Petri Net Models for Describing Multimedia Synchronization Requirements

S. Vuong, K. Cooper, and M. Ito
Departments of Electrical Engineering and Computer Science
The University of British Columbia
email: vuong@cs.ubc.ca

Abstract

Synchronization constitutes an important research field in multimedia communication. The synchronization problem has been addressed in the literature in two distinct levels: specification and design. On the specification level, several formal models have been proposed, which are mostly variations of the Petri net model. Although some models are deemed to be better than others in some aspects, there has not been a comprehensive comparison of these specification models. This paper provides a critical review of the existing Petri net specification approaches for specifying synchronization constraints, including object composition, extended object composition, dynamic timed, and time stream petri net models, and applies them to the specification of an interesting, relatively straightforward synchronization example for a qualitative and quantitative evaluation of their strengths and weaknesses.

1. Introduction

Multimedia (mm) refers to the integration of text, graphic, image, audio, and video in an application. The mm applications can be divided into two broad categories: distributed and non-distributed applications. Non-distributed mm applications include those where the source site is the same as the destination site. These presentations are composed of locally supplied data, for example, from CDROM, hard disk, or camera. Distributed mm applications, the more complex category, support the integration of various media, supplied by data sources over a network. For both categories the data may be from a real-time or pre-orchestrated media source. Examples of mm applications and their characteristics are provided in [1].

Since the multiple streams of data can be supplied by different sources, the various streams need to be coordinated both spatially and temporally for presentation. Synchronization refers to the temporal composition of these streams. The synchronization task involves coordinating among streams of distinct media (interstream synchronization) as well as coordinating within each stream of media (intrastream synchronization). Synchronization can be applied to the playout of streams of data (concurrent or sequential) and in response to external events from a human user or the supporting network. The problem of synchronizing time-ordered sequences of data elements is fundamental to mm data. The problem includes synchronizing data that are:

- heavily time dependent and require time-ordered playout during presentation for a fixed amount of time. For example, playout the audio and visual data in a motion picture with lip synchronization constraints.
- heavily time dependent and require time-ordered playout during presentation for a variable amount of time. For example, playout the audio and visual presentation until the user indicates to stop.
- not required to be stringently synchronized. For example, display the text annotation of a presentation within several seconds of the corresponding image being displayed.

The synchronization problem has been addressed in the literature in two distinct levels: specification and design (mechanism). The specification level aims at defining the multimedia application requirements and the associated synchronization constraints. It includes, for example, the use of formal methods such as petri nets [2,3,4,5], path operators [6], high level languages [8, 9], and time extended formal description technique LOTOS [18] as modelling techniques. The design level refers to the development of schemes and protocols to achieve the intrastream and interstream synchronization as well as controlling the network delay and jitter. Examples of synchronization mechanisms and protocols include synchronization marks and channels [12], elastic buffers [13], logical time system [14], the flow protocol [15], and the SRTDD protocol [16]. This paper focus on the specification level of the synchronization

0-8186-7216-1/95 $04.00 © 1995 IEEE

problem. The first objective of this paper is to review four Petri net specification approaches described in the literature. The interested reader may refer to [17] for an overview of Petri net modelling. The second objective is to evaluate these Petri net specification approaches in their application to an interesting and straightforward example. The example includes a variety of synchronization requirements including interstream and intrastream synchronization, time dependent and time independent data streams, fixed playout duration, coarse and fine grained synchronization requirements, user interactions, and jitter constraints. The evaluation includes criteria such as measuring the complexity and completeness of the specification(s). Not including the introduction, the paper is organized as follows: Section 2 outlines the specification approaches used; Section 3 summarizes and discusses the results; Section 4 provides conclusions and discusses future work. The natural language specification of the example is included in Appendix A.

