Design of the Data-Retrieving Engine for Distributed Multimedia Presentations

Chun-Chuan Yang
Multimedia and Communications Laboratory
Department of Computer Science and Information Engineering
National Chi-Nan University, Taiwan, R.O.C.
ccyang@csie.ncnu.edu.tw

***Abstract*— To provide the smooth progress of a SMIL-based distributed multimedia presentation, the data-retrieving engine for the player must pre-fetch each object before its playback time. In this paper, a smart data-retrieving engine is proposed, which adopts a retrieval policy named the just-in-time policy. The policy requires the retrieval process of an object to be finished right before the playback time of the object. By converting the synchronization relationship of objects in the SMIL document to the Real-Time Synchronization Model, which simplifies the handling of the synchronization relationship, and considering the network condition, the data-retrieving engine could determine the request time for each object. The engine then makes the request to fetch each object for the ongoing presentation according to the pre-computed request time, and provides the player with proper media objects for smooth playback of the presentation.**

***Index Terms*— Distributed multimedia presentation, SMIL, Real-time synchronization model (RTSM)**

I. INTRODUCTION

Distributed multimedia presentation [1-3] is concerning with viewing a presentation via the network as shown in Fig. 1. Some or all the media objects in the presentation could be somewhere else in the network instead of the site where the viewer is. There are two major issues to be addressed in the distributed multimedia presentation. First, it requires a script language to describe both the *spatial relationship* and the *temporal relationship* among distributed media objects in the presentation. The spatial relationship is about the visual layout of media objects in the presentation, while the temporal relationship which is also called *synchronization relationship* [4-6] is concerning with the timing control of media objects. The World Wide Web Consortium (W3C) developed the *Synchronization Multimedia Integration Language (SMIL)* [7-10] that allows the use of a text editor to write multimedia presentations. With the efforts of W3C, SMIL will become the most popular language in authoring multimedia presentations. This paper is thus focus on the SMIL-based presentations.

Second, since the media objects in the presentation are distributed over the network, the browser or the playback application (player) for the presentation must deal with the random behavior of the network and provide a smooth playback to the viewer. The quality of the playback depends on the data-retrieving engine, which should retrieve the proper media object before the object's playback time according to the synchronization relationship of the presentation. Two extreme policies for data retrieving could be adopted. First, retrieve all objects before starting the presentation, or second, do not make the request to retrieve

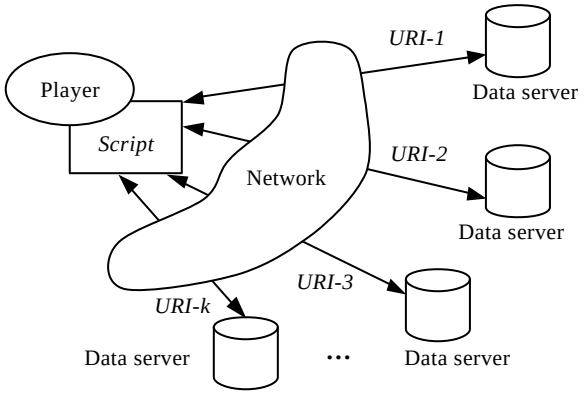


Fig. 1. Distributed multimedia presentation

the object until the object's playback time. The first policy is called the *pre-loading policy*, and the second policy is called the *passive-loading policy* in the paper. The pre-loading policy guarantees the smooth playback in the cost of a long initial delay and a large buffer for all objects. Moreover, since the viewer could activate hyperlinks in an ongoing presentation as in SMIL, it is improper to pre-fetch all media objects in advance. Minimal buffer is required for the second policy, but it introduces gaps between the request time and the finish time of object retrieval because of the network delay. It implies that the smooth playback is impossible for the passive-loading policy.

The proposed data-retrieving engine adopts a better policy that is called the *just-in-time policy*. The policy requires 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. Under such policy, the data-retrieving engine only buffers necessary objects for the smooth progress of the presentation, thus it has a better buffer utilization and network bandwidth efficiency.

In order to estimate the request time for an object under the just-in-time policy, the data-retrieving engine has to extract the synchronization relationship from the SMIL document to determine the playback time for each object in the presentation. By considering both the playback time of each object and the transmission delay of the network, the request time for each object could be determined. The data-retrieving engine then makes the request to the data server for an object according to the pre-computed request time for that object.

