マルチメディア通信と分散処理 97−33
コンピュータセキュリティ 8−33
(2000. 3. 22)

SMIL 風シナリオ群からの QoS を考慮したプロトコル合成

梅津 高朗† 北道 淳司† 船曳 信生† 山口 弘純† 安本 慶一†† 東野 輝夫†
†：大阪大学大学院基礎工学研究科情報数理系
††：滋賀大学経済学部管理科学科
umedu@ics.es.osaka-u.ac.jp

本稿では，複数ユーザからなる分散マルチメディアプレゼンテーションシステムの設計・開発を目的に，ネットワークの各リンクの利用可能な未使用帯域を最大にするような QoS 保証機構を組み込んだプロトコル合成法を提案する．提案する手法では，各ユーザノード上で表示したいマルチメディアプレゼンテーションの内容とその実行順序を SMIL 風のシナリオとして記述する．システム全体の動作 (サービス仕様) は，これら全ノードでの SMIL 風シナリオとシステム全体の実行制御などに関する制約を記述したシナリオの集合として記述する．このサービス仕様とネットワークのトポロジー，各リンクの使用可能帯域，各マルチメディアデータの配送に必要な帯域などの QoS 情報から，各ノードの動作内容と実行順，ならびに，各マルチメディアデータの配送経路などを記述した各ノードの動作仕様群 (プロトコル仕様) を自動生成する．

QoS Based Protocol Synthesis from SMIL-like Scenarios

Takaaki Umedu† Junji Kitamichi† Nobuo Funabiki† Hirozumi Yamaguchi†
Keiichi Yasumoto†† Teruo Higashino†
†：Dept. Info. and Math. Sci., Osaka Univ., Japan
††：Fac. of Economics, Shiga Univ., Japan

In order to design and develop multi-user multimedia presentation systems, we propose a protocol synthesis method considering QoS requirements for network. In this method, the scenario of each entity (node) in a multimedia presentation system is specified in a SMIL-like language. A specification of the total system (service specification) is described as a set of all entities' SMIL-like scenarios and scenarios specifying the interaction between entities. For a given service specification, network topology, link capacity and required bandwidth for each media, a protocol specification is automatically derived. The derived protocol entity specification includes (1) the behavior of each entity, and (2) network paths through which media are sent.

1 まえがき

近年のネットワークの高速化やマルチメディア技術の発展に伴い，様々な分散マルチメディアシステムが設計・開発されている．このような分散システムの信頼性を高め，効率よくシステムを設計するための手法としてプロトコル合成法に関する研究が活発に行われている [4]．プロトコル合成法では，処理を行っている計算機は実際にはネットワークを介して分散していても，それらの処理があたかも単一の計算機上で行われているようにシステム全体の動作を記述し，その記述 (サービス仕様と呼ばれる) からサービス仕様通りに動作する各計算機の動作仕様 (その計算機で実行する処理の内容と手順のみならず，サービス仕様に書かれた順に動作を実行するための同期メッセージやデータの送受信動作を加えた仕様) の集合 (プロトコル仕様と呼ばれる) を自動生成する．

一方，特に分散マルチメディアプレゼンテーションシステムなどでは，プレゼンテーションのシナリオの変化やディスプレイ上に表示する話者の変更などに伴いマルチメディアデータの送信元ノードが動的に変化する場合も多い．それらの変化に対しても安定したメディア送受信を行うための適切な End-to-End の QoS 保証も望まれる．

本稿では，ネットワークの各リンクの未使用帯域を最大にすることで，バーストなどによる帯域変動に対してもなるべく安定したメディア送受信を保証できるマルチメディアプレゼンテーションシステムの設計・開発のためのプロトコル合成法に基づく手法を提案する．提案する手法では，まず各計算機 (ノード) 上で表示したいメディアとその表示制御の実行順序をマルチメディアプレゼンテーションなどのシナリオ記述言語としての標準化が進められている SMIL[1] を拡張した言語で記述する．複数ノードからなるシステム全体の動作仕様 (サービス仕様) は，これら全ノードでの SMIL 風シナリオ群とシステム全体の実行制御などに関する制約を記述したシナリオの集合として記述する．このサービス仕様とネットワークのトポロジー，各リンクの使用可能帯域，各メディアの配送に必要な帯域などの情報から，各ノードの動作内容と実行順，ならびに，各メディアの配送経路などを記述

した各ノードの動作仕様群 (プロトコル仕様) を自動生成する．また，各メディアの配送経路の候補 (あるいはサーバーの候補) が複数ある場合，それらのメディアをどの経路で配送すれば特定のリンクに配送が集中しないかなどを線形計画法の手法を用いて解析することで，各リンクの未使用帯域を最大にするようなプロトコル仕様を生成する．

