マルチメディアシステムに対するQoS機能試験の一手法

孫 弢[†] 安本慶一[†] 森 將豪[††]

本論文では，分散マルチメディアシステムにおけるクライアントプログラムにおいて，メディアオブジェクトの再生機構が，あらかじめ設計者により与えられた品質基準（フレームレートの範囲やメディア同期の精度など）に対して，正しく実現されているかどうかをテストする方法を提案する．提案手法では，テストのためのシナリオを時間 EFSM と呼ぶ形式モデルを用いて記述する．テストシナリオには，テスト対象プログラム（IUT）への入力フローの動作（特性）と，実現されるべき再生品質（フレームレートの変動がある範囲に収まるなど）を満たすような再生機構の動作を記述する．一般に，マルチメディアシステムでのビデオの再生やメディア同期においては，フレームレートや再生場面の時間のずれを一定範囲にコントロールする必要がある．しかし，なんらかの要因による一時的な乱れ（範囲からの離脱）は，人に知覚できないことも多く，許容できると考える．そこで，本論文では，ごく短い周期で計測したフレームレート（あるいは場面の時間的ずれ）を標本として登録し，比較的長時間の間に記録したすべての標本の分布から，IUT がテストに合格したかどうかを統計的に判定する方式を採用する．テストの合格基準として，設計者は，フレームレートの許容下限となる閾値と，それを下回る標本の数の総標本数に対する割合の最低値（信頼レベル）を与える．以上のテスト判定法を含むテスト系列をテストシナリオから自動生成する．テスト系列の実行系を Java で作成し，あるビデオ再生システムに対して適用し，本手法の有効性を確認した．

A Method for Testing QoS in Multi-media Systems

Tao Sun,[†] Keiichi Yasumoto[†] and Masaaki Mori[††]

In this paper, we propose a testing method for QoS functions in distributed multi-media systems, where we test whether a playback mechanism of media objects is correctly implemented or not in a client side program according to the quality designated in advance, such as allowable range of the frame rate and/or the maximum time lag between parallel playbacks of multiple media objects. In the proposed technique, we describe test scenarios in timed EFSMs where we specify behavior of an input flow transfered to a given IUT (implementation under test) and behavior of playback with certain QoS functions observed from the IUT (e.g., the range of fluctuation of frame rates). In general, when playing back media objects, the frame rate and the time lag between multiple objects should be controlled within a specified range. However, temporal and sporadic exceptions (e.g., the frame rate is out of the range for one second) may not be perceived by humans. Therefore, we can allow such exceptions in multi-media systems. In the proposed test method, we use a statistical approach where actual frame rates and/or time lags are taken as samplings while an IUT is executed, and test results are reported from the ratio of the samplings with low quality below a threshold in a normal distribution of all samplings. From the scenarios, we generate test sequences to test whether a given IUT realizes the QoS functions specified in the scenarios. We have implemented a test system for test sequence execution in Java, and applied it to a video playback system to evaluate the proposed method.

1. はじめに

近年，インターネットのブロードバンド化にともない，高信頼性を有するマルチメディア通信システムの開発手法の確立が望まれている．エンドユーザに対して，所期の QoS 機能を有するマルチメディアサービスを提供するシステムは重要なものであり，フレームレート（単位時間あたりの描画フレーム数）や複数の並行メディアオブジェクト間におけるリップ同期[8]のための制御機構などは，とりわけ重要である．高い信頼性を有するマルチメディアシステムを開発するためには，これらの機構が，与えられたプログラム（Implementation Under Test，以下 IUT と呼ぶ）に

† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†† 滋賀大学経済学部情報管理学科
Faculty of Economics, Shiga University

おいて正しく実現されているかどうかをテストする方法を確立することが望まれる．

従来，プロトコル工学において成功をおさめた伝統的な通信ソフトウエアのテスト手法は，主として入出力対応関係の正しさを扱ったものであり[11]，それゆえに音声や画像のマルチメディアオブジェクトの再生タイミングなどに関する QoS 機能のテストには直接適用することはできない．マルチメディアシステムのテストでは，入出力事象の対応関係だけでなく，対応する生起事象間の時間的乖離の度合いや，生起する事象の継続時間がテストの本質的な問題となるからである．したがって従来の方法では，たとえ入出力事象の対応関係が正しくても，入力事象と出力事象間の時間関係によりテストに通ることが必ずしも保証されないのが普通である．たとえ，マルチメディアオブジェクト間の時間に関する関係を，リアルタイムシステムでよく用いられる Timed CTL のような記述形式[2] を用いて詳細に仕様記述したとしても，テスト系列の複雑さが爆発的に増すだけであり，マルチメディアシステムの QoS 機能テストとしては現実的ではない．たとえば，ビデオ再生の仕様として，ビデオフレームの再生が正確に 33 ± 5 ミリ秒間隔で行われるように記述されていたと仮定しよう．このとき，与えられた IUT がこの仕様を少しばかり満たさなかった（たとえば 1 つのフレームが 10 ミリ秒遅れて再生された）としても，それが突発的なものであり，メディアオブジェクトがなめらかに再生されている限り問題は少ないと考えられる．したがって，マルチメディアオブジェクトの再生に関する QoS テストにおいては，メディアオブジェクトの再生に関する時間関係の結果を統計的に考察してテスト結果の判断を行うのが妥当であると考えられる．

マルチメディア通信システムのテストに関しては，伝送系[5] やコンテンツの品質[6]，および分散システムの相互操作性やパフォーマンステストに関する研究[15]，などがあるが，いずれも時間関係のテストは含まれていない．統計的アプローチを採用したものとしては，SMIL 言語[16] で記述されたマルチメディアプレゼンテーションシナリオにおけるメディアオブジェクト間の時間関係を対象としたテスト方法に関する研究がある[4),12]．ここでは，メディアオブジェクト対の間の再生開始時刻と終了時刻の時間的な二項関係を統計的手法を用いてテストする方法が提案されているが，十分に長い時間幅の中でのメディアストリーミングの再生中の品質をテストしたものではない．さらに，コンカレント時間オートマトンを用いてメディア同期プロトコルの機能テストを行う方法の提案[17] もあるが，そこでは，あらかじめ計算された適当な時間間隔内において各入力動作を実行し，出力動作の実行タイミングが適当な時間間隔内に収まっているか否かを観察することにより，IUT がテストされている．その他のアプローチとして，複数の並行メディアオブジェクト間のリップ同期のようなメディア制約に対してモデルチェッキングを用いる方法も提案されている[3]．

本論文では，ネットワークで接続されたサーバとクライアントからなる分散マルチメディアシステムを想定し，このシステム上でのオブジェクトストリーミングにおける再生中の品質（フレームレートやリップ同期など）を統計的手法を用いてテストする方式を提案する．提案手法では，所与の IUT に対する QoS 機能テストのシナリオを時間 EFSM と呼ばれる形式モデル（EFSM と時間オートマトン[1] の混成モデル）を用いて記述する．ここでは，パケットのジッタやパケットロス率などの入力トラフィック特性，および与えられたトラフィック特性に対して実現されるべき再生品質を満たすようなフレームの再生動作を指定する．さらに，制約指向記述スタイル[14] を用いることにより，メディア間同期で実現すべき精度（場面間の時間的ずれの最大許容値）を，それぞれのメディアオブジェクト（たとえば映像とそれに対応する音声）の再生機構をテストするサブシナリオ間に成り立つべき依存関係として記述することで，メディア同期機構の品質をテストする．これらのテストシナリオより，所与の IUT が指定された品質基準を満たしているか否かをテストするためのテスト系列を生成する．

