SMILへのQoS保証文の導入とその柔軟な実装法

寺島芳樹[†] 安本慶一[††] 東野輝夫[†]
安倍広多[†††] 松浦敏雄[†††] 谷口健一[†]

本論文では，種々の機能拡張に柔軟に対応できるQoS制御機構の実装法を提案し，SMILへの適用例を示す．提案手法では，仕様記述言語E-LOTOSのサブクラス（時間拡張LOTOSと呼ぶ）を中間言語として用いる．QoS制御機能は，システムを動画，音声の再生動作などを記述した主プロセスと，メディアスケーリング，メディア同期など追加したい機能のみを記述した制約プロセスとで構成し，それらを並列に同期実行させるという，制約指向スタイルを利用して実現する．またSMILに動的メディアスケーリング，メディア同期の精度指定の機構を追加したQOS-SMILを定義し，実装方法の適用例として示す．QOS-SMIL記述は対応する時間拡張LOTOS仕様に変換され，我々が開発している時間拡張LOTOSコンパイラを用いて実行される．いくつかの実験結果から，提案手法はQoS制御機能の開発コストに優れ，実行効率の点でも十分な性能を持っていることを確かめた．

Integration of QoS Requirements into SMIL and Its Flexible Implementation

Yoshiki Terashima,[†] Keiichi Yasumoto,[††] Teruo Higashino,[†]
Kota Abe,[†††] Toshio Matsuura[†††] and Kenichi Taniguchi[†]

In this paper, we propose a flexible implementation technique for QoS control mechanisms and apply it to SMIL language. In the proposed technique, we use a subclass of E-LOTOS (called real-time LOTOS). We implement QoS control mechanisms using the constraint oriented style where a system is composed of a main process (e.g., video/audio playback) and several constraint processes (e.g., media scaling, inter-media synchronization and so on). Using the multi-way synchronization mechanism of real-time LOTOS, those processes run in parallel satisfying the specified constraints. We define QOS-SMIL as an example extension of SMIL where it has dynamic media scaling and explicit inter-media synchronization among objects, and show the applicability of our implementation technique. QOS-SMIL documents are converted to executable programs with our real-time LOTOS compiler. Through some experiments, we have confirmed that the proposed technique has some advantages w.r.t. development cost for QoS control mechanisms and the derived programs have relatively good performance for practical use.

1. はじめに

近年，複数メディアオブジェクト（動画，静止画，音声，テキストなど，以下**オブジェクト**と呼ぶ）を利用者側で効果的に表示することを目的として，それらの再生順序や画面上の表示位置・再生時刻などをあらかじめシナリオとして記述するための言語とその処理系が注目されている[5),11)]．

† 大阪大学大学院基礎工学研究科
Graduate School of Engineering Science, Osaka University
†† 滋賀大学経済学部情報管理学科
Faculty of Economics, Shiga University
††† 大阪市立大学学術情報総合センター
Media Center, Osaka City University

シナリオ記述言語SMIL 1.0[11)]は，個々のオブジェクトの表示位置，再生範囲，再生開始・終了時刻の指定や，それらの並行・連続再生などの制御機構，さらにはシステム（表示系/ネットワーク）の性能や機能の制限に応じた代替メディアの指定などの構文を持つ．

一般に，マルチメディアアプリケーションにおいては，システムのリソース不足やパケットの消失・遅延などに対応するため，あらかじめ与えられた各ユーザのオブジェクトに対する許容品質や優先度などのQoS要求をもとに，メディアスケーリング[3)]により必要なリソース量を調整したり，動画と音声のずれを一定範囲に収めるメディア同期[2),8)]を行うなどのQoS制御を必要とする場合が多い[9)]．

しかしSMIL 1.0では，そのようなQoS制御はす

べて実装依存と見なし，シナリオ記述にQoSを陽に指定することはできないため，システム資源が変動する環境での一般の分散マルチメディアアプリケーションの実現に適用するには問題があった．

これらの問題に対処するには，SMILにQoSを要求するための何らかの言語機能を追加し，SMILの実行系において，それらを実現するためのQoS制御機構を実装する必要がある．新たなQoS制御機構の追加や今後の標準化の動向に対応できるように，SMIL PlayerのQoS制御機構の実装はその追加・変更に対して柔軟であることが望ましい．

本論文では，SMILへの将来の種々のQoS制御機構の拡張にも容易に対応可能とする実装法を提案し，QoS制御機構のうち代表的であると考えられる，(1) オブジェクトの再生途中でも代替メディアへの切替えを可能にする動的選択機構，(2) 並行再生オブジェクト間で再生シーンのずれを指定した時間範囲内に収めるメディア同期機構を追加したQOS-SMILを定義する．

提案手法では，SMILの時間制御や並行同期制御を扱うのに優れたE-LOTOS[7]のサブクラス（時間拡張LOTOS[13]と呼ぶ）を中間言語として用いる．QOS-SMIL記述は対応する時間拡張LOTOS仕様に変換され，我々が開発している時間拡張LOTOSコンパイラ[13]を用いて実行される．

QoS制御機構は，システムを動画，音声の再生動作などを記述した主プロセスと，輻輳検出，メディアスケーリング，メディア間同期など追加したい機能のみを記述した制約プロセスとで構成し，それらを時間拡張LOTOSのマルチランデブ機構により並列に同期実行させるという，制約指向スタイル[10]を利用して実現する．このため，各QoS制御機構の変更・追加などが容易である．

提案手法に基づき，与えられたQOS-SMIL記述を時間拡張LOTOS仕様に変換する処理系を作成した．生成された時間拡張LOTOS仕様は時間拡張LOTOSコンパイラにより，実時間スレッドプログラムとして実現される．プログラム中，各並行再生オブジェクトは我々が開発した実時間スレッド機構[1]の各1スレッドに割り当てられ，それぞれの時間優先度に応じてEDF（Earliest Deadline First）によりスケジューリングされる．

