社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE
SS2002-29(2003-1)

拡張時間オートマトン群による実時間システムの記述および検証

加藤雄一郎† 山口 弘純†† 岡野 浩三†† 谷口 健一††

† 大阪大学 大学院基礎工学研究科 情報数理系専攻 〒560-8531 大阪府豊中市待兼山町 1-3
†† 大阪大学 大学院情報科学研究科 〒560-8531 大阪府豊中市待兼山町 1-3
E-mail: †y-katou@ics.es.osaka-u.ac.jp, ††{h-yamagu,okano,taniguchi}@ist.osaka-u.ac.jp

あらまし この論文では，QoS 制御機能をもった動画像再生システムを例に，拡張時間オートマトン群による実時間システムの記述及び検証について述べる．動画像再生システムの記述に検証のための追加や変更を行い，ジッターやフレーム再生間隔といったいくつかの性質を CTL で記述した．そして，チェッカー Uppaal を用いて，記述した動画像再生システムが安全性と上記の性質を満たすかの検証を行った．また，QoS 制御機能の Uppaal による検証についても触れる．

キーワード 実時間システム，時間オートマトン，検証

Specification and Verification of Real-Time System with Extended Timed Automata

Yuichiro KATO†, Horozumi YAMAGUCHI††, Kozo OKANO††, and Kenichi TANIGUCHI††

† Department of Infomatics and Mathmatical Science, Graduate school of Engineering Science, Osaka University Machikaneyama-cho 1–3, Toyonaka-shi, Osaka, 560–8531 Japan
†† Graduate School of Information Science and Technology, Osaka University Machikaneyama-cho 1–3, Toyonaka-shi, Osaka, 560–8531 Japan
E-mail: †y-katou@ics.es.osaka-u.ac.jp, ††{h-yamagu,okano,taniguchi}@ist.osaka-u.ac.jp

Abstract In this paper, we present specification and verification examples of a real-time system (a video player with QoS controller) modeled in networks of (extended) timed automata, where they communicate via channels and global variables. We described sevearl properties, such as frame rate and jitter in CTL and test timed automata. We have verified that the player satisfies safety and those properties using Uppaal. We also discuss verification of QoS properties using Uppaal.

Key words real-time system, timed automaton, verification

1. はじめに

マルチメディアデータ配信サービスを実現するためには，サーバでのマルチメディアデータの送信とクライアントでのマルチメディアデータの再生が適時行われる必要がある．例えば，ビデオ会議システムにおいては，サーバでデータが生成されてから，クライアントで再生されるまでの遅延時間が大きすぎると同一ストリーム内の同期性 (intra-stream synchronization) が失われることとなる．また，また映像と音声を同時に扱うので，クライアントでデータの再生を行うときには，それらの異なるストリーム間の同期性 (inter-stream synchronization) を保つ必要がある．

このように，マルチメディア配信サービスシステムでは時間に関する制約が重要な役割を果たすこととなり，システムを設計する場合には，マルチメディアシステムのリアルタイム性を考慮しなければならない．しかしながら，リアルタイム性の記述や検証は困難な問題である．

この問題に対して，マルチメディアシステムの仕様を形式的に記述し，その記述上でリアルタイム性を検証する手法が提案されている．文献 [1] では，シングルストリームを対象に，スループットやサーバ/クライアント間の遅延時間が設定されている QoS を満たすか検証することで，intra-stream synchronization の検証を行う手法を紹介している．この手法では，ストリームを時間オートマトンで記述し，検証ツールとしてリアルタイムモデルチェッカである Uppaal [2] を使用している．また文献 [3] では，Lip Synchronisation を対象に，inter-stream synchronization の形式的仕様と検証を行う手法を紹介している．この手法では，提案されているある Lip Synchronisation アルゴリズムを Uppaal の拡張時間オートマトンで記述し，アルゴリズムがパケットの遅延のばらつきであるジッターやオー

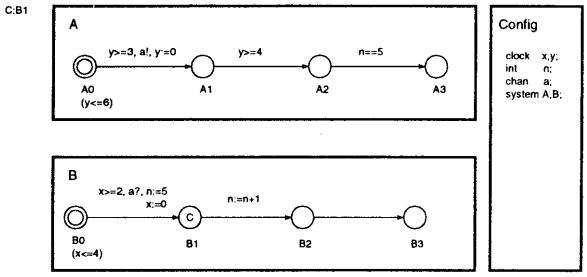


図 1 拡張並列時間オートマトンの記述例

Fig. 1 An example Uppaal model

ディオとビデオの再生時間のずれが設定されている QoS を満たすかを検証している.

