited object cannot go through a transition until it has prepared all its resources.

3 Prioritized Petri Nets

3.1 Definitions

A *P-Net structure*, D, is a five-tuple, $D = (P, T, I, I_p, O)$. $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of *places*, $n \geq 0$. $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of *transitions*, $m \geq 0$. The set of places and the set of transitions are disjoint, $P \cap T = \varnothing$. $I : T \rightarrow P^{\infty}$ is the input function, a mapping from transitions to bags of places. $I_p : T \rightarrow P^{\infty}$ is the priority input function, a mapping from transitions to bags of places. $O : T \rightarrow P^{\infty}$ is the output function, a mapping from transitions to bags of places.

The difference of P-nets from traditional Petri nets lies in the introduction of priority into input functions. Input functions are treated unequally at transitions, as shown in Fig. 1. A priority input event arriving at a transition may force firing without waiting for the arrival of nonpriority events.

Firing rules: A transition with nonpriority input events only will fire when all events are complete and ready. A transition with a priority input event concurring can fire with the arrival of that priority input event, without waiting for other nonpriority events. For the same priority events concurring at a transition, we apply the "AND" rule. A place with a token and several transitions enabled from this place will fire the transition with a priority arc from this place. If there are more than one priority arcs outgoing from a place enabling more than one transition, then the firing choice is nondeterminate.

3.2 Examples and Application

The mathematical representations for the example in Fig. 2 is shown as below:

$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$, $T = \{t_1, t_2, t_3\}$,
$I(t_1) = \{p_1\}$, $I_p(t_1) = \{p_7\}$, $O(t_1) = \{p_2, p_3, p_8\}$,
$I(t_2) = \{p_2, p_3\}$, $I_p(t_2) = \{p_9\}$, $O(t_2) = \{p_4, p_5\}$,
$I(t_3) = \{p_5, p_8\}$, $I_p(t_3) = \{p_4\}$, $O(t_3) = \{p_6\}$

The arrival of input event p7 at transition t1 will force t1 to fire. The arrival of input event p9 at transition t2 will force t2 fire. Note the forced firing at t2 will not affect p8 under our definition. But, for implementation purposes, it may be good if a notification mechanism can be added to inform places/processes concurring in the near future to speed up processing whenever a transition is being fired prematurely.

We illustrate the application of P-nets to real-world examples. Imagine a scene we usually see at a bus stop. When people come to a bus stop, they wait until a bus arrives, while a bus arriving at an empty bus stop will usually drive away without waiting further. Another scene we commonly see is in meetings: A meeting usually will not start until the chairman arrives. People need to wait if they come in early.

There are also many examples in the real or computer world where time schedule dominates an event transition. Even though some conditions are not ready yet, an event will occur when its time schedule is due. This will happen when real time constraint is concerned and when a downgraded service can be achieved without some prespecified resources. P-nets can be applied to these cases by using a clock or time schedule and priority arcs driving those time-sensitive transitions.

3.3 Analysis—Modeling Power, Reachability, and Liveness

In this section, we show the following:

THEOREM 1. *P-nets can simulate Turing machines. Agerwala [7] and others have shown that an extended Petri Net model with the ability to test a place for zero token can simulate a Turing machine.*

PROOF. We show in Fig. 3 how a P-net can be constructed to test for zero. To test if P (place **b**) has a token, we place a token in p = 0? (place **a**) initially and define two priority arcs from places **a** and **b** to transition t1. Now, if P has no token, then transition t1 won't fire. Transition t2 will fire. If P has tokens in it, then both t1 and t2 are enabled. But, according to our firing rule (a place enabling several transitions will fire the transition with a priority arc from it), t1 will fire because there is a priority arc from place **a**. □

Now that the P-net model has the power of Turing machines, the issues of reachability and liveness [4] are undecidable. However, for applications that have P-nets free from cycles, it will be live (i.e., free from deadlocks). A Petri Net is conservative if the number of tokens in the net is conserved. For general P-nets, the number of tokens in the net will depend on the concurrency of the modeled system and, therefore, is not conservative.