提案方式によりいくつかのQOS-SMIL記述から実行コードを生成し実行した結果，システム資源の動的変動を吸収し各ユーザの嗜好に合った形でオブジェクトの再生が行えることなどを確認した．また，中間言語を用いずにC言語とスレッド機構を用いて直接実行系を設計・開発した場合と比較した結果，制約指向を採用する提案手法では，機能拡張にともなう変更にかかる手間が大幅に少なく，また実行効率においてもそれほど遜色ないことを確かめた．

2. SMILとそのQoS拡張

2.1 SMIL 1.0の概要と対象とするサブクラス

SMILでは，オブジェクトのレイアウトの情報と，オブジェクトの挙動とが別々にXMLに基づき，エレメント（`<element>...</element>`），およびエレメント属性（`<element attr_name=attr_value>...</element>`）を用いて記述される．

オブジェクト間の制御機構としてpar，seqエレメントがあり，複数のオブジェクトをそれぞれ並行，逐次に再生することができる（入れ子構造を許す）．また，代替のオブジェクトを複数指定しておき，システムで利用可能なリソース（帯域幅や画面サイズなど）に応じて静的に1つを決定し再生するためのswitchエレメントがある．

本論文では，SMIL 1.0のサブクラスとして次のエレメントと属性を対象とする．

エレメント	エレメント属性
region	id, left, top, width, heigh, title
video, audio, img,text	id, begin, end, dur, clip-begin, clip-end, src, region, fill, repeat
par	id, begin, end, dur, endsync, repeat
seq	id, begin, end, dur, repeat

ここで,オブジェクトの場所を示すsrcはファイル名あるいはハイパーリンク(http://somehost.somedomain/video1.mpgなど)で指定する．その他のroot-layout，fit，ref，altといったレイアウトに関する属性などについては，簡単のため本論文では扱わない（SMIL 1.0の詳細は文献11)，12)参照)．

2.2 QOS-SMIL

上記のSMIL 1.0のサブクラスに対し，(1) 複数オブジェクトの動的切替えを指定する*dswitch*エレメント，(2) 複数オブジェクト間の優先度指定のための属性*pri*，(3) オブジェクト間の同期精度指定のための属性*maxskew*を拡張した言語**QOS-SMIL**を定義する．

(1) 複数オブジェクトの動的切替えの指定 *dswitch*では，システムの負荷に応じて動的に代替メディアを選択・再生する，切替え時に内容の連続性が維持される，という点で従来のswitchと異なる．

システム資源の変動を吸収できるように，(i) 時間解像度の異なる同一オブジェクト，(ii) 異なる品質でエンコードされた同一内容オブジェクト，(iii) 異なる

表 1 *dswitch* エレメントの使用例
Table 1 Example of dswitch element.

```
<dswitch>
  <video src="video.mpg" />
  <video src="video.mpg" tempcrop=50 />
  <video src="video_half.mpg" />
  <text  src="text.txt" />
</dswitch>
```

(*tempcrop*=*n* は，オブジェクトの時間解像度（動画の場合フレームレート）を *n*%間引くことを指定する属性で，QOS-SMIL で新たに拡張した)

表 2 *dswitch* エレメントの使用例（URL 指定）
Table 2 Example of dswitch element with URLs.

```
<dswitch>
  <video src="URL1/video.mpg" />
  <video src="URL2/video.mpg" />
  <video src="URL3/video.mpg" tempcrop=50 />
</dswitch>
```

URL で指定された同一内容オブジェクト（品質は異なってもよい）の間の動的な切替えが指定できる．

dswitch を用いた記述例を**表 1** に示す．代替メディアは必要なリソースの量の降順（品質が高い順）に記述する．最初の行は動画 video.mpg をフルモーションで再生することを指定し，それに続く行でリソース不足時の代替メディアとして，video.mpg の時間解像度（フレームレート）を 50%間引いたもの，画像サイズを半分にした動画 video_half.mpg，テキスト文書 text.txt を順に指定している．

また**表 2** の URL を用いた記述例では，最初に URL1 で示されるサーバから動画 video.mpg を受信，再生することを指定している．ネットワークの輻輳などにより URL1 からの受信，再生が不可能な場合には URL2 の示すサーバから，URL2 からも不可能な場合，より下位の行のオブジェクト（URL3 から半分のフレームレートで受信）に動的に切り替えられる．

(2) 複数オブジェクト間の優先度指定 システムの負荷が高いときに優先度の低いオブジェクトから先に品質を下げるようにするため，par エレメント内のオブジェクトに優先度を設定する属性 *pri* を追加した．*pri* には 1 以上の任意の整数を与えることができる．ただし，1 が最も優先度が高く，数が大きいほど優先度は低いものとする（値の差は意味を持たない）．優先度指定は，通常 par エレメント内の各オブジェクトについて，*dswitch* による代替オブジェクトを指定することによって行う．*dswitch* 内における最下行（最も低品質）のオブジェクトの優先度はつねに 1 となる（さらなる代替メディアがない）ため，優先度の指定は省略してもよい．

優先度指定の記述例を**表 3** に示す．この例では最初にオブジェクト A (*pri*=100)，B (*pri*=70) がフルモーションで並行再生される．負荷が高まるにつれ，B より優先度の低い A が，代替メディアとして指定されている A *tempcrop*=33 (*pri*=90)，A *tempcrop*=66 (*pri*=80)，A *tempcrop*=86 での再生に動的に切り替えられる．A *tempcrop*=86 の優先度は B よりも高いので，さらに負荷が高まったときには B が B *tempcrop*=33 (*pri*=60)，B *tempcrop*=66 (*pri*=50)，B *tempcrop*=86 と切り替えられる．