このように, マルチメディアシステムなどのリアルタイム性をシステム開発の初期段階で分析することは非常に有益である.

そこで本論文では, Uppaal を用いた, QoS 制御機能をもつ動画像再生システムを例題にして, 実時間システムの拡張時間オートマトン群による仕様記述および検証について提案する. 動画像再生システムの特徴として, 受信端末にシステムリソースの状況を把握しながら自律的に品質調整機能を設けている点があげられる. これは, インターネットに代表されるコンピュータネットワークでは, 要求される品質は各アプリケーションにより異なり, また利用者が知覚する品質はネットワーク性能だけでなく, 端末の性能にも大きく依存すると考えられるためである.

システムの仕様記述に関しては、システムの負荷に応じて QoS 制御を実行する機能もつデコーダ, 表示デバイスといった動画像再生システムの各プロセスごとに Uppaal の拡張時間オートマトンで記述する. そして, 各プロセス間はチャネルを通じて同期し, システム全体は各コプロセスのオートマトン群で構成されている.

システムの検証に関しては, まず, 動画像再生システム全体としての安全性の検証を行った. つぎに, システムの負荷に対処した QoS 制御が実行されているかを様々な項目から検証を行うため, 動画像再生システムの各プロセスを各検証項目に適した形に記述し, 検証を行った. この検証を行うことで, 単位時間あたりに再生されるフレーム数や, 各フレーム間の再生間隔といった, システム利用者の観点から見た QoS 制御が行われているかを確認できる.

以降, 2. では, Uppaal で用いられている拡張並列時間オートマトンの定義を述べる. 3. では, 本研究で対象としている QoS 制御機能を持つ動画再生システムのシステム構成および QoS 制御機能について述べる. 4. では, 時間オートマトンによるシステムの記述に関して説明する. 5. では, 不変性検証によるシステムの安全性およびサービス品質の検証法について述べ, 6. でその実験結果を述べる. Uppaal を用いた QoS 性質の検証法についても議論する.

2. 記述モデル

Uppaal はリアルタイムシステムの仕様記述および自動検証に用いるツールである. Uppaal では, 論理型変数や整数型変数といった一般的なデータ変数を Alur の時間オートマトン [4] に付加した拡張時間オートマトンの並列動作モデルでシステムを記述し, そのシステムの振る舞いをシミュレーション機能 (トレース機能) により解析できる. またシステムの到達可能性や不変性の検証をモデルチェック機能により検証でき, 満たされない場合は, その反例となるトレースを表示することができる.

2.1 拡張時間オートマトンによるシステム記述

図 1 に Uppaal で用いられている拡張時間オートマトンによるシステム記述の例を示す. このモデルは 2 つの拡張時間オートマトン (プロセスと呼ばれる) A および B から構成されている. これらのプロセス群に加えて, このモデルは 2 つのクロック x, y と, 1 つの整数型変数 n と, 1 つの通信チャネル (communication channel) a を用いている. クロックは動作開始時は値 0 であり, 単位時間ごとに 1 増加する.

Uppaal における遷移条件はガード式 (guard) とよばれ, 例えば, 図 1 において, クロック y の値が 3 以上である場合, A_0 から A_1 の遷移が可能である. 同様に, 整数型変数 n の値が 5 である場合, A_2 から A_3 の遷移が可能である. ここで, 時間制約は自然数との比較式であり, データ制約は整数との比較式である.

また, 遷移が起こると, クロック変数およびデータ変数には新たな値を割り当てることができる. 図 1 において, クロック y は A_0 から A_1 の遷移で値 0 にリセットされる. 同様に, 整数型変数 n は B_1 から B_2 の遷移でその値が 1 増加する. なお, データ変数は全プロセスで共有される.

プロセス群は, 通信チャネルを用いて同期通信を行う. 図 1 において, 2 つのプロセスは, 遷移 $A_0 \rightarrow A_1, B_0 \rightarrow B_1$ において, 通信チャネル a を経由した同期が指定されている. プロセスが互いに同期する動作を記述するために, Uppaal では, $a!$と $a?$を用いる (これはそれぞれ通信チャネル a を介した送信・受信動作を表し, 一対一対応である) . また, 2 つのプロセスが同期になった瞬間に同期通信が実行されるような通信チャネルを緊急チャネル (urgent channel) として宣言することができる.

各ノードには, そのノードにとどまる必要のある時間をクロック変数の制約式で表した不変式 (invariant) ラベル, およびそのノードで時間経過が起きないことを表すコミット (committed) を指定できる. 図 1 において, 現在のノードが A_0 であるとき, クロック y の値が 6 以下である限りは, 制御は A_0 でのみ維持される. また, ノード B_1 はコミット (c で表される) であるので, B_1 から B_2 の遷移は, B_0 から B_1 に遷移した直後に起こる.