2. Petri Net Specification Approaches

Petri nets (PN) are modelling techniques suitable for describing concurrent, asynchronous, distributed, parallel, non-deterministic and/or stochastic systems [17]. The model offers both a graphical representation and a mathematical theory basis for the analysis of the system's behavior. Pictorially, a marked Petri net is represented by indicating its places by circles, transitions by bars, arcs by arrows, and the token by small black dots [17]. Formally, a marked Petri net is a bipartite, directed graph defined by the tuple (P,T,A,M) where:

$P = \{p_1, p_2, p_3, \ldots, p_m\}$ is a set of places,

$T = \{t_1, t_2, t_3, \ldots, t_n\}$ is a set of transitions,

$A \subseteq \{T \times P\} \cup \{P \times T\}$ is a set of arcs,

$M : \mathrm{P} \rightarrow I$, $I = \{0, 1, 2, 3 \ldots\}$ associates a marking to each place in the net.

Extensions to the marked Petri net model include the object composition petri net, extended object composition petri net, dynamic timed petri net, and the time stream petri net models. These extensions are described below.

2.1 Object Composition Petri Net

The OCPN [2] is a model for specifying the processing and temporal requirements at the presentation level. The model is an enhanced version of Timed Petri nets with resources and durations associated with places. The durations are the fixed playout presentation times for the object represented by each place. The transitions of the net represent the explicit inter-object synchronization points. Formally, the OCPN is defined as a tuple (T, P, A, D, R, M), where:

(P, T, A, M) defines a marked Petri Net,

$D : \mathrm{P} \rightarrow R$ is a mapping from places to durations,

$R : \mathrm{P} \rightarrow \left(r_1, r_2, r_3, \ldots r_k\right)$ represents the resources required at each place.

The OCPN uses thirteen temporal relations to define the relationships between the multimedia data and to produce presentation scenarios. These are before, meets, overlaps, during, starts, finishes, equal and their inverse relations (equal does not have an inverse). These relations are derived from the logic of intervals presented in [10].

In addition to describing complex multimedia presentation scenarios, the OCPN specification is amenable to constructing a logical multimedia database model. This facility for composing and storing multimedia data in an integrated and comprehensive model is a major strength of the method. As a drawback of the model, the inter-object level of synchronization is too coarse for modelling isochronous data synchronization requirements.

2.2 Extended Object Composition Petri Net

The XOCPN [3] model is an enhanced version of the OCPN method. It allows both coarse and fine grained synchronization for continuous media objects. The XOCPN overcomes the limitation of the coarse grained, object level synchronization found in the OCPN. In addition, the XOCPN addresses modelling the quality of service (QoS) specification requirements. QoS requirements cannot be specified in the OCPN.

To achieve finer synchronization control the temporal interval associated with an object is divided into a sequence of smaller units, called synchronization interval units (SIUs). For example, a video object may be divided into multiple SIUs, each one holding information for a video frame. The size (duration) of a SIU depends on the type of the object. In addition, a number of intermediate synchronization points, Interstream Pacing Points (IPP), are inserted at appropriate, precalculated points in the string of SIUs. These IPP places are used to represent fine grained interstream synchronization control.

Processing in the places is also described in more detail as compared to the OCPN model. The places in the XOCPN can be of two types: object places and control places. Object places specify actions for the playout or the transmission of the object. Control places denotes controlling actions according to the XOCPN semantics. These actions are used to setup a virtual channel, negotiate QoS parameters, delete release resources upon the playout, and define interstream synchronization policies.

Unlike the OCPN, the number of specifications models described in XOCPN is two: distinct models are defined at source and receiver sites. The transmitter model schedules the transmission of the multimedia objects. The

receiver model plays out the multimedia objects and introduces actions for maintaining the synchronization.

Formally, the XOCPN is defined as a tuple (T, P, A, D, R, M, Y, Z) where

(P, T, A, M) defines a marked Petri Net,

$D : \mathrm{P} \rightarrow (d_1, d_2)$ represents the delay before an action and the action duration, respectively,

$R : \mathrm{P} \rightarrow \left(r_1, r_2, r_3, \ldots r_n \right)$ is a mapping from places to objects,

$Y : \mathrm{P} \rightarrow$ (Resource_setup, Resource_release, SIU_playout, SIU_transmit, Interstream_synch) is a mapping from places to actions to be performed during communication,

$Z : \mathrm{P} \rightarrow$ (address_for_QoS, address_for_SIU, address_for_synchronization_requirements) are the address locations for the parameters.