以下，2 章で提案する SMIL 風言語と記述モデル，並びに，提案する言語を用いたマルチメディアプレゼンテーションシステムの記述例について述べる．3 章でサービス仕様を実現する一つのプロトコル合成法を与え，4 章で QoS を考慮したメディアの配送経路の決定法について述べる．

2 サービス仕様の記述モデルと言語

本稿では複数ユーザからなるマルチメディアプレゼンテーションシステムなどのシナリオを SMIL に幾つかの機能拡張した言語 (SMIL-EX1 と呼ぶ) で記述する．

2.1 SMIL-EX1

一般に，SMIL ドキュメントは XML エレメント [2] により記述する．SMIL ドキュメントでは，動画や音声，静止画，テキストなどのメディアデータを**オブジェクト**と呼び，これらのオブジェクトについて，表示されるエリアの幅，高さ，座標などの**レイアウト情報**の指定と，再生オブジェクトの実行タイミング (開始時刻，終了時刻，再生期間) など**オブジェクトの挙動**が別々に記述される．レイアウトは主に region エレメントで表示範囲やレイアウトなどを指定する．オブジェクトの挙動を記述するために par，seq エレメントがあり，オブジェクトを並行，逐次に再生することができる．また，代替コンテンツを指定しておくために switch エレメントなどがある．オブジェクトを示すためには src 属性が利用でき，オブジェクトをファイル名 (ハイパーリンク名) で指定する．オブジェクトの再生開始，終了や再生期間，再生範囲は begin，dur，end，clip-begin，clip-end 属性で指定する．

しかし，SMIL ではいわゆる while 文に相当する制御や変数に外部入力などの値をセットする代入文，変数値に依存して選択を行う if 文に相当する制御機構を指定できない．本稿ではこれらの指定のためのエレメントを SMIL に加えた言語として SMIL-EX1 を定義し，それを用いてサービス仕様を記述する．SMIL-EX1 では，while，if，input，set の 4 つのエレメントを導入する．例えば，<input var="x" from="region_A" val="A B C"/>で領域 region_A からの (ボタン) 入力 (val で指定された A，B，C のいずれか) を変数 x にセットすることを表し，<set var="x" val="A"/>で変数 x に定数 A をセットすることを表す．変数 x の値の参照は%変数名; のように行う．参照はその SMIL-EX1 テキスト中のいずれの場所でも可能で，実行時に%変数名; の部分が変数の値で置き換えられる．例えば，http://%x;/video.mpg のように指定することで表示するビデオのソースを動的に変更するようなシナリオを記述できる．また，<input>エレメントでは，入力元の指定に他のノードの region を指定することが出来る．そのための機能として，各ノードにはノードの ID が設定されており，ノード ID/region の形で他ノードの region を表す．

さらに，複数人の協調作業での同期を指定するために，<sync-in id="SYNC-ID" var="x"/>，<sync-out id="SYNC-ID" val="y"/>エレメントを定義する．これは同じ ID を持つ<sync-out>，<sync-in>エレメント同士が同期し，その際<sync-out> の属性値 val で指定された値 y が，<sync-in>の属性値 var の変数 x へ代入される．同期には n 個の<sync>エレメント中の任意の k 個の<sync>エレメントが同期するような指定も可能である．また，同期に必ず参加するマスターを指定することも可能で，その場合は属性として master="true"を加える．マスターノードは同期が実行されるための条件を指定することが可能で，それらは number，val 属性で指定する．number="2"の指定があった場合，同期しようとしているノードが多数あっても，そのうちマスターと他の一つのノードのみが同期することになり，排他制御などが簡潔に記述できる．マスターノード指定のある<sync-in>の属性値 val は受け付ける値を列挙し，列挙された値のうちのいずれかが転送される場合にのみ同期が行われる．

なお，上述の仕様は，概念的には安全活性な自由選択ネット [3] と呼ばれるペトリネット構造 (同期並行有限状態機械モデル) に変数を取り扱えるようにしたモデル [5] でモデル化できる．以下では理解を深めるため，SMIL-EX1 記述をペトリネットで表現する場合がある．なおその際，<par endsync="first"> (並列実行し，その内最初に終わったものに合わせて全体の並列実行を終了させる) のような割り込み動作はペトリネットで簡潔に記述できないので，図式を簡単にするため，割り込み部分は破線で囲む形で略記表現している．

2.2 サービス仕様の記述モデル

本稿では，複数ユーザからなるマルチメディアプレゼンテーションシステム (例えば，遠隔授業システムなど) の仕様は，各ユーザの端末にどのような画像をどのような順に表示するかといった各ユーザノードのシナリオと，どの生徒の映像をどのように切り替えて表示するかといった生徒画面の選択や複数ユーザ間の排他制御などシステム全体の挙動を記述したシナリオの集合で構成されると考える．