To deal with the synchronization relationship efficiently, a synchronization model [4, 11, 12], which provides a systematic view of the synchronization relationship specified in the SMIL script, is necessary for the data-retrieving engine. A Petri-net based model namely *OCPN (Object Composition*

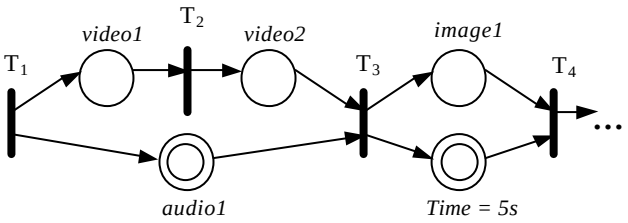


Fig. 2. Example of RTSM

Petri Net) was proposed to model the synchronization relationship among media objects [11]. However, we had shown that OCPN is not suitable for real-time network applications, and instead the *Real-Time Synchronization Model (RTSM)* was proposed [13]. Moreover, some synchronization behaviors that could be specified in SMIL make RTSM more suitable than OCPN. For example, SMIL allows the author to set the explicit beginning time, duration, or end time for a media object. RTSM could easily model the synchronization behaviors as will be explained in section III, but OCPN could not. Therefore, RTSM is adopted in the proposed data-retrieving engine. Before going through the details of the data-retrieving engine, we make a brief survey of RTSM in the following.

The elements in RTSM include *place*, *token*, and *transition* as in OCPN. However, there are two kinds of places in RTSM, *regular places* and *enforced places*. A different firing rule for enforced places is defined. The rule specifies that once an enforced place becomes unblocked (in other words, the related action with the place is completed), the transition following it will be immediately fired regardless the states of other places feeding the same transition. An example of RTSM is shown in Fig. 2 in which a single circle is for the regular place, a double circle is for the enforced place, and a bar is drawn for the transition. The RTSM in the figure requires that audio segment *audio1* and two video clips *video1* and *video2* be played simultaneously, in which *video2* follows *video1*. Since *audio1* is an enforced place, transition *T3* is fired right after *audio1* is finished, regardless of whether *video2* has finished or not. After firing *T3*, *image1* is displayed for 5 seconds then transition *T4* is fired. Note that the enforced place of "Time = 5s" in the figure is not a media object but a virtual medium that is called *Time Medium* [13]. The time medium is used to represent time duration. For more details of RTSM, please reference paper [13].

The remainder of the paper is organized as follows. The architecture of the proposed data-retrieving engine is explained in section II. Section III presents the algorithm of converting the SMIL temporal relationship to RTSM. In section IV, the method of determining the playback time and the estimated request time for each object is explained. Finally, section V concludes the paper.

II. Architecture of the Data-Retrieving Engine

The functional architecture of the proposed data-retrieving engine is shown in Fig. 3. The engine accepts the SMIL script from the player, and provides necessary media object data to the player for maintaining the smooth progress of the ongoing presentation. There are mainly four steps for the data-retrieving engine. First, the synchronization relationship is extracted from the SMIL script and is represented by

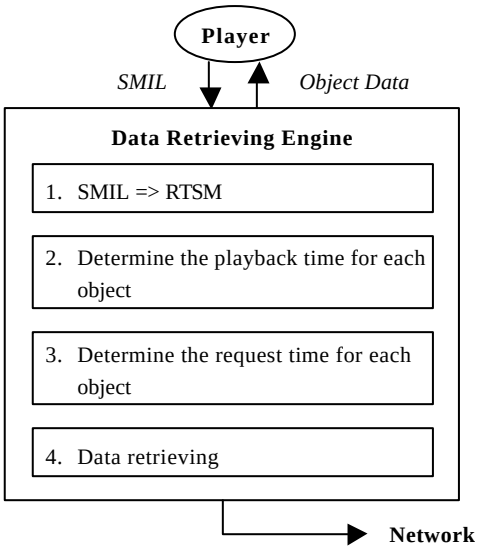


Fig. 3. Overview of the data retrieving engine

RTSM. Second, the playback time for each object must be computed, and third, the request time for each object is determined. Lastly, each object is requested according to the computed request time for the object. The conversion of SMIL to RTSM and the method that determining the request time are presented in the following sections.

