Protocol Synthesis from SMIL-Based Scenarios and Its Implementation in Distributed Environment

Takaaki Umedu[†], Hirozumi Yamaguchi[†],
Keiichi Yasumoto[††] and Teruo Higashino[†]
[†] *Dept. Info. and Math. Sci., Osaka Univ., Japan*
[††] *Fac. of Economics, Shiga Univ., Japan*

Abstract

In order to design and develop multi-user multimedia presentation systems, we propose a protocol synthesis method considering QoS requirements for networks. In this method, the scenario of each entity (node) is specified in a SMIL-like language. A specification of the total system (service specification) is described as a set of all users' SMIL-like scenarios and control scenarios specifying the interaction between entities. For a given service specification, network topology, link capacity and required bandwidth for each media-object, a set of protocol entity specifications (protocol specification) is automatically derived. The derived protocol entity specification includes (1) the behavior of each node, (2) communication among other nodes, and (3) each object's transmission path. We have been developing an execution engine for the SMIL-like language which supports dynamic QoS controls and user interaction.

1. Introduction

As high-speed network has become widespread, various methods for designing and developing distributed multimedia systems have been proposed. Protocol synthesis[1] is one of promising methods to develop reliable distributed systems efficiently. Generally when we develop distributed systems, we need to specify communication among computers for data transmission and synchronization. However, since such communication is rather complicated, we may often make mistakes if we describe such communication directly. In protocol synthesis methods, we describe the specification of a whole system as if it is a centralized program (called a *service specification*). The methods automatically synthesize a set of protocol entity specifications (called a *protocol specification*) which include not only nodes' behavior but also communication among them. Protocol synthesis methods for various computational models such as EFSMs, Petri nets, LOTOS and timed automata have been proposed [1, 2, 3, 4, 5, 6, 7].

We think it is important to apply protocol synthesis approaches to the development of distributed multimedia systems, since we have to deal with the following problems in the development: (1) a lot of interaction among users may occur, and (2) multiple media-object may compete with network resources. About the problem (1), it is very difficult to specify such interaction in a distributed environment, since they are implemented by complex communication. Also, about the problem (2), resource reservation such as RSVP [9] and QoS routing can be used to provide end-to-end QoS for users [10, 11, 12]. However, in such a situation that we know which and what size of media-object is transferred in the system, it is much efficient to analyze the concurrency of multiple media transmission in the service specification, and specify adequate routes in the protocol specification to avoid network congestion.

In this paper, we propose a design method based on protocol synthesis for distributed multimedia systems. In this method, the sequence of media presentation on each node (user's machine) is described in a SMIL-like language as a scenario. A specification of the total system (a service specification) is described as a set of all the users' scenarios and control scenarios specifying interaction between the users. For a given service specification, a distributed environment (where network topology and link capacities are known), and bandwidth requirement for each media-object, a set of protocol entity specifications (a protocol specification) is automatically derived. The protocol entity specifications are implemented using our QOS-SMIL player proposed in [15]. In the derived protocol specification, adequate routes for media transmission are determined automatically so that the minimum unused bandwidth is maximized by using a technique to solve linear programming problems.

2. Service Specification

2.1. Basic Policy

For development of multimedia presentation systems consisting of multiple users on a network, the following criteria should be considered.

(1) The whole system behavior should be easily described as a service specification where interaction among users (nodes) should be easily described with high-level communication primitives without considering complicated message exchanges .

(2) A given service specification should be implemented as a set of all nodes' protocol entity specifications (protocol specification) where each node's specification includes message exchanges among nodes for the interaction specified in the service specification.

For the above (1) in the proposed method, we define *SMIL-EX* language where a synchronous communication mechanism (m of n participants can synchronize with each other and exchange data) as well as some other useful mechanisms are introduced to SMIL [8].

For the above (2), we propose a protocol synthesis method where the synchronization mechanism is implemented as asynchronous communication among nodes in each node's entity specification. In the implementation level, we also consider an optimal route selection for avoiding network congestion due to multiple object data transmissions.

In this paper, we describe a service specification of a multi-user multimedia presentation system as a set of scenarios in SMIL-EX. Fig. 1 depicts the organization of the whole specification. A scenario is described for each user separately from others. And in *Presentation Definition File*, we specify assignment of all scenarios to the respective nodes, information on the target network, and so on.