4 DOCPN MODEL

Imagine we have multiple computers connected in a distributed environment to simulate a distributed computer symphony. In order to achieve this, we need to handle interaction and synchronization within and across platforms. Here, we propose a Distributed Object Composition Petri Net (DOCPN) model with the following properties:

1) Inheriting the characteristics of Petri Net, that is, waiting at a transition until all input signals arrived, then firing concurrently.
2) Extending Petri Nets with P-net attributes described in Section 3: The arrival of priority input at a transition may cause firing of a transition without waiting for other nonpriority input concurring at the same transition.

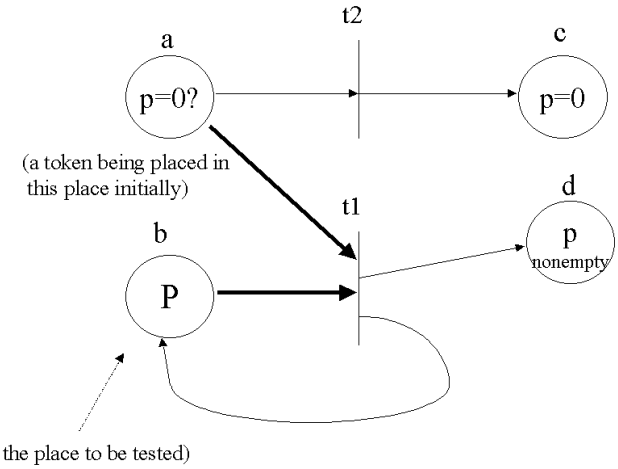

Fig. 3. Testing zero using prioritized Petri Nets.

3) Using the synchronous methods inherited from OCPN and XOCPN to achieve synchronization among intermedia-objects.
4) Extending OCPN to a distributed environment that handles asynchrony across platforms using a global clock.
5) Adding user interaction control into OCPN, thus user interaction can be a new and important factor in synchronization.

4.1 DOCPN Architecture

For each node to achieve synchronization within a distributed multimedia system, it must have DOCPN control mechanism built in. It communicates with other nodes to form an interconnected synchronization control system. DOCPN consists of three subcomponents: Multimedia Synchronization Control, Integrated Synchronization Control, and Interactive Control.

4.1.1 Multimedia Synchronization Control

This component exhibits multimedia according to a preprescribed schedule. DOCPN inherits the characteristics of Petri Net, that is, waiting at a transition until all input signals arrived, then firing concurrently. Therefore, DOCPN can present multimedia with time dependence on schedule. For example, a video channel and an audio channel to be lip-synchronized can be achieved easily if we define SIU as 1/30 second.

4.1.2 Integrated Synchronization Control

In a distributed environment, asynchrony of local clocks at various sites is another difficult issue for keeping a consistent time across platforms. In DOCPN model, "integrated synchronization control" is treated as part of preorchestration. We propose a Global Clock (GC) to adjust local time frame periodically. The time periods to adjust local time frame are defined in advance. If the clock in a client site is faster than GC, the current transition in this client will not fire until GC arrives. Otherwise, if the clock in a client site is slower than GC, the transition of this client will fire immediately once GC arrives. Actions unfinished before the transition will be skipped.

4.1.3 Interactive Control

Multimedia preorchestration is as described above if combined with user interaction allows on-demand application. In a distributed environment, interaction may come from participating group members local or remote contributing to a harmonious orchestration. Orchestration flow may be modified according to user feedback. This allows a hypermedia-like navigation. Presentation can be side-tracked, skipped, paused, or repeated.