III. Converting SMIL to RTSM

In this section, the synchronization elements in SMIL are examined, and the algorithm that converts the elements to RTSM is presented. There are three kinds of synchronization elements in SMIL to be converted: the <seq> element, the <par> element, and the class of media object elements such as <img>, <video>, <audio> and <text>, etc [7]. Besides, some synchronization related attributes such as "begin", "dur", and "end" could be associated with these synchronization elements. We assume that the player has checked the syntax of the SMIL document, so the data-retrieving engine only performs necessary conversion.

A. Converting the <seq> element

The <seq> element defines a sequence of elements in which elements play one after the other. The children elements of the <seq> element could be any of the synchronization elements such as <seq>, <par>, or the media object elements, so the conversion is a recursive procedure. Since the children of a <seq> element form a temporal sequence, we concatenate each child of <seq> one by one in RTSM as illustrated in Fig. 4. Note that there are *virtual places* (denoted by the dashed circle) in the figure. They are used to maintain the consistency of RTSM, since the arc could only be the link between a transition and a place. In fact, the virtual place is a regular place that maps to the time medium with zero duration.

B. Converting the <par> element

The <par> element defines a simple parallel time grouping in which multiple elements can play back at the same time. Thus, all children of <par> should be within the same pair of

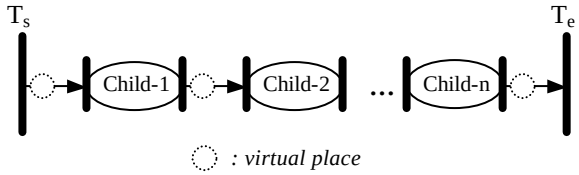


Fig. 4. Convert the <seq> element to RTSM

transition (T_s, T_e) as illustrated in Fig. 5. There are three variations for <par> since a special attribute, "endsync", could be associated with <par>. The "endsync" attribute controls the end of the <par> element, as a function of children. Legal values for the attribute are "last", "first", and "id-ref".

The value of "last" requires <par> to end with the last end of all the child elements, and the corresponding RTSM is shown in Fig. 5-(a), in which transition T_e could not be fired unless all the children end. The value of "first" requires <par> to end with the earliest end of all the child elements. Therefore, we should change the places between each child element and transition T_e to *virtual enforced places* as illustrated in Fig. 5-(b) so that the child that ends first will fire transition T_e. A virtual enforced place is an enforced place that maps to the time medium with zero duration. The value of "id-ref" requires <par> to end with the specified child. So we change the place between the specified child and transition T_e to the virtual enforced place as shown in Fig. 5-(c).

Other synchronization attributes, such as "begin", "end', and "dur", could also be associated with <seq> and <par>, but the conversion is similar to that in the media object elements that we present in the following.

C. Converting the media object elements

The media object elements allow the inclusion of media objects into a SMIL presentation. Media objects are included by reference (using a URI). One media object element naturally represents a regular place in RTSM. However, the attributes associated with the element require some more complex conversion. We examine and convert attributes "begin", "end", "dur" respectively in the following.

(1) Converting the "begin" attribute

This attribute specifies the time for the explicit begin of an element. Two types of values could be contained in the attribute: *delay-value* and *event-value*. A delay value is a clock-value to postpone the playback time of the element by the delay value. Therefore, one enforced place representing the delay time with the specified duration is added in front of the element as illustrated in Fig. 6-(a). The event-value requires the element begin when a certain event occurs. According to the specification of SMIL 1.0, the element *X* generating the event must be "in scope", in other words, *X* must be a sibling of the element that contains the "begin" attribute. There are two variations for the event-value, which is shown in Fig. 6-(b) and 6-(c) respectively.

In Fig. 6-(b), the "id(*X*)(n s)" value of "begin" attribute means that element *Obj* begins after its sibling *X* has begun for n seconds. So one enforced place representing the delay time with value of summation of "*X*'s begin time" and "n seconds" is added in front of *Obj* element. Actually, from the semantic point of view, the case in the Fig. 6-(b) is only valid when elements *X* and *Obj* are child elements in a <par> element. The other value of event-value for "begin" attribute is "id(*X*)(end)", which means the *Obj* element begins right after element *X* ends. The case is actually the function of <seq>. Therefore, the value is only valid when *X* is the direct predecessor of the element *Obj* in a <seq> element, and it