本論文で提案する統計的手法に基づくテスト手法においては，フレームレートや複数オブジェクトに関する最新のフレーム間のタイムラグについて，IUT からのごく短い時間（たとえば 1 秒間）でのテスト出力（実測データ）を 1 つの標本とし，テスト出力を長時間（たとえば 1 時間）にわたって集約した実測データをもとに標本空間の平均と標準偏差を計算して統計分布を仮定する．そして，得られた分布よりテスタ側で設定しておいた閾値 ϵ（IUT に対する最大許容度）以下の割合を計算し，所期の信頼レベルとを比較してテストにパスしたか否かを判定する．

上述のテスト系列をリアルタイムに実行するためのテストシステムを Java 言語を用いて構築し，それをビデオ再生プログラムに適用することにより，本方式が QoS 機能テストに有効であることを確認した．

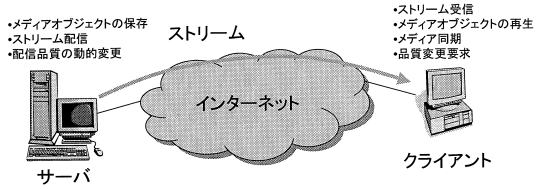


図 1 分散マルチメディアシステム
Fig. 1 Distributed multi-media systems.

2. QoS 機能テストの概要

本論文では，**図 1** に示すように，サーバ計算機（以下，サーバ）がクライアント計算機（以下，クライアント）からの要求に応じて指定されたメディアオブジェクトのストリーミングを行う際に，クライアントでのオブジェクトの再生品質が，入力フローの特性と照らし合わせたうえで適切かどうかを判定するためのテスト手法を提案する．

従来の実時間システムに対するテスト手法[7),10)]では，IUT へデータを適切な時刻（仕様を満たす時刻）に入力し，データが出力される時刻を観測して，それが仕様で指定された時間制約を満たしているかどうかをテストするのが一般的である（**図 2** 上）．しかし，マルチメディアシステムにおけるオブジェクトの再生品質のテストにおいては，パケットロスなどにともなう再生品質のゆらぎにはある程度の許容範囲が存在するため，実現される品質が許容範囲内かどうかをテストする方法が必要になる（図 2 下）．文献 4) では，SMIL 言語[16)]で指定された複数メディアオブジェクト間の時間的な同期関係（終了時刻が同じ，片方の終了 10 秒後に他方を開始など）に対し，IUT における実際の再生時刻間のずれの分布を求め，統計的な手法を用いて指定された同期関係が正しく実現されているかどうかをテストする手法が提案されている．

本論文では，これらの統計的手法を用いて，マルチメディアシステムのクライアントプログラムにおける，メディアオブジェクトの再生品質と複数オブジェクト間のメディア同期の精度をテストする方法を提案する．

2.1 提案手法の概要

提案手法では，与えられたストリームに対し，オブジェクトが再生されるべき品質を算出し，**図 3** に示すように，オブジェクトの各データユニット（動画フレームや音声の断片などで，以下では**フレーム**と呼ぶ）の出力時刻を観測することにより，指定した品質基準どおりに再生できているかどうかをテストする．

一般に，比較的短い周期（たとえば 1 秒）の間に出

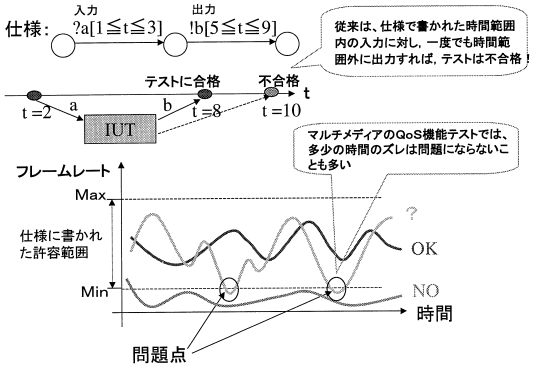


図 2 従来の時間システムのテスト手法と問題点
Fig. 2 Existing test methods for timed systems and problems.

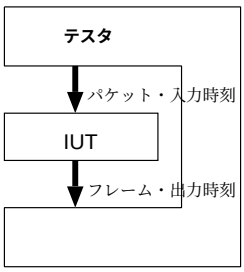


図 3 QoS 機能テストの環境
Fig. 3 Environment for QoS functional testing.

力されるフレーム数の単位時間あたりの平均（**フレームレート**と呼ぶ）は，パケットのジッタやロスなど，入力フローの特性によりつねに変化しうる．そこで，提案手法では，この短い周期（以下 SP で表す）でのフレームレートを標本として記録し，十分に長いモニタ周期（以下 LP で表す）の間に得られた標本の分布をもとに統計的に処理を行い，ある信頼性をもってテストにパスしたか否かを判定する方法を採用する．

マルチメディアシステムのテストを実行するため，まず，テストシナリオを記述する．テストシナリオは，入力フローの特性に従いパケットを適切な時刻に送出する動作を記述した**入力トラフィックテストシナリオ**と，そのフローに対して実現されるべき品質基準を満たすような再生機構の動作を記述した**再生品質テストシナリオ**で構成する．これらのシナリオから以下に示す手続きによりテストケース（テスト系列の集合）を生成する．

- テスト系列では，モニタ周期 LP とフローの出力レートから，周期的なパケット送出時刻に，入力トラフィックシナリオで許された範囲のジッタ

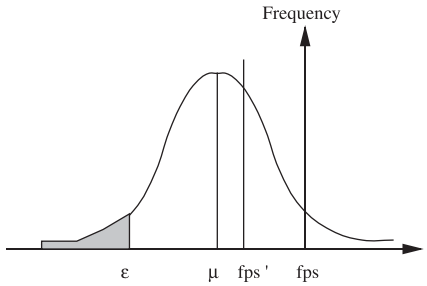


図 4 フレームレートの正規分布
Fig. 4 Distribution graph of frame rates.