表 3 属性 *pri* の使用例
Table 3 Example of pri attribute.

```
<par dur="60s">
  <dswitch>
    <video src="A"                pri="100" />
    <video src="A" tempcrop="33" pri="90"  />
    <video src="A" tempcrop="66" pri="80"  />
    <video src="A" tempcrop="86" />
  </dswitch>
  <dswitch>
    <video src="B"                pri="70" />
    <video src="B" tempcrop="33" pri="60" />
    <video src="B" tempcrop="66" pri="50" />
    <video src="B" tempcrop="86" />
  </dswitch>
</par>
```

表 4 属性 *maxskew* の使用例
Table 4 Example of maxskew attribute.

```
<par maxskew="0.2s">
  <video src="video.mpg" />
  <audio src="audio.wav" />
</par>
```

負荷が低くなった場合，品質を下げる場合と逆の順，すなわちより高い優先度を持つオブジェクトへ切替え可能なオブジェクトから，先に切替えを行う．

(3) メディア同期の精度指定 SMIL 1.0 では，オブジェクト間の同期の正確さは実装依存である．ある実験によると，ビデオと対応する音声とのずれは 80 ms 以内であれば違和感を感じないことが知られており[2]，一般には，ユーザやコンテンツにより許容値には差があると考えられる．

最新の SMIL Boston 仕様の草稿では，オブジェクト間のリップ同期を指定するための属性 *syncBehavior* が追加されているが，同期の精度については陽に指定することはできない．

そこで QOS-SMIL では，並行再生オブジェクト間の許容可能な最大のずれを指定する属性 *maxskew* を追加した．記述例を**表 4** に示す．この例では video.mpg と audio.wav の再生中に，再生のずれが 0.2 秒を超えることがないよう調整される．

3. QOS-SMIL の実行系の構築

本論文では，中間言語として仕様記述言語 E-LOTOS[7] のサブクラスである**時間拡張 LOTOS** を利用し，そのコンパイラ[13] を用いて QOS-SMIL 記述を実現する．

3.1 時間拡張 LOTOS の概要

LOTOS[6] は ISO により開発された通信プロトコル向けの仕様記述言語であり，並列性を含むシステムの挙動を簡潔に表すための強力な構文を有する．LOTOS では，システムをいくつかの並列プロセスとして記述し，各プロセスの動作（**動作式**）は，**イベント**と呼ばれるゲートを介したプロセス外部との相互作用（値の入出力）の実行系列として定義される．

イベント間の実行順序を指定するため，連接（$a; B$），選択（$B1[]B2$），同期並列（$B1|[gl]|B2$），非同期並列（$B1|||B2$），割込み（$B1[> B2$），逐次（$B1 >> B2$）などのオペレータが任意の動作式間に指定される（ここで，a はイベント，B は動作式）．$[boolexp] -> B$ と記述することで，論理式 $boolexp$ が成立するときのみ動作式 B を実行することも指定できる．

同期並列オペレータを用いて，複数のプロセスが指定されたイベントを同期実行しデータ交換を行う（**マルチランデブ**と呼ばれる）よう指定できる．

時間拡張 LOTOS は，イベント a の実行時刻（実行可能になってから実際に実行されるまでの経過時間）を変数 t に取得する $a@?t$，イベントの実行時刻に制約を指定する $a@?t[p(t)]$（$p[t]$ は $C_1 \leq t \leq C_2$ のような連続した時間範囲に制限），d 単位時間待つ **wait**(d)，動作式 B を繰り返し実行する **loop** B **endloop**，書換可能変数群を宣言する **var** V **in** B **endvar** などの E-LOTOS[7] の構文が従来の LOTOS に追加された言語である[13]．

3.2 QOS-SMIL 実現の基本方針

本論文では，QOS-SMIL 記述の動作を，オブジェクトの再生に関する基本動作（イベント）とその実行順序のみを指定した主プロセスと，イベント間の実行時間間隔や実行順序に対する制約を指定した制約プロセスとして互いに独立に時間拡張 LOTOS を用いて設計し，それらの間で関連するイベントを同期実行させて実現するという方法を用いる．これは制約指向記述スタイルと呼ばれ，システムの設計変更・保守に優れていることが知られている[10]．

以下，制約指向を使ったシステムの構成法を，オブジェクト Obj_i の再生プロセス $Play(Obj_i)$ を例に説明する（ここでは動画の例のみ与えるが音声など他のメディアでも同様）．

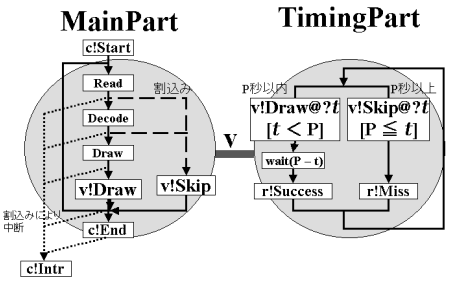


図 1 制約指向スタイルによるオブジェクト再生プロセスの記述
Fig. 1 Object playback process in constraint oriented style.

