

Provided for non-commercial research and educational use only.
Not for reproduction or distribution or commercial use.



This article was originally published in a journal published by Elsevier, and the attached copy is provided by Elsevier for the author's benefit and for the benefit of the author's institution, for non-commercial research and educational use including without limitation use in instruction at your institution, sending it to specific colleagues that you know, and providing a copy to your institution's administrator.

All other uses, reproduction and distribution, including without limitation commercial reprints, selling or licensing copies or access, or posting on open internet sites, your personal or institution's website or repository, are prohibited. For exceptions, permission may be sought for such use through Elsevier's permissions site at:

<http://www.elsevier.com/locate/permissionusematerial>



ELSEVIER

Available online at www.sciencedirect.com

 ScienceDirect

The Journal of Systems and Software 80 (2007) 1142–1155

 **The Journal of
Systems and
Software**

www.elsevier.com/locate/jss

Synchronization modeling and its application for SMIL2.0 presentations

Chun-Chuan Yang *, Yung-Chi Wang, Chih-Wen Tien

*Multimedia and Communications Laboratory, Department of Computer Science and Information Engineering, National Chi-Nan University,
1, University Road, Puli, Nantou County 545, Taiwan, ROC*

Received 25 November 2005; received in revised form 30 September 2006; accepted 30 September 2006

Available online 13 November 2006

Abstract

A novel synchronization model namely Extended Real-Time Synchronization Model (E-RTSM) for modeling SMIL2.0 temporal behaviors is proposed in this paper. E-RTSM deals with event-based/non-deterministic synchronization as well as schedule-based synchronization in SMIL2.0. Converting of the temporal relationship of a SMIL2.0 document to E-RTSM is presented. Moreover, design of the E-RTSM-based data-retrieving engine for SMIL2.0 presentations is also proposed in the paper. The data-retrieving engine estimates the worst-case playback time of each object at the parsing stage and applying an error compensation mechanism at run-time to adjust the estimated playback time as well as the schedule of the fetching requests for data retrieval. Performance measurements from the real implementation of the E-RTSM-based data-retrieving engine for SMIL2.0 presentations have demonstrated the efficiency of the proposed technique.

© 2006 Elsevier Inc. All rights reserved.

Keywords: Synchronized multimedia; SMIL2.0; Real-time synchronization model

1. Introduction

Multimedia presentation provides a good tool for business demonstrations as well as distance learning. Multimedia presentation is concerning with the integration of multimedia objects, which may be located at remote data servers. *Synchronized Multimedia Integration Language (SMIL)* (*Synchronized Multimedia Integration Language, 1998, 2001; Bulterman, 2001, 2002*) was developed by *WWW Consortium* to address the lack of HTML for multimedia over WWW. With the introduction of SMIL, Web multimedia creators have a new tool for building time-based multimedia presentations that combine audio, video, images, animations, text, etc. There are two versions of SMIL specification that had been released. The first version of SMIL (SMIL1.0) (*Synchronized Multimedia Integration Language, 1998*) is primarily a scheduling model, but

with some flexibility to support continuous media with unknown duration. The second version of SMIL (SMIL2.0) (*Synchronized Multimedia Integration Language, 2001; Bulterman, 2001, 2002*) enhances SMIL1.0 by providing a strong support for user interaction with a declarative event-based timing. Event-based timing in SMIL2.0 presents the non-deterministic synchronization behavior such that the player cannot get the accurate playback time (and duration) for a media object as well as the total length of a presentation before run-time.

In our previous work, we had developed modeling and converting techniques for parsing SMIL1.0 documents. *Real-Time Synchronization Model (RTSM)* (*Yang and Huang, 1996*) was used for modeling the temporal relationship in SMIL1.0 documents. Based on RTSM, an efficient data-retrieving engine was proposed and implemented (*Yang and Yang, 2004*). However, the proposed techniques in our previous work cannot be applied to SMIL2.0 directly. Extensions of RTSM as well as the modification of the converting algorithm are necessary to cope with

* Corresponding author.

E-mail address: ccyang@csie.ncnu.edu.tw (C.-C. Yang).

non-deterministic features in SMIL2.0. In this paper, we propose *Extended RTSM (E-RTSM)* and the converting algorithm to present the temporal relationship in a SMIL2.0 document.

The application of E-RTSM modeling for SMIL2.0 may include: (1) design of the SMIL2.0 player (Yang et al., 2003a), (2) design of the data-retrieving mechanism in SMIL2.0 players (Yang et al., 2003b), and (3) design of the authoring tools for SMIL2.0 presentations (Yang et al., 2004a,b). While viewing a multimedia presentation via a SMIL2.0 player, the user should be able to perform *VCR-like control functions* such as *play/stop*, *pause/resume*, *fast forward/backward*, and *sliding*. Most of the commercial products of SMIL2.0 player as well as the academic proposals only support simple operations such as play/stop and pause/resume. It seems none of them provides fast forward/backward and sliding operations for SMIL2.0 presentations. The reason is the non-deterministic playback behavior due to event-based timing in SMIL2.0 makes it difficult to provide the advanced VCR-like functions. Therefore, a good design of the SMIL2.0 player tackling the non-deterministic characteristic and providing proper VCR-like functions is a real challenge.

For SMIL2.0 authoring, the direct way to create a SMIL2.0 document is to use to text editor and starting writing SMIL tags as most of the programmers do. For non-professionals, it is much better to have an authoring system that helps users compose SMIL documents in a visualized (WYSIWYG) way. Two major categories of visualized SMIL1.0 authoring are (1) structure-based editing, and (2) timeline-based editing. Structure-based editing is primarily based on the visualization of SMIL temporal relations, and users need to organize nested `<seq>` and `<par>` blocks. On the other hand, timeline-based editing hides the language structure of SMIL by visualizing the playback time and duration of each object in the timeline manner providing users a more intuitive way to understand and easily control the timing of each object. Since traditional timeline-based editing can only provide schedule-based presentations and is insufficient for SMIL2.0 authoring, we are working towards the extension of non-determinism to timeline-based editing (Yang et al., 2004a). In this

paper, we focus on the application of E-RTSM modeling in the design and implementation of SMIL2.0 data-retrieving engine.

The remainder of the paper is organized as follows. First of all, E-RTSM is presented in Section 2. The converting algorithm from SMIL2.0 to E-RTSM is presented in Section 3. E-RTSM-based data-retrieving engine designed for SMIL2.0 presentations is presented in Section 4. Implementation of the proposed data-retrieving engine and performance measurement are presented in Section 5. Related work of multimedia synchronization as well as SMIL modeling is discussed in Section 6. Finally, Section 7 concludes this paper.

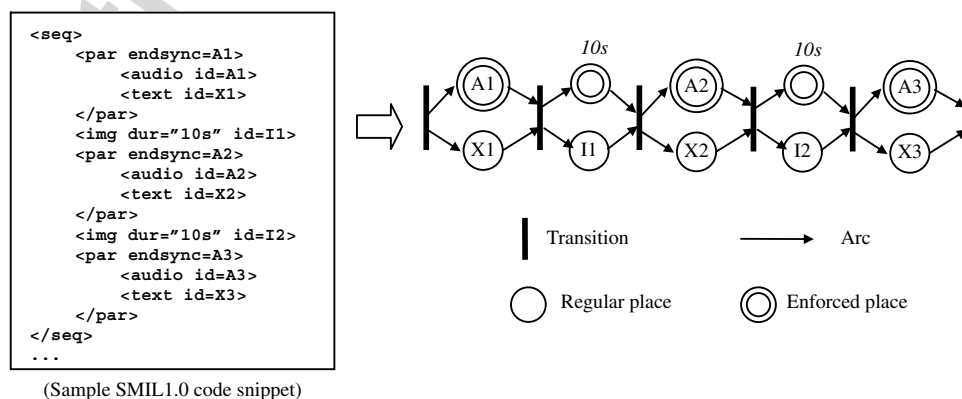
2. Extended Real-Time Synchronization Model (E-RTSM)

2.1. Brief survey of RTSM

RTSM was proposed to address the lack of *Petri net* based models such as *OCPN (Object Composition Petri Net)* (Little and Ghafoor, 1989) for dealing with real-time synchronization. There are two kinds of places in RTSM, *regular places* and *enforced places*. The firing rule of RTSM specifies that once an enforced place becomes unblocked (i.e. related action associated with the place is completed), the following transition will be immediately fired regardless the states of other places feeding the same transition. With the enforced firing rule, temporal relationship of objects in a SMIL1.0 document can be easily represented by RTSM. A sample SMIL1.0 code snippet and its associated RTSM are displayed in Fig. 1, in which an enforced place (denoted by a *double circle* in the figure) can be mapped to a medium unit (e.g. A1, A2, and A3) or a time unit (e.g. 10 s). Please refer to our previous work (Yang and Huang, 1996) for more detailed definition and properties of RTSM.