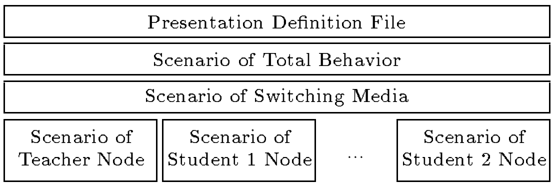
Presentation Definition File
Scenario of Total Behavior
Scenario of Switching Media
Scenario of Teacher Node
Scenario of Student 1 Node
...
Scenario of Student 2 Node

Figure 1: Service Specification

2.2. SMIL-EX

SMIL documents are composed of XML elements [13]. In a SMIL document, multi-media objects such as movies and sounds are called **objects**. For each object, **(1) layout information** such as displaying position (called *region*) and its **(2) behavior** such as starting/ending time of its playback are specified separately. The layout information is described mainly by `region` elements. The behavior of objects is described by `par` and `seq` elements. These elements are used for parallel and sequential playbacks of objects, respectively. Objects are identified by file names (URLs) using `src` attributes. The starting/ending time of object playbacks and their durations are specified by `begin, dur, end, clip-begin` and `clip-end` attributes.

However, SMIL does not support iterative statements, variables nor branch statements. Such control statements are useful for specifying scenarios of multimedia presentations simply and comprehensibly. Therefore, in SMIL-EX, we introduce four new elements : `while, if, input` and `set`. `<while` *condition*`> ... </while>` specifies that the statements between `<while>` and `</while>` are repeatedly executed while *condition* is true. `<input var="x" from="region_A" val="A B C"/>` specifies that a value of `A`, `B` or `C` is input from region `region_A` and stored to variable `x`. `<set var="x" val="A"/>` simply sets value `A` to variable `x`. The value of variable `variable-name` is referred as `"%variable-name;"`. Every reference `"%variable-name;"` is replaced by its actual value when the corresponding `<input>` or `<set>` is executed. This notation can be used in `if` and `while` statements, `src` attributes of `<video>` elements, and so on. For example, a video object whose source is dynamically changed, can be specified as `http://%variable-name;/video.mpg`. We can use any `regions` specified in other documents as sources of `<input>` elements. For this purpose, each document has an unique ID and each `region` is referred to as `document-ID/region`.

In order to specify cooperative work among multiple users, we introduce `<sync-in id="SYNC- ID" var="x"/>` and `<sync-out id="SYNC-ID" val="y"/>` elements for synchronization among documents. All `<sync-out>` and `<sync-in>` elements with the same `ID` must synchronize with each other. When they synchronize, the output value `y` in `<sync-out>` element is set to variable `x` in `<sync-in>` element. We can also specify synchronization among any k of n `<sync>` elements. A master for each synchronization can be specified by `master="true"` attribute where the master must join the synchronization.

The conditions for synchronization can be specified by `number` and `val` attributes in master's `<sync>` element. When `number="2"` is specified, even if more than two documents try to synchronize, only the master document and another one document can synchronize. Using this notation, we can specify mutual exclusion easily.

In this paper, we only focus on the above elements although there are some other elements in SMIL. Note that the above specification can be modeled as a live and safe Free-Choice net [14] with variables where the live and safe Free-Choice net is a sub-class of Petri nets or concurrent extended finite state machines with synchronization. Therefore, hereafter, SMIL-EX doc-

```
 1: <?xml version="1.0"?>
 2: <!DOCTYPE smil-ex SYSTEM "SmilEx.dtd">
 3: <smil-ex>
 4:  <head>     <!-- details are omitted -->
 5:   <layout>
 6:    <region id="TEACHER"/>
 7:    <region id="TEXT"/>
 8:    <region id="STUDENT"/>
 9:    <region id="BUTTON"/>
10:   </layout>
11:  </head>
12:  <body>
13:   <sync-in id="SYNC-START" var="condition"/>
14:   <while var="condition" op="ne" val="STOP">
15:    <par endsync="first">
16:     <while>
17:      <par endsync="first">
18:       <sync-in id="SYNC-STUDENT" var="student"/>
19:       <video src="%student;/student.mpg"/>
20:      </par>
21:     </while>
22:     <while>
23:      <input var="request" region="BUTTON" val="REQUEST"/>
24:      <sync-out id="SYNC-REQUEST" val="%node-id;"/>
25:     </while>
26:     <video src="*Text/text.mpg"/>
27:     <video src="Teacher/teacher.mpg"/>
28:     <sync-in id="SYNC-END" var="condition"/>
29:    </par>
30:   </while>
31:  </body>
32: </smil-ex>
```

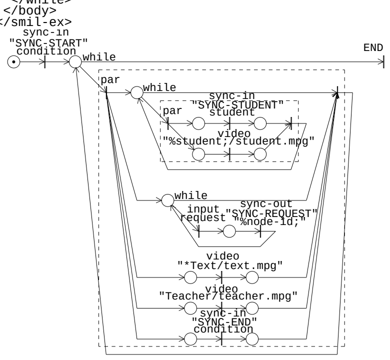

Figure 2: Scenario of Student

ument is often represented as the Petri net model so that we can easily understand its control structure. In SMIL, we can specify interruption such as `<par endsync="first">` to execute statements concurrently and stop all the statements when one of them is finished. Since such interruption cannot be easily expressed in Petri net structure, we denote each interruption as a dotted rectangle as shown in Fig. 2.