The interstream synchronization among multimedia objects at the receiver in the XOCPN approach is handled at two levels of granularity: the object level and at a finer grained level. For the object level synchronization the XOCPN defines the Inter-object Synchronization Point (ISP), which corresponds to the transition points in the OCPN. This kind of synchronization imposes a strict timing constraint among the multimedia objects. For isochronous data this may be too coarse. Interstream Pacing Points (IPP) provide the capability for a much finer grained synchronization by introducing an appropriate number of intermediate synchronization points between two ISPs. The type of media and the level of human perception involved dictates the interval between two ISPs.

The intrastream synchronization makes use of the concept of blocking and restricted blocking policies. In the blocking policy the playout action can be suspended until the late SIU arrives. In the restricted blocking policy the playout process blocks for a pre-specified period of time only while it waits for the current SIU to arrive. If a time-out occurs, the most recently stored SIU is played back, and the late SIU is skipped. This synchronization policy is not general enough to describe the requirements for some applications. For example, if restricted blocking is used to ensure the playout of a presentation continues, replaying the previous audio SIU may not be appropriate. Blocking and restricted blocking can also be applied to interstream synchronization.

2.3 Dynamic Timed Petri Net Model

The Dynamic Timed Petri Net (DTPN) approach [11] is an extension of the OCPN approach which supports the specification of user interactions. The DTPN extends the OCPN to model typical user inputs such as skip, reverse, freeze, restart, and scaling the speed of a presentation to make is faster or slower. Additional user interactions such as restart from the beginning, terminate, forward are not supported. User interactions are supported by allowing pre-emption and modification of the playout durations in the places. Recall that the OCPN model has fixed playout durations in the places. Processing is represented by places and synchronization points are represented by transitions. Firing transitions takes no time, as in the OCPN model. The DTPN model is simplified by encapsulating user interactions in a hierarchy.

The formal definition of the DTPN is a tuple = (T,P,A,D,R,M,C,E)

(P, T, A, D, R, M) defines the OCPN,

$C \subseteq P \times T$; $A \cap \mathrm{C} = 0$ C represents a set of escape arcs,

$E : \mathrm{P} \rightarrow R$ represents the remaining duration as a mapping from the set of places to the set of real numbers. E maintains the remaining duration for which execution is to be carried out. D maintains the 'normal' allowed duration. Initially D and E have the same value.

The execution of active places are pre-emptable and their execution duration values may be modified. When a place is pre-empted, its execution duration may be modified temporarily, modified permanently, terminated, or deferred. Temporary modification of the execution time refers to setting the execution duration for a particular instance of the execution. Permanent modification refers to setting the execution time duration for every activation of the petri net place. Termination of the execution is accomplished by zeroing out the execution duration for the place; the result is a premature ending for the execution. The deference of execution retains the execution duration value to reflect the amount of time that the execution has played out before it was pre-empted.

The data streams are classified according to which stream the user interaction is applied. The stream to which the interaction is applied is called the primary stream. The remaining are called secondary streams. The classification is dynamic, as a user interaction may be applied to more than one data stream in the representation. The secondary streams are classified into dependent or independent secondary streams, depending on their synchronization requirements with the primary stream. A dependent secondary stream is one where events in the secondary stream must occur before or in synchronization with the event in the primary stream. An independent secondary stream is one where no events in the secondary stream occur after the event in the primary stream. Note that the playout durations of the "independent" streams are affected by the user interaction operations. The "independent" secondary streams are those which do not have the pre-emption and termination operations applied, but do have playout duration adjustments made.

The user interactions and their impact on synchronizing data streams are summarized below. Figure 1 (a) pre-

sents the graphic notation for the composite of the skip ahead, freeze and restart, and scaling user interactions. Figure 1 (b) presents the notation for the reverse user interaction, which builds on the composite from 1 (a).