2.2 検 証

Uppaal のモデルチェッカにより検証できる性質は, 不変性 (invariant) および到達可能性 (reachability) である. それらは次の形式で記述される.

$$\Phi ::= A\Box\beta \mid E\Diamond\beta$$
$$\beta ::= a \mid \beta_1 \text{ and } \beta_2 \mid \beta_1 \text{ or } \beta_2 \mid \beta_1 \text{ implies } \beta_2 \mid \text{not } \beta$$

a は原子式 (atomic) であり, $A_i.l$ または $v_i \sim n$ である. ここで, A_i はプロセスであり, l は A_i のノードを示している. また, v_i はデータ変数またはクロック変数であり, n は自然数, $\sim$ は比較演算子 $\{\leq, \geq, ==, >, <\}$ である. 時相論理式は, $A\Box\beta$ と $E\Diamond\beta$ のみである. $A\Box\beta$ はすべての到達可能な状態は β を満たさなければならないこと (不変性), $E\Diamond\beta$ は少なくとも 1 つ β を満たす状態に到達しなければならないこと (到達可能性) を表している.

3. QoS 制御機能を持つ動画像再生システム

本研究で対象とする QoS 制御機能を持つ動画像再生システムのシステム構成を図 2 に示す.

3.1 基本処理

この節では, 動画像再生の基本処理を行うプロセスの各機能について述べる.

- ビデオソース:エンコードされたビデオストリームを生成する.
- パケットフィルタ:デコーダ制御機能によって決定される, 単位時間あたりに送出すべきフレームの数 (送出 fps とよぶ) に応じてビデオパケットをデコーダに送出する. なお, パケットをバッファリングする機能を持ち合わせている.

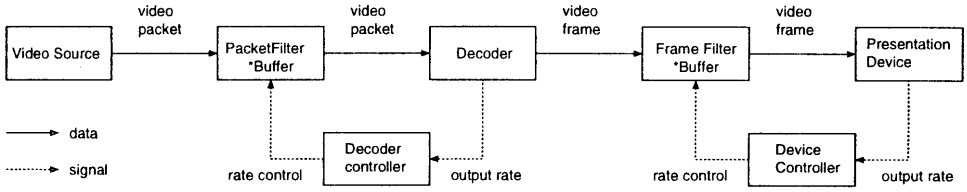


図 2　QoS 制御機能を持つ動画像再生システム
Fig. 2　Vidoe Player with QoS Contoller

- デコーダ:エンコードされたビデオパケットをデコードし, ビデオフレームを生成する.
- フレームフィルタ:デバイス制御機能によって決定される送出 fps に応じてビデオフレームを表示デバイスに送出する. なお, フレームをバッファリングする機能を持ち合わせている.
- 表示デバイス:ビデオフレームを再生する.

3.2 QoS 制御機能

動画像再生システムにおける品質制御について説明を行う. なお, 簡単のため, 本論文ではビデオストリームの 1 つのパケットが 1 つのフレームの符号化データすべてを含むものとする. 主にパケット, パケットフィルタおよびデコーダの品質制御について述べるが, 同様の品質制御をフレーム, フレームフィルタおよび表示デバイスについても行う.

（1） パケット補完: パケットフィルタは次のパケットを送出するまで, 直前に送出したパケットを保存しておき, パケットフィルタからデコーダに送信するパケットが不足する (バッファアンダーフローが生じる) 場合は, この保存しているパケットをデコーダに送出するものとする.

（2） パケット棄却: パケットフィルタへ到着するフレームの数 (到着 fps とよぶ) がパケットフィルタの送信 fps に対して大きい (バッファオーバーフローは生じる) 場合, およびデコーダにおけるパケットの処理の遅れによる再生時間のずれが生じた場合は, パケットフィルタはあるパケットがを送出してから, 次に送出するまでの時間に, 複数のパケットが到着することがある. このとき, 最初に到着したパケットのみをデコーダに送出し, 他のパケットは破棄する. よって, システムの負荷に応じたフレーム数をデコーダで処理できるので, フレームの生成に対する再生時刻の遅れも生じにくくなると考えられる.

（3） 送出 fps の制御: クライアントの負荷はデコーダおよび表示デバイスで生じる負荷のみとする. ここで負荷とは, パケットおよびフレームを処理するのに要する時間とする. それぞれで負荷が生じた場合は, パケットフィルタまたはフレームフィルタの送出 fps を下げ, また負荷が生じていない場合は, 送出 fps を上げる. これにより, クライアントの負荷に応じたサービス品質が提供できると考えられる.