成分およびバースト成分，パケットロス率を加味した時刻にパケットを IUT に与える．

- 周期 MP ($SP \leq MP \leq LP$) ごとに IUT における実際のパケットロス率を計測し，2.2 節で定義する式で表される当該周期間で実現されるべき平均フレームレート fps' を算出する．
- 周期 SP ごとに IUT で実現された平均フレームレートを計測し，多くの繰返しテスト試行よりモニタ周期 LP にわたってフレームレートの値の集合を標本として確保する．この標本より単位時間 (SP) ごとのフレームレートの平均と標準偏差を求め，ユーザが設定しておいた IUT に対する最大許容度 ϵ（maximum tolerance acceptable）より決まる信頼レベルとの比較によりテスト結果を判定する（**図 4** 参照）．

簡単のため，フレームレートの分布は図 4 に示すような正規分布に従うものと仮定すると，テストの判定の計算手順は以下のようになる．

(1) 周期 LP の間のテスト試行より得られたフレームレートの標本平均 μ および標準偏差 s を求める．
(2) 標準化の式 $z = (x - \mu)/s$ を用いて ϵ を変換し，標準正規分布表より $[-\infty, (\epsilon - \mu)/s]$ 区間の面積 C の値を求める．
(3) C の値が小さければ（0 に近いほど），標本分布において ϵ 程度の値を持つ fps の再生は稀なことであったと解釈し，テストに合格と判定する．逆に C の値が大きければ（0.5 に近いほど）標本分布において ϵ 程度の値を持つ fps の再生は稀なことではなかったと解釈し，テスト結果は否定される．

一般には，外的または内的な負荷などにより，図 4 に示すようにモニタ周期 LP で計測したフレームレート fps' と 1 秒あたりのフレームレートの標本平均 μ は一致しないことが予想される．この乖離をテストの判定にどのように反映させるかについても考慮する必要がある．

2.2 オブジェクトの再生品質に関する考察

各メディアオブジェクトの再生品質に影響を与える外部要因には，以下が考えられる．

- 入力ストリームにおけるパケット到着時刻のジッタおよびパケットのロス
- クライアントの負荷

たとえば，あるサーバがあるクライアントに，30 fps（フレーム/秒）の動画データを固定送出レートで伝送する場合を想定する．クライアントはサーバからパケットを受信し，適切なフレームレートでの動画の再生を試みる．この際，実現される再生品質（フレームレート）は，受信レートと，パケットロス率，パケット到着時刻のジッタ，クライアントの負荷に依存すると考えられる．

もし，平均送出レートと同等の受信レートで受信できており，パケットロス率が 0 に近く，クライアント計算機の負荷も低い場合には，実現されるべき再生品質は，サーバにおいて送出したデータのフレームレート（すなわち 30 fps）ということになる．パケットロス率が高い場合，あるいはシステムの負荷が高い場合には，再生されるべきフレームレートは 30 fps 未満になる．本論文では，実現されるべき再生品質 fps' を以下のように定義する．

$$fps' = fps \cdot (1 - f(Loss)) \cdot \beta$$

ここで，fps は元の（サーバが送出したデータの）フレームレート，$Loss$ はパケットロス率，$f(x)$ $(0 \leq f(x) \leq 1)$ は各パケットの消失が画像フレームの消失に関与する割合である．ここで，1 つの画像フレームが 1 つのパケットで送られる場合 $f(x) = x$ になる．1 つの画像フレームが複数パケットで送信される場合や，MPEG などフレーム間の依存関係がある場合には $f(x) > x$ となる．ここで，β $(0 < \beta \leq 1)$ はクライアントシステムにおけるネットワーク以外の変動要因（CPU 負荷など）である．以下では，簡単のため，$\beta = 1$ と仮定する．

3. マルチメディアシステムのテストシナリオ

2 章で説明したようにマルチメディアシステムに対するテストシナリオを複数のサブシナリオから構成する．各シナリオは，時間付き拡張有限状態機械（以下，**時間 EFSM** と呼ぶ）を用いて記述する．時間 EFSM は，時間オートマトン[1] に EFSM で用いられる変数やガード式の記述を組み入れたものである．また，複数の並行に動作する時間 EFSM 間の同期や動作に関

する制約は，仕様記述言語 LOTOS[9] のマルチランデブ機構を用いて，それらの時間 EFSM 群を同期並列実行させる形（制約指向スタイル[14] と呼ぶ）で記述する．

各時間 EFSM は，6 項組 $M =< S, A, C, V, \delta, s_0 >$ で記述する．ここで，$S = \{s_0, s_1, ..., s_n\}$ は状態の有限集合，A はアクション（イベント）の有限集合，C はクロックの有限集合である．また，V は変数の集合，$\delta : S \times A \times C \times V \rightarrow S \times V$ は状態遷移関数，s_0 は初期状態である．G をアクションが実行されるゲートの集合，IO をゲートへの入出力の集合とするとき，$A \subseteq G \times IO$ であり，$g?x$ でゲート g から変数 x への値の入力を表し，$g!E$ でゲート g への式 E の値の出力を表す．状態遷移関数 δ は，$s \overset{g?x[Guard]}{\underset{Def}{\longrightarrow}} s'$ のように表記する．

ここで，s は現在の状態，s' は状態遷移後の状態である．このアクションを実行するための条件 $Guard$ は，C のクロック変数と V の変数および定数からなる線形不等式の論理結合で記述する．Def は変数への代入文（{x:=x+1,clock:=0} のように表記する）を表し，この状態遷移の実行時にこの代入文が実行される（クロックのリセットを含む）．

複数の時間 EFSM 間のインタラクションと同期は LOTOS の並列合成オペレータ $|[gate\_list]|$ または $|||$ を用いて記述する．すなわち，システム全体に対するテストシナリオ S は次のように定義される．

$$S ::= S\ |[gate\_list]|\ S \mid S\ |||\ S \mid (S) \mid EFSM$$

ここで，$EFSM$ はある時間 $EFSM$ の名前であり，$|[gate\_list]|$ はオペレータの両側のイベントを同期実行されなければならないイベントのゲートリストである．オペレータ $|||$ は両側のイベントが同期することなく並列実行できることを表す．

3.1 オブジェクト再生機能に対するテストシナリオ

ここで，映像や音声など各メディアオブジェクトを再生する IUT を考える．IUT の再生機能をテストするシナリオ $Player$ は入力トラフィックテストシナリオ S_T と再生品質テストシナリオ S_Q の 2 つの時間 EFSM を用いて次のように記述可能である．

$$Player := S_T|||S_Q$$

入力トラフィックテストシナリオ S_T は，IUT に入力されるトラフィックの特性をパケットの受信アクションなどの系列として記述し，再生品質テストシナリオ S_Q は，IUT で実現されるべき品質基準を満足するようなフレームの出力アクションなどの系列として記述される．これらのシナリオは，外部環境から通信を行うことによって IUT をテストすることを想定している．

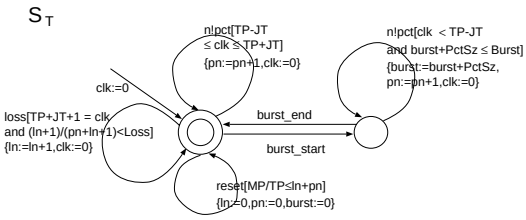


図 5 入力トラフィックテストシナリオの例
Fig. 5 Traffic testing scenario.