- skip ahead

  When a skip interaction occurs, the primary data stream and all the dependent secondary streams are pre-empted and terminated until the events of the dependent secondary stream synchronize with the primary stream. This specification approach assumes that a dependent secondary sources playout duration ends at the same time as the pre-empted and terminated place in the primary stream. The playout execution times of the independent data streams' events are decreased to synchronize with the primary stream.

- reverse

  When a reverse presentation interaction occurs, the primary data stream is pre-empted and terminated. The events of dependent secondary streams are pre-empted and terminated until they synchronize with the primary stream. The remaining duration of execution of the event in the independent secondary data streams are modified for the reverse presentation. The execution time is set to the difference between time spent till the pre-emption and the roll-back time for synchronizing with the pre-empted event in the primary stream. The presentation is played out in reverse. Once playing in reverse, there is no user interaction defined to return to playing out the presentation in the forward direction.

- freeze and restart

  When a freeze interaction occurs, the data streams are pre-empted and execution is deferred until a restart user interaction occurs. When a restart user interaction occurs, the presentation continues.

- scaling

  The speed of the presentation can be scaled by a factor to make the presentation faster or slower in its entirety. When a scale presentation speed interaction occurs, the data streams are pre-empted and permanently modified.

2.4 Time Stream Petri Net Model

The TSPN model [4,5] is a variant of Time Petri nets which uses labelled arcs and typed transitions to describe multimedia presentation scenarios. Like the XOCPN model, the TSPN approach aims at extending the OCPN by addressing the finer grained synchronization requirements for isochronous data. A set of firing rules is used to enforce the continuity of the presentation and to preserve the temporal semantics.

In the model, arcs leaving the places are labelled with a tuple [α, η, β] representing the earliest firing time (EFT), the nominal duration, and the latest firing time (LFT), respectively. This is used to define the maximum tolerable jitter. The places are the processing points and the transitions with more than one ingoing arc are typed and drive the interstream synchronization strategy.

Formally, a TSPN is defined as a tuple (P, T, B, F, M, IM, SYN) where:

(P, T, B, F, M) defines a marked Petri net,

$A = \{a = \langle p, t\rangle \in P \times T / (B(\langle p, t\rangle \neq 0))\}$ set of outgoing arcs,

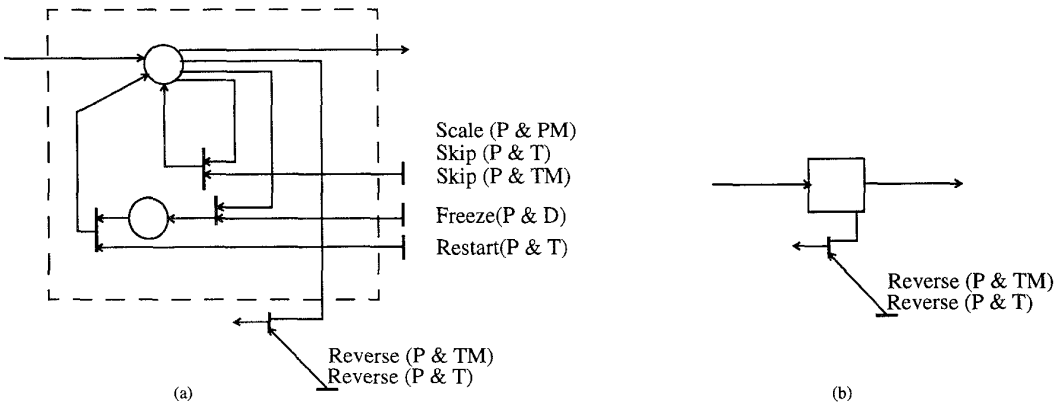
$IM : A \rightarrow Q^{+} \times Q^{+} \times \left(Q^{+} \cup \infty\right)$ IM maps arcs entering transitions into real numbers representing the EFT, the duration of the processing at the incoming place, and the LFT, respectively. The duration defined here has the same meaning as the duration described in the OCPN model,

$SYN : T \rightarrow$ {or, strong_or, and, weak_and, master, or-master, and-master, weak-master, strong-master} SYN is the typing function that distinguishes between the different firing rules (i.e. the different semantics) for interstream synchronization. Note that intrastream synchronization transitions are not typed, but use the standard Petri Net firing rules.