具体的には以下の方針で送出 fps を調整する.

- パケットフィルタでの送出 fps ($= R_{in}$) とデコーダでの送出 fps ($= R_{out}$) の差があるしきい値より大きくなれば, デコーダに負荷が生じていると判断できるため, パケットフィルタでの fps を下げる.
- デコーダの送出 fps がパケットフィルタの送出 fps と同じであり, かつ, パケットフィルタへの到着 fps ($= R_{buff}$) とパケットフィルタの送出 fps の差があるしきい値以上であれば, デコーダに負荷が生じていなく, かつ十分なパケットがパケットフィルタに到着しているのでパケットフィルタの送出 fps を上げる.

これにより, デコーダの負荷が大きくなれば, パケットフィルタの送出 fps を下げることで負荷を減らし, かつ負荷に応じたパケットをデコーダで処理することができる. また, デコーダに負荷が生じていない場合は, 送出 fps を上げることで, より高いサービス品質が提供できる.

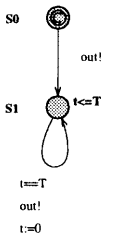


図 3　ビデオソースプロセス
Fig. 3　Video Source Process

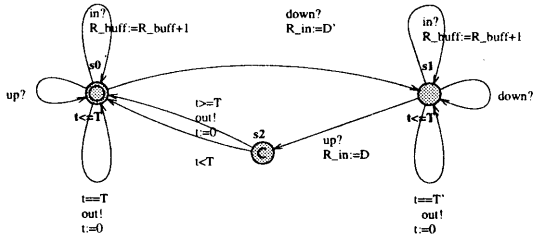


図 4　パケットフィルタ, フレームフィルタプロセス
Fig. 4　Packet Filter, Frame Filter Process

4. 拡張並列時間オートマトンによるシステムの仕様記述

3 章で説明した QoS 制御機能を持つ動画再生システムの各コンポーネントを Uppaal の拡張時間オートマトンで記述する.

図 3 はビデオソースプロセスを記述したものである. これは, 最初のパケットを生成 (out!) した後, ビデオパケットを一定の時間間隔 T で生成する.

図 4 は, パケットフィルタプロセスを記述したものである. これは, ビデオソースで生成されたパケットをバッファリング (in?) するたびに, バッファリングしたパケット量を表す R_{buff} を 1 つずつ増加する. これは, パケットフィルタへの到着 fps を評価するのに用いる. また QoS 制御プロセスから, デコーダへの送出 fps を下げる合図 (down?) を受信した場合は, 送出 fps を下げ ($R_{in} := D'$) , その送出 fps に対応した一定の時間間隔 T' ($T' > T$) でパケットをデコーダに送信する (例えば, 送出 fps が 20fps であれば, 時間間隔を 50ms) . 逆に, QoS 制御プロセスからデコーダへの送出 fps をあげる合図 (up?) を受信した場合は, 送出 fps を上げ ($R_{in} := D$) , その送出 fps に対応した一定の時間間隔 T でパケットをデコーダに送出する. この際, 直前にパケットを送出してから時間が T 以上経過していれば, ただちにパケットを送信するものとする (ノード s_2 はコミットとしている) . これは, 送出 fps に応じたパケットをデコーダに送出するために行う処理である. フレームフィルタについても同じプロセスが利用できる.

図 5 はデコーダプロセスを記述したものである. パケットをパケットフィルタから取りこんだ (in?) 後, パケットを時間 T

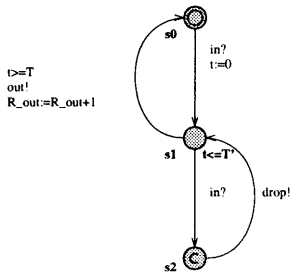


図 5　デコーダ, 表示デバイスプロセス

Fig. 5　Decoder, Presentation Device Process

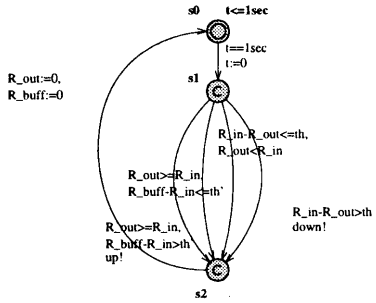


図 6　QoS 制御プロセス

Fig. 6　QoS Control Process

以上 T' 以下の間に処理する (out!) . T,T' はそれぞれ 1 パケットを処理するのにかかる最小および最大の時間である. この時間によりデコーダの負荷の幅を表す. また, パケットを処理するたびに, 処理したパケット量を表す R_{out} を 1 ずつ増加する. これは, デコーダで処理されたパケットの数を評価するのに用いる. また, あるパケットを処理中に次のパケットがパケットフィルタから到着する場合がある. このときは, そのパケットをただちに破棄する (drop!) (ノード s_2 はコミットとしている). 表示デバイスについても同じプロセスが利用できる.