2.2. E-RTSM

Major differences between SMIL1.0 and SMIL2.0 in timing control include: (1) Values of `<begin>` and `<end>` attributes for an object (or time containers `<par>` and `<seq>`) can be non-deterministic events, i.e. events with unknown



(Sample SMIL1.0 code snippet)

Fig. 1. An example of RTSM.

occurring times such as *Mouse-Click* events or *Key-Pressed* events. (2) Multiple values for $\langle begin \rangle$ and $\langle end \rangle$ attributes are allowable for media objects and time containers, i.e. the start or the end of a SMIL2.0 object can be controlled by more than one event. (3) Some complicated synchronization features such as $\langle restart \rangle$ and $\langle min/max \rangle$ attributes are also defined in SMIL2.0.

In order to cope with the non-deterministic synchronization behaviors of SMIL2.0, two new features are added in E-RTSM: (1) allowing a place in E-RTSM to be mapped to a non-deterministic event (denoted by a “?” in a place). (2) *Run-time controllers* for complicated synchronization features are defined.

Introducing association of non-deterministic events with E-RTSM places increases the flexibility of the model, but it also increases the difficulty in processing the model, such as the estimation of the firing time of each transition that is addressed in Section 4. On the other hand, as presented in Section 3, a non-deterministic event is always associated with an enforced place in the application of converting SMIL2.0 scripts to E-RTSM. However, from the viewpoint of modeling, a regular place can also be mapped to a non-deterministic event, and in such case the non-deterministic event is not dominating the firing of the following transition.

Run-time controllers are used to model complicated timing features in SMIL2.0 that are difficult or impossible to be represented by the combination of other basic elements (arc, transition, and place). A run-time controller can be placed in between any two transitions (the start transition and the end transition) as places in E-RTSM. *Bi-directional arcs* are used to connect the start transition to the run-time controller and the run-time controller to the end transition. A run-time controller is associated with a set of rules that control the firing of the start and the end transitions. Therefore, the operation of a run-time controller overrides the operation of the places/transitions in between the start and the end transitions of the run-time controller.

By using run-time controllers in E-RTSM, handling of these complicated timing features is delayed until run-time rather than the parsing phase. More specifically, the play-

back process of a SMIL2.0 document is divided into the parsing phase and run-time phase in this paper, in which the player parses and converts the script into E-RTSM in the parsing phase and perform the actual playback in the run-time phase. Three run-time controllers are defined in E-RTSM: *Restart* controller, *Min* controller, and *Repeat* controller. Usage of these run-time controllers in modeling SMIL2.0 elements is presented in the next section. For completeness, we give the list of the terminologies used in E-RTSM in Table 1. The complete definition of E-RTSM is given as follows:

Definition 1. E-RTSM is a 10-tuple $\{T, P, E, \underline{R}, A, \underline{B}, D, M, \underline{N}, X\}$, where (*Note that the differences between E-RTSM and RTSM are underlined.*)













$T = \{t_1, t_2, \dots, t_n\}$ transitions
 $P = \{p_1, p_2, \dots, p_m\}$ regular places (single circles)
 $E = \{e_1, e_2, \dots, e_k\}$ enforced places (double circles)
 $S = P \cup E$ all places
 $R = \{r_1, r_2, \dots, r_i\}$ run-time controllers
 $A = \{T \times S\} \cup \{S \times T\}$ unidirectional arcs
 $B = \{T \times R\} \cup \{R \times T\}$ bi-directional arcs
 $D = S \rightarrow$ Real number time duration of places
 $M = S \rightarrow \{m_1, m_2, \dots, m_j\}$ regular types of medium
 $N = S \rightarrow \{n_1, n_2, \dots, n_i\}$ non-deterministic events
 $X = S \rightarrow \{0, 1, 2\}$ state of places

Each place may be in one of the following states:

- 0: no token
- 1: token is blocked A “cross” in the place
- 2: token is unblocked A “dot” in the place

The firing rules of E-RTSM are the same as those of RTSM in the absence of run-time controllers. When a run-time controller is presented between two transitions, the firing of the transitions (the start transition and the end transition) is controlled by the run-time controller, which can override the firing rules associated with places.

Table 1
Terminology used in E-RTSM

Terminology	Graph notation	Meaning
Regular place		Representation of a medium object or time duration
Enforced place		A specialized place that dominates the firing of transition
Transition		Synchronization control for a group of places
Run-time controller		Modeling of the medium object with complicated timing attributes
Unidirectional arc		Used for connecting a place to a transition
Bi-directional arc		Used for connecting a run-time controller between two transitions
Place is blocked	 or 	The action associated with the place is unfinished
Place is unblocked	 or 	The action associated with the place is finished
Non-deterministic event	 or 	Places that map to non-deterministic events like mouse-click

3. Converting SMIL2.0 to E-RTSM

In this section, we focus on converting major differences between SMIL2.0 and SMIL1.0 that are mentioned in Section 2. The rest of converting of SMIL2.0 is similar to that of SMIL1.0 in our previous work (Yang and Yang, 2004).

3.1. Converting begin and end attributes

We classify the value of $\langle begin \rangle$ and $\langle end \rangle$ attributes to two types of event values: *Time value event* (event with known occurring time, e.g., *Clock-value*) and *Non-deterministic event* (*Mouse-Click*, *Key-Pressed*, etc). A time value event is converted to an enforced place with duration specified by the event as in SMIL1.0 conversion. On the other hand, a non-deterministic event is converted to an enforced place that maps to that event.

E-RTSM for an element (media element or time container) with multiple $\langle begin \rangle$ values and $\langle end \rangle$ values is illustrated in Fig. 2. Note that in the figure, the start transition of non-deterministic event values in *End-value-list* is different from that of time value events. The reason is: SMIL2.0 specifies that a non-deterministic event in *End-value-list* does not have any effect on an element until the element has been activated.

3.2. Using run-time controllers

Currently, three run-time controllers as displayed in Fig. 3 are defined in E-RTSM: *Restart controller*, *Min controller*, and *Repeat controller*. SMIL2.0 allows an element to be restarted multiple times during the element’s active duration. The behavior is controlled by the $\langle restart \rangle$ attribute. Restart controller is used when the value of the $\langle restart \rangle$ attribute of an element equals “*always*” or “*whenNotActive*”.

SMIL2.0 also allows the author to control the lower and upper bound of the element active duration by using the $\langle min/max \rangle$ attributes. Min controller is used when the $\langle min \rangle$

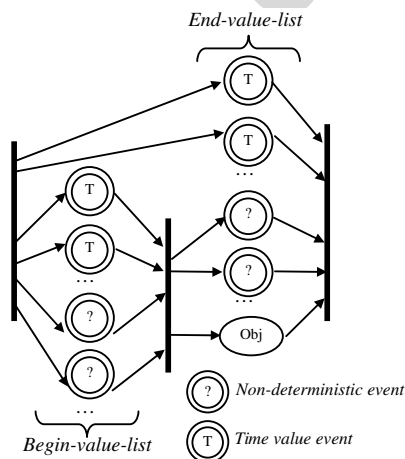


Fig. 2. Converting *Begin-value-list* and *End-value-list*.

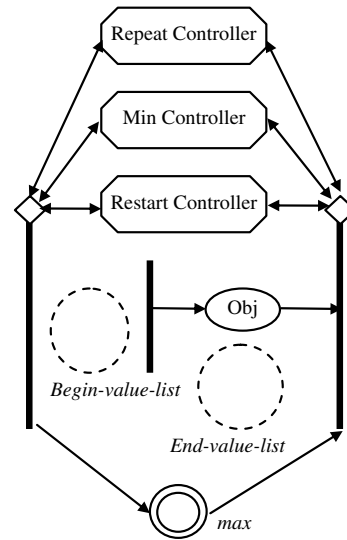


Fig. 3. Run-time controllers in E-RTSM.

attribute is presented for an element. The effect of the $\langle max \rangle$ attribute is similar to that of a time value event in *End-value-list* as shown in Fig. 3. Repeat controller is used when either the $\langle repeatCount \rangle$ attribute or the $\langle repeatDur \rangle$ attribute is presented for an element.