我々は，時間拡張 LOTOS で動画や音声を単位データごとに処理するためのプリミティブを作成している[13]．各プリミティブをイベントに対応づけることにより，動画再生の主プロセスは，**図 1** MainPart に示すように，(1) フレームデータの読み込み（?frame := Read(fp)），(2) そのデコード（?pic := Decode(frame)），(3) 描画（vout!Draw(pic); v!Draw），あるいはデータ読み込み後，輻輳時には，(4) デコード，描画の処理を省く（v!Skip）ことの繰返しで表せる．ここで (4) の v!Skip は (2)，(3) の処理に対する割込み動作として表せる．

一方，これらの時間制約のプロセスとしては，図 1 TimingPart に示すように，v!Draw というイベントを指定した周期 P（1/動画のフレームレート）で実行する（v!Draw@?t[t<P]; wait(P−t)）か，あるいは v!Draw が P 秒以内にできない場合にはイベント v!Skip を実行する（v!Skip@?t[P≤t]）というように表せる．

MainPart と TimingPart でゲート v に関して同期実行することで，全体として MainPart の 1 フレームに対する読み込み，デコード，表示の一連のイベント系列が周期 P で実行（その動画のフレームレートで再生）される．なんらかの要因で P 時間以内に実行できなければ，MainPart のデコード，描画処理は v!Skip により割り込まれ，その周期の描画は省かれる（図 1 中の c!Start，c!End，c!Intr は再生開始，終了，中断の制御に，r!Success，r!Miss は周期処理が時間どおりに行えたかどうかの通知に使用）．

このような構成手法を用いることで，主プロセス MainPart をほとんど変更する必要なく，時間制約プロセス TimingPart のような制御機構を付け加えていくという形で，望むシステムを設計することができる．

3.3 SMIL の基本機能の実現

par，seq エレメントは LOTOS の並列（|||），逐次オペレータ（>>），および前節で述べた再生プロセス

表 5 モニター機構
Table 5 Example of a monitoring process.

```
Monitor[r1,s] :=
  loop
    var suc, mis: intarray in
      loop
         r1?id:int!Success; ?suc:=Add(suc,id,1)
      [] r1?id:int!Miss; ?mis:=Add(mis,id,1)
      endloop
  [>
    wait(T);
    ?rate := MaxMissRate(suc,mis);
    ( [0.3 < rate] ->       s!Congestion
   [] [0.1 < rate <=0.3] -> s!Tolerable
   [] [rate <= 0.1] ->      s!Stable
    endvar
  endloop
```

($Add(x,i,val)$ は配列 x の i 番目の要素の値に val を加算した後の配列を返す関数，$MaxMissRate(suc,mis)$ は各オブジェクトごとに周期処理ミス率 $mis/(suc+mis)$ を計算し，そのうちの最大値を返す関数)

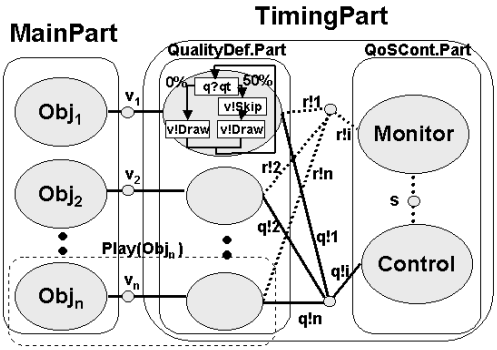


図 2 動的切替えの実現
Fig. 2 Dynamic switching mechanism.

表 6 品質定義部
Table 6 Example of QualityDef.Part for controlling timing resolution in video playbacks.

```
QualityDef.Part[v,c,r,q](P:time, i:int): exit :=
  loop
    q!i?qt:int;
  []
    ([qt==1]->(v!Draw@?t[t<P];
                r!i!Success; wait(P-t)
              [> wait(P); r!id!Miss )
  [] [qt==2]->(v!Skip@?t1[t1<2*P];
                v!Draw@?t2[t1+t2<=2*P];
                r!i!Success; wait(2*P-t1-t2)
              [> wait(2*P); r!id!Miss )
  endloop
```

$Play(Obj_i)$ の呼び出しを用いて実現する．再生開始・終了時刻の制御は，c!Start，c!Intr を適切な時刻に実行する制約プロセスをゲート c に関して再生プロセスと同期させることで実現できる．

3.4 QOS-SMIL における拡張機能の実現

3.4.1 代替オブジェクトへの動的切替え

本機能の実現には，(1) 輻輳状態の検出法，(2) 代替メディアへのシームレスな切替え法が問題となる．

(1) 輻輳状態の検出　本論文では，ユーザシステムの資源不足による輻輳を各オブジェクトの再生状況（周期処理が時間制約を満たしているかどうか）を調べることで検出することとした．基本的には，全再生プロセスの再生状況を監視し，状況がよくないオブジェクトが存在する場合には，輻輳が生じていると見なす．

表 5 に示したプロセス Monitor の記述例では，各オブジェクトの再生状況を再生プロセスのイベント r?i:int!Success，r?i:int!Miss により取得し（i はどのオブジェクトかを識別），それぞれの回数を記録する．周期 T ごとに，取得した値を基に周期処理ミス率の最大値を求め，その値が 30%より大きい場合に輻輳状態（s!Congestion），10〜30%の場合許容範囲（s!Tolerable），10%以下では安定状態（s!Stable）であると判断するよう記述した．

URL で指定されたオブジェクトの再生においては，パケットのサイズ，受信時刻などから平均転送レートを計算し，オブジェクトの再生に必要な帯域と比較することによってネットワークの輻輳を検出する．

(2) オブジェクトの切替え　輻輳を検出した場合，QOS-SMIL の定義より，現在再生されている中で最も優先度の低いオブジェクトを $dswitch$ 内の直下の代替メディアに切り替える．なお，輻輳検出プロセスが一定の回数連続して s!Stable と判断したときは，輻輳が一時的であったと見なし，上位品質へのメディアスケーリングを実施することとした．