The model defines six synchronization time instants which are used to state time correctness. These instants can be distinguished between two points of view: a modelling and a run-time point of view. The synchronization instants include τ^{tok} the instant the token enters place p_i, τ^{min} the instant the token in place p_i reaches the minimum temporal bound, τ^{end} the instant the presentation associated with the place ends, τ^{max} the instant the token reaches the maximum temporal bound, τ^{tra} the instant when the transition becomes enabled, and τ^{fir} the instant the transition fires [5].

The firing rules for intrastream synchronization use these concepts to guarantee that, for each medium, $\tau^{fir} \in [\tau^{min}, \tau^{max}]$ even if $\tau^{end} \notin [\tau^{min}, \tau^{max}]$. In the case of interstream synchronization this condition must hold for any arc entering the transition, i.e., $\forall i, \tau_i^{end} \cap i[\tau_i^{min}, \tau_i^{max}]$. In the later case, if the condition doesn't hold, interstream strategies labelled at the transitions must be applied in order to preserve the temporal semantics of the stream.

Nine models for interstream synchronization are described in [5]: or, strong or, and, weak and, master, or-master, and-master, weak-master, and strong-master. The three basic models of interstream synchronization are provided by the strong or; weak and; and master transition types. In the strong-or strategy the entire presentation of one media is sufficient to satisfy the global presentation semantics. The first ending object triggers the synchroniza-



P & T: Pre-emption and Termination

P & D: Pre-emption and Deference

P & TM: Pre-emption and Temporary Modification

P & PM : Pre-emption and Permanent Modification

Figure 1. DTPN Model for User Interactions

(a) Combined User Interaction Operations in Forward Direction plus Reverse Interaction
(b) Hierarchy Replacement for Combined User Interaction Operations plus reverse interaction

tion and it induces the speed-up of later streams. If the event does not occur within the valid time interval, the transition fires to ensure the continuity of the presentation. In the weak-and strategy the last ending object triggers the synchronization. The entire presentation of all media is required to satisfy the global presentation semantics. If one or more of the triggers does not occur during the valid time interval, the transition fires. As in the previous case, the continuity of the presentation is ensured. The master strategy ensures the temporal correctness for the master stream. The entire presentation of at least one (the master) medium is necessary in order to satisfy the global presentation semantics. If the trigger does not occur within the valid time interval, the transition fires to insure the presentation continues.

3. Evaluation of the specification models

An example set of requirements for a multimedia presentation are specified in each of the four Petri net extensions described in Section 2. Due to their complexity, the XOCPN and the TSPN models are revised using standard Petri Net modelling techniques, to produce $XOCPN_2$ and $TSPN_2$. The modifications used are the inclusion of 'for loops' and hierarchy. The 'for loop' removes repetition from the model and encapsulates it conveniently for reuse. One diagram for each unique initialization counter is necessary. For example, separate diagrams are necessary for a counter initialized for looping 30 times and another initialized for looping 15 times (refer to Figure 2).

The criteria used for evaluating the specification models include the complexity of the model for describing the

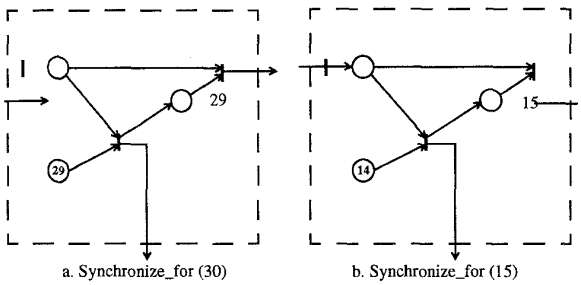


Figure 2. Synchronize_for(n) Diagrams

a. Synchronize_for (30)
b. Synchronize_for (15)