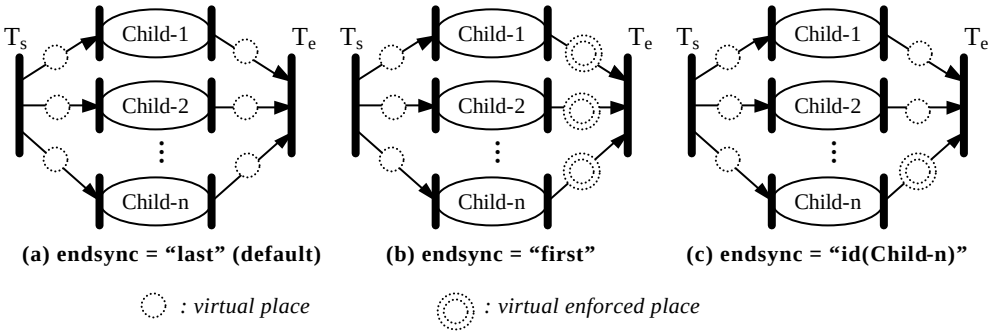


Fig. 5. Convert the <par> element to RTSM

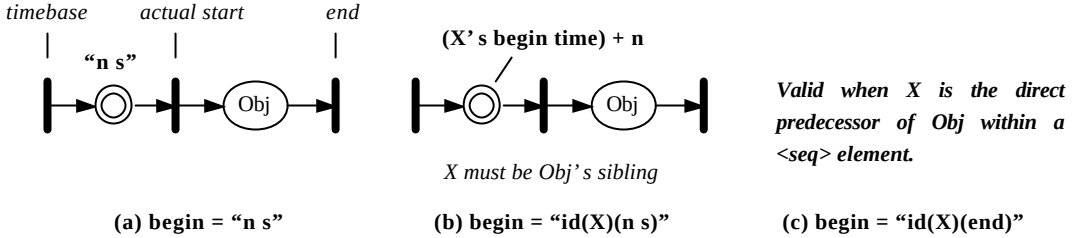


Fig. 6. Effect of "begin" attribute on RTSM

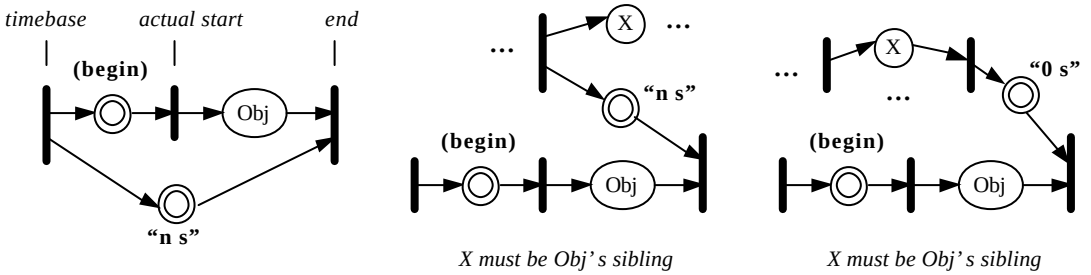


Fig. 7. Effect of "end" attribute on RTSM

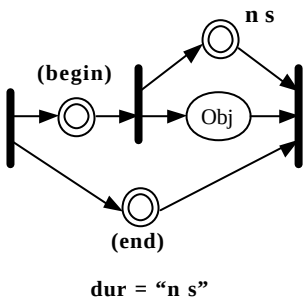


Fig. 8. Effect of "dur" attribute on RTSM

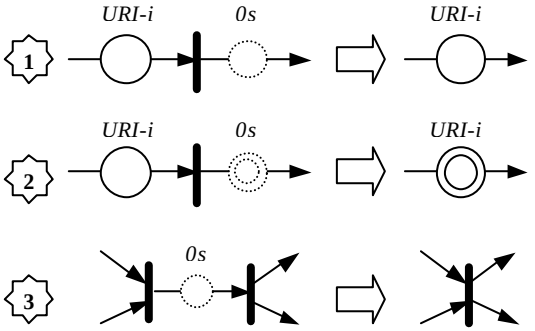


Fig. 9. Three rules to simplify RTSM

introduces nothing to RTSM.

(2) Converting the "end" attribute

This attribute specifies the explicit end of an element. There are also three possible values for the attribute as in the "begin" attribute. We illustrate the conversion of them in Fig. 7. In Fig. 7-(a), a clock value represents the end time from the original timebase of the element. So one enforced place with the clock value is added between the original start transition and the end transition. In Fig. 7-(b), the "id(*X*)(n s)" value of "end" attribute means that element *Obj* ends when his sibling *X* has begun for n seconds. Therefore, there is a enforced place between the actual start transition of *X* and the end transition of *Obj.* Finally, Fig. 7-(c) illustrates the case of value "id(*X*)(end)", which means element *Obj* must end when element *X* ends.