2.3. Example of Describing Service Specification

In this paper, we assume that a specification of a multimedia presentation system (e.g., remote lectures) consists of a set of the respective scenarios for all users (e.g., a teacher and students) and a set of scenarios specifying interaction among those users. Each user's scenario specifies what objects should be played back on his/her terminal. The scenarios for interaction among users specify the total system's behavior and constraints among users, that is, the temporal ordering of object playbacks, mutual exclusion among users for selecting some speakers among users who want to speak, and so on.

In the following, we explain how to describe service specifications using an example of remote lectures. First, we show an example scenario of a student in Fig. 2. In this example, the lecture begins with the synchronization element of `SYNC-START` (line 13 of Fig. 2)

```
 1: <?xml version="1.0"?>
 2: <!DOCTYPE smil-ex SYSTEM "SmilEx.dtd">
 3: <smil-ex>
 4:  <head>  <!-- details are omitted -->
 5:  </head>
 6:  <body>
 7:   <sync-in id="SYNC-START" var="condition"/>
 8:   <while var="condition" op="ne" val="STOP">
 9:    <par endsync="first">
10:     <while>
11:      <par endsync="first">
12:       <sync-in id="SYNC-STUDENT" var="student"/>
13:       <video src="%student;/student.mpg"/>
14:      </par>
15:     </while>
16:     <video src="Text/text.mpg"/>
17:     <sync-in id="SYNC-END" var="condition"/>
18:    </par>
19:   </while>
20:  </body>
21: </smil-ex>
```

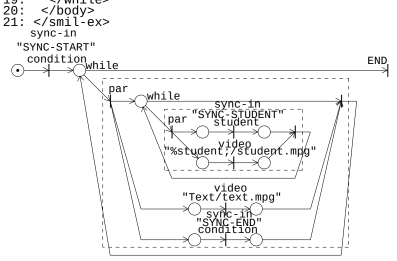

Figure 3: Scenario of Teacher

and repeats the following behavior until the synchronization element of `SYNC-END` (line 28 of Fig. 2) is executed. During the lecture, (1) the object used for lectures (`*Text_1/text.mpg` in line 26), (2) the live video object of the teacher (`Teacher/teacher.mpg` in line 27) and (3) the live video object of the current speaker (i.e., one of students) (`%student;/student.mpg` in line 19) are played back on the student's terminal in parallel. `<while>` element between lines 16 and 21 specifies that the current speaker is changed by the synchronization element of `SYNC-STUDENT` in line 18. Also, `<while>` element between lines 22 and 25 specifies the synchronization element of `SYNC-REQUEST` for a request to speak. When the request is accepted, the value of variable `%node-id;` for identifying the student is received at other users' scenarios.

We can also specify the source of each object as follows.

- If an object is placed on or transferred from a fixed server, the server name is specified directly in the corresponding URL. In the above example, `Teacher` denotes the URL for "teacher.mpg".
- If an object is placed on multiple servers, the mark "`*`" can be added like `*Text` to indicate that any available server can be selected. The candidates of servers for `*Text` are specified in `Presentation Definition File` in Fig. 1.
- In cases that the object source may be dynamically switched, we use a variable to indicate the object source (e.g., `%student;/student.mpg`) so that the object is dynamically switched when the new source is set to variable (`%student;`).