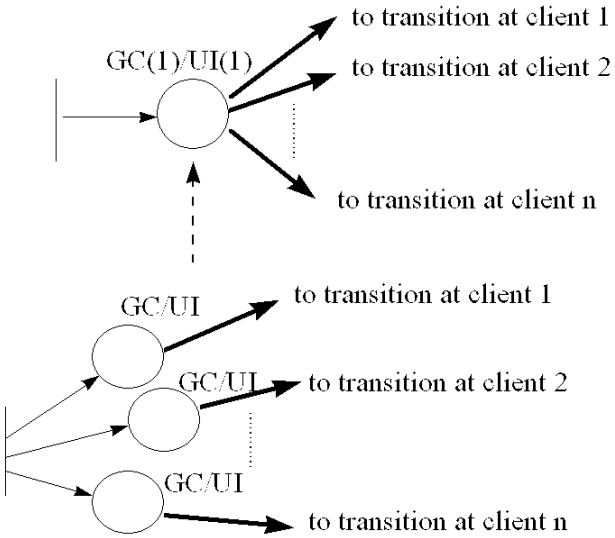

Fig. 4. An example of GC/UI replacement.

4.2 DOCPN Definitions and Examples

4.2.1 Places

TIP (Total Initial Place): TIP represents a unique starting-mark of a DOCPN map.

TFP (Total Final Place): TFP represents a unique ending-mark of a DOCPN map.

Conductor: Conductor indicates a site with the ability of sending out global control signals GC.

a: client identifier, b: time sequence

CIP(a) (Client Initial Place): CIP represents the starting-mark of a DOCPN map at client site a.

CFP(a) (Client Final Place): CFP represents the ending-mark of a DOCPN map at client site a.

O(a, b), O_j'(a, b) (Object): objects sitting at client site a to be synchronized at time b.

GC(b): Global clock sent at time b.

UI(a): User Interaction from user a.

To simplify our diagram drawing: If there are more than one client transitions that receive a priority arc from GC/UI, we use a single GC/UI place to represent several identical GC/UI places connected to the transition at each client. An example is shown in Fig. 4.

4.2.2 Input, Priority Input Function, and Transitions

Output function

The output function of DOCPN is the same as that of Petri nets.

Priority input function and Transitions

DOCPN categorizes input events in two priorities:

- priority: global clock events, user input events
- nonpriority: Input events for the orchestration of objects.

T(a, b) was represented as a transition sitting in client **a** to be synchronized at time **b**. With the new firing rules specified earlier, GC and UI events have leading roles in synchronization. They can guide the schedule of a DOCPN map.

4.2.3 Distributed Synchronization Using a Global Clock

We explain the above, using Fig. 5 as an example. Here, a "Conductor" plays the leading role as in an orchestra. It issues global clock signals regularly to each client, fixing problems arising from asynchrony. For example, if objects O(1, 3) and O'(1, 3) are displayed too fast at client site 1, then they have to wait at T(1, 3) until the arrival of GC(3). Or, if objects O(1, 3) and/or O'(1, 3) are too late for presentation, then the arrival of GC(3) will force T(1, 3) to fire. This will ignore the late schedule of O(1, 3) and/or O'(1, 3), preparing the whole system for the next transition.

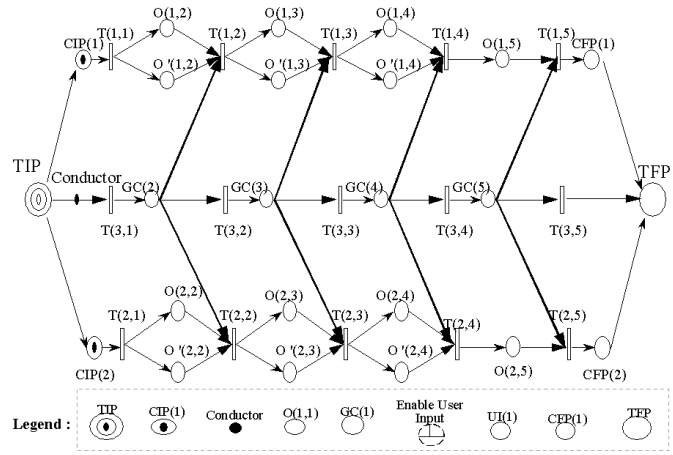

Fig. 5. A DOCPN map with a global clock controlling the global schedule.