図 6 はデコーダの負荷によりパケットフィルタの送出 fps の制御を実行するデコーダの QoS 制御コンポーネントを記述したものである. これは, 単位時間 (1 秒) ごとにパケットフィルタの送出 fps である R_{in}, パケットフィルタへの到着 fps である R_{buff}, およびデコーダの送出 fps である R_{out} を評価することにより, デコーダの負荷を評価する. QoS 制御方針に関しては, 3 章で述べたとおりである. デコーダの負荷が大きいときは, パケットフィルタの送出 fps を下げる合図 (down!) を出し, 逆に負荷が無く, かつパケットフィルタに十分なパケットが到着していれば, デコーダへの送出 fps を上げる合図 (up!) をそれぞれパケットフィルタに対して出す. R_{buff} および R_{out} に関しては, 単位時間あたりの値を評価するので, 評価の後はそれぞれの値をリセットする. QoS 制御コンポーネントによる負荷判定に関しては, 時間の経過がないものとし, したがってノード s_1 および s_2 はコミットとしている. 表示デバイスの QoS 制御についても同じプロセスが利用できる.

5. システムの安全性及びサービス品質の検証

表 1 に示すパラメータで, 動画像再生システムの安全性およびサービス品質の検証を行った.

5.1 システムの安全性および制御プロセスの正しさの検証

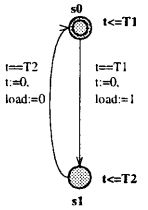
システムの安全性の検証に関しては, システムが到達可能なすべての状態でデッドロック状況に陥ることがないの検証を次式が満たされるかの検証を行うことで可能である.

$$A \,\square\, \mathbf{not}\ (deadlock) \qquad (1)$$

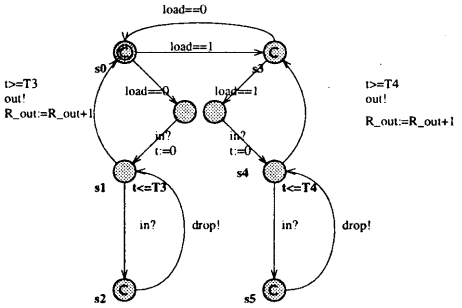
プロセス	項目	値
ビデオソース	送出 fps	20
パケットフィルタ	送出 fps の初期値	20
	送出 fps 制御	10 または 20
デコーダ	処理時間	0ms から 150ms
フレームフィルタ	送出 fps の初期値	20
	送出 fps 制御	10 または 20
表示デバイス	処理時間	0ms から 150ms

表 1　動画像再生システムのパラメータ

Table 1　parameter for video player



(a) 負荷発生用プロセス



(b) 負荷に応じて処理時間が異なるデコーダプロセス

図 7　デコーダの負荷を変更するプロセス群

Fig. 7　Decoder Processes with various load

また, デコーダの負荷に応じて, パケットフィルタの送出 fps の制御が正しく行われているかの検証を行う. 図 7(a) は, デコーダに与える負荷を定期的に変更するプロセスである. このプロセスは, デコーダの負荷を下げた (load:=0) 後, 時間 T1 が経過後にデコーダの負荷を上げ (load:=1) , その後, 時間 T2 が経過後に再びデコーダの負荷を下げることを繰りかえす. 図 7(b) は, 負荷に応じたパケットの処理時間をもつようなデコーダの記述である. デコーダはパケットを処理後 (out!) , 負荷の状態である load の値を確認する. そして, 負荷が小さい状態 (load ==0) であれば, 時間間隔 T3 でパケットの処理を実行する. 逆に負荷が大きい状態 (load==1) であれば, 時間間隔 T4 (T3 < T4) でパケットの処理を実行する. 表示デバイスについても同じプロセスが利用できる.