3.3. An example of the conversion

The sample SMIL2.0 code snippet in Fig. 4 is used to illustrate the converting process. Note that the sample SMIL2.0 code snippet is similar to the sample in Fig. 1 except some non-deterministic events and a $\langle min \rangle$ attribute are presented. Text objects X1, X2, and X3 in the code snippet of Fig. 4 are displayed only when button *Btn1* is clicked during the playback of audio objects A1, A2, and A3 respectively. Image objects I1 and I2 are displayed for 10 s after the playback of A1 and A2, but the viewer can terminate the display of the images before reaching 10 s by clicking button *Btn2*. The $\langle min \rangle$ attribute with value

```

<seq>
  <par endsync=A1>
    <audio id=A1>
      <text begin="Btn1.Click" id=X1>
    </par>
  <img end="Btn2.Click dur="10s" id=I1>
  <par endsync=A2>
    <audio min="15s" id=A2>
      <text begin="Btn1.Click" id=X2>
    </par>
  <img end="Btn2.Click dur="10s" id=I2>
  <par endsync=A3>
    <audio id=A3>
      <text begin="Btn1.Click" id=X3>
    </par>
</seq>
...
    
```

Fig. 4. A SMIL2.0 code snippet.

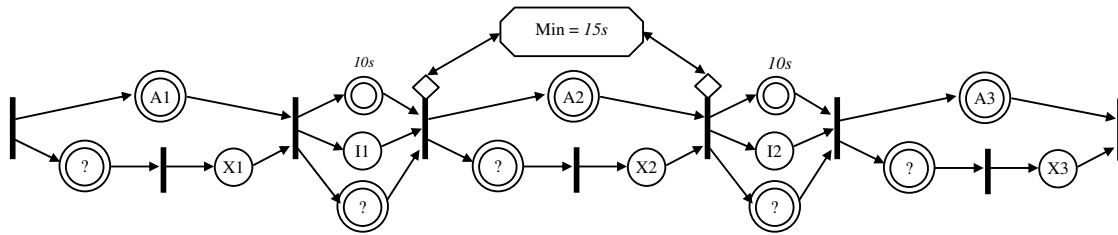


Fig. 5. E-RTSM for the sample SMIL2.0 code snippet.

of 15 s is associated with audio object A2 requiring A2 to be played at least 15 s. The E-RTSM for the sample SMIL2.0 code snippet is displayed in Fig. 5.

4. Design of the SMIL2.0 data-retrieving engine

4.1. Architecture of the engine

The data-retrieving engine is responsible for retrieving proper media data for the playback of a presentation. The concept of just-in-time data retrieving was proposed in our previous work for SMIL1.0 presentations (Yang and Yang, 2004). As illustrated in Fig. 6, just-in-time data retrieving expects the retrieval process for an object to be finished right before the playback time of the object so that the player could continue the presentation smoothly. Thus, the data-retrieving engine needs to compute the playback time for each object. The request time for an object is then calculated according to its playback time and bandwidth estimation.

For SMIL2.0 presentations, the accurate playback time for an object cannot be obtained before run-time because of the non-deterministic events. Instead, the data-retrieving engine calculates the worst-case playback time for each object at the parsing stage, and at run-time applying an error compensation mechanism for adjusting the estimated playback time as well as the request time. Note that the worst-case playback time of an object is defined from the perspective of data-retrieval to be the earliest playback time of the object by assigning zero time duration to each non-deterministic event (i.e. the non-deterministic event occurs instantaneously) in the script. In the actual playback of the document (run-time), the playback time of each object is revised according to the actual occurrence time of the non-deterministic events, and the revised playback

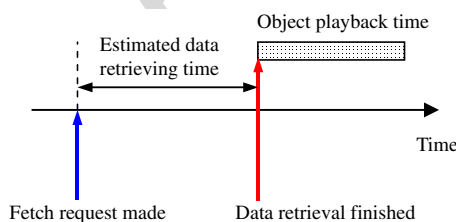


Fig. 6. Just-in-time data retrieval.

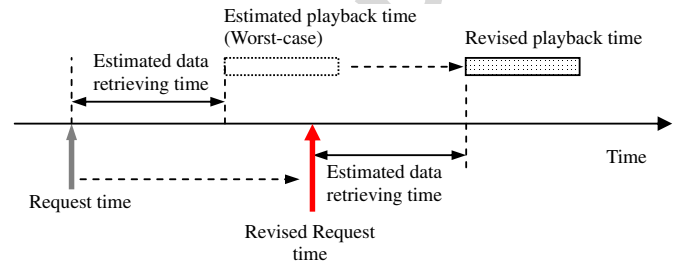


Fig. 7. Error compensation for data retrieval.

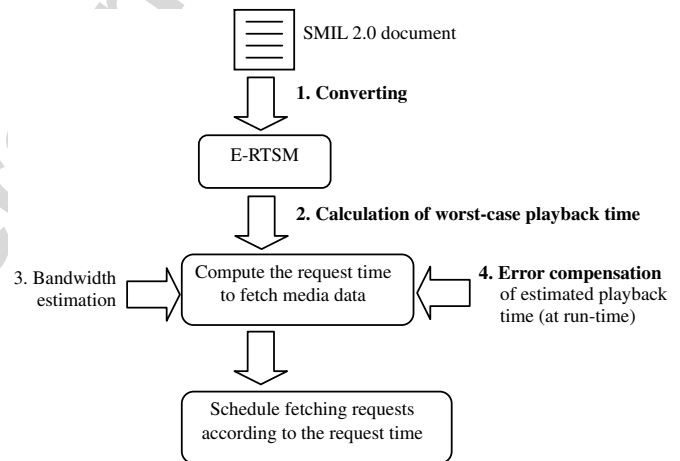


Fig. 8. Data retrieving for SMIL2.0 documents.

time of an object is used for revising the request time of the object as illustrated in Fig. 7. Overview of the proposed data-retrieving process for SMIL2.0 documents is illustrated in Fig. 8.

4.2. Worst-case estimation for the playback time

At the parsing stage, the input document is first converted to E-RTSM. The data-retrieving engine reduces E-RTSM by removing the regular places that have no effect on the firing time of a transition. As an example, Fig. 9 is the reduced E-RTSM for Fig. 5. The data-retrieving engine calculates the worst-case playback time for each object by assigning the duration of all non-deterministic events to zero and traversing the reduced E-RTSM.

The playback time for an object is the firing time of its start transition. There are only two possibilities for a tran-

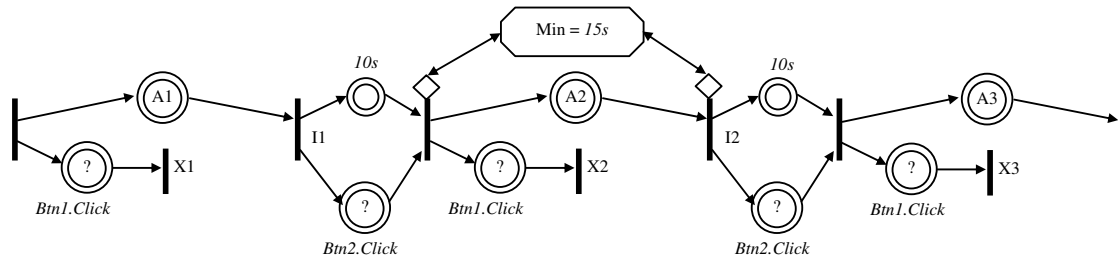


Fig. 9. Reduced E-RTSM for the sample SMIL2.0 code snippet.

sition without considering run-time controllers in the reduced E-RTSM: (1) places feeding to the transition are all enforced places, or (2) places feeding to the transition are all regular places. For case (1), the firing time of the transition is the minimal value of “the firing time of the preceding transition” + “the duration of the following place”, which is illustrated in Fig. 10(a). The firing time of the transition for case (2) is instead the maximum value of its predecessors as illustrated in Fig. 10(b). The duration of each place depends on the type of the media object. For static media objects such as ** and *<text>*, the duration of the place is zero. For continuous media object like *<audio>* and *<video>*, the duration of the place is the implicit duration of the object that is provided by the data server. Since the objects stored in a data server are all pre-orchestrated, it is easy for the data server to obtain the implicit duration of a continuous object. As mentioned in the last paragraph, the duration of places that map to a non-deterministic event is set zero in the worst-case calculation.