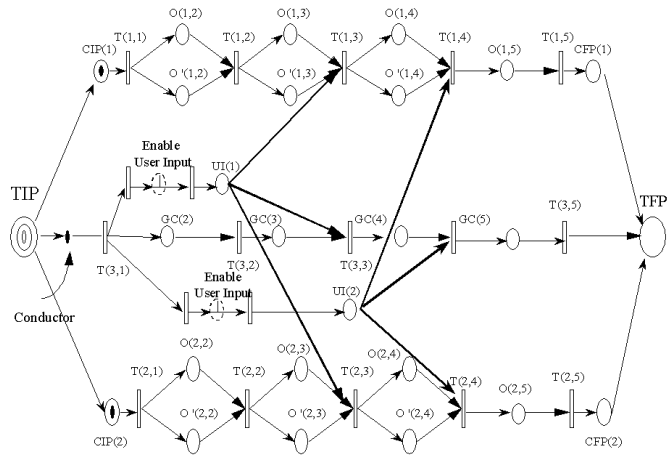

Fig. 6. A DOCPN map with UI controlling the global schedule.

4.2.4 User Interaction

Similarly, as shown in Fig. 6, a user interaction event (e.g., button on a mouse clicked, menu selected) will also change the global schedule. In Fig. 6, the output of UI(1) event will force the firing of transitions T(1, 3), T(2, 3), and T(3, 3). UI(1) may mean "skip the first two parts of a presentation." Incorporating UI events into a DOCPN map has an effect that it allows the global schedule to be adjusted interactively. A UI event may leave out a certain part of the orchestration and thus force a change of the global clock schedule. The input of UI(1) at transition T(1, 3), T(2, 3), and T(3, 3) also means that even with the completion of O(1, 3), O'(1, 3) and/or O(2, 3), O'(2, 3) the orchestration will halt temporarily waiting for user input.

Combining GC and UI, we form a complete DOCPN Map, such as in Fig. 7. If a GC edge meets with a UI edge at a transition, this transition would apply the "AND" rule for these two edges because they are both of the same priority.

4.3 Late Arriving Tokens

When a transition is forced to fire by an incoming priority arc, how should we handle the other late-coming tokens in places leading to nonpriority arcs? We regard this as an implementation issue related to the system being modeled. If these late-coming tokens stand for video frames, now, because they are late for presentation,

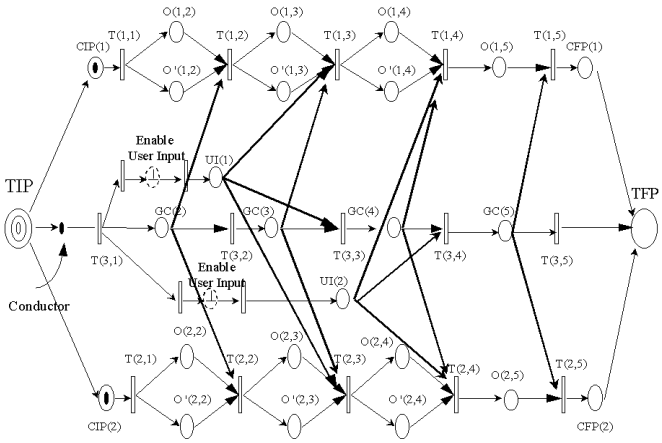

Fig. 7. A complete DOCPN map (with GC and UI).