The whole specification of the remote lecture is given as the scenario of (1) the teacher in Fig. 3, (2) the scenarios of n students (Student $_1$,..., Student $_n$) in Fig.

```
<?xml version="1.0"?>
<!DOCTYPE smil-ex SYSTEM "SmilEx.dtd">
<smil-ex>
 <head/>  <!-- details are omitted -->
 <body>
  <input var="condition" region="TEACHER/BUTTON" val="START"/>
  <sync-out master="true" id="SYNC-START" val="%condition;"/>
  <while var="condition" op="ne" val="STOP">
   <set var="condition1" val="" />
   <set var="condition2" val="" />
   <par endsync="first">
    <input var="condition1" region="TEACHER/BUTTON" val="STOP REPEAT"/>
    <input var="condition2" region="STUDENT1/BUTTON" val="STOP REPEAT"/>
   </par>
   <if var="condition1" op="ne" val=""><then>
     <set var="condition" val="%condition1;"/>
   </then><else>
     <set var="condition" val="%condition2;"/>
   </else></if>
   <sync-out master="true" id="SYNC-END" val="%condition;"/>
  </while>
 </body>
</smil-ex>
```

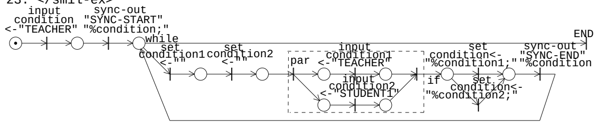

Figure 4: Scenario of Total Behavior

2, (3) the scenario for the whole behavior (i.e., interaction and constraints among the teacher and students) in Fig. 4, (4) the scenario of how to switch the current speaker among all students shown in Fig. 5, and (5) `Presentation Definition File` (here, for limitation of space, its details are omitted).

The scenario of the whole behavior in Fig. 4 specifies that synchronization of `SYNC-START` is executed after the teacher pushes the start button (START). When the teacher or Student$_1$ (who is the leader of the students) pushes the repeat (REPEAT) or end (STOP) button, the synchronization of `SYNC-END` is executed.

The switching control scenario in Fig. 5 specifies how to switch to the current speaker dynamically depending on the requests from the students. The synchronization of `SYNC-REQUEST` is executed when a student wants to speak. Here, we assume that the teacher is a master for synchronization. Therefore, the teacher always joins each synchronization and only one student can join the synchronization among all students exclusively. The teacher can push the accept (ACCEPT) or reject (REJECT) button. When accepted, the synchronization of `SYNC-STUDENT` is executed to tell the selected student to all students.

3. Synthesizing Protocol Specification

In order to execute the scenarios on P actual nodes (entities), the synchronous communication between scenarios must be realized with asynchronous communication between the P nodes.

For this purpose, we introduce `<send>` and `<receive>` elements in SMIL-EX. For example, `<send id="MessageID" to="NodeID" val="Value">` represents that `"Value"` is sent to node `"NodeID"` as a message with `"MessageID"`. If multiple nodes are specified in `"NodeID"`, the messages are sent to all of them.

```
<?xml version="1.0"?>
<!DOCTYPE smil-ex SYSTEM "SmilEx.dtd">
<smil-ex>
 <head/>  <!-- details are omitted -->
 <body>
  <while>
   <sync-in master="true" id="SYNC-REQUEST" var="request" number="2"/>
   <input var="accept" region="TEACHER/BUTTON" val="ACCEPT REJECT"/>
   <if var="accept" op="eq" val="ACCEPT">
    <sync-out master="true" id="SYNC-STUDENT" val="%request;"/>
   </if>
  </while>
 </body>
</smil-ex>
```

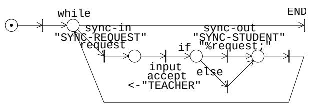

Figure 5: Scenario of Switch among Speaking Students

3.1. Synthesis Algorithm

In our synthesis algorithm, we derive a set of P protocol entity specifications. Let $pspec_k$ denote the protocol entity specification of node k.

At the starting point of derivation, node k has (a) a set of scenarios allocated to node k in the given `Presentation Definition File` and (b) a set of scenarios which have at least one element whose *responsible node* (defined later) is node k, as $pspec_k$. Also node k has all the variables used in the scenarios of (a).

We assume a responsible node for each element executed by more than one node. For example, `<input var="x" from="region_A"/>` element may be executed by two nodes, since the node which has `"region_A"` where an input value is given may be different from the one which has variable `"x"`. We decide a responsible node for such an element as follows.

- `<input>`: the node which has the region where an input value is given
- `<video>`: the node which receives the video.
- `<sync-in>`, `<sync-out>`: the master node of the synchronization (if master is not specified, one node is selected at random)

Then $pspec_k$ is derived as follows.

1. For each element (except `<input>`, `<sync>` and `<par endsync="first">`) in $pspec_k$, add `send/receive` elements after or before the element if the responsible node of its next or previous element is not node k.
2. For each `<input>` element in $pspec_k$, if node k is its responsible node, add `<send>` after the element. This sends the input value to the node which has the variable `"x"` specified in `<input>` element. If node k is the node which has variable `"x"`, add `<receive>` element to receive the input value and `<send>` element to let the next responsible node know the execution has been finished.
3. For each `<sync>` element in $pspec_k$, replace it with a set of elements. The details are explained later.
4. For each `<par endsync="first">` element in

$pspec_k$, replace it with a set of elements. The details are also explained later.