S_T に対して，パケットの受信アクションの許容時間範囲を記述する．ここで，様々なネットワーク環境に対応できるようにするため，メディアの送出ビットレート $AvRT$，最大バースト長 $Burst$，最大パケットロス率 $Loss$，パケットの最大遅延変動（ジッタ）JT の 4 つのパラメータを用いる．簡単のため，ここではすべてのパケットのサイズは同一であると仮定し，これを $PctSz$ で表す．**図 5** に入力トラフィックテストシナリオ S_T の例を示す（2 重丸は初期状態）．

図 5 において，clk はクロック変数，TP はメディアオブジェクトをストリーミングする際の標準パケット送信間隔とし，$TP = PctSz/AvRT$ である．

ln，pn は，それぞれ，ある周期 MP の間に失われたパケットの数，送出されたパケットの数を表す．動作系列は次のいずれかである：(1) 通常モード：各パケットを最大遅延変動 JT の範囲内，すなわち $TP - JT \leq clk \leq TP + JT$，にゲート n を介して送信する（n!pct）；(2) バースト転送モード：不定期にこのモードに入り（burst_start），最大バースト長 $Burst$ まで，短い時間間隔 $clk < TP - JT$ でパケットを送信し，バーストが終わると通常モードに戻る（burst_end）；(3) パケットロス：通常モードで，ある周期 MP でのパケットロス率（$(ln+1)/(pn+ln+1)$）が $Loss$ に満たない場合，パケットを消失させる（loss），あるいは，(4) 周期 MP の間隔で，pn，ln および $burst$ を 0 に初期化（reset）する．

同様に，再生品質テストシナリオ S_Q を定義する．そのために，フレームの表示間隔のゆらぎの最大値 FJT と本来の再生予定時刻（たとえば，フレームレートが 30 fps であるビデオの 1000 番目のフレームは，再生開始時から 1000×33 ms=33 秒時点で表示されるべき）とのずれの最大値 MD の 2 つのパラメータを用いる．再生品質テストシナリオ S_Q の例を**図 6** に示す．ここで，TF はオブジェクトを再生する際のフレームの表示時間間隔であり，周期 MP の間に表示された

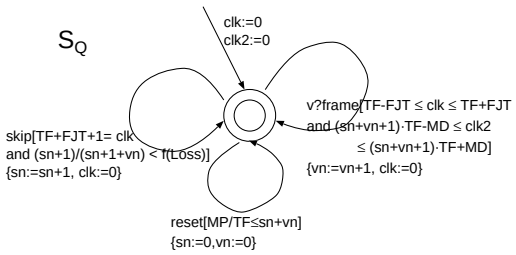


図 6 再生品質テストシナリオの例
Fig. 6 Quality testing scenario.

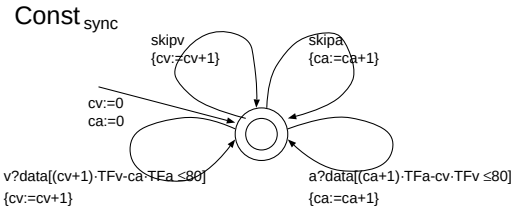


図 7 リップ同期のためのテストシナリオ
Fig. 7 Test scenario for lip synchronization.

フレームと，スキップされたフレームの数を，それぞれ vn，sn で表している．

図 6 には，次の 3 つの動作系列が指定されている：(1) フレーム再生：現在時刻が許容時間範囲（$TF - FJT \leq clk \leq TF + FJT$）以内であり，かつ現在のフレーム $(sn + vn + 1)$ が予期された時間（$(sn + vn + 1) \cdot TF - MD \leq clk2 \leq (sn + vn + 1) \cdot TF + MD$）より早すぎたり遅すぎたりしていない場合，そのフレームを再生する（$v?frame$）；(2) フレームスキップ：現在時刻が許容時間範囲を越えるまで，フレームが表示されずかつ周期 MP で計測中のスキップされたフレームの割合 $(sn+1)/(sn+vn+1)$ が 2.2 節で与えた $f(Loss)$ より小さいとき，フレームの表示をスキップする；(3) 周期 MP の間隔で sn，vn を 0 に初期化する．

3.2 オブジェクト間のリップ同期のテストシナリオ

複数オブジェクト間のリップ同期をテストするためのシナリオを記述する．複数の再生品質テストシナリオの間で，指定した精度でのリップ同期が実現できているかをチェックするための制約を，1 つの時間 EFSM として記述する．

たとえば，動画再生機構に対する再生品質テストシナリオを $Player[n_v, v, skip_v]$，音声再生機構に対する再生品質テストシナリオを $Player[n_a, a, skip_a]$ とし，これらを非同期に並列実行する場合を想定する．ここで，IUT で動画あるいは音声フレームが出力されたことを受け取るゲート名をそれぞれ v，a とする．また，動画，音声のフレームに対する $skip$ 動作をそれぞれゲート $skip_v$，$skip_a$ で受け取れると仮定する．さらに，最新の動画フレーム，音声フレームの先頭からの通し番号を c_v，c_a で表し，各フレームの表示間隔を TF_v，TF_a と表記する．このとき，動画と対応する音声のずれがつねに 80 ms 以内でなければならないという制約 $Const_{sync}$ は，**図 7** のように記述できる．図 7 では，動画フレームの表示アクション $v?$ と音声フレームの再生アクション $a?$ のずれが 80 ms を超えてしまう場合には，対応するアクションが実行できないような制約が記述されている．

上記のリップ同期のテストシナリオ $Const_{sync}$ を加えたシステム全体のテストシナリオは以下のような制約指向スタイル[14] で表すことができる．

$(Player[n_v, v, skip_v]|||Player[n_a, a, skip_a])$
$|[v, a, skip_v, skip_a]|Const_{sync}$

4. テスト系列の生成

3 章で述べた入力トラフィックテストシナリオと再生品質テストシナリオ，またリップ同期のためのテストシナリオから，テスト系列を生成する．以下，テスト系列を次の正規表現で表される系列 $Tseq$ として表記することにする．

$$Tseq := a.Tseq|Tseq +_p Tseq|(Tseq)|Tseq^{*K}$$

ここで，$a.Tseq$ はアクションの逐次実行であり，$Tseq1 +_p Tseq2$ は，系列 $Tseq1$ を $1 - p$ の確率で実行し，系列 $Tseq2$ を p の確率で実行することを意味する．また，$Tseq^{*K}$ は，系列 $Tseq$ を K 回繰り返し実行することを意味する．これらの値の指定がない場合，テスタ（テスト系列の実行系）は，デフォルト値（$p = 0.5$，$K = 100$ など）をもとに，テスト系列を実行する．

4.1 単一オブジェクトの再生機構に対するテスト系列の生成

提案手法では，テスタが IUT の各フレームの再生時刻を観測し，周期 SP ごとフレームレートをサンプルとして収集する．そして，2.1 節で説明した方法を用いて，正規分布を仮定し，最大許容度 ϵ 以下の標本の全体に対する割合を計算し，テストの合否を判定する．

3 章で記述したテストシナリオ S_T および S_Q に対し，周期 SP での標本の取得，モニタ周期 LP 満了後のテスト結果判定のための系列を追加し，分岐 “+” の確率，繰返し “*” の回数を決めることで，テスト系列 $Test_T$ および $Test_Q$ を得る．$Test_T$ および $Test_Q$ の例を，それぞれ，**表 1**，**表 2** に示す．

表 2 において，$Open()$，$Read()$ はファイル操作用のプリミティブであり，$Packet()$ はファイルストリームからパケットを生成するプリミティブである．そして，$Sampling(x)$，$CalcStatistics$，$Judgesult$ は，それぞれ，x を標本として記録，標本の集合から統計値を計算，統計情報からテスト結果を判定，を行うプリミティブである．