(3) Converting the "dur" attribute

This attribute specifies the explicit duration of an element. Therefore, the value of the "dur" attribute, which is a clock value, forms an enforced place between the actual start transition and the end transition. We illustrate the effect of the attribute in Fig. 8.

D. Hyperlinks in SMIL

As specified in the SMIL 1.0 specification, the hyperlink element <a> is transparent when playing the presentation until the user activates the link and starts a new presentation. Thus, only child elements of the element <a> are converted to RTSM, which is the same as previous sections.

E. Simplifying the obtained RTSM

As mentioned in the above section, some virtual (enforced) places are added in the RTSM during the converting process. However, there are some cases of RTSM that could be simplified by applying three rules, which are shown in Fig. 9. First, if the only input of a transition is a media place and the only output of the transition is a virtual place, we could naturally replace the case with the media place only since the virtual place is actually dummy (Fig. 9-1). Second, if the only output of the transition in rule (1) is a virtual enforced place, it means the firing of the media place will enforce to fire the following transition of the virtual enforced place. We could replace the case by changing the media place to an enforced one as shown in Fig. 9-2. Third, if there is only one virtual place between two transitions, the two transitions could be combined into one transition as shown in Fig. 9-3.

F. An example for the conversion

Fig. 10 shows a sample SMIL document, in which only temporal information is displayed. To make the sample clearer, the lifetime for each object in the example is shown in Fig. 11. The initial RTSM right after the converting process is shown in Fig. 12 and the simplified RTSM for the example is shown in Fig. 13.

SMIL 1.0 also introduced the "repeat" attribute, which is used to repeat a media element or an entire time container, such as <seq> or <par>. With the presence of the "repeat" attribute, the RTSM model for the element is copied for the number of times specified by the value of the "repeat" attribute.

```
<seq>
    <par>
        <seq>
            <img src=URI-1 dur="5s" />
            <img src=URI-2 dur="5s" />
        </seq>
        <audio src=URI-3, begin="2s" />
    </par>
    <video  src=URI-4 begin="3s" end="10s" />
</seq>
```

Fig. 10. Sample SMIL Document

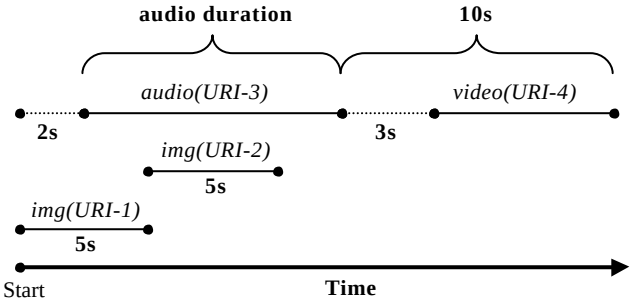


Fig. 11. Lifetime for each object in the sample

IV. Determining the Object Request Time

In order to compute the request time for each object, the data-retrieving engine has to know the implicit duration (intrinsic playback time) for each media object and has to estimate the transmission delay from the data server to the data-retrieving engine. Therefore, after the converting procedure, the data-retrieving engine sends the *probe packet* to each data server that provides the media object for the presentation. When receiving the probe packet, the data server has to estimate the transmission rate (bandwidth) [14-17] that it could support to transmit the requested object to the data-retrieving engine. We denote the estimated bandwidth as $EstBW_{URI\text{-}i}$ for object *URI-i*. The data server then acknowledges the probe packet by sending three parameters about the object back to the data-retrieving engine: (1) $EstBW_{URI\text{-}i}$, (2) the implicit duration of the object, and (3) the size of the object, which is denoted by $Size_{URI\text{-}i}$.

When the data-retrieving engine receives the acknowledgement from the data server, it measures the round-trip delay, which is denoted by $RT\text{-}Delay_{URI\text{-}i}$, to the data server. With the information provided by the data server, we could then determine the schedule for each object in the presentation and estimate the request time for each object.