During the traversal process of calculating the firing time of each transition, we also need to deal with the run-time controllers to get more accurate values for the worst-case playback time. The Restart controller deals with events that could restart an element during the active duration of the element, so the worst case would be no restart at all. Thus, the restart controller is ignored in the worst-case calculation. The Min controller specifies a lower bound of the duration between the start transition and the end transition of an element, so the firing time of the end transition should be updated if the estimated duration is smaller than the *<min>* value. The Repeat controller deals with the

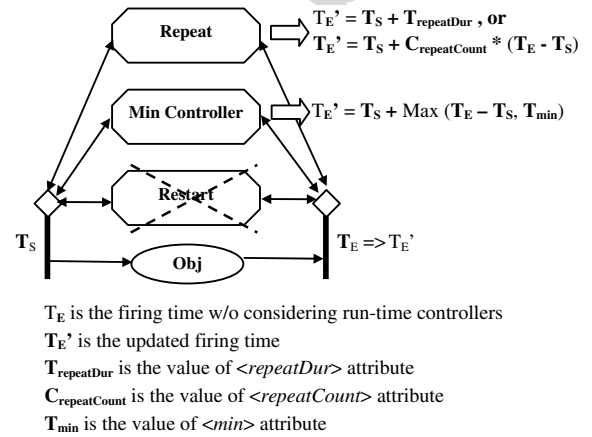


Fig. 11. Impact of the run-time controllers on the worst-case firing time.

<repeatDur> attribute as well as the *<repeatCount>* attribute. The *<repeatDur>* attribute sets the duration of repeating an element, so the firing time of the end transition should be the end of the repeat duration. The *<repeatCount>* attribute specifies times of repeating for an element, thus the final firing time of the end transition is extended as many times as specified by the *<repeatCount>* value. Update of the firing time of a transition with the run-time controllers is illustrated in Fig. 11.

For example, assuming the intrinsic durations for A1, A2, and A3 are all 10 s, the worst-case playback time for each object in Fig. 9 is A1 = 0 s, X1 = 0 s, I1 = 10 s, A2 = 10 s, X2 = 10 s, I2 = 25 s, A3 = 25 s, X3 = 25 s. (Note that these values are relative to the start time of the presentation).

4.3. Calculation of the object request time

The objects that should be retrieved (played) depend on the user action since different user actions (e.g. *Fast Forward/Backward*, *Sliding*, etc.) result in different playback patterns and different playback times of objects. In this paper, we only present the case of normal playback after the SMIL script has been loaded at the player for simplicity. Moreover, in order to determine the object request time, the data-retrieving engine has to collect some meta-information of each object from the data server. Therefore, when the data-retrieving engine accepts the SMIL script, it sends probe packets to all data servers to collect the object

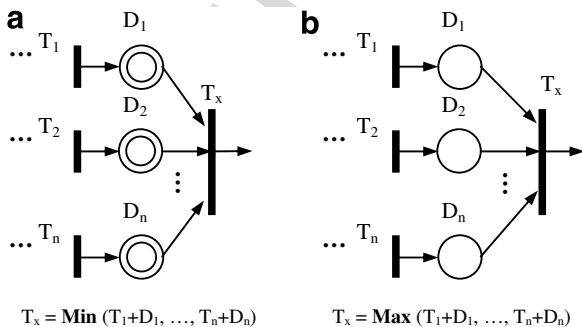


Fig. 10. Determine the firing time for transition T_x .

information, which includes *object size* and *estimated bandwidth*. We denote the object size for object $URI-i$ as $Size_{URI-i}$, the estimated bandwidth as $EstBW_{URI-i}$.

The data server estimates the bandwidth for transmitting the object to the data-retrieving engine. The *estimated bandwidth* is used to estimate the total delay of an object from the server to the client. There are two factors that affect the item: (1) the transmission rate of the server for the requested object, and (2) the effective network bandwidth of the path from the server to the client. The transmission rate depends on the load of the server and the capacity of the outgoing link. The effective network bandwidth can be measured by bandwidth measuring mechanisms (Bolliger et al., 1999; Lai et al., 1999; Prasad et al., 2003; Liu et al., 2003; Huang et al., 2004). Discussion on the estimation of the end-to-end bandwidth for retrieving an object is beyond the scope of the paper. If the server cannot provide information about the estimated bandwidth, it should inform the client to perform the estimation by itself.

In addition to object related information, the data-retrieving engine also has to estimate the time for the request packet arrived to the data server. We use the round trip delay, denoted by $RTDelay_{URI-i}$ as the estimated value for the delay of the request packet to the server. Thus, the total time to retrieve an object is the summation of the delay of the request packet and the transmission time of the object from the data server to the client site. That is, the retrieving time for object $URI-i$ is estimated as $(Size_{URI-i}/EstBW_{URI-i}) + RTDelay_{URI-i}$. If the data server supports the streaming mechanism for continuous objects, it is not necessary to retrieve the whole content of the object before its playback time. Only the amount of data to support the streaming operation is required. Thus, the transmission time of the object is $BufferSize_{URI-i}/EstBW_{URI-i}$, in which $BufferSize_{URI-i}$ is the amount of data to buffer. The value of $BufferSize_{URI-i}$ depends on the streaming operation and is not addressed in this paper.

The accuracy of end-to-end bandwidth estimation affects the performance of the data-retrieving engine as well as the quality of the presentation. Since the network behavior is very dynamic, it is impossible to exactly estimate the time required to finish the retrieving process for a media object. We discuss the impact of the accuracy of estimated time to finish the retrieving process for a media object on the performance of the data-retrieving engine.

If the estimated time is more pessimistic (bandwidth is underestimated) than the actual status, the object will be buffered for some time before its playback time. On the other hand, if the estimated time is more optimistic (bandwidth is overestimated) than the actual status (e.g. network is congested), the presentation will probably be paused to wait for the object. Furthermore, if the network bandwidth could be reserved in advance by some booking method, the estimated object request time will be more precise. Hence, the quality of the presentation and the buffer utilization will also be improved.

4.4. Run-time error compensation

Apparently, the actual playback time for an object at run-time is no earlier than the worst-case estimation. The difference (error of estimation) between the actual playback time and the worst-case estimation can be used to adjust the estimated playback time of the following objects that are not played yet. For example, if the event *Btn2.Click* in Fig. 9 has occurred when *I1* has been played 8 s, we can then update the estimated playback time of the following objects: $A2 = 18$ s, $X2 = 18$ s, $I2 = 33$ s, $A3 = 33$ s, $X3 = 33$ s. New estimation of the playback time is then used to re-calculate the new request time. The new request time for an object may not change the schedule of the object's fetching request since the request probably had already been issued to retrieve the data. But the error compensation mechanism does make the estimated playback time of later objects more close to reality and it also makes data retrieval more intelligent.

5. Implementation and performance measurement

We have implemented an experimental system for the feasibility and performance evaluation of the proposed modeling technique as well as the data-retrieving engine. The network environment for the experimental system is shown in Fig. 12. There are three data servers in the system. The performance criterion is the jitters between the playback time and the arrival time of the object (i.e. $jitter = actual\ playback\ time - arrival\ time$). A positive value of the jitters implies the buffering time for the object before playback, while a negative value of the jitters indicates the pause time of the playback to wait for the object to arrive. Thus, a positive value of the jitters closer to zero implies a better performance.