example and the ability to express the requirements of an application in the model. The metric for measuring the complexity of the model is the number of symbols in the specification used to describe the requirements. For the Petri Net models, the number of symbols is the sum of the number of transitions, arcs, places, and labels. For the natural language specification, the number of symbols is the line count in the specification. Although these measures of

complexity may be considered rough, they do provide an order of magnitude measurement for comparison purposes. A larger model is more difficult to manually review and analyze than a smaller, more concise model.Complete examples for each of the specification approaches are available in [7]. The results of the evaluation are summarized in Table 1. Since the original and modified versions of the XOCPN and TSPN models can describe the same requirements, only the modified versions are used in the following comparison. The modified versions simplify the specification by an order of magnitude.

Three clusters are described based on the number of symbols required to describe the example. The first group required less than fifty symbols; the second group requires between one hundred and three hundred symbols; the third group requires almost one thousand symbols. The first group includes the natural language approach. The natural language specification is very concise, and can specify all of the requirements in this example. Drawbacks of this approach are that the informal specification is not amenable to formal verification and the specification is open to interpretation during the development cycle.

The second category includes the OCPN, DTPN, and $TSPN_2$ approaches. The OCPN approach cannot describe the jitter, user interactions, or the fine grained synchronization requirements, but does allow the specification of the playout duration. The DTPN model has the same jitter and level of granularity deficiencies as the OCPN model, but does allow a partial description of the user interaction requirements. The user interactions the DTPN approach can model include skip ahead, scaling the presentation speed, freezing and resuming, and reversing the presentation playout. The DTPN model cannot model start, skip back, play forward once the presentation has been reversed, restart or terminate user interactions. The $TSPN_2$ approach can describe all of the requirements in the example except for the user interactions.

The third category contains the $XOCPN_2$ approach. The number of symbols is not distributed evenly between the transmitter and the receiver model. The transmitter requires 248 symbols, while the receiver requires 713 symbols. For the example worked out, the only requirement explicitly stated for the transmitting sites is that each data stream is provided by a unique site. This requirement is modelled twice in the XOCPN approach: once in the transmitter model and once in the receiver model. The $XOCPN_2$ specification cannot model the jitter requirements or the user interactions in the example.

Requirement	Synchronization Specification Models						
	OCPN	XOCPN	$XOCPN_2$	TSPN	$TSPN_2$	DTPN	Natural Language
Pre-orchestrated Source Data	yes	yes	yes	yes	yes	yes	yes
Source data time dependency	implicit	implicit	implicit	implicit	implicit	implicit	implicit
Source data kind of medium	yes	yes	yes	yes	yes	yes	yes
Multiple source sites	yes	yes	yes	yes	yes	yes	yes
Duration of Presentation	yes	yes	yes	yes	yes	yes	yes
Single destination site	yes	yes	yes	yes	yes	yes	yes
Jitter	no	no	no	yes	yes	no	yes
User Interaction	no	no	no	no	no	partial	yes
Level of granularity	too high	adequate	adequate	adequate	adequate	too high	adequate
Number of models	1	2	2	1	1	1	1
Total number of symbols	86	19332	961	1117	121	299	44

Table 1: Summary of Specifications

4. Conclusions and future work

Existing Petri net models for specifying synchronization constraints in distributed multimedia applications are reviewed in this paper. A fairly interesting distributed multimedia example is specified in OCPN, XOCPN, TSPN and DTPN, as well as in $XOCPN_2$ and $TSPN_2$, which are respecifications of the XOCPN and TSPN models, respectively, using the standard Petri net formalism. The $XOCPN_2$ and $TSPN_2$ specifications for the example are re-

duced in complexity by a factors of 20 and 9 (as compared to the XOPCN and TSPN specifications), respectively, without reducing their power to express requirements. Standard Petri Net modelling techniques including hierarchy and a 'for loop' are used to achieve the reduction in complexity. The reduction in complexities, however, are only based on one example, so additional work is necessary in order to generalize the results.

