QoS Functional Testing for Multi-media Systems

Tao Sun[1], Keiichi Yasumoto[1], Masaaki Mori[2], and Teruo Higashino[3]

[1] Graduate School of Information Science, Nara Institute of Science and Technology,
{song-t,yasumoto}@is.aist-nara.ac.jp
[2] Department of Information Processing and Management, Shiga University,
mori@biwako.shiga-u.ac.jp
[3] Graduate School of Information Science and Technology, Osaka University,
higashino@ist.osaka-u.ac.jp

Abstract. In this paper, we propose a testing method for QoS functions in distributed multi-media systems, where we test whether playback of media objects is correctly implemented or not in a client side program according to the quality designated in advance, and/or whether a time lag between parallel playbacks of multiple media objects is controlled within the specified time interval. 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) . From the scenarios, we generate test sequences to test whether a given IUT realizes the QoS functions specified in the scenarios. In the proposed test method, we use a statistical approach where test sequences take samplings of actual frame rates and/or time lags when an IUT is executed, and report test results from ratio of samplings with low quality below a threshold in a normal distribution of all samplings. We have implemented a test system for test sequence execution in Java, and applied it to a video playback system.

1 Introduction

As broadband infrastructure is widespread in the Internet in recent years, it has been needed to establish a high-reliable development method for multi-media communication systems. It is important for such systems to provide multi-media services with certain QoS (Quality of Service) functions to end users. Among various QoS functions, control mechanisms for the frame rate (the number of frames displayed every second) and for the lip synchronization [8] among multiple concurrent media objects seem most important ones. For development of multi-media systems with high-reliability, it is desirable to establish a method for testing if those QoS functions are correctly implemented in a given IUT (Implementation Under Test).

Traditional software testing methods which have succeeded in protocol engineering, focus mainly on the correctness of input / output correspondences[11]. Therefore those methods could not be directly applied to QoS testing such as video / audio playback timings. Essentially important test problems of multi-media systems are not only correspondence relations of input / output actions, but time difference between an input and the corresponding output or the time duration in executing a sequence of actions. Then,

H. König, M. Heiner, and A. Wolisz (Eds.): FORTE 2003, LNCS 2767, pp. 319–334, 2003.

even if all the correspondence relations of input / output actions hold, it is not necessarily guaranteed to pass a test due to the time difference between input and output actions.

Even if temporal relations among multi-media objects are specified in detail by a formalism such as Timed CTL [2] used in real time systems, it makes test sequences be explosively complicated and is not realistic to test multi-media systems. Suppose that a specification for a video playback system specifies that a video frame is drawn exactly every 33 msec plus/minus 5msec. In general, when an IUT does not satisfy such a specification a little (e.g., only a frame were delayed for 10msec), it may not be considered a problem as long as media objects are played back naturally. So, for QoS testing for the playback of media objects, it is desirable to statistically analyze the temporal relations and give test results based on statistically calculated information[14].

There exist some studies on multi-media testing such as testing of multi-media transmission systems [5], the quality of multi-media contents [6] and interoperability and performance testing of distributed systems [14]. However they do not deal with testing on temporal relations of input / output actions. A few researches, which deal with temporal relations in multi-media systems explicitly, have been reported [4, 12]. These studies propose a method to test binary temporal relations on the starting time and/or the ending time between two objects by using a statistical approach. However they do not deal with the quality during playback of an object. Moreover, [16] proposes a method for functional testing of media synchronization protocols using a concurrent timed I/O automata model, where a given IUT is tested by executing each input action within an appropriate time interval calculated in advance, and by observing whether execution timings of output actions are within the appropriate time interval. However, this approach does not consider statistical aspects of multi-media systems. As a different approach, [3] proposes a model checking method for media constraints such as lip synchronization among multiple media objects.

In this paper, we suppose typical distributed multi-media systems consisting of servers and clients connected via a network, and propose a method for testing QoS of media playback functions w.r.t. frame rates and lip-synchronization for a client program (IUT). In the proposed method, we specify scenarios for QoS functional tests for an IUT in timed EFSM (a hybrid model of EFSM and timed automata [1]), where we designate the input flow characteristics like jitter / packet loss ratio and the play-back quality of frames to be realized for the given traffic. Furthermore, we specify the quality of inter-media synchronization as a constraint to be satisfied among sub-scenarios corresponding to concurrent playbacks of different media objects (e.g., video and corresponding audio) by using the constraint oriented description style [13]. From those test scenarios, we generate test sequences to test whether a given IUT realizes the specified QoS.