In addition to the *just-in-time (JIT)* policy for data retrieving, we also implemented two extreme policies for performance comparison: *pre-loading* policy and *passive-loading* policy. In the pre-loading policy, the data-retrieving engine retrieves all objects before starting the presentation. Hence, the pre-loading policy guarantees the smooth playback in the cost of a long initial delay and a large buffer for all objects in the presentation. On the other hand, the passive-loading policy does not make the request to retrieve the object until the object's playback

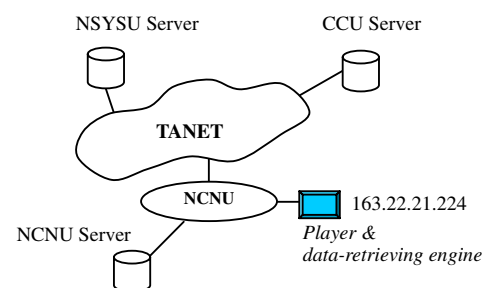


Fig. 12. Network environment for the experimental system.

time. Thus, minimal buffer is required for the passive-loading policy, but it introduces gaps between the request time and the finish time of data retrieval because of the network delay. It implies that the smooth playback is impossible for the passive-loading policy.

Since the data servers cannot provide the advanced service like bandwidth estimation, the data-retrieving engine in our implementation has to estimate the bandwidth by itself. In the just-in-time retrieving policy, the data-retrieving engine measures the average bandwidth from the data server to the client by requesting some test files from the server. The *EstBW* obtained is then used in the calculation of the object request time as presented in Section 4.

A test SMIL2.0 document (denoted by T1) for performance measurement is displayed in Fig. 13, in which 8 text objects, 6 images objects, and 3 audio objects are included in the presentation. As shown in Fig. 14, the size of the audio objects is above 1 Mbytes, the size of the image objects is within the range of 100–200 Kbytes, and the size of the text objects is within the range of 1–20 Kbytes. In order to model the user behavior in viewing the document, the occurrence of the non-deterministic “*mouse.Click*” event is controlled by two random numbers. The first random number determines whether the event happens or not, and the second random numbers determines the occurring time of the event, which is uniformly distributed in between 0 and 10 s.

Fig. 15 shows the average jitters of 10 measurements for the test SMIL2.0 documents (T1). Note that the error compensation mechanism is not applied to the JIT policy in Fig. 15. The pre-loading policy retrieves all the objects before starting the playback of the presentation, so the later the playback time of an object (which has a larger ID in the test SMIL2.0 script T1), the larger jitters (more buffering time) the object will experience. Therefore, the jitters for the pre-loading policy form a monotonic ascending curve in the figure. On the other hand, the jitters for the passive-loading policy are always negative (although some of the values are close to zero) as shown in the figure, and the value of the jitters depends on the traffic condition and the size of the requested object. Therefore, the proposed JIT policy is better than the two extreme policies in terms of both fewer buffers and smooth playback.

Fig. 16 shows the jitter results of the JIT policy with and without error compensation respectively. The jitter curve of the JIT policy without error compensation tends to go up for objects with later playback time because of the worst-case estimation of the playback time. On the other hand, the proposed error compensation mechanism demonstrates the benefit of run-time amendment and saves more buffers. Fig. 17 displays the buffer utilization (the size of the required buffers as a function of time for the test document) during the worst-case playback process for each scheme. As we expected, the pre-loading scheme requires the most

```

<body>
  <seq>
    <text id="x1" dur="5s" src="http://studentweb.ncnu.edu.tw/91321516/picture/x8.txt">
      <par>
        
        <text id="x2" src="http://studentweb.ncnu.edu.tw/91321516/picture/x4.txt">
      </par>
      <par endsync="A3">
        <audio id="A3" dur="10s" begin="mouse.Click" src="http://www.cs.ccu.edu.tw/~hyhs91/temp/a2.wav">
        
        <text id="x3" src="http://studentweb.ncnu.edu.tw/91321516/picture/x5.txt">
      </par>
      <par>
        
        <text id="x4" src="http://dip2.cse.nsysu.edu.tw/~dio/x1.txt">
      </par>
      <par endsync="A5">
        <audio id="A5" dur="10s" end="mouse.Click" src="http://dip2.cse.nsysu.edu.tw/~dio/a1.wav">
        
        <text id="x5" src="http://studentweb.ncnu.edu.tw/91321516/picture/x6.txt">
      </par>
      <par>
        
        <text id="x6" src="http://studentweb.ncnu.edu.tw/91321516/picture/x7.txt">
      </par>
      <par endsync="A7">
        <audio id="A7" begin="mouse.Click" src="http://studentweb.ncnu.edu.tw/91321516/picture/a3.wav">
        
        <text id="x7" src="http://www.cs.ccu.edu.tw/~hyhs91/temp/x2.txt">
      </par>
      <text id="x8" dur="5s" src="http://dip2.cse.nsysu.edu.tw/~dio/x3.txt">
    </seq>
  </body>

```

Fig. 13. The code snippet of test SMIL2.0 document T1.

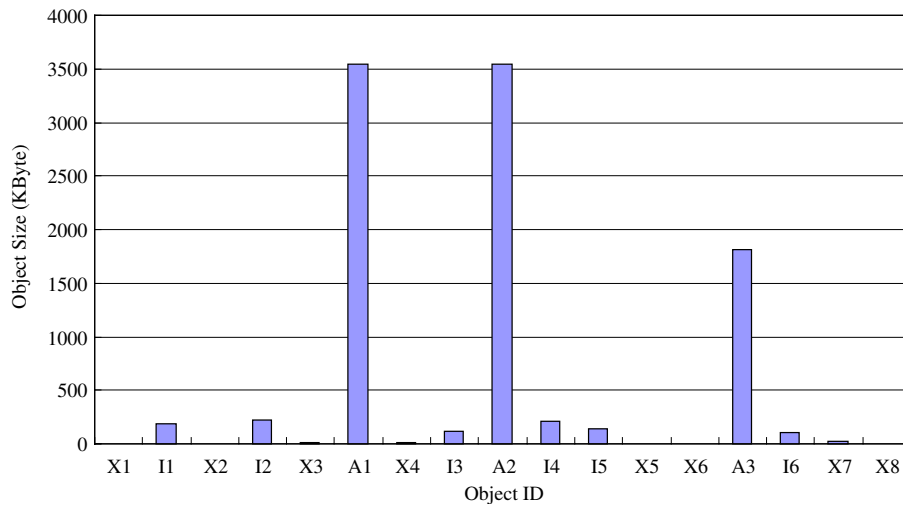


Fig. 14. Size of each object in test document T1.

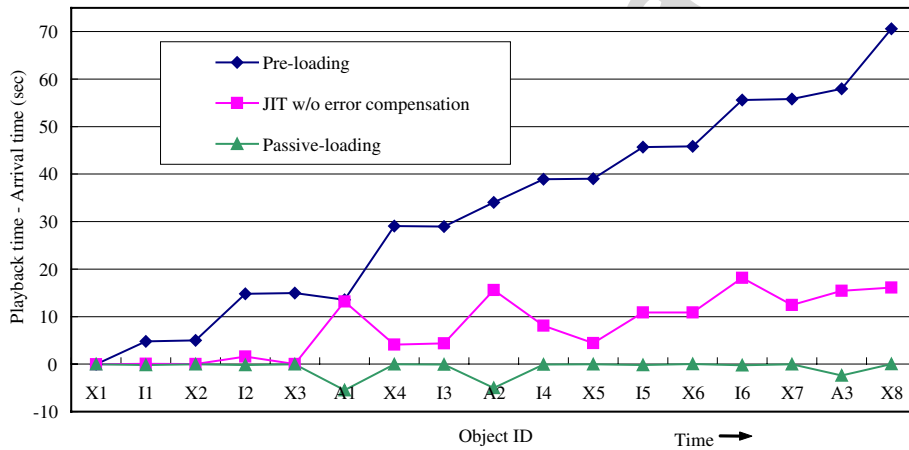


Fig. 15. Average jitters for objects in T1.