どのオブジェクトの品質を上げる（下げる）かを判断し制御するために，**図 2** に示すようなプロセス Control を追加した．

メディアスケーリングに関して，同一オブジェクトの時間解像度を下げる場合は，TimingPart の制約部分を，時間解像度に応じた複数の制約を用意しておくことで実現する．図 2 および **表 6** に示した Quality-Def.Part では，通常の再生動作と，$tempcrop$=50 の 2 つの制約を持つ例を示している．どちらの制約を選択するかは外部プロセス Control がゲート q を通して設定する．

異なる画像サイズや色数でエンコードされたファイルの切替えでは，現在動作しているオブジェクトの再生プロセスを中断し，新たなオブジェクトの再生プロセスを起動，中断時点に相当するフレーム分をスキップ後，再生する．ある程度のバッファを用意することで上記操作にかかるオーバヘッドを吸収し，シームレスな切替えを実現した．

全体として，図 2 のように，(1) オブジェクトの再生プロセス $Play(Obj_i)$ からなる主動作部（MainPart），(2) オブジェクトの品質を定めるための品質定義部（QualityDef.Part），(3) 輻輳検出のための Monitor，切替えの指示を出す Control からなる QoS 制御部（QosCont.Part）のプロセスにより動的切替えを実現する．ここでは，ゲート r，s，q に関するマルチランデブによりそれぞれ再生状況，輻輳状態，品質の変更の通知が行われる．

一方，異なる URL で指定されたオブジェクト間の動的サーバ選択について，本論文では簡単のために，輻輳時には単純に *dswitch* 内の次のサーバを選択するとしたが，文献 4) ではサーバの負荷の状態に基づいての選択方法を提案しており，こういった他の制御手法についても，それに関連するプロセス（ここでは Control）のみを変更すればよく，容易に採用することができる．

シームレスな切替えには，サーバ側では途中フレームからの配信，メディアデータのスケーリングなどといった機構が，クライアント側ではサーバへのデータ送信・停止の要求などを行う機構が必要になる．ここでは時間拡張 LOTOS を用いてサーバプログラムを実装したが，これらの機能追加は比較的小規模であり，既存の httpd サーバに組み込むことも容易に実現可能と思われる．

3.4.2 精度を指定したメディア同期機構の実現

各オブジェクト Obj_i における単位データ処理の周期を P_i，属性 $maxskew$ により指定される最大のずれを $skew$ とする．提案方式では，メディア同期すべきオブジェクト群の現時間再生位置（開始時からの時間）を記憶し，あるオブジェクトで単位データの処理を実行すると，最も遅れているオブジェクトとの時間差が $skew$ を超えてしまう場合には，その処理を待たせることにより実現する．

表 7 に示すような制約プロセスを，3 つの並行再生プロセスとの間に，ゲート v1，v2，v3 に関して同期指定することで，再生プロセスを変更することなくメディア同期が達成される（v!any が再生プロセス内の v!Draw あるいは v!Skip と同期）．表 7 の例では，Obj_1 の単位データの処理（v1!any）は，ずれが $skew$ 以内であるという制約（$t1$+P1-Min($t2$,$t3$) $\leq skew$）が満たされるときのみ実行される．

3.5 時間拡張 LOTOS コンパイラの概要

我々は，時間拡張 LOTOS 仕様を通常の UNIX 上でソフトリアルタイム方式で実現するためのコンパイラを作成してきた[13]．時間拡張 LOTOS 仕様は，各

表 7 メディア同期のための制約
Table 7 Example of TimingPart for Inter-media synchronization.

```
Mediasynch[v1,v2,v3](P1, P2, P3, skew: time):=
  loop
    var t1:=0, t2:=0, t3:=0: time in
    [t1+P1-Min(t2,t3) <= skew]->
       v1!any; ?t1:=t1+P1
 [] [t2+P2-Min(t1,t3) <= skew]->
       v2!any; ?t2:=t2+P2
 [] [t3+P3-Min(t1,t2) <= skew]->
       v3!any; ?t3:=t3+P3
    endvar
  endloop
```

プロセスの生成時刻があらかじめ予測できない，プロセスの動作が周期的であるとは限らないなどの特徴を持つため，各処理のデッドラインの動的設定・変更に適した EDF（Earliest Deadline First）スケジューリングを組み込んだ実時間スレッド機構 RT-PTL を開発した[1]．

本コンパイラは，与えられた時間拡張 LOTOS 仕様を，イベントの逐次・選択・繰返しからなる並列動作を含まない動作式（**単位動作式**）の集合に分割し，各単位動作式を，RT-PTL の 1 つのスレッドに割り当て，スレッド間の共有変数領域を用いて，マルチランデブなどの並列プロセス間におけるインタラクションを実現する．

各実時間スレッドはイベントの時間制約から実行開始時刻，デッドラインを設定し，EDF により全体として早いデッドラインを設定したスレッドが先にスケジュールされるよう制御する（詳細は文献 13) を参照）．

ただし，ソフトリアルタイム方式のため，動画再生などの周期処理においては，システムの性能以上の処理要求があると，各フレームの処理が想定されているデッドラインを超え，フレームの表示時刻がずれることもある．

4. 評価実験

3 章の方法に基づき，QOS-SMIL 記述を対応する時間拡張 LOTOS 仕様として設計，時間拡張 LOTOS への変換系を作成し，変換系および時間拡張 LOTOS コンパイラ[13] を用いて，いくつかの QOS-SMIL 記述を実装した．

前述したように，ソフトリアルタイム方式では，システムの性能以上の処理要求に対し，動画が間延びして表示されるなど，オブジェクトの再生品質が指定した品質と異なってしまう．そこで，3 章で述べた制約指向スタイルでの QoS 制御機構により，過負荷時にメディアの品質をうまく制御できるかなど，実行

表 8 最高フレームレート
Table 8 Maximum frame rates.