5. Delete any other element in $pspec_k$ which is not executed in node k.
6. Make a new scenario which has all the scenarios in $pspec_k$ as parallel components and let the scenario be $pspec_k$.

The idea is that we put `<send>` and `<receive>` elements at the points where the responsible nodes are changed between nodes. Also, they are put if the value should be transferred.

`<sync>` is implemented by the following message exchange.

- *Synchronization request message* : Each node except the synchronization master node sends this message to the master node to synchronize.
- *Synchronization accept message* : When node k is a synchronization master node, the node waits until receiving synchronization request messages from all the nodes that have the possibility of synchronization. Then when each synchronization request message is received, the master node evaluates the condition and sends synchronization accept messages to all the nodes which can join the synchronization if the condition is acceptable.

Those messages include values if needed.

Also we explain the implementation of `<par endsync="first"> ... </par>`. First, let us consider the case that no `<sync>` elements. are included The following message transmissions are added.

- *End message* : A message exchange from the node (node h) which executes the last action of each action sequence in concurrent action sequences to the responsible node (node p) of `<par endsync="first"> ... </par>` is added.
- *End notification message* : The responsible node sends the message to all the nodes concerned to stop their execution of `<par> ... </par>` .
- *Confirmation message* : After stopping their execution, the nodes send the confirmation messages to the responsible node. The responsible node p stops the execution of the actions and sends the execution control message explained above to the node where the next action is executed after receiving all the confirmation messages.

By adding those message exchanges in the last of each action sequence, the execution of `<par>` element is completed when one of action sequences has finished. There is a case that an interruption has occurred during the input value is transmitted after an input action is executed. Even in such a case, the input value will be sent to the responsible node (on which the variable is placed) and the update of the variable will be completed before the confirmation message to the end notification message is returned. We also explain the way for the case that `<par endsync="first"> ... </par>` includes `<sync>` element. In this case, the synchronization master node enters the *synchronization cancel state* when the node receives the end notification message. And when the node is in the synchronization cancel state, the node sends *synchronization cancel messages* to all the nodes from which the synchronization master node has received the synchronization request messages in `<par> ... </par>` (also the nodes from which the node receives the messages after it enters the synchronization cancel state). The actions of the other nodes depend on if the node has already sent the synchronization request message before receiving the end notification message.

- If the node has already sent the message, the node waits for the reply message from the synchronization master node. If the reply message is the synchronization accept message, the corresponding synchronization is executed. If the reply message is the synchronization cancel message, nothing is done. Then the interruption is completed after sending the confirmation message to the responsible node.
- In the other case, the node sends the confirmation message immediately and completes the interruption.

The responsible node sends *complete messages* to all the nodes that have synchronization elements in `<par>` after receiving all the confirmation messages. The synchronization master nodes leaves the cancel state after receiving the complete messages. The actions of `<par>` are all completed by the above way.

By deleting actions which are not executed in node k, we can obtain the scenarios corresponding to node k. Finally, the protocol entity specification of node k is synthesized from those scenarios by making all the scenarios run in parallel.

4. Deciding Routes for Object Transmission

Suppose that at most m objects may be transmitted at the same time in a protocol specification. In such a case, we formulate a mixed linear programming problem to maximize the minimum of unused bandwidth on each link for efficient utilization of network resources.

For this purpose, we introduce the following variables. Here, we assume that for each object transmission from one node to another, there are k possible routes.

- $st\_h (h = 1, \ldots, m)$: its value is 1 if object h is being transmitted.

- $path\_h\_ij\_q (q = 1, \ldots, k)$: its value is 1 if q-th route is selected for transmitting object h from node i to node j.
- $Min\_un\_used\_band$: the minimum of the bandwidth left (unused) on each link
- $un\_used\_band(L\_x)$: the unused bandwidth on link L_x
- $max(L\_x)$:the link capacity of L_x

Also we use the following terms.