A. Calculation of the playback time

The playback time for object *URI-i*, which is denoted by $PB\text{-}Time_{URI\text{-}i}$, is actually the firing time of the start transition of *URI-i* in RTSM. To compute the firing time for each transition, we have to follow the progress of RTSM. Since there are usually more than one place that feeds to a transition, the behavior of the transition depends on the type of places that feed into it. If a transition is fed by some enforced places,

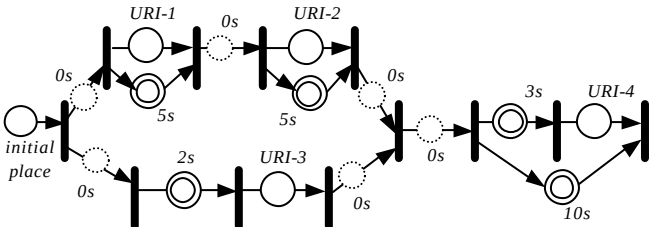


Fig. 12. Initial RTSM for the sample SMIL document

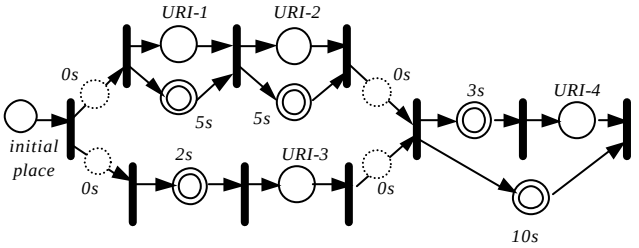


Fig. 13. Simplified RTSM for the sample SMIL document

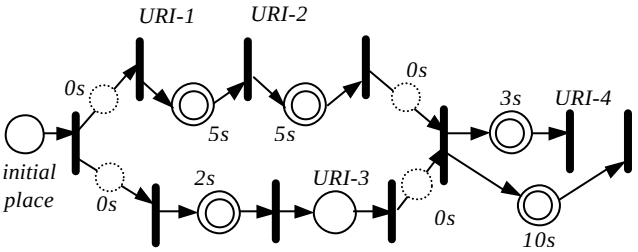
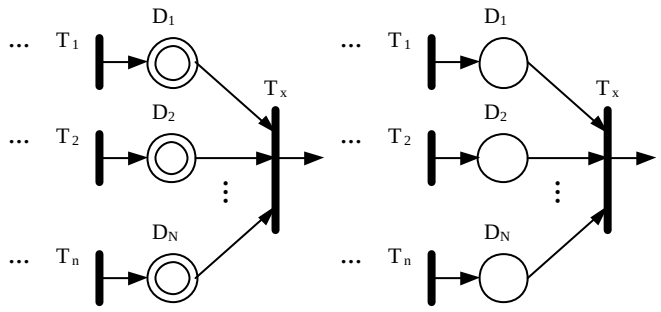


Fig. 14. Reduced RTSM for the sample SMIL document

the enforced places will dominate the behavior of the transition. In other words, if a transition is fed by some enforced places, other regular places can not affect the firing time of the transition at all. Thus, we reduce the RTSM model by removing the regular places that feed to a transition with enforced places. The reduced RTSM for the example in Fig. 13 is shown in Fig. 14.

The playback time for each object is computed by traversing the reduced RTSM transition by transition from the initial place (i.e. the start of the presentation). There are only two possibilities for one transition in the reduced RTSM: (1) places that feed to the transition are all enforced places, or (2) places that feed to the transition are all regular places. For possibility (1), the firing time of the transition is the minimal value of "the firing time of the preceding transition" plus "the duration of the following place of the preceding transition", which is illustrated in Fig. 15-(a). Fig. 15-(b) shows the case of possibility (2), in which transition T_x is fired only after all its preceding regular places finish playing. Therefore, for possibility (2), the firing time of transition T_x is the maximum value of "the firing time of the preceding transition" plus "the duration of the following place of the preceding transition". The duration of each place depends on the type of the media object. For an enforced place of time medium, the duration of the place is the value of the duration. For static media objects, such as <img> and <text>, the duration of the place is zero. For continuous media objects, such as <audio> and <video>,



(a) $T_x = Min(T_1+D_1, \ldots, T_n+D_n)$ (b) $T_x = Max(T_1+D_1, \ldots, T_n+D_n)$