	(a) QOS-SMIL（制約指向）			(c) 時間拡張 LOTOS（制約指向なし）			(b) C 言語+スレッド機構		
動画数	$n=1$	$n=2$	$n=3$	$n=1$	$n=2$	$n=3$	$n=1$	$n=2$	$n=3$
352 × 240	26.16	13.28	8.81	27.59	13.46	8.87	29.30	14.56	9.04
176 × 120	114.92	57.06	36.76	126.11	61.78	40.20	143.09	70.36	45.32

系の性能を調べた（RedHat Linux 5.2，Pentium III 500 MHz，128 MByte RAM）．

4.1 基本性能

提案手法では，機能拡張時の開発コストが少ないという特徴を持つが，実装時，制約指向スタイルに起因する並列プロセス間の同期のオーバヘッドやコンパイラの生成コードの効率が問題となる．

352×240，176×120 ピクセルのモーション JPEG 動画に対し，(a) QOS-SMIL 記述から設計・変換された時間拡張 LOTOS 仕様を経て生成したプログラム，(b) C 言語とスレッド機構で作成したプログラム，(c) 動画再生プロセスを制約指向を用いず 1 つのプロセスとして記述した時間拡張 LOTOS 仕様から生成したプログラムを実装，実行効率の比較を行った．結果を**表 8** に示す（すべて同じ実時間スレッド機構，プリミティブを使用）．表 8 (a) の値から，提案手法に基づいた実装において，176×120 の動画 3 つを 30 fps で並行再生する程度には十分な性能を持っているといえる．

また (a)，(b) の値から，(a) では 20%程度のプロセッサパワーが (b) より多く消費されており，(a)，(c) の値の差から，プロセス間の同期制御にそのうち約半分のプロセッサパワーが消費されていることが分かる．なお，ここでは動画の 1 フレームの表示ごとに 4 回同期している．

一方，時間拡張 LOTOS を中間言語に用いた場合の設計のしやすさの目安として，時間拡張 LOTOS での実現と C 言語での実現を比較するため，(a) の QOS-SMIL 記述から変換された時間拡張 LOTOS 記述と，(b) の C 言語とスレッド機構とのソースコード量を調べたところ，絶対的な記述量はアクション（プロセス，関数，変数の宣言部を除く，代入文，選択文，繰返し文，プロセス/関数呼び出し）の数で見ると，(a) では (b) に比べて半分程度であった．

また，一定時間ごとに同期をとりながら再生する機構を実装した際の，システムの追加，変更にともなうコード量は，(b) では全体の 5 割程度に増加するのに対して，(a) では 2 割程度の増加にとどまった．なお，このような新たな機能を追加することにより増加するプロセス間の同期の頻度は，基本的に制御する再生プロセスの数に比例した程度と考えられる．

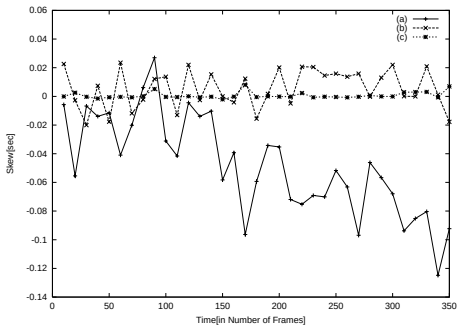


図 3 メディア同期における 2 動画間のフレームのずれの推移
Fig. 3 Skew deviation in inter-media synchronization.

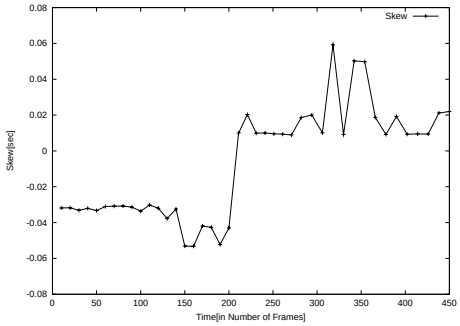
4.2 メディア同期とオーバヘッド

ソフトリアルタイム方式においては，複数のリアルタイムメディアの並行再生において，過負荷時に再生シーン間にずれが発生する恐れがある．

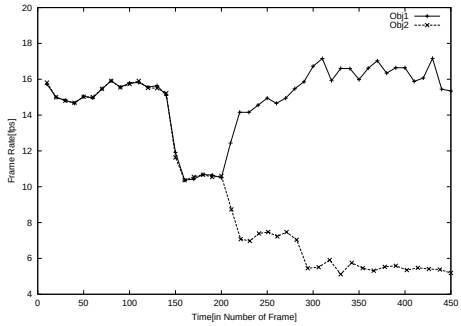
提案手法によるメディア同期機構の性能を調べるために，176×120 ピクセル，30 fps でエンコードされた 2 つの動画の並行再生に対し，同期の精度を，(a) 何も指定しない場合と，(b) $maxskew$="0.08s" と指定した場合とで動画の各フレームのずれ（ある n 番目のフレームを表示した時刻の差）を比較した．なお，実験では，動画再生においてジッタがある程度発生するよう，適度な負荷をシステムに与えている．結果を**図 3** に示す．

精度指定のない (a) ではずれが変動，開いてしまうのに対し，(b) では指定した 0.08 秒以下におさえられていることが分かる．なお，負荷が低いときには，メディア同期制御を行わなくてもほとんどずれは生じなかった（図 3 (c)）．

この例ではメディア間のずれはおさえられるものの，各々のメディアに関しては間延びが発生する恐れがある．このため，メディアスケーリングを併用し，2 つの動画 Obj_1 と Obj_2（15 fps でエンコード）をメディア同期させる実験を行った．ここで Obj_2 は $dswitch$ により 7.5 fps，5 fps，2.5 fps の順に時間解像度を減らした代替メディアが指定されている．実験では 2 つ



(a) フレームのずれの推移
Skew deviation.