以下では，教師と複数の生徒からなる遠隔授業システムを例に，仕様記述の仕方を説明する．まず，図 1は各生徒画面のシナリオの例である．この例では，SYNC-START(図 1の 13 行目) に同期して授業が開始され，SYNC-END(図 1の 28 行目) に同期して繰り返し (REPEAT)，または，終了 (STOP) する．授業中は，(1) 教材となる映像 (26 行目の*Text_1/text.mpg) と (2) 教師の映像 (27 行目の Teacher/teacher.mpg)，(3) 発言を求めている生徒の映像 (19 行目の%student;/student.mpg)，の 3 つを並行して表示する ("*"の意味は後述)．また，16〜21 行目までの<while>では，18 行目で SYNC-STUDENT で新しい生徒名が変数 student に代入されるごとに該当する生徒の映像への切り替えが行われる．22〜25 行目の<while>では，生徒が発言を希望する場合に SYNC-REQUEST 同期でその要求を伝える．SYNC-REQUEST で同期が成功した場合，変数%node-id; に書かれたそのノードの ID が他ノードに伝えられる．

また，各映像の所在地の指定は 3 通り可能で，

- 具体的にメディアのロケーションが決定している場合 (上記の例では教師の映像を表すリンク先 Teacher)，そのロケーションを直接指定する．
- 教材となる映像がネットワーク上の複数のサーバーにある場合，*Text のように，複数のサーバーの中から 1 つを選択してよいことを表すマーク"*"を付けておく．*Text の選択肢は別に定義ファイルを作成して記述する (図 2参照)．
- 発言を求めている生徒の映像 (%student; /student.mpg) のようにどの生徒の映像を表示するかが動的に変化する場合，変数%student; を用いて具体的な生徒名を実行時に動的に決定する．

次に，図 3に教師画面のシナリオ例を示す．教師画面のシナリオでは，SYNC-START 同期の後，教材となる映像 (*Text/text.mpg) と指名されている生徒の映像 (%student;/student.mpg) を並行に表示する．

いま，図 3の教師画面のシナリオと図 1の生徒画面のシナリオ n 個 (生徒 1,..., 生徒 n のシナリオと呼ぶ) を用いて遠隔授業を行うことを考える．この場合，これら

```
 1: <?xml version="1.0"?>
 2: <!DOCTYPE smil-ex1 SYSTEM "SmilEx1.dtd">
 3: <smil-ex1>
 4:  <head>          (詳細は省略)
 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-ex1>
```

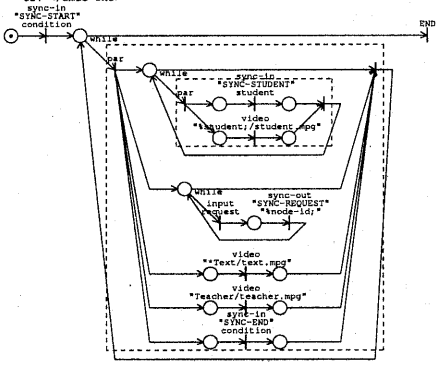

図 1: 生徒ノードのシナリオとペトリネット構造

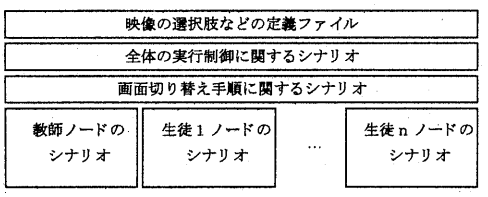

図 2: サービス仕様

のシナリオに加え，全体の実行制御を記述した図 4のようなシナリオと，生徒の画面の選択方法に関する制約を記述した図 5のようなシナリオ，並びにメディアに複数のサーバー候補がある場合の選択肢 (前述の"*")，ノード ID に関する情報などを記した定義ファイル (詳細は省略) の集合で全体の挙動を表すと考え，それらの集合をサービス仕様と呼ぶ (図 2).

図 4の全体の実行制御用シナリオでは，教師が開始 (START) のボタンを押すと SYNC-START の同期が実行される．教師もしくは生徒 1 (クラス委員長に相当) が繰り返し (REPEAT) か終了 (STOP) を選択することが可能で，いずれかがこれらのボタンを押した時点で SYNC-END の同期が実行される．

図 5 の生徒画面の選択方法に関する制約記述部では，どの生徒が発言を求めているかに応じて生徒の映像 (**%student;/student.mpg**) を動的に切り替えている．発言を求める場合には，SYNC-REQUEST 同期が実行される．この同期は，先生 (同期に必ず参加するマスター) と一人の生徒の間で行われるため，常に生徒のうちの一人が排他的に選ばれる．その後，先生が許可 (ACCEPT) ボタンを押すか，却下 (REJECT) ボタンを押す．許可された場合には全員に対してその生徒の画像を表示する

```
 1: <?xml version="1.0"?>
 2: <!DOCTYPE smil-ex1 SYSTEM "SmilEx1.dtd">
 3: <smil-ex1>
 4:  <head>  (詳細は省略)
 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-ex1>
```