- $band\_h$: the bandwidth required for the transmission of object h
- ST : the set of objects transmitted concurrently
- $path(L\_x)$: the set of $path\_h\_ij\_q$ containing link L_x

The problem is formulated as follows.

$$max : Min\_un\_used\_band \tag{1}$$

subject to:

$$st\_h = 1 (\text{for each } st\_h \in ST) \tag{2}$$

$$path\_h\_ij\_1 + \cdots + path\_h\_ij\_k = 1 \tag{3}$$

$$\sum_{path\_h\_ij\_q \in path(L\_x)} (band\_h \cdot path\_h\_ij\_q) \leq max(L\_x) \tag{4}$$

$$max(L\_x) \; - \sum_{path\_h\_ij\_q \in path(L\_x)} (band\_h \cdot path\_h\_ij\_q) = un\_used\_band(L\_x) \tag{5}$$

$$Min\_un\_used\_band \leq un\_used\_band(L\_x) \tag{6}$$

The equations (2) and (3) are obvious. (4) should hold due to the limitation of bandwidth. Then, the above problem can be solved using techniques for solving mixed LP problems.

We also consider the following three cases where streams are multicast streams, objects are replicated to several nodes, and the nodes where objects are placed are dynamically changed.

Multicast Here, if stream st_h is a multicast stream transmitted from node i through link L_x, we introduce 0-1 integer variables $multi\_h\_x$ whose value is 1 if either one of the routes $path\_h\_ij_1\_q_1, \ldots, path\_h\_ij_w\_q_w$ for multicast stream st_h uses link L_x. Then, we add the following inequality.

$$path\_h\_ij_y\_q_y \leq multi\_h\_x \qquad (1 \leq y \leq w) \tag{7}$$

Also, inequality (4) is modified by using variable $multi\_h\_x$.

Replicated Objects An object may be replicated and placed on several nodes. In such a case, we modify the mixed LP problem so that a route from one of those nodes is selected. Here suppose that stream $st\_h \in ST$ has v candidates of source nodes $i_1, \ldots, i_v$. Then we introduce v variables $st\_h\_i_1, \ldots, st\_h\_i_v$ and add the following equation.

$$st\_h\_i_1 + \cdots + st\_h\_i_v = st\_h \tag{8}$$

Also equation (3) is replaced by the following v equations $(1 \leq u \leq v)$.

$$path\_h\_i_u j\_1 + \cdots + path\_h\_i_u j\_k = st\_h\_i_u \tag{9}$$

Dynamic Change of Object Source Objects such as the video of a current speaker in a video-conference, the object source is changed dynamically. In this case, the routes should be decided so that used bandwidth on each link does not exceed its capacity even if any node is selected as the source node. Now suppose that a stream $st\_h \in ST$ has v possible sources. Let $i_1, \ldots, i_v$ denote the source nodes, and assume that $i_1, \ldots, i_v$ are different nodes for simplicity of discussion. Then we introduce v variables $st\_h\_1, \ldots, st\_h\_v$ and add the following equation.

$$st\_h\_i_1 = \cdots = st\_h\_i_v = st\_h \tag{10}$$

Also equation (3) is replaced by the following equation.

$$path\_h\_i_u j\_1 + \cdots + path\_h\_i_u j\_k = st\_h\_i_u \tag{11}$$

In these equations, the value of variable $path\_h\_i_u j\_k$ becomes 1 when k-th candidate of stream h is used to transmit the object from node i_u to node j. By this change, the routes of all streams $st\_h\_i_1, \ldots, st\_h\_i_v$ can be decided. Then we introduce a new variable $select\_h\_j\_x$ for each pair of st_h and L_x, which represents the bandwidth used by $path\_h\_ij\_q$ and add the following relation.

$$select\_h\_j\_x \geq band\_h\_i \cdot path\_h\_ij\_q \quad (path\_h\_ij\_q \in path(L\_x)) \tag{12}$$

In the above equation, $band\_h\_i$ represents the bandwidth of the source that is placed on node i in stream h. Also equation (4) is changed to replace the stream whose source is dynamically switched by variable $select\_h\_j\_x$. DYNAMIC denotes the set of streams whose sources are dynamically switched.

$$\sum_{st\_h \in \mathrm{DYNAMIC}} select\_h\_j\_x + \sum_{\substack{path\_h\_ij\_q \in path(L\_x) \\ st\_h \notin \mathrm{DYNAMIC}}} (band\_h \cdot path\_h\_ij\_q) \leq max(L\_x) \tag{13}$$

5. Experimental Results and Conclusion

5.1. Experimental Results

We have implemented the algorithm proposed in Sections 3 and 4, and we have also checked the relation between the number of candidates of the routes

Num. of Candidates	1	2	3	4
Min. unused bandwidth (%)	18.3	18.4	21.2	28.3
Ave. unused bandwidth (%)	28.6	39.7	46.8	53.6
Ave. Calculation time (sec.)	0.01	0.66	2.75	9.77