(b) フレームレートの推移
Deviation of frame rates.

図 4 メディアスケーリングとメディア同期を併用したときの性能
Fig. 4 Performance of media-scaling and inter-media synchronization.

のオブジェクトの再生開始 10 秒後からシステムに継続して負荷を与えた．実験結果を**図 4** に示す．

図 4 (a) は 2 つの動画間の再生のずれ，(b) は各動画で達成されているフレームレートの推移をそれぞれ示している．図 4 (b) では，再生開始後 10 秒の時点では，負荷により，Obj_1，Obj_2 のどちらも一時的に 10 fps 程度に落ちており，その後，Obj_2 が 7.5 fps, 5 fps と落とされるに従い，Obj_1 が当初の 15 fps に回復しているのが分かる．図 4 (a) では，このような過負荷時のメディアスケーリング処理を行いつつ，2 つの動画間の再生のずれが指定した範囲 0.08 s 以内に制御できていることが分かる．

4.3 優先度に基づいた QoS 制御

優先度によるメディアスケーリング制御の性能を調べるため，352×240 ピクセル，15 fps でエンコードされた動画 A，B を用意し，表 3 の QOS-SMIL 記述を実行し，過負荷時の A，B のフレームレートの時間的変化を計測した．結果を**図 5** に示す．なお，ここでは A，B の再生開始後 10〜50 秒にあるアプリケーションを実行，20〜40 秒にさらに別のアプリケーションを実行することで負荷を与えている．

初めは A，B ともに 15 fps で並行再生されるが，負荷が与えられると，優先度の低い A が 10（*tempcrop*=33），5（*tempcrop*=66），2 fps（*tempcrop*=86）と下げられ，優先度の高い B のフレームレートは保たれていることが分かる．さらに負荷が高まると B も同様に 10，5，2 fps と下げられているが，負荷が低くなると A，B ともに元の 15 fps まで徐々に戻っている．

4.4 複数サーバからの動的選択

受信サーバを動的に選択・切替えする場合の動画再生の様子を調べるため，同一内容の 10 fps の動画を提供するサーバ A，B，C と，表 2 の QOS-SMIL 記述（C からは半分のレート 5 fps で受信）を実行するクライアントをネットワーク接続し，利用可能帯域幅変化時の再生フレームレートの変化を計測した（サーバ A，B，C への回線には重なりはなく，通常 10 fps でデータの受信が可能とする）．

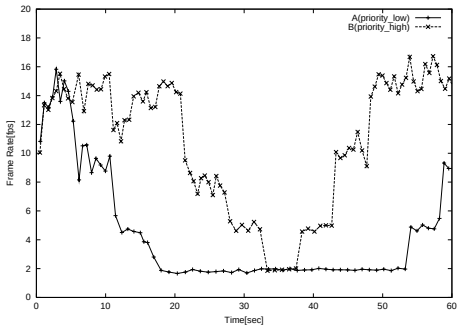


図 5 フレームレートの変化
Fig. 5 Deviation of frame rates.

QOS-SMIL 記述を実行開始後 15 秒以降に，サーバ A からの受信レートが 2 fps まで落ちるよう A への回線に負荷を与え，25 秒後以降は B への回線にも同様に負荷を与えた．フレームレートの推移を**図 6** に示す．なお，サーバ切替え時のオーバヘッドを吸収するためのバッファを (a) 2 秒間，(b) 7 秒間の 2 種類用意し，切替え時の様子を比較した．

図 6 の (a) から，開始 17 秒付近でサーバ A から B への切替えが，27 秒付近でサーバ B から C への切替えが発生していることが，フレームレートの急減により分かる．これは，再生開始 15 秒後にサーバ A からの受信レートが落ちた直後サーバ B への切替え作業が始まるが，(a) では B への切替え完了までにバッファ内のデータが尽きてしまうためである．一方，十

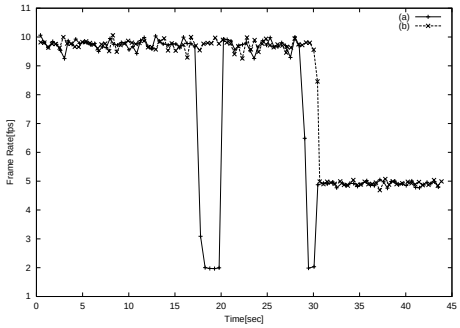


図 6 サーバ切替え時の再生フレームレートの変化
Fig. 6 Deviation of frame rates in dynamic server switching.

分なバッファ容量を持たせた場合，サーバの切替えがシームレスに行えることが分かった（図 6 (b)）．

5. おわりに

本論文では，仕様記述言語である E-LOTOS のサブクラス（時間拡張 LOTOS）の制約指向スタイルを利用した，QoS 制御機構に柔軟に対応可能な実装法を提案した．また，マルチメディア記述言語 SMIL 1.0 に，代替メディアへの動的切替え機構とメディア同期機構を拡張した QOS-SMIL を定義し，提案手法に基づいた QOS-SMIL の実装方式を与えた．

提案した実装法では，リアルタイムメディアに対してフレームコントロールなどの何らかの QoS 制御を課したいとき，新たに追加したい機能のみを独立したプロセスモジュールとして設計開発し，元となる時間拡張 LOTOS 仕様に追加し，関連するイベントに対し同期実行を指定するだけで，その機能を含んだシステムを実現できるという特徴を持つ．また，時間拡張 LOTOS と制約指向記述スタイルを用いずに設計・開発したプログラムと比べても実行効率の面で遜色ないことから，提案する実行方式は，QOS-SMIL などの処理系のプロトタイピングなどに有用であると思われる．