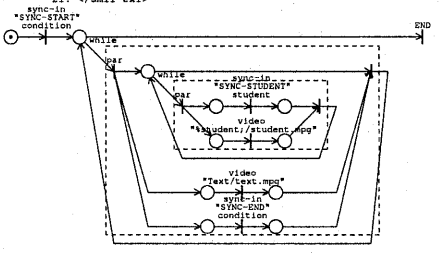

図 3: 教師ノードのシナリオ

```
 1: <?xml version="1.0"?>
 2: <!DOCTYPE smil-ex1 SYSTEM "SmilEx1.dtd">
 3: <smil-ex1>
 4:  <head/>
 5:  <body>
 6:   <input var="condition" region="TEACHER/BUTTON" val="START"/>
 7:   <sync-out master="true" id="SYNC-START" val="%condition;"/>
 8:   <while var="condition" op="ne" val="STOP">
 9:    <set var="condition1" val="" />
10:    <set var="condition2" val="" />
11:    <par endsync="first">
12:     <input var="condition1" region="TEACHER/BUTTON" val="STOP REPEAT"/>
13:     <input var="condition2" region="STUDENT1/BUTTON" val="STOP REPEAT"/>
14:    </par>
15:    <if var="condition1" op="ne" val=""><then>
16:     <set var="condition" val="%condition1;"/>
17:    </then><else>
18:     <set var="condition" val="%condition2;"/>
19:    </else></if>
20:    <sync-out master="true" id="SYNC-END" val="%condition;"/>
21:   </while>
22:  </body>
23: </smil-ex1>
```

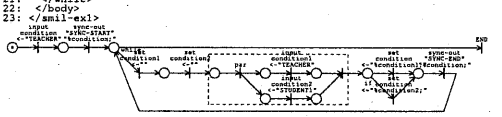

図 4: 全体の実行制御に関するシナリオ

ように同期 SYNC-STUDENT が実行され，選ばれた生徒名が皆に知らされる．

3 プロトコル仕様

3.1 プロトコル仕様の記述

2 章で述べたシナリオ群では複数ノードが関わる動作列や割り込みを一つのシナリオで記述したりシナリオ間のデータ交換は同期通信を仮定したりしている．一方，実際のネットワーク環境ではこれらを非同期通信で実現する必要がある．そこで，実装レベルのプログ

```
 1: <?xml version="1.0"?>
 2: <!DOCTYPE smil-ex1 SYSTEM "SmilEx1.dtd">
 3: <smil-ex1>
 4:  <head/>
 5:  <body>
 6:   <while>
 7:    <sync-in master="true" id="SYNC-REQUEST" var="request" number="2"/>
 8:    <input var="accept" region="TEACHER/BUTTON" val="ACCEPT REJECT"/>
 9:    <if var="accept" op="eq" val="ACCEPT">
10:     <sync-out master="true" id="SYNC-STUDENT" val="%request;"/>
11:    </if>
12:   </while>
13:  </body>
14: </smil-ex1>
```

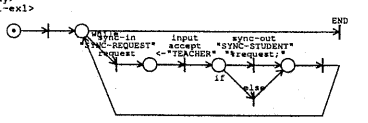

図 5: 表示する生徒画像の切り替え手順

ラム (プロトコル仕様と呼ぶ) は，SMIL-EX1 を少し変更した言語 (SMIL-EX2 と呼ぶ) で記述する．この言語は，前述の SMIL-EX1 から<sync>エレメントを省き，<input>動作をそのノードからの入力のみに限定し，その代わりに送受信動作<send>と<receive>を加えたものである．<send id="MessageID" to="NodeID" val="Value">で，ノード"NodeID"に対して"Value"を送信する．"NodeID"に複数ノードの ID が指定された場合には，指定された全てのノードに対してメッセージが送られる．各メッセージは"MessageID"によって一意に区別される．受信動作 (<receive>) も同様に定義される．

3.2 プロトコル仕様の導出

プロトコル仕様の導出に際して，まずすべてのエレメントに対してその動作の実行に責任を持つノード (責任ノードと呼ぶ) を決める．映像の表示や入力動作などは該当するノードを責任ノードとする．各シナリオ中の変数はシナリオ毎に一つのノードに保持されるものとし，<if>や<while>，<set>，<sync>エレメントも変数が配置されるノードを責任ノードとする．<par endsync="first">エレメントも変数が配置されたノードを終了割込に責任を持つ責任ノードとして設定する．また，同期エレメントでマスターが指定されていない場合，適当にマスターを決める．