Table 1: Relation Between Number of Candidates and Unused Bandwidth

and the unused bandwidth . In order to evaluate our route selection algorithm in Section 4, we have generated some random networks with 25 nodes and placed all users on three nodes on each generated network. Then, we have calculated the results for the cases that the number of candidates of routes is changed from 1 to 4. When only one candidate is given, all objects are transmitted through their shortest paths. When the number of candidates is larger than one, the candidate which maximizes the minimum of unused bandwidth on each link is selected. We have calculated the results for 10 instances. The results are shown in Table 1. We have used Pentium III 600MHz with 256MB memory. This table shows how much unused bandwidth is left after applying our route selection algorithm. Here, the average unused bandwidth and the minimum unused bandwidth are shown. By increasing the number of the candidates from 2 to 4, the selection among several alternative routes becomes possible, which also makes the unused bandwidth large. The unused bandwidth can be increased not only by the selection among alternative routes but also by the use of multicast which decreases waste bandwidth by using common links. From those results, we believe that our method enables to use the network resources efficiently.

5.2. Implementation of SMIL-EX documents

Generally, in order to cope with shortage of network/system resources or packet losses/delays, QoS control mechanisms are required in multi-media applications. For the purpose, in [15], we have proposed an extension of SMIL (called *QOS-SMIL*) with some QoS control facilities, and implemented its player.

In QOS-SMIL, new statements for two major QoS requirements (1) a dynamic switching facility among alternative objects, and (2) a precise inter-media synchronization facility, are introduced to a sub-class of SMIL 1.0. The above facility (1) enables media-scaling of objects and dynamic server selection, which selects the optimal server among several available servers distributing the same content objects. And (2) allows users/designers to specify the maximum skew deviation in inter-media synchronization depending on user's allowance/preference or nature of objects.

In the proposed implementation technique, we use a subclass of E-LOTOS [16] (called *real-time LOTOS* [17, 18]) as an intermediate language since it can nicely treat timing constraints for actions and synchronization among parallel processes used in SMIL documents. We implement QoS control mechanisms using the constraint oriented style [19] of real-time LOTOS where a system is composed of a main process (e.g., video/audio playback) and several constraint processes (e.g., congestion detection, media scaling, inter-media synchronization, and so on). Those processes run in parallel satisfying the specified constraints. There are a lot of QoS control mechanisms to be implemented and for each control mechanism, several implementations can be considered depending on the system resources. Using the above technique, we can select and implement different combinations from such mechanisms/implementations suitable for target environments. Thus, the proposed technique can be used for rapid prototyping of the specified combination of QoS control mechanisms.

Based on the proposed technique, we have developed a converter from QOS-SMIL documents to the corresponding real-time LOTOS specifications. Using this converter and our real-time LOTOS compiler [17, 18], QOS-SMIL documents are implemented as executable programs with a real-time thread mechanism. In each generated program, object playback processes are assigned to the corresponding real-time threads, and all the threads are scheduled in EDF (Earliest Deadline First) manner.

In our experiment, the generated programs could playback each Motion-JPEG video with 176×120 pixels in 115 frames/sec. on a RedHat5.2 PC with Pentium III 500MHz and 128MB memory (also, 57 frames/sec. for two parallel video playbacks and 37 frames/sec. for three) where we used JPEG library of a free-ware called XAnim. In another experiment, we have also confirmed that the generated programs with inter-media synchronization could regulate the maximum skew deviation between two parallel video playbacks in about 20msec. on normal UNIX (RedHat5.2). For details of QOS-SMIL, its implementation technique and experimental results, see [15].

For implementation of SMIL-EX documents in this paper, we are currently trying to add some new elements used in SMIL-EX to the above QOS-SMIL tool so that the elements such as `<send>`, `<receive>`, `<while>` and `<if>` can be treated.

5.3. Conclusion

In this paper, we have proposed a design method based on protocol synthesis for multi-user multimedia presentation systems. In this method, we have defined a SMIL-like language called SMIL-EX and a service specification of the whole system is described as a set of scenarios corresponding to users' respective behavior and interaction among those users using multi-way synchronization. We have also proposed a protocol synthesis technique which derives each node's protocol entity specification including asynchronous message exchanges for implementing multi-way synchronization among nodes. To maximize the unused bandwidth on the network, we have also proposed a route selection algorithm for each object transmission when several routes exist for the transmission. As a total, in the proposed method, we can describe abstract specifications in SMIL-EX, while in the implementation level, our protocol synthesis method and route selection algorithm for avoiding congestion can make the specified distributed multi-media presentation systems work efficiently.