表 1 入力トラフィックテストシナリオから得られるテスト系列 $Test_T$ の例
Table 1 Example of test sequence $Test_T$ derived from traffic testing scenario.

$$
\begin{aligned}
&Test_T := \\
&\quad \{fp := Open(file)\}. \\
&\quad \{clk := 0, Loss := 0.0, ln := pn := burst := 0\}. \\
&\quad (\{pct := Packet(Read(fp))\}. \\
&\qquad n!pct[TP - JT \le clk \le TP + JT] \\
&\qquad\qquad \{pn := pn + 1, clk := 0\} \\
&\qquad +[MP/TP \le pn + ln]\{pn := ln := 0\} \\
&\qquad +([TP + JT + 1 = clk \\
&\qquad\quad and\ (ln + 1)/(pn + ln + 1) \le Loss] \\
&\qquad\quad \{ln := ln + 1, clk := 0\} \\
&\qquad\quad +(n!pct[clk < TP - JT\ and\ burst + PctSz \le Burst] \\
&\qquad\qquad \{burst := burst + PctSz, pn := pn + 1, clk := 0\} \\
&\qquad\quad)^{*(Burst/PctSz)} \\
&\qquad) \\
&\quad)^{*(LP/TP)}
\end{aligned}
$$

表 2 再生品質テストシナリオから得られるテスト系列 $Test_Q$ の例
Table 2 Example of test sequence $Test_Q$ derived from quality testing scenario.

$$
\begin{aligned}
&Test_Q := \\
&\quad \{clk2 := 0\}. \\
&\qquad (\{clk := 0, vn := sn := 0\}. \\
&\qquad\quad (v?frame[TF - FJT \le clk < TF + FJT\ and \\
&\qquad\qquad (sn + vn + 1) \cdot TF - MD \\
&\qquad\qquad \le clk2 \le (sn + vn + 1) \cdot TF + MD] \\
&\qquad\quad \{vn := vn + 1, clk := 0\} \\
&\qquad\quad +skip[TF + FJT + 1 = clk] \\
&\qquad\quad \{sn := sn + 1, clk := 0\} \\
&\qquad\quad +[MP/TF \le sn + vn]\{sn := vn := 0\}. \\
&\qquad\quad)^{*(SP/TF)} \\
&\qquad\quad .Sampling(vn/SP)\{vn := 0, sn := 0\} \\
&\qquad)^{*(LP/SP)} \\
&\qquad .CalcStatistics.JudgeResult
\end{aligned}
$$

得られたテスト系列 $Test_T$ と $Test_Q$ は IUT に対し並列に動作しなければならない．それぞれのアクションの実行時間に応じて，様々な組合せがあるため，これらの直積としての系列を構成すると系列が長大になる．そこで，テスタに並列処理機能を持たせることで対処する．

4.2 複数オブジェクトの並行再生時のリップ同期に対するテスト系列の生成

提案手法では，複数のオブジェクトを並行に再生する場合に，それぞれのオブジェクトを再生するプロセス（再生機構）でのフレームの再生時刻の差を標本として求め，2.1 節で述べた統計的手法により，テスト結果を判定する．3.2 節のリップ同期のテストシナリオ $Const_{sync}$ から，**表 3** のようなテスト系列 $Test_{sync}$ を導出できる．

ビデオと音声オブジェクトの再生品質をテストするためのテスト系列をそれぞれ，$Test_v$，$Test_a$ とする．ここで，$Test_v := (Test_{T_v}|||Test_{Q_v})$ である．これらのテスト系列と $Test_{sync}$ を用いて，IUT に対し，リップ同期および再生品質に関するテストを行うためのテスト系列を以下のように構成できる．

$(Test_v[n1, v, skip_v]|||Test_a[n2, a, skip_a])$
$||[v, a, skip_v, skip_a]|Test_{sync}$

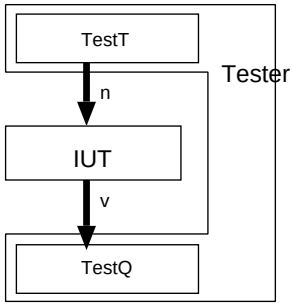
このテスト系列は並列動作および同期を含むため，それらの機能を提供するテスタを実現する必要がある．テスタの実装法については，5 章で述べる．

5. テスト系列の実行

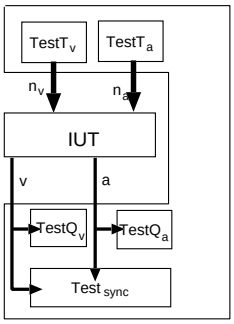
テストは IUT と 4 章で導出したテストケースを実行するプログラムとで行われる．このプログラムを以後テスタと呼ぶ．IUT をブラックボックスとしてテストするために，テスタと IUT は別々のプログラムとして実行し，相互に通信させる．

表 3 リップ同期のテストシナリオから得られるテスト系列 $Test_{sync}$ の例
Table 3 Example of test sequence $Test_{sync}$ derived from lip synchronization test scenario.

$$
\begin{aligned}
&Test_{sync} := \\
&\quad \{c_a = c_v = 0\}. \\
&\quad (v?data\{c_v := c_v + 1\}.Sampling((c_v + 1) \cdot TF_v(q_v) - c_a \cdot TF_a(q_a)) \\
&\quad +skip_v\{c_v := c_v + 1\} \\
&\quad +a?data\{c_a := c_a + 1\}.Sampling((c_a + 1) \cdot TF_a(q_a) - c_v \cdot TF_v(q_v)) \\
&\quad +skip_a\{c_a := c_a + 1\} \\
&\quad)^{*(LP/Min(TF_v(q_v), TF_a(q_a))} \\
&\quad .CalcStatistics.JudgeResult
\end{aligned}
$$



(a) 単一オブジェクトの再生機構のテスト



(b) 2 つのオブジェクトの並行再生とリップ同期のテスト

図 8 テストの実行環境
Fig. 8 Test environment.