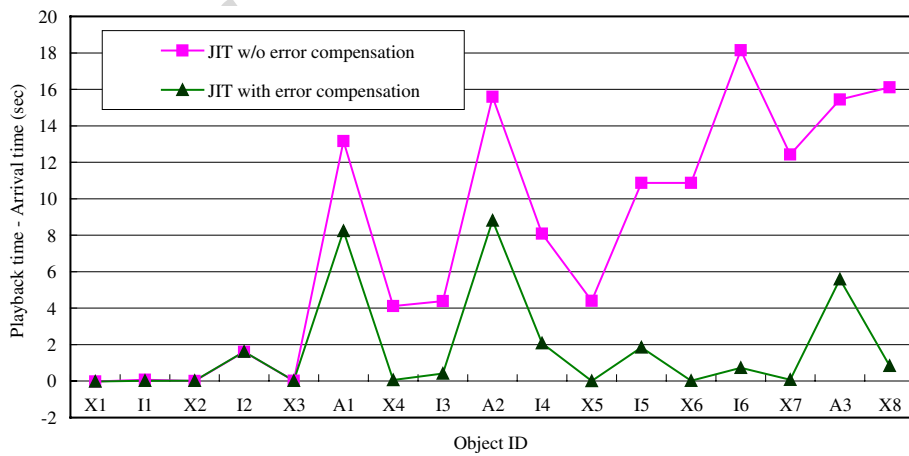


Fig. 16. Jitters: JIT with error compensation vs. JIT without error compensation for T1.

buffers among all schemes, and the passive-loading scheme requires the fewest buffers. Moreover, Fig. 18 demonstrates

that the error compensation mechanism can effectively reduce buffer utilization for the JIT scheme.

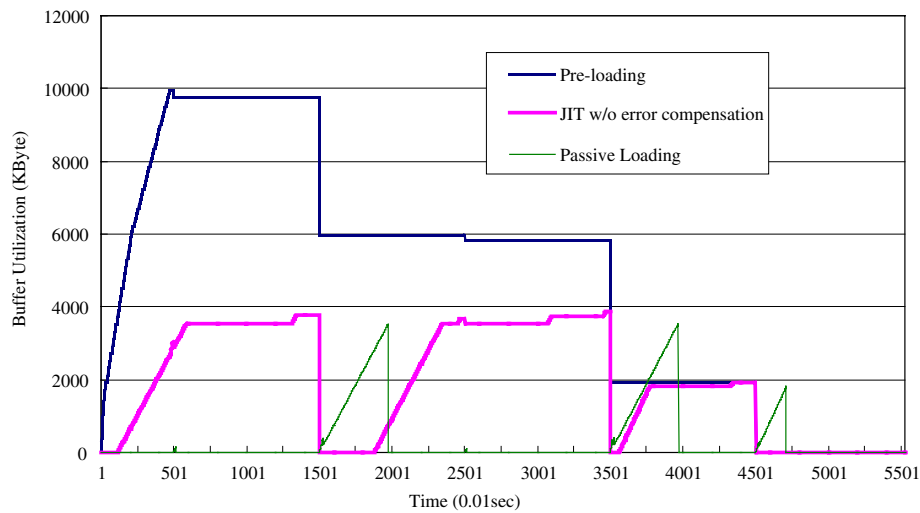


Fig. 17. Buffer utilization in the three schemes for T1.

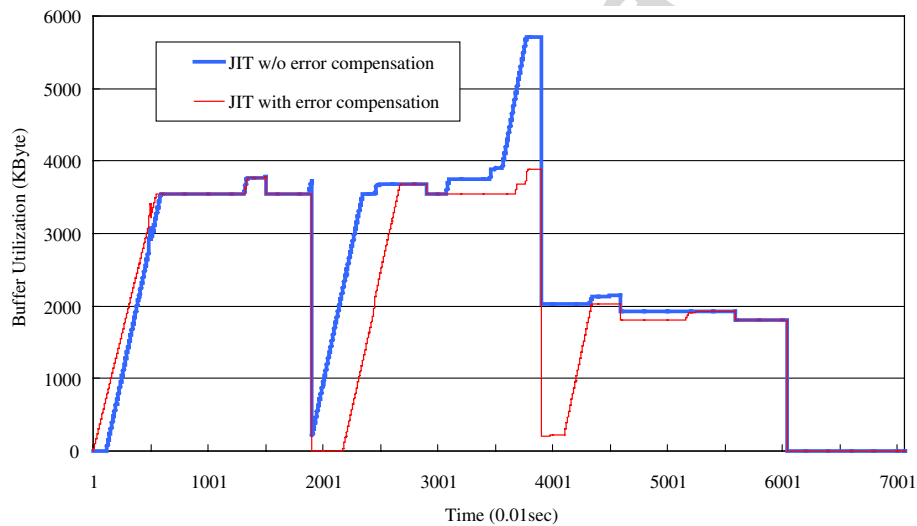


Fig. 18. Buffer utilization: JIT with error compensation vs. JIT without error compensation for T1.

```

<body>
  <seq>
    <text id="x1" dur="3s" src="http://studentweb.ncnu.edu.tw/91321516/picture/x8.txt">
    
    <audio id="A1" dur="5s" src="http://homepage.ntu.edu.tw/~d93944007/a1.wav">

    <par endsync="A2">
      <audio id="A2" dur="10s" begin="mouse.Click" src="http://homepage13.seed.net.tw/web@5/barucia/pic/a2.wav">
      
      <text id="x2" src="http://studentweb.ncnu.edu.tw/91321516/picture/x5.txt">
    </par>

    <par>
      <audio id="A3" begin="mouse.Click" src="http://studentweb.ncnu.edu.tw/91321516/picture/a3.wav">
      
      <text id="x3" src="http://studentweb.ncnu.edu.tw/91321516/picture/x6.txt">
    </par>

  </seq>
</body>

```

Fig. 19. The code snippet of test SMIL2.0 document T2.

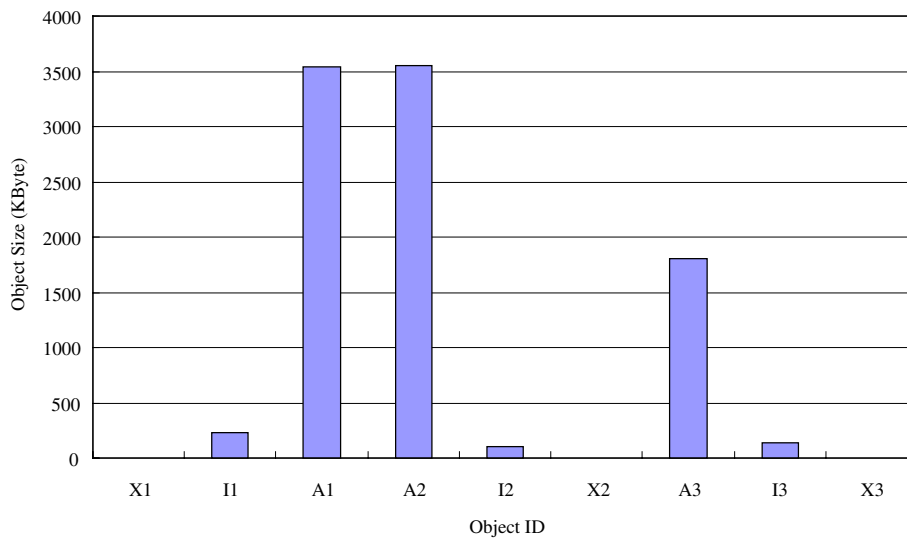


Fig. 20. Size of each object in test document T2.

Another set of performance measurements for a simpler test document T2, of which the script is shown in Fig. 19 and the size of each object is shown in Fig. 20, is displayed from Figs. 21–24. Once again, the results have demonstrated the benefit of the proposed JIT scheme with error compensation over the contrasts in terms of buffer utilization.

6. Related work

For media synchronization, Manvi and Venkataram (2006) proposed an adaptive synchronization agency for synchronization of multimedia streams by using an agent-based approach. The synchronization agency is aiming for adaptation to the run-time and life-time presentation requirements of an application by using static and mobile agents to estimate the network delays/jitters and monitor the loss and playout times of the presentations units. Adaptive synchronization mechanism adjusts play-

out times in accordance with changes in network conditions and offers better quality presentation by maintaining the sustainable losses.

Timing issues in multimedia formats had been addressed intensively in the work of Rogge et al. (2004), in which 10 criteria were proposed in their reference model for comparing existing multimedia formats including *SMIL*, *Quick-Time*, *Shockwave Flash*, *Realmedia*, *Advanced Streaming Format*, and *MPEG-4*. The authors concluded that *SMIL* was the only document model supporting all 29 temporal relationships in the reference model. Moreover, *SMIL* were also designed to have good properties in terms of fine granularity, interactivity with users, extensibility, reusability, adaptability, etc. In other words, from the academic viewpoint, *SMIL* does have what it takes to become one of the most important formats in multimedia presentations.