以下では，プロトコル合成を行う上で議論を簡単にするため，次のような仮定を置く．

- サービス仕様はデッドロックを含まない
- 同期エレメントのそれぞれの ID の同期マスターは単一のノードとする
- <if>エレメントなどで非決定的な選択動作が行われる場合，直後の動作は共に同じ責任ノードで実行される
- <par>文などで並行に実行可能な<input>エレメントや<set>エレメントで同時に同じ変数に値を代入することがない
- <par endsync="first"> ... </par>中に同期マスターが指定された<sync>エレメントがない
- 各ノード i からノード j への送信でメッセージの紛失はなく，送信メッセージは送信順に到着する

これらの制約はプロトコル合成を単純化するための制約であり，本質的な制約ではない．これらの制約は，例えば，仮想同期マスターノードを設定したり，トークンを関係ノードで巡回させたり，適当な排他制御機構を組み入れるなどの工夫で解消できる．

各ノード k のプロトコル仕様の導出は以下の手順で行う．ただし，以下では一つのノード k に配置されている変数の集合を *Var(k)* で表す．

- すべてのシナリオからノード k が責任ノードとして含まれるシナリオ群と *Var(k)* の変数を含むシナリオ群，ノード k が同期マスターであるような<sync>エレメントを含むシナリオ群を取り出す (それらの集合を *scenario(k)* とする)
- ノード k の動作の前後に，前後の動作を行うノードとの送受信動作を追加する
- *Var(k)* の変数の入力を行う<input>エレメントの後に，そのエレメントの責任ノードからその変数が配置されているノードに更新後の変数の値を知らせる送受信動作と，変数の値を更新する動作，変数値の更新が行われたノードからこのエレメントの次の動作を行うノードへの送受信動作を追加する
- ノード k の各同期動作をそれを実現する他ノードとの送受信動作の列に置き換える (詳細は後述)
- ノード k の各割り込み動作をそれを実現する他ノードとの送受信動作の列に置き換える (詳細は後述)
- ノード k 以外のノードの動作を削除する
- 得られたシナリオ群を並行に動作させることで一つのシナリオにまとめる

上述の手順では，まずノード k と関係のあるシナリオ群 *scenario(k)* を取り出し，次に同期に必要な送受信動作を追加している．あるノード k1 の動作 a と他ノード k2 の動作 b が引き続いて実行される場合，a の実行に引き続いて b の動作を実行させるためのメッセージ交換 (以下，実行制御メッセージと呼ぶ) が必要となる．

また，<input>エレメントが実行されると変数の値が入力値に更新される．これらの変数を持つノードが入力を行うノードと異なる場合，更新後の変数の値を通知する必要がある (以下，変数値通知メッセージと呼ぶ)．また，変数の値を更新したノードから次の動作を行うノードへ更新終了メッセージを転送してから次の動作を実行するようにすることにより，変数の値の更新が終了する前に次の動作が実行されることを防いでいる．なお，更新終了メッセージが転送される場合，上述の実行制御メッセージを省略できる．また，<if>や<while>エレメントの条件節に含まれる変数はそれらの変数が配置されるノードを責任ノードとしているので，上述の操作により，<if>や<while>エレメントの条件節を評価しなければならない場合，その責任ノードに保持している変数の値を用いて条件節の真偽を判定できる．

また，<sync>エレメントが用いられている場合，これらの同期制御機構を非同期の送受信動作で実現する必要がある．本稿では以下のような方法で同期制御機構を実現する．まず，同期マスター以外のノードの場合，同期マスターへ同期希望メッセージを送信し，同期マスターから同期許可メッセージを受信することで同期を行う．この際，同期条件の判定に用いる値や自身の ID を同期希望メッセージに含めて送信する．ノード k が同期マスターノードの場合，同期に参加する可能性のある全てのノードから同期希望メッセージを受信できるようにしておき，同期希望メッセージを受信するごとに同期の条件を判定し，条件が整い次第，同期に参加できるノードへ同期許可メッセージを送信する．これらの同期を行う際に変数の値を交換するよう指定されている場合，同期マスターが必要な値を同期許可メッセージに含めて送信する．同期マスターが同期可能になる前に送られてきた同期希望メッセージや同期に参加できなかったノードからの同期希望メッセージは同期マスターノードのバッファに保存され，次の同期判定に利用されるものとする．