Development of the SMIL-EX player and evaluation of the proposed method by applying to more practical examples is part of future work.

[1] K. Saleh : "Synthesis of Communication Protocols: an Annotated Bibliography", *ACM SIGCOMM Computer Communication Review*, Vol.26, No.5, pp.40–59 (1996).

[2] P. -Y. M. Chu and M. T. Liu : "Protocol Synthesis in a State-transition Model", *Proc. of COMPSAC '88*, pp. 505–512 (1988).

[3] R. Gotzhein and G. v. Bochmann : "Deriving Protocol Specifications from Service Specifications Including Parameters", *ACM Trans. on Computer Systems*, Vol.8, No.4, pp.255–283 (1990).

[4] T. Higashino, K. Okano, H. Imajo and K. Taniguchi : "Deriving Protocol Specifications from Service Specifications in Extended FSM Models", *Proc. of 13th Int. Conf. on Distributed Computing Systems (ICDCS-13)*, pp.141–148 (1993).

[5] H. Yamaguchi, K. Okano, T. Higashino and K. Taniguchi : "Synthesis of Protocol Entities' Specifications from Service Specifications in a Perti Net Model with Registers", *Proc. of 15th Int. Conf. on Distributed Computing Systems (ICDCS-15)*, pp.510–517 (1995).

[6] R. Langerak : "Decomposition of Functionality; a Correctness–Preserving LOTOS Transformation", *Proc. of 10th IFIP WG6.1 Symp. on Protocol Specification, Testing and Verification (PSTV-10)*, pp.229–242 (1990).

[7] A. Khoumsi, G. v. Bochmann and R. Dssouli : "Protocol Synthesis for Real-Time Applications", *Proc. of IFIP Joint Int. Conf. On 12th Formal Description Techniques and 19th Protocol Specification, Testing and Verification (FORTE/PSTV'99)*, pp.417–433(1999).

[8] W3C : "Synchronized Multimedia Integration Language", (SMIL) 1.0 Spec., *http://www.w3c.org/TR/REC-smil/*

[9] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala : "RSVP : A New Resource ReSerVation Protocol", *IEEE Networks*, Vol.7, pp.8–18 (1993).

[10] C. Diot, W. Dabbous and J. Crowcroft : "Multipoint Communication : A Survey of Protocol, Functions, and Mechanisms", *IEEE Journal on Selected Areas in Communications*, Vol.15, No.3, pp.277–290 (1997).

[11] Z. Wang and J. Crowcroft : "Quality-of-service routing for supporting multimedia applications", *IEEE Journal on Selected Areas in Communications*, Vol.14, pp.1288–1234 (1996).

[12] C. Pornavalai, G. Chakraborty and N. Shiratori : "QoS Based Routing Algorithm in Integrated Services Packet Networks", *Proc. of IEEE 1997 Int. Conf. On Network Protocols (ICNP'97)*, (1997).

[13] W3C : "Extensible Markup Language", (XML) 1.0 W3C Recommendation, *http://www.w3c.org/TR/REC-xml/*

[14] T. Murata : "Petri Nets : Properties, Analysis and Applications", *Proc. of IEEE*, Vol.77, No.4, pp.541–580 (1989).

[15] Y. Terashima, K. Yasumoto, T. Higashino, K. Abe, T. Matsuura and K. Taniguchi : "Extension of SMIL with QoS Control and its Implementation", to appear in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME2000)* (2000).

[16] ISO : "Final Committee Draft 15437 on Enhancements to LOTOS", *ISO/IEC JTC1/SC21/WG7* (1998).

[17] T. Higashino : "Design and Implementation of real-time LOTOS compiler", *IPA/MITI (Japanese Ministry of International Trade and Industry), http://www-tani.ics.es.osaka-u.ac.jp/IPA/* (1999).

[18] H. Tatsumoto, K. Abe, K. Yasumoto, T. Higashino, T. Matsuura, H. Yamaguchi and K. Taniguchi : "Development of Realtime LOTOS Compiler and Its Application to Multimedia Systems", *Trans. of IPSJ*, Vol. 41, No. 2, pp.424–434 (2000) (in Japanese).

[19] C. Vissers A., G. Scollo and M. v. Sinderen : "Architecture and Specification Style in Formal Descriptions of Distributed Systems", *Proc. of 8th Int. Conf. on Protocol Specification, Testing, and Verification (PSTV'88)*, pp.189–204 (1988).