本来, QoS 制御プロセスの正しさを検証するのであれば, “デコーダの負荷が大きくなると, (一定時間内に, あるるいは条件を緩めて, いつか) パケットフィルタの送出 fps が下がる” ことを検証しなければならない. しかし, 2. 2 で述べた Uppaal の検証式ではこのことは検証できない. 仮に次の式 (2) (デコーダに負荷を与えている状態 LoadClock.s1 にあるならその期間中は送出 fps が低い状態 PacketFilter.s1 になっている) が成り立てば, QoS の観点からは満足である. しかし, 後述するように、式 (2) は, 記述したシステムでは, 成立しない (負荷を検出して送出 fps を下げるように記述しているが, 式 (2) では, 負荷を与えるときにはすでに送出 fps が低い状態に入っているこ

とを要求している).

$$A \ \Box\ (t > 0 \textbf{ and } t < T2 \textbf{ and } LoadClock.s1 \textbf{ imply } PacketFilter.s1) \quad (2)$$

なお, この式は, デコーダの負荷が大きい場合は, パケットフィルタの送出 fps が下がることをすべての状態で満たすことを表している. ここで, LoadClock は図 7(a) のプロセスであり, PacketFilter は図 4 のプロセスである. また, t は LoadClock で用いるクロック変数である.

5.2 サービス品質の検証

以下に示す項目を検証することで, システム利用者の観点から見た QoS 制御が正しく行われているかの検証を行う. ここでは, フィルタの送出 fps に応じたパケットが提供できていることを検証するものとする.

5.2.1 フィルタの送出 fps を上げた場合

システムでは QoS 制御として, パケットフィルタにパケットがバッファリングされていない場合は, 直前に送出したパケットを再度送出するものとしている. これにより, デコーダへの送出 fps に応じてパケットがデコーダに提供できる. しかし, 重複したパケットをデコーダに繰り返し送出されると, デコーダへの送出 fps に応じたサービス品質が提供されているとはいえない. 記述したシステムでは, デコーダへの送出 fps を上げた場合には, パケットフィルタに十分なパケットが到着しており, 重複パケットがデコーダに送出されることはない. 次式が満たされればそのことが保証される.

$$A \ \Box\ (t > 0 \textbf{ and } t < T4 \textbf{ and } LoadClock.s0 \textbf{ and } PacketFilter.s0 \textbf{ imply not } PacketEvaluator.s4) \quad (3)$$

この式は, デコーダの負荷小さく, パケットフィルタの送出 fps が上がっている場合は重複パケットがないことをすべての状態で満たすことを表している. ここで, PacketEvaluator は後述する図 8 の評価プロセスである.

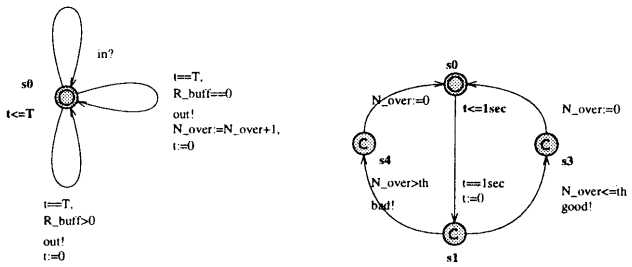
5.2.2 フィルタの送出 fps を下げた場合

デコーダの負荷に応じてパケットフィルタでの送出 fps の制御が行われていなければ, パケットフィルタでパケットの棄却を行うだけでなく, デコーダにおいてもパケットの棄却が行われることがある. このため, パケットフィルタでデコーダへの送出 fps に応じてパケットを棄却しているのにもかかわらず, デコーダでもパケットの棄却が行われ, 送出 fps に応じたサービス品質が提供されるとはいえなくなる. デコーダでパケットの棄却が行われるのは, デコーダの負荷が高い場合である. システムでは, デコーダの負荷に応じて, デコーダへの送出 fps を制御している. そのため, デコーダへの送出 fps が下がった後は, デコーダでのパケットの棄却は行われない. そのことを, 次式が満たされるかで検証する.

$$A \ \Box\ (t > 0 \textbf{ and } t < T3 \textbf{ and } LoadClock.s1 \textbf{ and } PacketFilter.s1 \textbf{ imply not } PacketEvaluator.s4) \quad (4)$$

この式は, デコーダの負荷大きく, パケットフィルタの送出 fps が下がっている場合は, 棄却パケットがないことをすべての状態で満たすことを表している. ここで, PacketEvaluator は後述する図 9 の評価プロセスである.