The natural language is the only approach powerful enough to describe all the requirements for the example. If a formal specification is required for proving safety and liveness properties, the TSPN approach is the best choice among the formal Petri net approaches, but it does not model the user interaction requirements. The only Petri net approach reviewed which supports the specification of user interactions is the DTPN approach. This approach is limited to a few user interactions, though, and cannot model all the user interactions in this example. If the DTPN approach is used, jitter constraints may not be modelled, and the level of granularity for synchronization control is too coarse to describe lip synchronization requirements.

The XOCPN approach produced a large, complex specification which did not model the example completely. The transmitter model does not add value to the specification for this example, as the one requirement it explicitly models is also modelled in the receiver model. Design details such as QoS addresses are also components of the specification model, which impose design constraints at an early stage of development.

Since the evaluation shows that $TSPN_2$ is the best choice among the Petri net approaches, except for its lack of support for user interactions, current work on its extension to allow for the specification of user interaction requirements is under way. Other useful extensions include support for multiple destination sites and transmission error handling.

Appendix A. Natural Language Specification

The presentation is composed of six video sequences (VS), three audio sequences (AS), two images (I), and two text strings (T). The presentation has a total duration of thirty seconds at the single destination site.
Each of the VS has a playout duration of five seconds.
Each of the AS has a playout duration of ten seconds.
Each of the I has a playout duration of ten seconds.
The maximum allowed jitter is 2 s for each image.
The first T has a playout duration of ten seconds.
The second T has a playout duration of twenty seconds.
The maximum allowed jitter is 2 s for each string.
The first VS, first AS, first I, and the first T all start to be presented at the same time.
When the first VS finishes, the second VS starts.
The second VS and the first AS finish at the same time. When they finish, the third VS and the second AS start at the same time.
When the third VS finishes, the fourth VS starts.
The fourth VS and the second AS finish at the same time. When they finish, the fifth VS and the third AS start at the same time.
When the fifth VS finishes, the sixth VS starts.
The AS and VS sequences need to be synchronized every one third of a second. The maximum allowed jitter is 75 ms for each synchronization point.
The first I and the first T finish at the same time. When they finish, the second I and the second T start at the same time.
When the second I finishes, the display area for the image is cleared for the remainder of the presentation.
The sixth VS, third AS, and the second T all finish being presented at the same time.
Each of the data streams originates from a unique site.
The data sources are all pre-orchestrated data.

The user requests to start the presentation. This user interaction is not allowed after a presentation has started being presented.
The user requests to skip ahead to the next presentation unit in a data stream. If the presentation is currently being played out in the forward direction, the presentation unit skipped to is the next presentation unit in the forward direction. If the current playout direction is in the reverse direction, then the presentation unit skipped to is the next presentation unit in the reverse direction. This user interaction is not allowed after a presentation has finished or been terminated. This user interaction is not allowed after a freeze request has been made and before a resume request is made.
The user requests to skip back to the previous presentation unit in a data stream. If the presentation is currently being played out in the forward direction, the presentation unit skipped to is the previous presentation unit in the reverse direction. If the current playout direction is in the reverse direction, then the presentation unit skipped to is the previous presentation unit in the forward direction. This user interaction is not allowed before a presentation has started to be presented.This user interaction is not allowed after a freeze request has been made and before a resume request is made.
The user requests to start the presentation from the beginning in the current playout direction. If the presentation is currently being played out in the forward direction, the presentation unit skipped to is the first presentation unit that is played out in the forward direction. If the current playout direction is in the reverse direction, then the presentation unit skipped to is the first presentation unit that is