5.1 テスタの実現

4 章で導出したテストケースは以下の特徴を持つ．

(1) 各アクションの実行の許容時刻が時間範囲として指定されており具体的な実行時刻が指定されていない．
(2) 選択動作において選択確率が指定されているとともに，部分系列に繰返し回数が指定されている．
(3) 単一オブジェクトの再生機構のテストに 2 つのテスト系列を並列実行するよう指定されている．
(4) 複数オブジェクト間のリップ同期のテストでは，複数のオブジェクトの再生機構に対するテスト系列の並列実行に加えて，それらをリップ同期のためのテスト系列と同期実行するよう指定されている．
(5) Sampling(), CalcStatistics(), JudgeResult() などの統計処理用プリミティブが含まれている．

上記の特徴を持つテストケースを実行するテスタを Java 言語を用いて開発した．4 章の文法で与えられるテストケースの構文解析プログラムは JavaCC で作成し，構文解析後のテスト系列の並列実行およびそれらの間の同期は，文献 18) で提案している実時間 LOTOS コンパイラの手法をもとに実装した．

上記 (1) について，テスト系列の各アクションにはその実行を許容する時間範囲が指定されている．原則的には，この時間範囲のどの時刻にアクションが実行されても，IUT が正しく実行されることを調べる必要があるが，各テスト系列に複数のアクションが含まれる場合には，実行時刻の組合せが無数に発生し，すべてをテストするのは現実的ではない．そこで，テスタは，次に実行すべきアクションが，出力アクション（IUT への出力）の場合には，指定された時間範囲からランダムな時刻を選択しそのアクションを実行する．一方，次のアクションが入力アクションの場合には，IUT からの入力を待ち，実際に入力が実行された時刻を計測し，そのアクションが指定時間範囲に実行されたかどうかをチェックする．また，(2) については，テスタは，指定された値に基づき，乱数を発生させて各分岐を指定確率で選択実行し，繰返しは指定回数の間反復実行する．(1) について，繰返しの回数を増やすことで，テストのカバレッジを改善できる☆．上記 (5) に関しては， 2.1 節で説明した方法に従い対応するプリミティブを Java のメソッドとして実装した．

5.2 テスト環境の構築

テスタは IUT でのフレームの描画時刻を観測できる必要がある．ここでは，IUT でフレームが再生されたときに，フレームの表示とその時刻が，ゲート v でのイベントにより，テスタ側で知ることができると仮定している．また，4 章で求めたテストケースは $Test_T$ と $Test_Q$ の並列実行を含む．作成したテスタでは，これらに対応する 2 つのスレッドを並列実行することで実現した．この場合，IUT を含むテストの実行環境の全体は**図 8** (a) に示すように構築できる．一方，2 つのオブジェクトの並行再生時のリップ同期のテストケースは 4 つのテスト系列（各オブジェクトの再生機構のテストに 2 つの系列が必要）と 1 つのリップ同期に対するテスト系列 $Test_{sync}$ を含む．テスタは，これらのテスト系列を同期並列実行する．この場合のテスト環境は図 8 (b) に示すように構築できる．

6. 実験結果と評価

6.1 オブジェクト再生品質に関する実験結果と評価

テストの目的は，与えられた IUT において，指定されたトラフィックの特性（パケットロス率 *Loss*，ジッ

☆ 文献 4)，12) では，時間範囲の境界付近を集中的にテストしている．提案手法でも同様の方法を採用することは可能である．

表 4 実験に用いたパラメータ
Table 4 Parameters used in experiments.

	JT	$Loss$	$Burst$（バイト）
実験 1	30 ms	0.0	0
実験 2	10 ms	0.0	0
実験 3	30 ms	0.1	0
実験 4	30 ms	0.1	58,800
実験 5	30 ms	0.2	58,800
実験 6	30 ms	0.0	0
実験 7	30 ms	0.0	58,800
実験 8	30 ms	0.1	0
実験 9	30 ms	0.2	0
実験 10	30 ms	0.2	58,800

タ JT，最大バースト長 $Burst$ で与えられる）に対し，所望の品質（実現されるべきフレームレート fps'，信頼レベル ϵ，信頼レベル ϵ 以下の割合の許容上限 r）が実現できているかどうかを調べることである．

本実験では，IUT として，(1) IUT_a：パケットを受信するとすぐにデコード，再生し，フレームレートの調整を行わない，(2) IUT_b：受信したパケットをある容量のバッファに保持し，遅延再生を行うことでパケットの到着時刻のジッタを吸収し，フレーム表示間隔を指定したフレームレートに合うよう制御する，の 2 種類のプログラムを用いた．これらは JMF2.1.1c（Java Media Framework）[13] を用いて実装されている．

IUT_a，IUT_b に対し，上述のパラメータの値を変化させて，5 章のテスタを用いてテスト系列を実行することにより，IUT の再生機構の品質を検査した（実験 1〜10）．パラメータ値の一覧を**表 4** に示す．すべての実験で，動画データとして，モーション JPEG ストリーム（320 × 240，25 fps，1 フレームのサイズは 5.88 K バイトで固定）を用いた．また，$SP = 1$ 秒，$MP = 60$ 秒，$LP = 30$ 分，$TP = 40$ ミリ秒（各フレームを 1 パケットで送信）として IUT に適用した．また，実験では，テストの合否結果ではなく，合否の判定のもととなるフレームレートの分布を測定した（これらの分布から合否判定を得ることは容易）．

実験 1〜5 までは IUT_a に対して行った．実験 1，2 はジッタのみ発生させた場合について実験し，ジッタの大きさによるフレームレートの分布の違いを調べた．実験 3〜5 では，パケットロスを含む場合，パケットロスとバースト転送の両方を含む場合についてフレームレートの分布を調べた．結果を，**図 9**，**図 10** に示す．図の x 軸と y 軸はそれぞれフレームレートと頻度（標本数）を示す．これらの図から，標本はおおむね $fps' = 25 \times (1 - Loss)$ の値を中心として左右に分布しており，fps' に近いほど標本数が多く，遠いほど標本数は少ない．また，ジッタ，パケットロス率，バースト転送のそれぞれの要因により，標本の分布が広範囲に及ぶことが分かる．

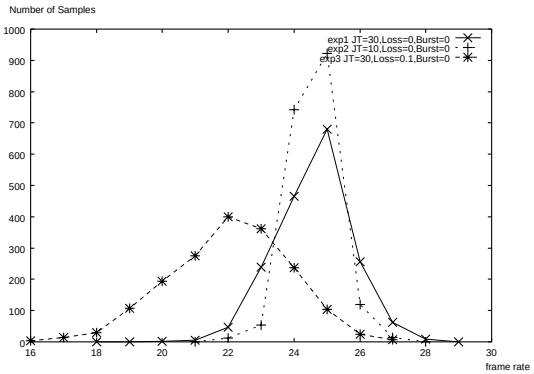


図 9 フレームレートの分布（実験 1〜3）
Fig. 9 Distribution for Experiments 1, 2, 3.

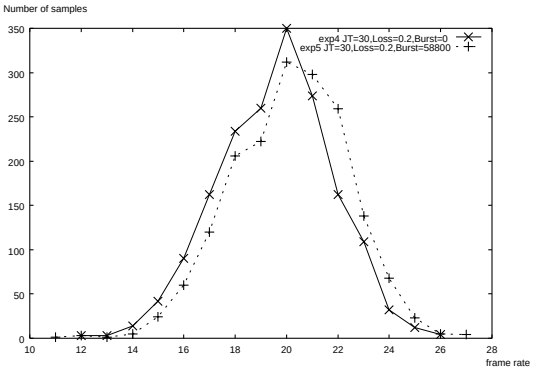


図 10 フレームレートの分布（実験 4，5）
Fig. 10 Distribution for Experiments 4, 5.