Fig. 15. Determine the active time for transition T_x

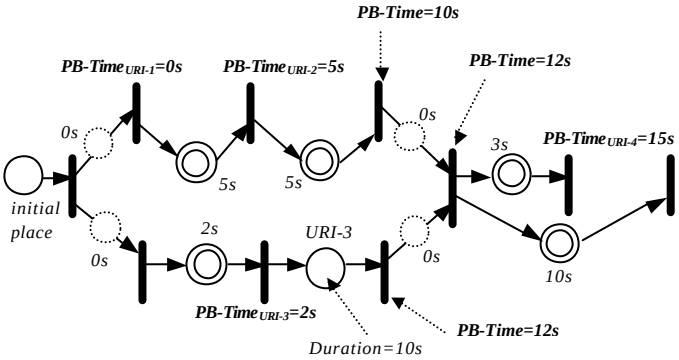


Fig. 16. Computing the playback time for each object

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. Assuming that the implicit duration for *URI-3* in Fig. 13 is 10 seconds, the playback time for each object in the example of Fig. 13 is shown in Fig. 16.

B. Determining the request time

Since the just-in-time policy requires the data-retrieving process to finish right before the playback time of the object, the latest request time will be the playback time of the object minus the total time to finish retrieving the object. The total time to finish retrieving object *URI-i* is the summation of the time for the request packet arrived to the server and the transmission time of the requested object from the server to the data-retrieving engine. The time for the request packet arrived to the server could be estimated as the measured round-trip delay from the server to the data-retrieving engine, i.e. $RT\text{-}Delay_{URI\text{-}i}$. The transmission time of object *URI-i* is estimated as $Size_{URI\text{-}i} / EstBW_{URI\text{-}i}$, in which $Size_{URI\text{-}i}$ and $EstBW_{URI\text{-}i}$ are provided by the data server as mentioned in section IV. Therefore, the latest request time for object *URI-i* is $PB\text{-}Time_{URI\text{-}i} - (RT\text{-}Delay_{URI\text{-}i} + Size_{URI\text{-}i} / EstBW_{URI\text{-}i})$. The negative value of the latest request time for an object means we have to retrieve the object before the presentation starts.

Note that if the object is a streaming object, 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

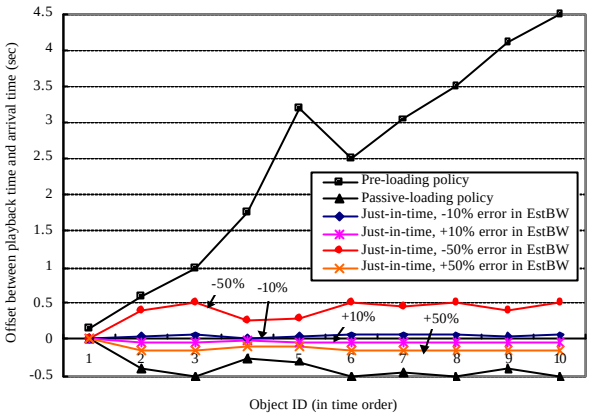


Fig. 17. Simulation result

of the object is $BufferSize_{URI\text{-}i} / EstBW_{URI\text{-}i}$, in which $BufferSize_{URI\text{-}i}$ is the amount of data to buffer. The value of $BufferSize_{URI\text{-}i}$ depends on the streaming operation and is not addressed in the paper.

C. Discussion

It is easy to see that the accuracy of bandwidth estimation affects the performance of the data-retrieving engine as well as the quality of the presentation. There are some research results [14-17] for the bandwidth estimation (bandwidth modeling) in the literature. 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. Thus, 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. On the other hand, if the network bandwidth from each data server to the data-retrieving engine could be reserved in advance (for example, by some booking method), the estimated request time for each object will be more precise. Hence, the quality of the presentation and the buffer utilization will also be improved.

Some simulations are conducted to investigate the impact of the bandwidth estimation. The test SMIL file used in the simulations is consist of ten media objects with similar size (2.5k bytes ~ 5k bytes). Three retrieving policies, pre-loading, passive-loading, and just-in-time policies as mentioned in section I, are compared in the simulations. The performance criterion is the offset between the playback time and the arrival time of the object (i.e. offset = playback time – arrival time). Positive value of the offset implies the buffered time for the object before playback, while negative value of the offset indicates the amount of pause time in the presentation to wait for the object to arrive. We simulated the just-in-time policy with four error cases in bandwidth estimation as shown in Fig. 17. Error cases of –10% and –50% in bandwidth estimation indicate the amount of underestimated bandwidth, while cases of +10% and +50% indicate the overestimated bandwidth. Fig. 17 (object ID for x-axis, and

offset for y-axis) shows that the just-in-time policy is better than the other two policies, even with worse bandwidth estimations.