QOS-SMIL では，メディアスケーリングやメディア同期などの実現に焦点をあてたが，SMIL 1.0 の拡張草案である SMIL Boston 仕様では，インタラクティブ性に関する拡張がなされている．ビデオ会議や遠隔講義など，インタラクティブ性が重要となるアプリケーションも数多く存在し，これらに対応していくことは今後の課題の 1 つである．

参考文献

1) Abe, K., Matsuura, T., Yasumoto, K. and Higashino, T.: Design and Implementation of an efficient I/O Method for a Real-time User Level Thread Library, *Proc. IEEE 5th Int. Workshop on Real-Time Computing Systems and Applications* (*RTCSA'98*), pp.117–120 (1998).
2) Blakowski, G. and Steinmetz, R.: A Media Synchronization Survey: Reference Model, Specification, and Case Studies, *IEEE J. Select. Areas Commun.*, Vol.14, No.1, pp.5–35 (1996).
3) Delgrossi, L., Halstrick, C., Hehmann, D., Herrtwich, R.G., Krone, O., Sandvoss, J. and Vogt, C.: Media Scaling for Audiovisual Communication with the Heidelberg Transport System, *Proc. ACM Multimedia'93*, pp.99–104 (1993).
4) Fei, Z., Bhattacharjee, S., Zegura, E. and Ammar, M.: A Novel Server Selection Technique for Improving the Response Time of a Replicated Service, *Proc. INFOCOM'98* (1998).
5) Fujikawa, K., Shimamura, H., Teranishi, Y., Shimojo, S., Matsuura, T. and Miyahara, H.: Application Level QoS Modeling for A Distributed Multimedia System, *Proc.1995 Pacific Workshop on Distributed Multimedia Systems*, pp.44–51 (1995).
6) ISO: *Information Processing System, Open Systems Interconnection, LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour* (1989). IS 8807.
7) ISO/IEC JTC1/ SC21/ WG7: *Final Committee Draft 15437 on Enhancements to LOTOS* (1998).
8) Qiao, L. and Nahrstedt, K.: Lip Synchronization within an Adaptive VOD System, *Proc. 1997 Intl. Conf. on Multimedia Computing and Networking* (*MMCN'97*) (1997).
9) Steinmetz, R. and Wolf, L.C.: Quality of Service: Where are We?, *Proc. IFIP 5th Intl. Workshop on Quality of Service* (*IWQOS'97*), pp.210–221 (1997).
10) Vissers, C.A., Scollo, G. and Sinderen, M.v.: Architecture and Specification Style in Formal Descriptions of Distributed Systems, *Proc. 8th Int. Symp. on Protocol Specification, Testing, and Verification* (*PSTV-VIII*), pp.189–204 (1988).
11) W3C: *Synchronized Multimedia Integration Language* (*SMIL*) *1.0 Specification*. http://www.w3c.org/TR/REC-smil/ (1998).
12) 廣津登志夫：マルチメディア記述言語 SMIL, *Bit*, Vol.30, No.12, pp.10–18 (1998).

13) 辰本比呂記，安倍広多，安本慶一，東野輝夫，松浦敏雄，山口弘純，谷口健一：時間拡張 LOTOS コンパイラの作成とマルチメディアアプリケーションへの応用，情報処理学会論文誌，Vol.41, No.2, pp.424–434 (2000).

(平成 12 年 5 月 19 日受付)
(平成 12 年 10 月 6 日採録)

寺島 芳樹

平成 11 年大阪大学基礎工学部情報工学科卒業．平成 13 年同大学大学院博士前期課程修了．モバイルアプリケーションの開発等に興味を持つ．

安本 慶一（正会員）

平成 3 年大阪大学基礎工学部情報工学科卒業．平成 7 年同大学大学院博士後期課程退学後，滋賀大学経済学部助手．現在同大学助教授．博士（工学）．平成 9 年モントリオール大学客員研究員．通信プロトコルや分散システムの形式仕様記述・実装法に関する研究に従事．

東野 輝夫（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院博士後期課程修了．同年同大学助手．平成 2，6 年モントリオール大学客員研究員．現在，同大学大学院基礎工学研究科教授，工学博士．分散システム，通信プロトコル等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．

安倍 広多（正会員）

平成 4 年大阪大学基礎工学部情報工学科卒業．平成 6 年同大学大学院博士前期課程修了．同年 NTT 入社．平成 8 年大阪市立大学助手．平成 12 年同講師．博士（工学）．マルチスレッド機構の実装，オペレーティングシステムの設計等に興味を持つ．電子情報通信学会会員．

松浦 敏雄（正会員）

昭和 50 年大阪大学基礎工学部情報工学科卒業．昭和 54 年同大学大学院基礎工学研究科（情報工専攻）博士後期課程退学後，同大学助手．平成 4 年同大学情報処理教育センター助教授，平成 7 年大阪市立大学教授．工学博士．ユーザインタフェース，マルチメディア，情報教育等に興味を持つ．ACM，IEEE，電子情報通信学会等会員．

谷口 健一（正会員）

昭和 40 年大阪大学工学部電子工学科卒業．昭和 45 年同大学大学院博士課程修了．同年同大学助手．現在，同大学大学院基礎工学研究科教授．工学博士．この間，計算理論，ソフトウェアやハードウェアの仕様記述・実現・検証の代数的手法および支援システム，関数型言語の処理系，分散システムや通信プロトコルの設計・検証法等に関する研究に従事．