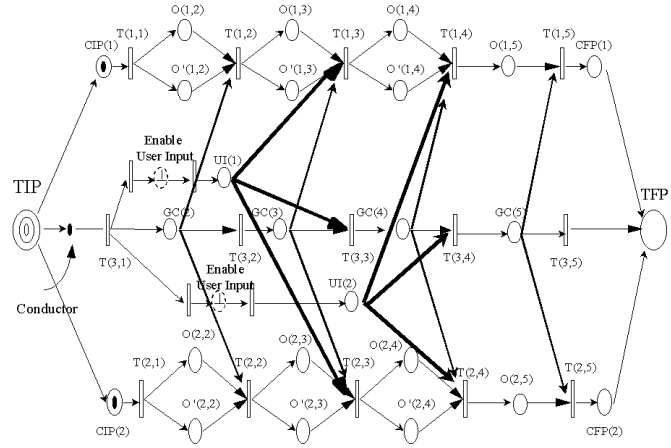

Fig. 8. A DOCPN map with two priorities.

they can be discarded. If these late-coming tokens stand for reusable resources, e.g., buffers, then they should be preserved for the next use.

5 Priority Petri Net Extension with More Priorities

The proposed model can be extended with more priorities. The arrival of the highest priority input event at a transition will force firing without waiting for the arrival of low-priority or nonpriority events. See Fig. 8 for an example: Here, UI has a priority higher than GC; the arrival of UI at T(1, 3), T(2, 3), or T(3, 3) will force firing without waiting for GC or others.

Applications of this extended model can be seen in some real-world examples, e.g., scheduling of production lines in a manufacturing plant can be adjusted by the section leader, production manager, or general manager, depending on the authorization. Some production lines can be controlled by the section leader. Other production lines are controlled by the production manager. The most important production lines are controlled by the general manager.

6 Conclusions and Future Work

In this paper, we have described an extended Petri Net model. It extends traditional Petri Nets with priority. We have illustrated the application of P-nets using solid examples. We have also shown that P-nets can simulate Turing machines. Properties of P-nets have been analyzed. Fields in which we feel that P-nets may find applications are: distributed control, distributed systems and applications, distributed simulation, and real-time scheduling.

We have presented the application of P-nets in distributed multimedia synchronization using DOCPN. DOCPN can be used to design the control mechanism for distributed multimedia systems, handling the synchronization and user interaction for the system. At the same time, it needs the help of a Global Clock to adjust asynchrony, which results from different timing among computer sites. An interactive mechanism is incorporated to handle the interactive operations of a multimedia system and the branching resulting from navigation. DOCPN finds application in distributed control or orchestration where user intervention and global timing are critical.

An independent work done at University of Ottawa by Li et al. has derived a software architecture for distributed data stream without the use of a global clock [5].

Acknowledgments

We are grateful for the support received from Accton Corp., the National Science Council of Taiwan, and the Australian Research Council. We also thank the reviewers for many valuable comments.

References

[1] P. Lougher and D. Shepherd, "The Design of a Storage Server for Continuous Media," *The Computer J.*, vol. 36, pp. 32-42, 1993.
[2] T.D.C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE J. Selected Areas Comm.*, pp. 413-427, Apr. 1990.
[3] M. Woo, N.U. Qazi, and A. Ghafoor, "A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information," *IEEE Network*, pp. 52-61, Jan./Feb. 1994.
[4] J.L. Peterson, *Petri Net Theory and Modeling of Systems*. Prentice Hall, 1981.
[5] L. Li, A. Karmouch, and N.D. Georganas, "Multimedia Teleorchestra with Independent Source: Part 1 and Part 2," *ACM/Springer-Verlag J. Multimedia System*, vol. 1, no. 4, pp. 143-165, Feb. 1994.
[6] T. Agerwala, "Putting Petri Nets to Work," *Computer*, vol. 12, pp. 85-94, Dec. 1979.
[7] T. Agerwala, "A Complete Model for Representing the Coordination for Asynchronous Processes," Hopkins Computer Research Report Number 32, Computer Science Program, Johns Hopkins Univ., Baltimore, Md., July 1974.