In the research of *SMIL* modeling, Yu et al. (2002) proposed a formal approach to modeling and analyzing temporal aspects of *SMIL* documents using the *Software*

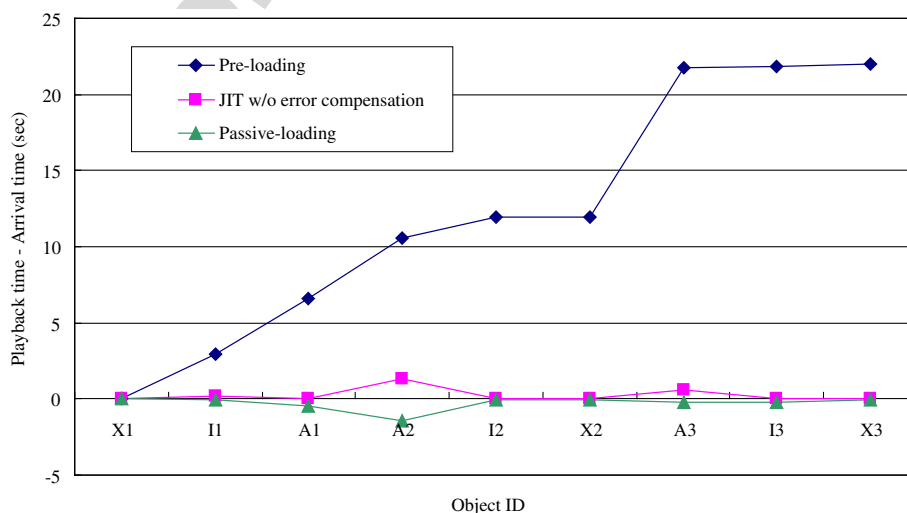


Fig. 21. Average jitters for objects in T2.

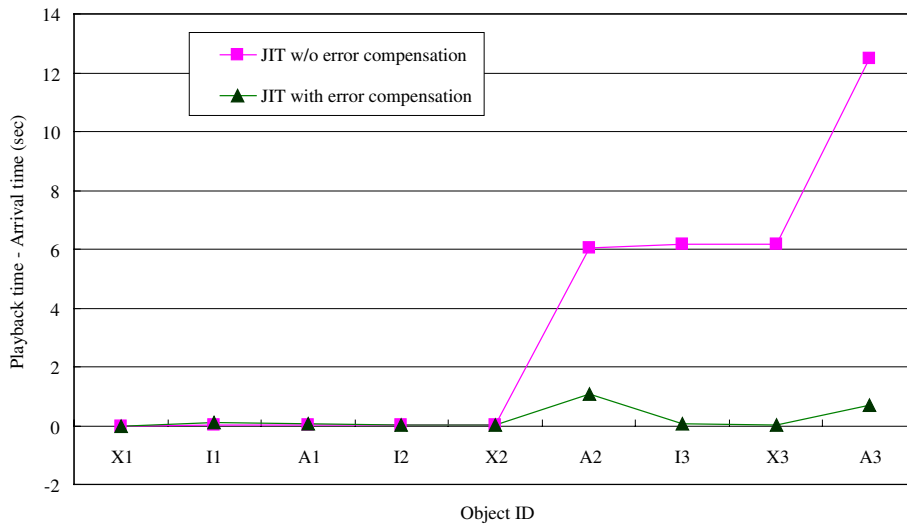


Fig. 22. Jitters: JIT with error compensation vs. JIT without error compensation for T2.

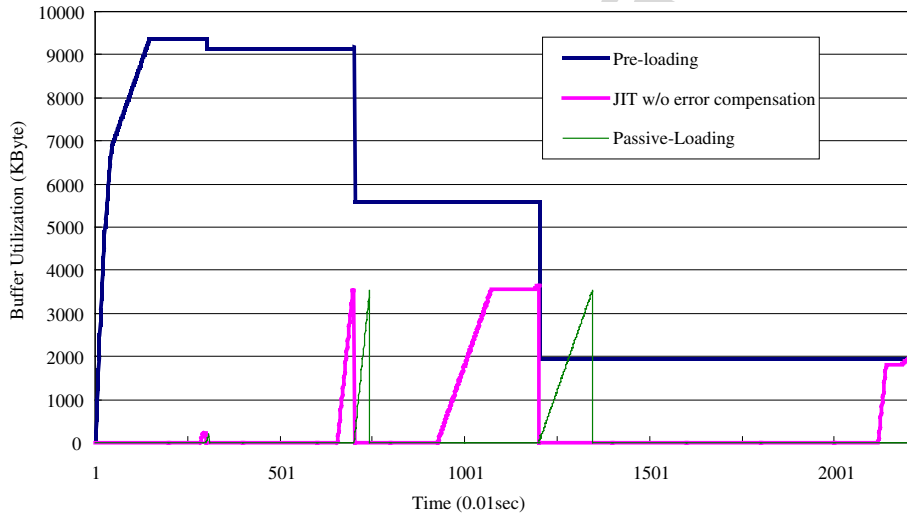


Fig. 23. Buffer utilization in the three schemes for T2.

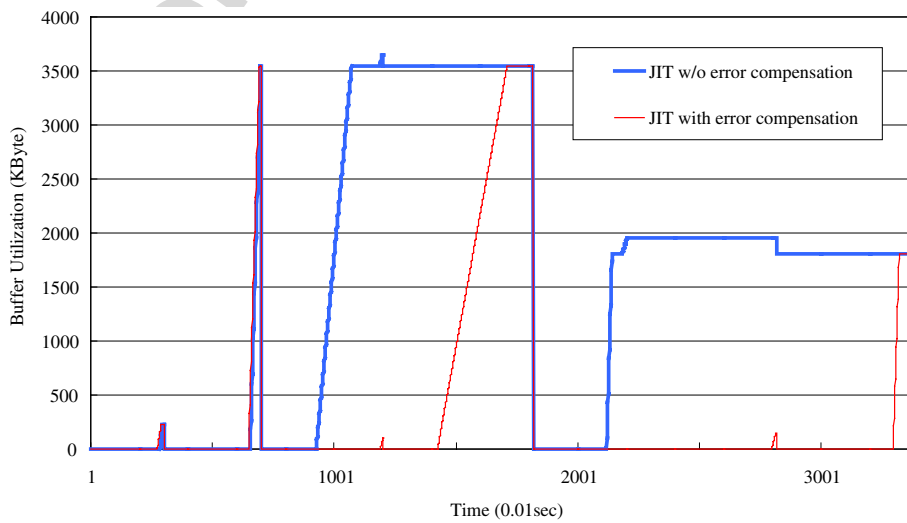


Fig. 24. Buffer utilization: JIT with error compensation vs. JIT without error compensation for T2.

Architecture Model (SAM). SAM is based on dual formalism combining Petri nets and temporal logic. Although SAM was claimed to use Petri nets, the time control information (e.g. the value of attributes $\langle begin \rangle$, $\langle dur \rangle$, and $\langle end \rangle$ for an element) in SAM is stored in transitions instead of places such that the transitions in SAM are not necessarily instantaneous. Moreover, Yu's work is basically focused on SMIL1.0. Sampaio et al. (2000) proposed a *RT-LOTOS (Real-Time Language of Temporal Ordering Specifications [Courtiat et al., 2000])* based mechanism for semantic verification of SMIL documents. Non-deterministic events were considered in their work, but their target was mainly SMIL1.0.

Chung et al. (2003) and Chung and Pereira (2005) proposed a technique based on *Timed Petri Net (TPN)* to capture the timing and synchronization information of multimedia objects specified in SMIL2.0. They incorporated a new type of transition (denoted by special transition), a couple of different types of state for places (two tokens can be placed in a place, and each token can be in one of three types), and complicated firing rules in TPN for SMIL2.0 modeling. As a matter of fact, their proposed model is not Petri net any more. Although some elaborate features were added in TPN, the authors did not explain how can the proposed scheme model some of the complex timing control behaviors associated with $\langle restart \rangle$, $\langle repeatCount \rangle$, and $\langle repeatDur \rangle$ attributes, etc. We doubt the feasibility of using a graph-based modeling mechanism such as OCPN, TPN, or E-RTSM, to completely capture the complex temporal relationships among media objects in SMIL2.0. Thus, the notion of run-time controllers is adopted in our paper.