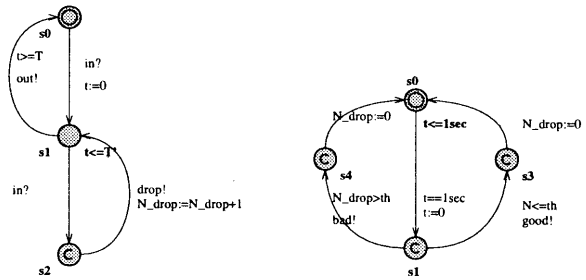
また, フレームフィルタでの送出 fps の制御が行われていなければ, フレームの再生間隔がフレームへの送出 fps から考えて, 望まれる間隔より大きくなる場合があり, 送出 fps に応じたサービス品質が提供されるとはいえなくなる. フレームの再生間隔は, フレームフィルタの送出 fps および表示デバイスの負荷に依存する. システムでは, 表示デバイスの負荷に応じて, 表示デバイスへの送出 fps を制御している. そのため, デバイスへ



(a) フィルタプロセス　　(b) 評価プロセス

図 8 重複パケット数の評価を行うプロセス群

Fig. 8 Evaluation Processes for a number of Duplicate Packets



(a) デコーダプロセス　　(b) 評価プロセス

図 9 棄却パケット数の評価を行うプロセス群

Fig. 9 Evaluation Processes for a number of Drop Packets

の送出 fps が下がったのちは, フレームの再生間隔は, 望まれる間隔より大きくならない. そのことを, 次式が満たされるかで検証する.

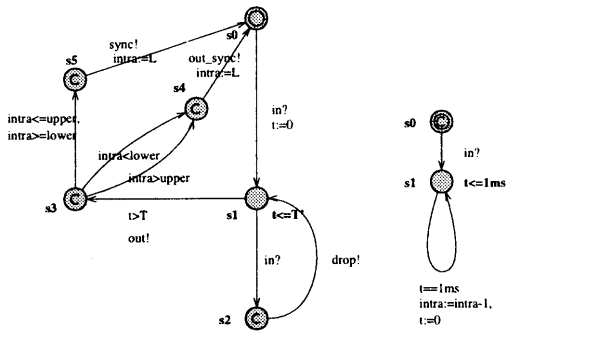
$$A \ \Box\ (t > 0 \textbf{ and } t < T3 \textbf{ and } LoadClock.s1 \textbf{ and } PacketFilter.s1 \textbf{ imply not } IntervalEvaluator.s5) \quad (5)$$

この式は, デコーダの負荷大きく, パケットフィルタの送出 fps が下がっている場合は表示デバイスでのフレームの処理間隔が望ましい間隔であることをすべての状態で満たすことを表している. ここで, IntervalEvaluator は後述する図 10 の評価プロセスである.

5.2.3 サービス品質検証用プロセス

図 8 は, パケットフィルタで重複して送出されるパケットを評価するプロセス群である. 図 8(a) は重複パケットの数を評価するパケットフィルタプロセスである. パケットがバッファリングされていなければ ($R_{buff} == 0$), 保存しておいたパケットを送出し, 重複して送出したパケットの数を保持する N_{over} を 1 ずつ増加させる. 図 8(b) は単位時間 (1 秒) ごとに, 重複されたパケットの数 (N_{over}) の値の評価をするプロセスである. 値がしきい値以下であれば, デコーダの負荷に応じたパケットがパケットフィルタから送出されていると評価できる.

図 9 は, デコーダで棄却されるパケットを評価するプロセス群である. 図 9(a) は棄却パケットの数を評価するデコーダプロセスである. あるパケットを処理中に次のパケットがパケットフィルタから到着したときは, そのパケットを破棄し, 破棄したパケットの数を保持する N_{over} を 1 ずつ増加させる. 図 9(b)



(a) 表示デバイスプロセス　　(b) タイマプロセス

図 10　フレーム再生間隔の評価を行うプロセス群

Fig. 10　Evaluation Model for the interval fo frames

は単位時間 (1 秒) ごとに, 破棄されたパケットの数 (N_{drop}) の値の評価をするプロセスである. 値がしきい値以下であれば, デコーダの負荷に応じたパケットがパケットフィルタから送出されていると評価できる.

図 10 は, 表示デバイスで再生されるフレームの間隔を評価するプロセス群である. 図 10(a) は, フレームを再生するときに, 直前のフレームを再生してからの時間経過を評価する表示デバイスのプロセスである. ここで, L はフレームフィルタの送出 fps から考えられるフレームの望ましい再生間隔を表している (例えば, 送出 fps が 20 であれば, $L = 50ms$) . 図 10(b) は, フレームの時間間隔を表す $intra$ を評価するプロセスである. 時間間隔が望ましい間隔であれば, $intra = 0$ となり, 短ければ, $intra > 0$ となり, 逆に長ければ $intra < 0$ となる. そして, 図 10(a) より, しきい値 $upper$, $lower$ を用いて, 再生間隔が適切であるか (sync!) , 適切でないか (out_sync!) を評価する.