次に，バッファおよびフレームレートの制御機構を有する IUT_b に対して実験 6〜10 を行った．実験 6，7 では，それぞれ，ジッタのみ発生させた場合，ジッタ，バースト転送の両方がある場合のフレームレートの分布を調べた．実験 8，9 では，パケットロスがある場合について，実験 10 では，パケットロスとバースト転送の両方がある場合についてフレームレートの分布を測定した．結果を**図 11** および**図 12** に示す．図 11（実験 6，7）では，バッファによりジッタが吸収されるため，すべての標本が $fps' = 25$ あるいはそれより 1 少ない値に集中していることが分かる．一方，図 12 の実験 8，9 との比較で分かるように，パケットロス率が高いほど，fps' を中心として左右に広く分布することが分かる．これは，$Loss$ が周期 MP=60 秒の間のパケットロス率の最大として指定されているため，各標本周期では，パケットロス率が $Loss$ より大きい場合と小さい場合が存在しうるためと考えられる．また，図 12 で実験 9，10 を比較すると，バース

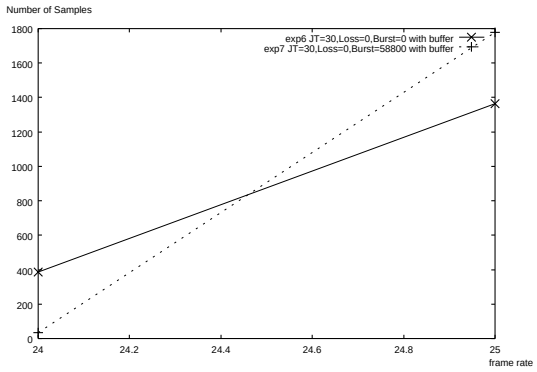


図 11 フレームレートの分布（実験 6，7）
Fig. 11 Distribution for Experiments 6, 7.

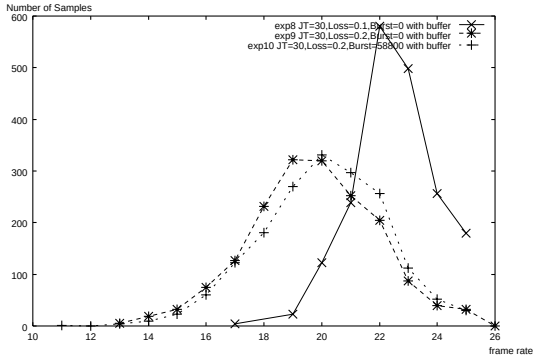


図 12 フレームレートの分布（実験 8〜10）
Fig. 12 Distribution for Experiments 8, 9, 10.

トの有無ではあまり差は見られないことが分かる．

次に，2.1 節で述べた方法により，テストの合否を統計的に判定する．標本の平均および標準偏差を求め，正規分布を仮定する．また，最大許容度 ϵ を $fps' \pm 20\%$ とし，信頼レベル以下（以上）の割合の許容上限 r を 2%と設定する場合を考える．IUT_a，IUT_b に対し，実験 1〜10 を適用した結果の，標本の平均値，標準偏差，信頼レベル以下の割合，およびテストの合否判定の一覧を**表 5** に示す．以上のテスト結果から，ジッタが 10 ミリ秒以内と比較的小さく，かつ，パケットロス率が 0 に近い環境では，IUT_a はテストに合格し，ジッタが 10 ミリ秒を超える場合やパケットロス率が 0.1 以上の環境では，テストに不合格となることが分かる．また，IUT_b は，パケットロス率が 0.1 以下で，かつ，ジッタが 30 ミリ秒以下であるような環境では，テストにパスすることが分かる．

正規分布を仮定することの妥当性を評価するため，例として，実験 3 の結果（図 9）について，ϵ を変化させた場合の，ϵ 以下の（$\epsilon > fps'$ のときは，ϵ 以上の）標本数の全体に対する割合と，正規分布における

表 5 テストの合否判定
Table 5 Test results.

	平均	標準偏差	ϵ	ϵ 以下の割合	合否
実験 1	24.6	2.65	20.0	4.2%	否
実験 2	24.8	1.71	20.0	0.0%	合
実験 3	22.0	3.50	18.0	12.3%	否
実験 4	19.6	4.36	18.0	21.0%	否
実験 5	20.1	5.03	16.0	20.3%	否
実験 6	25.0	—	20.0	0.0%	合
実験 7	25.0	—	20.0	0.0%	合
実験 8	22.5	4.23	18.0	1.8%	合
実験 9	20.0	4.86	16.0	16.6%	否
実験 10	19.7	3.80	16.0	20.3%	否

表 6 正規分布と実際の標本の分布との比較
Table 6 Comparison between normal distribution and distribution of actual samplings.

	ϵ 以下（以上）の割合	
フレームレート	正規分布	実測
16	0.04182	0.002277
17	0.07493	0.010245
18	0.12302	0.026750
19	0.19215	0.088218
20	0.27759	0.198633
21	0.38209	0.355150
22	0.49601	0.582242
23	0.39358	0.417758
24	0.28774	0.211725
25	0.20045	0.076836
26	0.12924	0.018213
27	0.07927	0.003984

$[-\infty, (\epsilon - \mu)/s]$ の区間の面積 C の全体の面積に対する割合を比較した（μ，s は平均および標準偏差）．結果を**表 6** に示す．

表 6 によれば，ϵ をどの値に設定した場合でも，正規分布で近似した値が，実測値よりも大きな値となっている．すなわち，テストに合格すれば，ϵ 以下の標本の割合はつねに r 未満となり，判定結果は正しい．したがって，提案手法のように，限られた標本数から統計的な処理を行うにあたって，正規分布を仮定することは妥当であると結論できる（正規分布以外の，実測により近い分布を採用することも可能である）．

6.2 オブジェクト間リップ同期に関する実験結果と評価

次に，複数オブジェクトを並行再生する際に，与えられた IUT のリップ同期機構が所望の正確さで実現できているかどうかをテストする．

この実験には 2 つの動画を並行再生する IUT_c☆を用いた．IUT_c は IUT_b のように，バッファとフレー

☆ 簡単のために本実験では，2 つの動画間の再生位置のズレを調べたが，音声についても同様の方法でテストが可能である．

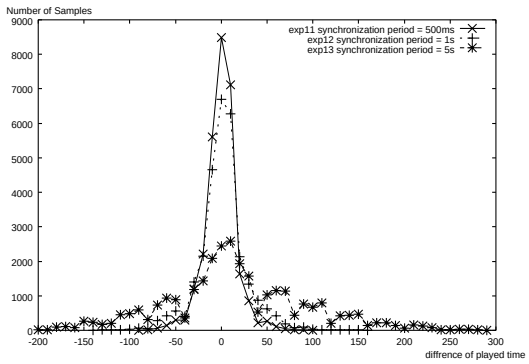


図 13 リップ同期のズレの分布（実験 11，12，13）
Fig. 13 Distribution of lip synchronization for Experiments 11, 12, 13.

表 7 リップ同期テストの合否判定
Table 7 Lip synchronization test results.