V. CONCLUSION

In this paper, a smart data-retrieving engine for SMIL-based distributed multimedia presentations is proposed. The just-in-time policy for data retrieval is adopted by the engine, which requires 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. To meet the goal of the policy, the synchronization relationship among objects in the presentation is extracted and presented by the Real-Time Synchronization Model, which provides a systematic point of view for the synchronization relationship. RTSM helps the data-retrieving engine be able to handle the temporal relationship more easily. By analyzing RTSM and considering the network condition, the request time of each object could be determined. The data-retrieving engine then make the request to the data server to fetch the proper object for the ongoing presentation according to the pre-computed request time. Simulation results show that the proposed data-retrieving engine with the just-in-time policy could achieve better performance for the distributed multimedia presentation.

REFERENCES

[1] Taeil Jeong, Jae Wook Ham, and Sung Jo Kim, "A pre-scheduling mechanism for multimedia presentation synchronization," Proceedings., IEEE International Conference on Multimedia Computing and Systems '97, 1997, pp. 379-386.

[2] R. Paul, M. F. Khan, S. Baqai, and A. Ghafoor, "Real-time scheduling for synchronized presentation of multimedia information in distributed multimedia systems," Proceedings., Third International Workshop on Object-Oriented Real-Time Dependable Systems, 1997, pp. 177-184.

[3] De Lima, R. M. et al., "SAMM: An integrated Environment to Support Multimedia Synchronization of Pre-orchestrated Data," Proceedings., IEEE International Conference on Multimedia Computing and Systems, 1999, pp. 929-933.

[4] Chung-Ming Huang and Chian Wang, "Synchronization for interactive multimedia presentations," IEEE Multimedia, Oct.-Dec. 1998, pp. 44–62.

[5] I. F. Cruz and P. S. Mahalley, "Temporal Synchronization in Multimedia Presentations," Proceedings., IEEE International Conference on Multimedia Computing and Systems, 1999, pp. 851-856.

[6] Z. Yang, et al., "A New Look at Multimedia Synchronization in Distributed Environments," Proceedings., 4th International Symposium on Parallel Architectures, Algorithms, and Networks, (I-SPAN '99), 1999, pp. 322 –327.

[7] *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*, W3C Recommendation, June 1998, http://www.w3c.org/TR/REC-smil.

[8] *Synchronized Multimedia Integration Language (SMIL) Boston Specification*, W3C Working Draft 20-August-1999, http://www.w3.org/TR/smil-boston.

[9] Larry Bouthillier, "Synchronized Multimedia on the Web- A New W3C Format is All Smiles," Web Techniques Magazine, September 1998, Vol. 3 Issue 9.

[10] Philipp Hoschka, "An introduction to the Synchronized Multimedia Integration Language," IEEE Multimedia, Oct.-Dec. 1998, pp. 84-88.

[11] T. D. C. Little, and A. Ghafoor, "Synchronization and storage models for multimedia objects," IEEE Journal of Selected Area in Communications, vol. 8, no. 3, pp. 413-427, Apr. 1989.

[12] InHo Lin, Chwan-Chia Wu, and Bih-Hwang Lee, "A synchronization model for multimedia presentation with critical overlap avoidance," Proceedings. International Workshop on Multimedia Software Engineering, 1998, pp. 36-43.

[13] C. C. Yang and J. H. Huang, "A Multimedia Synchronization Model and its Implementation in Transport protocols," IEEE Journal of Selected Area in Communications, vol. 14, No. 1, pp. 212-225, Jan. 1996.

[14] J. Bolliger and Th. Gross, "A Framework-based Approach to the Development of Network-Aware Applications," IEEE trans. Software Engineering, vol. 24, no. 5, pp. 376-390, May 1998.

[15] J. Bolliger and Th. Gross, "Bandwidth Modelling for Network-Aware Applications," Proceedings., IEEE INFOCOM' 99, pp. 1300-1309, 1999.

[16] K. Lai and M. Baker, "Measuring Bandwidth," Proceedings., IEEE INFOCOM' 99, pp. 235-245, 1999.

[17] Vern Paxson, "End-to-End Internet Packet Dynamics," IEEE/ACM trans. Networking, vol. 7, no. 3, pp. 277-292, June 1999.