次に，<par endsync="first"> ... </par>のような終了割り込み動作が指定されている場合について考える．まず，<par endsync="first"> ... </par>中に<sync>エレメント (同期制御機構) が含まれていない場合を考える．この場合，<par endsync="first"> ... </par>中で並行に動作する各動作列の最後に，最後の動作を実行するノード (ノード h とする) から<par endsync="first"> ... </par> の責任ノード (ノード p とする) への終了メッセージの送信動作を追加する．そのメッセージを受信した責任ノードは *scenario(k)* に含まれるすべてのノードにその動作列の終了を通知するメッセージ (終了通知メッセージと呼ぶ) を送信する．また，これらの終了通知メッセージを受信したノードは責任ノードに受信確認メッセージを送信し直ちに<par> ... </par>の実行を停止する．責任ノード p はそれらのノードからのメッセージを全て受信した後，<par> ... </par>を終了し，上で述べた実行制御メッセージを次の動作を行うノードへ送る．これらの送受信動作を<par>で並列に実行される系列の最後にはさむことにより，いずれかの動作列の実行が終了した場合，関係するすべてのノードが<par>エレメントから抜け出すことになる．なお，この方法では変数値が入力され，その値が変数を保持するノードに送信中に終了割り込みが発生する場合がある．そのような場合でも，終了通知メッセージに対する受信確認メッセージが返信されるまでに責任ノード (変数が配置されているノード) に変数値が送られ，変数の値の更新が完了する．次に，<par endsync="first"> ... </par>中に<sync>エレメントが含まれる場合を考える．この場合には，同期マスターは終了通知メッセー

ジを受けると同期キャンセル状態になり，その状態の場合，それまでにこの<par> ... </par>から受信した(及び，それ以降に受信した)同期依頼メッセージ全てに対して同期キャンセルメッセージを返す．同期マスター以外のノードは，終了通知メッセージを受けた時点で同期依頼メッセージを送信しているかどうかで動作が変わる．すでに送信している場合は，同期マスターノードからの返信を待つ．返信は，同期許可か同期キャンセルかの2通りあり，許可の場合は通常どおりに同期の処理が行われ，キャンセルの場合は何もしない．その後，受信確認メッセージを責任ノードへ返信し割り込み終了する．同期依頼メッセージを未送信のときには，直ちに受信確認メッセージを責任ノードへ返信し割り込み終了する．責任ノードは全ての受信確認メッセージを受信した後に，その<par>内に含まれる同期の同期マスターに対して完了メッセージを送信する．完了メッセージを受信した同期マスターは同期キャンセル状態を終了する．以上の動作で，<par>内の動作は全て完了する．なお，<par endsync="first">と指定された場合には，<par endsync="first" id="ParID">のように id 属性を追加して"ParID"でそれぞれを一意に区別する．終了メッセージ，受信確認メッセージ，完了メッセージには<par>エレメントの ID を付与することで，どの<par>エレメントに対するメッセージなのかを明確に区別する．

4 マルチメディアデータの配送経路の決定

4.1 各ストリームの転送経路の決定

以下では，シナリオの実行において，高々m個のメディアストリームがネットワークを介して同時に転送される状況を考える(必ずしもm個のストリームが同時に転送されるとは限らない)．このとき，各ストリームがネットワークに流されているとき真になるような論理変数を$st_1,\ldots,st_m$とし，各ストリームst_hの転送に必要な帯域を$band_h$とする．また，ストリームst_hがノードiからノードjへのストリームで，そのストリームを転送するのにk個の経路候補があるとする．各ストリーム毎に候補の数kが変化してもよいが，ここでは簡単のためすべてのストリームがk個の経路候補を持つと仮定する．それらの経路にストリームst_hが流されるとき真になる論理変数を$path_h_ij_1,\ldots,path_h_ij_k$とする．いま，$ST$をネットワーク上を同時に流れるストリームの集合とすると，次の関係が成り立つ．

$$st\_h = 1(\text{すべての } st\_h \in ST\text{に対して}) \quad (1)$$

さらに，$st\_h \in ST$ なる各 st_h に対して，次の関係が成り立つ．

$$path\_h\_ij\_1 + \cdots + path\_h\_ij\_k = st\_h \quad (2)$$

また，各ストリームst_hの経路$path_h_ij_q$がネットワーク上のあるリンクL_xを通るとき$path_h_ij_q \in path(L_x)$であるとし，$max(L_x)$をリンク$L_x$の最大転送可能帯域とすると，リンク$L_x$には$max(L_x)$以上の帯域のストリームを流すことができないので，次の関係が成り立つ．

$$\sum_{path\_h\_ij\_q \in path(L\_x)} (band\_h \cdot path\_h\_ij\_q) \leq max(L\_x) \quad (3)$$

この式の左辺を$used_band(L_x)$とすると，$used_band(L_x)$はリンクL_xを使用する帯域を表す．

もし，あるストリームst_hがノードiからノード$j_1,..,j_w$へのマルチキャストストリームである場合，$path_h_ij_1_q_1,\ldots,path_h_ij_w_q_w$が同じリンク$L_x$を通過する($path\_h\_ij_1\_q_1,\ldots,path\_h\_ij_w\_q_w \in path(L\_x)$)可能性がある．その場合，上の(3)式のままではストリームst_hを送るのに必要な帯域$band_h$をw倍に計算していることになる．そこで，st_hがノードiからのマルチキャストストリームでリンクL_xを通過する場合，そのマルチキャストストリームが通過することを表す論理変数$multi_h_x$を新たに導入し，転送経路$path_h_ij_1_q_1,\ldots,path_h_ij_w_q_w$のいずれかが$L_x$を使用するとき$multi_h_x$が真になるよう，次の式を追加する．