	平均	標準偏差	ϵ	ϵ 以下の割合	合否
実験 11（周期 =500 ms）	−0.520	12.842	40	0.082%	合
実験 12（周期 =1,000 ms）	1.279	16.141	40	0.84%	合
実験 13（周期 =5,000 ms）	16.057	20.815	40	12.5%	否

ムレートの制御機構を持ち，かつ，複数のオブジェクトの再生時に周期的にオブジェクト間の再生位置のズレを補正する同期機構を含む．また，ズレを補正する周期を 500 ミリ秒，1 秒，5 秒と変化させ，5 章のテスタを用いて 2 つのオブジェクトに対応する 2 組のテスト系列とリップ同期の分布を記録するテスト系列を同期並列実行することにより，IUT_c のリップ同期機構を検査した（実験 11〜13）．すべての実験で，6.1 節の実験と同じ動画データを用い，テスタのパラメータ値には 2 組のテスト系列とも $JT = 30$，$Loss = 0$，$Burst = 0$ を用いた．結果を**図 13** に示す．この図から，標本はおおむね 0 を中心として左右に分布して，同期周期が短くなればなるほど分布が 0 に集中してくることが分かる．

得られた標本の分布から，2.1 節で述べた方法により，テストの合否を統計的に判定する．標本の平均および標準偏差を求め，正規分布を仮定する．また，最大許容度 ϵ を 40 ms とし，信頼レベル以下（以上）の割合の許容上限 r を 2%と設定する場合を考える．実験 11〜13 を適用した結果の，標本の平均値，標準偏差，信頼レベル以下の割合，およびテストの合否判定の一覧を**表 7** に示す．表 7 から，与えられた環境において IUT_c がテストに合格するには，リップ同期機構におけるずれ補正周期を 1 秒以下にすればよいことが分かる．

7. おわりに

本論文では，分散マルチメディアシステムのクライアントプログラムにおける QoS 機能を統計的にテストする方法を提案した．提案手法では，IUT におけるメディアオブジェクトの基本的な再生品質をテストする．そのため，IUT に入力されるトラフィックの特性と，そのトラフィックに対して実現されるべき再生品質を規定したテストシナリオを時間 EFSM モデルにより記述し，それらのシナリオから，実行可能なテスト系列を生成する．生成したテスト系列は，IUT が，与えられたトラフィックに対し，所期の再生品質を実現しているかどうかを統計的にテストする．

与えられたテスト系列を用いて，IUT をリアルタイムでテストするテスタプログラムを Java により実装した．JMF で実装されたいくつかの IUT に対し，実験を行ったところ，提案手法により，パケット到着時刻のジッタ，パケットロス率，バースト長などで特徴づけられる様々なネットワーク環境に対し，所期の QoS が実現できているかどうかを，テストシナリオを用いることにより効率良くテストすることができた．

今後の課題として，高度な QoS 制御機構（優先度付きメディアスケーリング機構など）を実装した IUT に対し，本テスト手法を適用することを検討している．これにより，提案手法の有効範囲がより明確化すると思われる．

謝辞 本研究を行うにあたり，大阪大学大学院情報科学研究科の東野輝夫教授より貴重なご意見をいただきました．ここに，感謝の意を表します．

参考文献

1) Alur, R. and Dill, D.L.: Theory of timed automata, *Theoretical Computer Science*, Vol.126, pp.183–235 (1994).
2) Alur, R. and Henzinger, T.A.: Logics and Models of Real Time: A Survey, *Real Time: Theory in Practice*, LNCS 600, pp.74–106 (1992).
3) Bowman, H., Faconti, G. and Massink, M.: Specification and verification of media constraints using UPPAAL, *Proc. 5th Eurographics Workshop on the Design, Specification and Verification of Interactive Systems*, pp.261–277 (1998).
4) Cheung, S.C., Chanson, S.T. and Xu, Z.: Toward Generic Timing Tests for Distributed

Multimedia Software Systems, *IEEE Int'l. Symp. on Software Reliability Engineering* (2001).
5) Fibush, D.K.: Testing multimedia transmission systems, *IEEE Design & Test of Computers*, Vol.12, No.4, pp.24–44 (1995).
6) Grabowski, J. and Walter, T.: Testing Quality-of-Service Aspects in Multimedia Applications, *Proc. 2nd Workshop on Protocols for Multi-media Systems* (*PROMS*) (1995).
7) Higashino, T., Nakata, A., Taniguchi, K. and Cavalli, A.R.: Generating Test Cases for a Timed I/O Automaton Model, *Proc. 12th IFIP Workshop on Testing of Communicating Systems* (*IWTCS'99*), pp.197–214 (1999).
8) Huang, C.M. and Wang, C.: Synchronization for Interactive Multimedia Presentations, *IEEE MULTIMEDIA*, Vol.5, No.4, pp.44–62 (1998).
9) ISO: Information Processing System, Open Systems Interconnection, LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807 (1989).
10) Kone, O.: A Local Approach to the Testing of Real-time Systems, *The Computer Journal*, Vol.44, No.5 (2001).
11) Lee, D. and Yannakakis, M.: Principles and Methods of Testing Finite State Machines — A Survey, *Proc. IEEE*, Vol.84, No.8 (1996).
12) Misic, V.B., Chanson, S.T. and Cheung, S.: Towards a Framework For Testing Distributed Multimedia Software Systems, *Proc. 1998 Int'l. Symp. on Software Engineering for Parallel and Distributed Systems* (*PDSE98*) (1998).
13) Sun Microsystems: Java Media Framework Home Page. http://java.sun.com/products/java-media/jmf/
14) Vissers, C.A., Scollo, G. and Sinderen, M.v.: Specification Styles in Distributed Systems Design and Verification, *Theoretical Computer Science*, Vol.89, No.1, pp.178–206 (1991).
15) Walter, T., Scheferdecker, I. and Grabowski, J.: Test Architectures for Distributed Systems — State of the Art and Beyond (Invited Paper), *Testing of Communication Systems*, Vol.II, Chapman & Hall (1998).
16) W3C: Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. http://www.w3c.org/TR/REC-smil/
17) Yamada, M., Mori. T., Fukada, A., Nakata, A. and Higashino, T.: A Method for Functional Testing of Media Synchronization Protocols, *Proc. 16th Int'l. Conf. on Information Networking* (*ICOIN-16*) (2002).
18) Yasumoto, K., Umedu, T., Yamaguchi, H., Nakata, A. and Higashino, T.: Protocol animation based on event-driven visualization scenarios in real-time LOTOS, *Computer Networks*, Vol.40, No.5, pp.639–663 (2002).

(平成 15 年 5 月 19 日受付)
(平成 15 年 12 月 2 日採録)

孫　　弢（学生会員）
平成 7 年中国青島科学技術大学自動制御学科卒業．平成 15 年 4 月滋賀大学大学院経営学修士課程修了．現在，奈良先端科学技術大学院大学情報科学研究科博士後期課程在学中．

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．平成 9 年モントリオール大学客員研究員．平成 14 年より奈良先端科学技術大学院大学情報科学研究科助教授．博士（工学）．分散システム，マルチメディア通信システムに関する研究に従事．IEEE/CS 会員．

森　　將豪（正会員）
昭和 46 年名古屋工業大学工学部電子工学科卒業．昭和 48 年大阪大学大学院修士課程修了．現在，滋賀大学経済学部情報管理学科教授．工学博士．平成 4 年ブリティッシュコロンビア大学客員研究員．分散システムや，通信プロトコルの検証・テスト等に関する研究に従事．