played out in the reverse direction.This user interaction is not allowed before a presentation has started to be presented. This user interaction is not allowed after a freeze request has been made and before a resume request is made.
The user requests to terminate the presentation. After a termination request is requested, the user may only request to start the presentation.This user interaction is not allowed until a presentation has started to be presented.
The user requests to freeze the presentation. The playout of the presentation may resume when the user initiates a resume request. This user interaction is not allowed until a presentation has started to be presented.
The user requests to resume the presentation. The playout direction is the playout direction that was current when the presentation was stopped. The resume request is only allowed after a freeze request has been made.
The user requests to toggle the current presentation direction. If the current presentation direction is forward, the direction is changed to reverse. If the current presentation direction is reverse, the direction is changed to forward. This user interaction is not allowed until a presentation has started to be presented. This user interaction is not allowed after a freeze request has been made and before a resume request is made.
The user requests to scale the presentation speed of the presentation. The presentation may be scaled either slower or faster relative to the normal playout speed. The scaling factors allowed are: 1/4, 1/2, 3/4, 1, 5/4, 3/2, 7/4, where fractions less than one indicate slower presentation speeds, the factor one indicates the normal presentation speed, and fractions greater than one indicate faster presentation speeds. This user interaction is not allowed after a freeze request has been made and before a resume request is made
When the user interacts with the presentation, all components of the presentation respond to the request. For example, if a stop request is made, all the components stop being presented.

References

[1] Little T. and Ghafoor A., Network considerations for distributed multimedia object composition, IEEE Network Magazine, pp 32-49, November 1990.

[2] Little T. and Ghafoor A., Synchronization and storage models for multimedia objects, IEEE Journal on Selected Areas in Communications, pp 52-61, Volume 8, No 3, April 1990.

[3] Woo M., Qazi N. and Ghafoor A., A synchronization framework for communication of pre-orchestrated multimedia information, IEEE Network, January/February 1994

[4] Diaz M. and Senac P., Time stream petri nets: a model for multimedia streams synchronization, Proceedings of the International Conference on Multi-Media Modelling, Singapore, November 1993.

[5] Senac P., Diaz M., and Saqui-Sannes, P., Toward a formal specification of multimedia synchronization scenarios, Annals of telecommunications, pp 297-314, Volume 49, No 5-6, May, 1994.

[6] Hoepner P., Synchronizing the presentation of multimedia objects, Computer Communications, pp 557-564, Volume 15, No 9, November 1992.

[7] UBC CICSR Technical Report, Formal Methods for Modelling Multimedia Synchronization Requirements, to be published 1995.

[8] Horn, F. and Stephani, B., On Programming and Supporting Multimedia Object Presentation, The Computer Journal, pp 4-18, Volume 36, No1, 1993.

[9] Stefani, J.B., Hazard, L., and Horn, F., Computational Model for Distributed Multimedia applications based on a synchronous programming language, Computer Communications, pp 114-128, Volume 15, No 2, March 1992.

[10] Allen, J.F., Maintaining knowledge about temporal intervals, Communications of the ACM, pp832-843, Volume 26, No 11, November 1983.

[11] Prabhakaran, B and Raghavan, S.V., Synchronization Models For Multimedia Presentation With User Interaction, ACM Multimedia '93, California, USA, June, 1993, pp 157-166.

[12] Shepherd, D. and Salmony, M., Extending OSI to support synchronization required by multimedia applications, Computer Communications, pp 399-406, Volume 13, May, 1990.

[13] Sreenan, J.C., Synchronisation Services for Digital Continuous Media, Ph.D. thesis, Christ's College, University of Cambridge, 1992.

[14] Anderson, D.P. and Homesy, G.A., A continuous I/o server and its synchronization mechanisms, IEEE Computer, pp 51-57, Volume 24, No 10, October, 1991,

[15] Escobar, J., Deutsch, D. and Partridge, C., Flow synchronization protocol, Internal report, BBN, 10 Moulton Street, Cambridge MA, 1992.

[16] Li, L. and Karmouch, A., Georganas, N.D., Real-time synchronization control in multimedia distributed systems, Proceedings IEEE ICC '92, Chicago, June, 1992.

[17] Murata, T., Petri nets: properties, analysis, and applications. Proceedings of the IEEE, pp541-580, Volume 77, No 4, April, 1989.

[18] Regan, T., Multimedia in Temporal LOTOS: a Lip-Synchronization Algorithm, pp 127-142, Protocol Specification, Testing and Verification, XII, eds. Dantine, A., Leduc, G., and Wolper, P., North-Holland, 1993 IFIP.