6. 検証結果と考察

この章では, 5 章の検証プロセスを用いて行った, システムの安全性及びサービス品質の検証結果について述べる. なお, Uppaal 3.2.9, Pentium 3 (500Mhz), 512MB の RAM, Vine Linux 2.5 の環境下で検証を行った.

検証結果として、式 (1), (2) に関してはそれぞれ真、偽であることが約 2 秒でわかった. 式 (3) は偽であることが約 20 秒, 式 (4), (5) に関してはそれぞれ真であることが, 約 20 秒でわかった. 以下、式 (2), (3) の結果について考察する.

式 (2) の結果について, QoS 制御プロセスに関しては, システムの負荷が変わった直後のタイミングでは, デコーダ負荷の評価が行われないことがある. そのため, パケットフィルタへの送出 fps を下げる合図を出すことができず, パケットフィルタが送出レートの高い状態 (packetfilter.s0) にとどまったためである. よって, 式 (2) の結果は偽となった. しかし, Uppaal のトレース機能から, しばらくしてデコーダ負荷の評価が行われた後は, 送出 fps を下げる合図を出し, パケットフィルタが送信レートが低い状態 (packetfilter.s1) に遷移していることが確認できた. よって, QoS 制御プロセスは正しく実行されていると判断した.

式 (3) の結果については, パケットフィルタに新規パケットが到着した直後に, QoS 制御プロセスによる負荷の評価が行われた場合, パケットフィルタでバッファリングしているのにもかかわらず, QoS 制御プロセスにより, バッファリングしたパケットの数は値 0 にリセットされる. そのため, その後パケットフィルタがパケットをデコーダへ送出するときには, パケットがバッファリングされていないものと解釈され, 重複パケットをデコーダに送出する. よって, 式 (3) の結果は偽となった.

しかし, 以下の点から考えて, 重複するパケットはほとんど生じておらず, パケットフィルタの送出 fps に応じたパケットが送出されていると判断した. 今回の検証プロセスでは, デコーダの負荷を 3 秒ごとに変更している. したがって, 先ほど説明したパケットフィルタでの重複パケットの誤った判断が最大で 3 回生じること考えられる. このことを, 以下の検証式を調べることとした. 実際,

$$A\ \Box\ (t > 0 \textbf{ and } t < 3000 \textbf{ and } LoadClock.s1 \textbf{ and } PacketFilter.s1 \textbf{ imply } N\_over < 3) \quad (6)$$

$$A\ \Box\ (t > 0 \textbf{ and } t < 3000 \textbf{ and } LoadClock.s1 \textbf{ and } PacketFilter.s1 \textbf{ imply } N\_over < 4) \quad (7)$$

を検証した結果, それぞれ偽, 真となり, パケットフィルタでの重複パケットの誤った判断が毎秒最大で 1 回生じていることが確認できた. 以上の点から考えて, 重複するパケット数がほとんどないと判断できる.

以上示したように, QoS 制御プロセスの正しさ, およびパケットフィルタの送出 fps に応じたパケットが送出されているかをそれぞれ式 (2), 式 (3) の検証式では検証できないことがわかった.

そこで, 本稿では, QoS 制御プロセスの正しさを検証を Uppaal のトレース機能を併用することで対処し, また, パケットフィルタの送出 fps に応じたパケットが送出されていることの検証に関しては, 別の検証式を併用することで対処した.

どのようなクラス、検証性質であれば, 記述の変形の正しさを保証しつつ, Uppaal の検証式で検証できる等は残された問題である.

7. おわりに

本論文では, QoS 制御機能をもつ動画像再生システムを例題にして, Uppaal を用いて, 実時間システムの各プロセスを拡張時間オートマトンで記述し, システム全体を通信チャネルを介して同期するオートマトン群として記述した. そして, システム全体としての安全性および, システム利用者の観点から見た QoS 制御が行われているかを検証した. その結果, 安全性だけでなく QoS 制御の検証が不変性検証により可能であることを示した.

文　献

[1] Howard Bowman, Giorgio P. Faconti, and Mieke Massink. Spesification and verification of media constraints using uppaal. In *the Proceedings of the 5th Eurographics Workshop on the Design, Specification and Verification of Interactive Systems*. Springer-Verla, August 1998.

[2] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal of Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

[3] H. Bowman, G. Faconti, J-P. Katoen, D. Latella, and M. Massink. Automatic verification of a lip synchronization algorithm using uppaal -exteded version-. In *the Proceedings of the 3rd International Workshop on Formal Methods for Industrial Crtial Systems*, pages 97–124, May 1998.

[4] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.