Deng et al. (2002) also proposed a TPN-based technique for SMIL2.0 modeling. Instead of focusing on modeling complex temporal relationships, they aimed at offering a framework of web-based multimedia presentation system that includes user-concerned adaptive adaptation operations as well as authoring process. Chang et al. (2004) developed a temporal algebra system to unify media presentation time and interaction event and deal with qualitative and quantitative inconsistency in SMIL2.0 documents. Chang's work can be used in semantic verification during parsing stage, which is helpful in authoring process. Some other efforts were made for the design of SMIL2.0 player (Shin and Shin, 2002; Hieda et al., 2003). However, none of the above efforts provide an effective way of modeling temporal behavior of SMIL2.0 and propose an efficient data-retrieving mechanism as in this paper.

7. Conclusion and future work

In this paper, modeling of the non-deterministic synchronization behaviors in SMIL2.0 presentations has been proposed. The proposed model namely Extended Real-Time Synchronization Model (E-RTSM) is the extension of our previous work for SMIL1.0 modeling. We propose

the converting mechanism of the temporal relationship in SMIL2.0 to E-RTSM. Moreover, one application of E-RTSM-based modeling, the design of the just-in-time (JIT) data-retrieving engine for SMIL2.0 documents, is also presented. To cope with the users' random behavior in viewing SMIL2.0 presentations, we propose the idea of the run-time error compensation for data retrieving. Implementation of the proposed modeling technique as well as the JIT data-retrieving engine has been finished. The performance measurements have demonstrated that by combining the worst-case estimation of the playback time at parsing stage and error compensation at run-time, the proposed JIT data-retrieving engine outperforms the other two data-retrieving schemes. The future work of this paper includes modeling and handling of other complicated timing features in SMIL2.0 such as the $\langle excl \rangle$ element, the $\langle priorityClass \rangle$ element, etc.

The contributions of the research in the paper are listed as follows:

- (1) E-RTSM is proposed for modeling non-deterministic as well as deterministic temporal relationship among multimedia objects.
- (2) The converting algorithm of SMIL2.0 synchronization relationship to E-RTSM is proposed to provide an easier and systematic way for dealing with the temporal relationship of the objects in a SMIL2.0 presentation.
- (3) The application of E-RTSM-based modeling in SMIL2.0 data retrieving is proposed. Mechanisms of calculating the playback time as well as the request time for objects in a presentation are presented in the paper.
- (4) The feasibility and better performance of the proposed just-in-time policy for data retrieving have been proved by system implementation and performance measurements.

References

- Bolliger, J., Gross, Th., 1999. Bandwidth modelling for network-aware applications. In: Proceedings, IEEE INFOCOM, pp. 1300–1309.
- Bulterman, D.C.A., 2001. SMIL 2.0 part 1: Overview, Concepts, and Structure. IEEE Multimedia 8 (4), 82–88.
- Bulterman, D.C.A., 2002. SMIL 2.0. 2. Examples and Comparisons. IEEE Multimedia 9 (1), 74–84.
- Chang, A.Y., 2004. Design of an intelligent distributed multimedia presentation system using temporal algebra and SMIL. In: Proceedings, IEEE International Conference on Multimedia and Expo (ICME), pp. 2211–2214.
- Chung, S.M., Pereira, A.L., 2005. Timed petri net representation of SMIL. IEEE Multimedia 12 (1), 64–72.
- Chung, S.M., Pereira, A.L., 2003. Timed petri net representation of the synchronized multimedia integration language (SMIL) of XML. In: Proceedings, International Conference on Information Technology: Coding and Computing (ITCC), pp. 711–716.
- Courtiat, J.-P., Santos, C.A.S., Lohr, C., Outtaj, B., 2000. Experience with RT-LOTOS, a temporal extension of the LOTOS formal description technique. Computer Communications 23 (12), 1104–1123.

- Deng, L.Y., Chen, R.-X., Chang, R.-C., Huang, T.-S., 2002. Adaptive content model for multimedia presentation. In: Proceedings, First International Symposium on Cyber Worlds, pp. 209–216.
- Hieda, S., Saida, Y., Chishima, H., Sato, N., Nakamoto, Y., 2003. Design of SMIL browser functionality in mobile terminals. In: Proceedings, 6th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 143–146.
- Huang, Y.C., Lu, C.S., Wu, H.K., 2004. Reliable available bandwidth estimation based on distinguishing queuing regions and resolving false estimations. In: Proceedings, IEEE Global Telecommunications Conference (GLOBECOM), pp. 4081–4086.
- Lai, K., Baker, M., 1999. Measuring Bandwidth. In: Proceedings, IEEE INFOCOM, pp. 235–245.
- Little, T.D.C., Ghafoor, A., 1989. Synchronization and storage models for multimedia objects. *IEEE Journal of Selected Area in Communications* 8 (3), 413–427.
- Liu, Q., Hwang, J.N., 2003. End-to-end Available bandwidth estimation and time measurement adjustment for multimedia QOS. In: Proceedings, IEEE International Conference on Multimedia and Expo (ICME), pp. III-373-6.
- Manvi, S.S., Venkataram, P., 2006. Agent based synchronization scheme for distributed multimedia applications. *Journal of Systems and Software* 79, 701–713.
- Prasad, R., Dovrolis, C., Murray, M., Claffy, K., 2003. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network* 17 (6), 27–35.
- Rogge, B., Bekaert, J., Van de Walle, R., 2004. Timing issues in multimedia formats: review of the principles and comparison of existing formats. *IEEE Transactions on Multimedia* 6 (6), 910–924.
- Sampaio, P.N.M., Santos, C.A.S., Courtias, J.P., 2000. About the semantic verification of SMIL documents. In: Proceedings, IEEE International Conference on Multimedia and Expo (ICME), pp. 1675–1678.
- Shin, Dongkyoo, Shin, Dongil, 2002. Design and implementation of the SMIL (synchronized multimedia integration language) Player. *IEEE Transactions on Consumer Electronics* 48 (3), 575–578.
- Synchronized Multimedia Integration Language, 1998. (SMIL) 1.0 Specification, W3C Recommendation, June 1998. Available from: <<http://www.w3c.org/TR/REC-smil>>.
- Synchronized Multimedia Integration Language, 2001. (SMIL) 2.0 Specification, W3C Recommendation. Available from: <<http://www.w3.org/TR/smil20>>.
- Yang, C.C., Huang, J.H., 1996. A multimedia synchronization model and its implementation in transport protocols. *IEEE Journal of Selected Area in Communications* 14 (1), 212–225.
- Yang, C.C., Yang, Y.Z., 2004. Design and implementation of the just-in-time retrieving policy for schedule-based distributed multimedia presentations. *Journal of Systems and Software (SCI)* 71 (1-2), 49–63. Available from: <<http://www.csie.ncnu.edu.tw/~ccyang/Publication/JSS2004.pdf>>.
- Yang, C.C., Tien, C.W., Wang, Y.C., 2003a. Supporting VCR-like operations in SMIL2.0 players. In: Proceedings, IEEE International Conference on Multimedia and Expo (ICME), vol. 2, 6–9 July 2003, pp. 761–764.
- Yang, C.C., Tien, C.W., Wang, Y.C., 2003b. Modeling of the non-deterministic synchronization behaviors in SMIL2.0 documents. In: Proceedings IEEE International Conference on Multimedia and Expo (ICME), vol. 3 6–9 July 2003, pp. 265–268.
- Yang, C.C., Chu, C.K., Wang, Y.C., 2004a. Dividable dynamic timeline-based authoring for SMIL2.0 presentations. In: Proceedings, IEEE International Conference on Multimedia and Expo (ICME), 27–30 June 2004.
- Yang, C.C., Wang, Y.C., Chu, C.K., 2004b. Reuse of SMIL2.0 scripts in dividable dynamic timeline-based authoring. In: Proceedings, IEEE International Conference on Multimedia and Expo (ICME), 27–30 June 2004.
- Yu, H., He, Z., Gao, S., Deng, Y., 2002. Modeling and analyzing SMIL documents in SAM. In: Proceedings, 4th International Symposium on Multimedia Software Engineering, pp. 132–139.