$$path\_h\_ij_y\_q_y \leq multi\_h\_x \quad (1 \leq y \leq w) \quad (4)$$

また，(3)式でw倍に重複して計算している部分を$band_h \cdot multi_h_x$に置き換える．なお，一つの転送先ノードj_yに対して$path_h_ij_y_q_y, path_h_ij_y_q'_y$のように2つ以上の転送経路が通過する場合もあれば，ある転送先ノードへのマルチキャストストリームのいずれの経路もL_xを通過しない場合もある．前者の場合，$path(L_x)$に$path_h_ij_y_q_y, path_h_ij_y_q_y$の2つを含めればよいし，後者の場合，その目的ノードj_yに対する論理変数$path_h_ij_y_q_y$は$path(L_x)$に含まれないものとすればよい．

上述のような制約条件のもと，すべてのL_xに対して

$$max(L\_x) - used\_band(L\_x) = un\_used\_band(L\_x) \quad (5)$$

$$Min\_un\_used\_band \leq un\_used\_band(L\_x) \quad (6)$$

とすると，$un_used_band(L_x)$はリンクL_xでの未使用帯域を表し，$Min_un_used_band$はすべてのリンクの未使用帯域の最小値を表す．例えば，この最小値$Min_un_used_band$を大きくすることができれば，各リンクの使用可能帯域に変動が生じた場合やジッタなど短期間の帯域変動に対してもストリームの転送がスムーズに行われる可能性が高くなる．$band_h$や$max(L_x)$，$used_band(L_x)$などの帯域を表す変数を実数変数とすると，(1)〜(6)の制約条件は0-1整数変数を含む線形計画問題に帰着でき，線形計画法の解法を用いて，$Min_un_used_band$の値を最大にするような解を求めることができる．その解を用いれば，すべてのリンクの未使用帯域の最小値$Min_un_used_band$の値を最大にするような各ストリームの転送経路の決定が行える．

4.2 送信元の変動のあるストリームの取り扱い

4.2.1 複数の送信元から一つの送信元を選ぶ場合

使用帯域の分散のために，1つのマルチメディアデータをネットワーク上の複数箇所に置いておくことが考えられる．受信者はそれらの内，最も適切な任意のノードを選択して受信を行う．これらは異なる送信元からの複数の経路のうち一つの経路を選択するように制約を記述すればよい．いま$st_h \in ST$なる一つのストリームst_hに対して，その送信元がノード$i_1,\ldots,i_v$のv個の可能性がある場合を考える．その時，$st_h_i_1,\ldots,st_h_i_v$なるv個の変数を導入し，次の式を追加する．

$$st\_h\_i_1 + \cdots + st\_h\_i_v = st\_h \quad (7)$$

また，前節の(2)の式を次のようなv個の式($1 \leq u \leq v$)に置き換える．

$$path\_h\_i_uj\_1 + \cdots + path\_h\_i_uj\_k = st\_h\_i_u \quad (8)$$

このように変更することで，$st_h_i_1,\ldots,st_h_i_v$のうち最も適当な一つのストリームが選択される．

4.2.2 複数の送信元が動的に切り替えられる場合

例えば，ビデオ会議で生徒の映像を配信する場合など，どの生徒が選ばれるかによって一つのビデオストリームの送信元が動的に変動する．その場合，複数の送信元のいずれから映像が送られても各リンクの使用帯域を上まわらないように経路を決定できることが望ましい．以下

では，その一つの解決策を与える．いま $st\_h \in ST$ なる一つのストリーム st_h に対して，そのソースが v 個存在し，動的に切り替えられる場合を考える．これらの送信元を ノード $i_1, \ldots, i_v$ とし，簡単のため $i_1, \cdots, i_v$ は異なるノードとする．その時，$st\_h\_1, \ldots, st\_h\_v$ なる v 個の変数を導入し，次の式を追加する．

$$st\_h\_i_v = \cdots = st\_h\_i_v = st\_h \quad (9)$$

また，前節の (2) の式を次のようなv個の式 ($1 \leq u \leq v$) に置き換える．

$$path\_h\_i_u j\_1 + \cdots + path\_h\_i_u j\_k = st\_h\_i_u \quad (10)$$

ここで，$path\_h\_i_u j\_k$ はストリーム h の送信元のうちノード i_uからノード jへ送信する際に，k 番目の経路候補を使用する場合に真になるような変数である．このように変更することで，$st\_h\_i_1, \ldots, st\_h\_i_v$ の全てを送信する経路の決定を行うことが出来る．しかし，ここで，$st\_h\_i_1, \ldots, st\_h\_i_v$ の送信は同時には起こらないためリンクの使用帯域の計算に無駄が含まれる．そこで，マルチキャストの際と同様の変数の置き換えを行い，これを解決する．送信元が動的に変更される可能性のあるストリーム st_h とリンク L_x の組に対して，新たに変数 $select\_h\_j\_x$ 用いて，

$$select\_h\_j\_x \geq band\_h\_i \cdot path\_h\_ij\_q \quad (path\_h\_ij\_q \in path(L\_x)) \quad (11)$$

