A Multimedia Synchronization Model Based on Timed Petri Net

LIANG Yongquan and SHI Zhongzhi

1 Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, P.R. China
2 Shandong Institute of Mining and Technology, Taian 271019, P.R. China

Received November 9, 1997; revised March 6, 1998.

Abstract Multimedia synchronization is a significant requirement for distributed multimedia applications. In this paper, a multimedia synchronization model based on timed Petri nets is presented. Using this model one can give the abstraction and formal description for multimedia object compositions with time-constrained relations. Algorithms for asynchronous user interactions are also presented.

Keywords timed Petri net, multimedia synchronization, asynchronous user interaction

1 Introduction

Advanced multimedia systems are characterized by the integrated computer-controlled generation, storage, communication, manipulation and presentation of independent discrete (i.e., time independent, like text and graphics) and continuous (i.e., time dependent, like audio and video) media data[1]. Multimedia presentations involve the integration and display of a variety of data types in the form of both static and continuous media. The introduction of continuous media brings with it certain time-related constraints that must be considered at presentation time[2].

The word synchronization refers to time. Synchronization in multimedia systems refers to temporal relations between media objects in a multimedia system. Researchers have addressed multimedia synchronization from various perspectives. The major issues include how to specify and how to implement synchronization. Especially in the specification area a variety of techniques have been published and implemented. There are hierarchical models, reference point models and timed Petri nets models to specify multimedia synchronization[3].

Petri net is a kind of system description and analysis tool, and well known for its ability to model concurrent and sequential activities[4]. Since the concept of Petri net was first introduced by D. C. A. Petri in 1962, the Petri Net theory has developed very rapidly in Europe and America and found its wide application in computer science and other fields. The Petri net is defined as a bipartite, directed graph \( N = \{T, P, A\} \) where \( T \), \( P \) and \( A \) represent a set of transitions (bars), a set of places (circles), and a set of directed arcs, respectively. A marked Petri net \( N_m = \{T, P, A, M\} \) includes a marking \( M \) which assigns tokens (dots) to each place in the net. For simple Petri nets, firing of a transition is assumed to be an instantaneous event. To represent the concept of nonzero time expenditure in the Petri net, extensions of the original model are required. A class of enhanced Petri net models has been developed which assign a firing duration to each transition. These models are generally called timed Petri net models, and map well to Markov performance analysis.

Thomas D. C. Little and Arif Ghafoor suggested a synchronization and storage model for multimedia objects, which is called Object Composition Petri Net (OCPN) and is mainly based on timed Petri net and temporal interval logic[5,6]. The basic idea is to represent
various components of multimedia objects as places and describe their inter-relationships in the form of transitions. While the concept of instantaneous firing of transition is preserved, a non-negative time parameter is assigned to each place in the net. The authors discussed the reachability, liveness, boundedness, conservation properties of OCPN, and presented the hierarchical storage model and corresponding object retrieval and presentation algorithms. This model has shown to be quite efficient for specifying multimedia synchronization requirement.

But several issues are not addressed by the OCPN model, one of which is that the user interaction is not well described by the OCPN model. When the user is allowed to interact freely with the presentation, some new synchronization problems may arise. In the presentations involving continuous media, there may be need for synchronization when the user requests for some special effects such as pause and user skips. So we augment the OCPN model in order to deal with the user interaction. In Section 2, we present the augmented OCPN model. In Section 3, we describe the model's support to user interaction and give corresponding algorithms.

2 The Augmented OCPN Model

In order to deal with user interaction, we augment the OCPN model by introducing a measurement representing the extent of resource consumption for each place in the net, and we call the augmented net as XOCPN. The XOCPN is a 7-tuple, defined as follows:

\[ PXOCPN = (T, P, A, D, E, R, M) \]

where \( T = \{t_1, t_2, \ldots, t_n\} \) is a finite set of transitions; \( P = \{p_1, p_2, \ldots, p_m\} \) is a finite set of places; \( A : \{T \times P\} \cup \{P \times T\} \) is a set of arcs representing the flow relation; \( M : P \rightarrow I \), \( I = \{0, 1, 2 \ldots\} \) is a marking; \( D : P \rightarrow R^+ \) is a mapping from a set of places to the non-negative real numbers and represents the presentation intervals; \( R : P \rightarrow \{r_1, r_2, \ldots, r_j\} \) is a mapping from the set of places to a set of multimedia objects; \( E : P \rightarrow R^+ \) is a mapping from a set of places to the non-negative real numbers, representing the extent of the resources so far consumed; \( P \cap T = \phi \).

Let \( I_p(t) \) be the set of input places to transition \( t \); \( O_p(t) \) be the set of output places to transition \( t \); \( I_t(p) \) be the input transition for place \( p \); \( O_t(p) \) be the output transition for place \( p \).

For each object/resource \( p_i \) we define a hierarchy of states that describe the status of the object and are dependent on the stage of the object's consumption. The four possible states are: Idle, In-Process, Finished and Complete. They are defined as follows:

1. Idle: An object is in the Idle state when \( \tau^c_{p_i} = 0 \). An object in the Idle state is not yet ready to begin play, typically because the presentation is still at an earlier point.

2. In-Process: An object is in the In-Process state when \( 0 < \tau^c_{p_i} < \tau_{p_i} \). The object is being displayed and the token is being locked in \( p_i \) through the duration of the In-Process state.

3. Finished: An object is in the Finished state when \( \tau^c_{p_i} \geq \tau_{p_i} \). Resource consumption is completed, token is not yet removed. Token in place \( p_i \) is ready to enable output transition \( O_t(p_i) \). If \( \tau^c_{p_i} > \tau_{p_i} \), it means that the resource is waiting to be synchronized with some other resource(s).

4. Complete: An object is in the Complete state when \( \tau_{p_i} = null \). The object has completed its playout, the token is removed and the presentation has moved on beyond the object.

Associated with the definition of the Petri net is a set of firing rules governing the semantics of the model. Since we define a transition to occur instantaneously, places rather