とする．ここで，$band\_h\_i$ はストリーム h に含まれるソース u の帯域を表す．さらに，(3) の式を変更し，送信が動的に変わるストリームは，$select\_h\_j\_x$ に置き換える．ここで，送信元が動的に変わるストリームの集合を DYNAMIC とすると，

$$\sum_{st\_h \in \mathrm{DYNAMIC}} (select\_h\_j\_x) + \sum_{\substack{path\_h\_ij\_q \in path(L\_x) \\ st\_h \notin \mathrm{DYNAMIC}}} (band\_h \cdot path\_h\_ij\_q) \leq max(L\_x) \quad (12)$$

このようにすることで，送信元ノードが $i_1, \ldots, i_v$ のいずれに動的に変わった場合でも，他のストリームの転送経路を変えずに，かつ，すべての転送ストリームの使用帯域が各リンクの使用可能帯域を上回らないような経路の決定が行える．

4.3 ストリームの集合に変動がある場合

一般にSMIL 風のシナリオ群でマルチメディア並行分散システムの動作を記述した場合，あるタイミングでは $ST1 = \{st\_f_1, \ldots, st\_f'_m\}$ なるストリームの集合が同時に転送され，別のタイミングでは $ST1$ とは異なるストリームの集合 $ST2 = \{st\_g_1, \ldots, st\_g''_m\}$ が同時に転送される可能性がある．このような場合，上述の (1) の式の ST を $ST1$ や $ST2$ のように変化させて制約式を作り，それぞれのストリームの集合毎に転送経路を求めることが可能である．

ただし，このような形で転送経路を求めた場合，たとえ $ST1$ と $ST2$ の中に共通するストリーム (例えば st_p) があった場合でも，$ST1$ から $ST2$ へストリームの集合が変化した場合にストリーム st_p の転送経路が異なってしまう可能性がある．このような場合経路の切り替え時に映像が途切れてしまうなどの影響が出る可能性があるので，$ST2$ の転送時に st_p の転送経路を $ST1$ と同じようにできることが望ましい．例えば，$ST1$ と $ST2$ の中に共通する各ストリーム (st_p とする) に対して，$ST1$ の転送経路の決定で求めた $path\_p\_ij\_1, \ldots, path\_p\_ij\_k$ の値 $p_1, \ldots, p_k$ が変化しないことを表す制約式，

用意する経路候補数	1 個	2 個	3 個	4 個
最小未使用帯域 (%)	18.3	18.4	21.2	28.3
平均未使用帯域 (%)	28.6	39.7	46.8	53.6
平均計算時間 (秒)	0.01	0.66	2.75	9.77

表 1: 経路候補数と未使用帯域の関係

$$\begin{aligned} path\_p\_ij\_1 &= p_1 \\ &\vdots \\ path\_p\_ij\_k &= p_k \end{aligned} \quad (13)$$

を追加することにより，ストリーム st_p の転送経路を $ST1$ と $ST2$ で共通化することができる．

5 まとめと考察

4 章の経路選択アルゴリズムを実装し，用意する経路候補の数と経路割り当て，経路の割り当て後の未使用帯域の関係を調べた．実験は，乱数を用いて 3 人のユーザを 25 ノードからなるランダムなネット上に配置し，用意する経路候補数を 1〜4 個まで変化させて経路割り当てを行った．用意する経路候補の数が 1 個の場合は全ての経路について最短経路が用いられる．2 個以上の場合は最短経路から順にその個数だけの経路が候補として挙げられ，その内で最も効率よく未使用帯域を多く残すものが選ばれる．実験は10 個の問題を作成して行い，その結果が表 1である．実験には，512MB のメモリ，Pentium III の 600MHz を搭載した PC を使用した．この表には経路割り当てを行った結果，どれだけの未使用帯域が得られたかについて，全リンクの平均未使用帯域及び最小の未使用帯域を示している．経路候補が 1 個の場合は単純に最短経路を用いてマルチメディア情報の配信を行う通常の手法に相当する．経路候補を 2〜4 個に増やすことで 4 章で述べたような QoS を考慮した配送経路の選択がうまく行われ，未使用帯域が増加している．また，経路選択のみならずマルチキャストで出来るだけ同じリンクを利用し，無駄な帯域の予約を減らしていることも未使用帯域を増やしている要因と考えられる．この結果より，本手法を用いることでより効率よく帯域を利用できると考えられる．

参考文献

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

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

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

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

[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).