In the proposed test method, we use a statistical approach where test sequences take samplings of actual frame rates and/or time lags between the latest frames on multiple objects when an IUT is executed, and report test results from ratio of low quality samplings (e.g., frame rates less than a threshold) below a specified threshold in a normal distribution of all samplings.

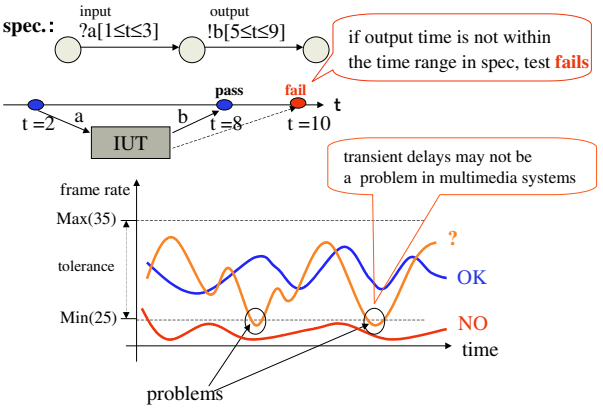


Fig. 1. Existing real-time testing methods

We have implemented a test system for executing test sequences in real-time in Java language, and applied it to a sample video playback program. Our experimental results show that our method works effectively for QoS functional tests.

2 Outline of Proposed QoS Functional Testing

As a target, we suppose distributed multi-media systems where a server transmits a stream of a requested media object to each client. Here, we test whether the playback quality of media objects at a client computer is feasible or not according to the characteristic of the given input flow.

In existing real-time testing methods as in [7, 10], tests are carried out by giving each input to the IUT at appropriate time, and by observing and testing if the corresponding output action is executed at appropriate time satisfying the constraints given in the specification (see Fig. 1). However, testing playback quality of media objects in multi-media systems should be different from those existing methods since some jitter in multi-media playback which may be caused by packet losses/delays is allowed to a certain extent. So, we need a new testing method for playback quality of multi-media objects.

In the proposed method, we adopt the statistical technique for testing playback quality of a single media object and of preciseness of inter-media synchronization among multiple object playbacks at a client computer of a client-server based system.

2.1 Outline of Proposed Method

In the proposed method, we calculate the ideal playback quality of an object for a given data stream (flow), and test whether or not the IUT works satisfying the constraints in the specified test scenario by observing the time at which the object outputs each data unit (called *frame*. e.g., a video picture, a unit of audio data, etc) as shown in Fig. 2.

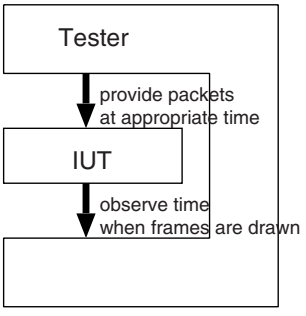


Fig. 2. Environment for QoS Functional Testing

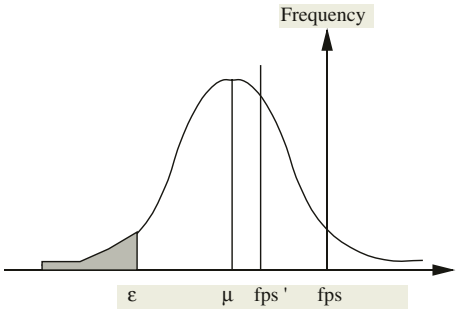


Fig. 3. Distribution Graph of Frame Rates

In general, when we measure the frame rate (the number of frames displayed every second) of an object in a relatively short time interval (denoted as SP), it can vary due to the characteristics of an input flow such as jitter in packet arrival time and packet losses. In the proposed method, we take a frame rate as a sampling every SP time interval for a sufficiently long time interval (denoted as LP), calculate statistic information from distribution of those samplings, and judge from the information whether or not the IUT correctly implements the control mechanism for the frame rate.

We compose a test scenario of the following two sub-scenarios: (1) *traffic testing scenario* which characterizes an input flow, and (2) *quality testing scenario* which represents behavior of the playback with certain quality. From those scenarios, we generate the following test cases (a set of test sequences).

- each test sequence transmits packets to the IUT at time within the allowable time range considering jitters and bursts specified in the traffic testing scenario.
- each test sequence measures actual packet loss ratio in the IUT for a time interval MP (such that $SP \leq MP \leq LP$), and calculates the ideal frame rate fps' which is defined in Sect. 2.2.
- each test sequence measures average frame rate every time interval SP by observing output from the IUT, and keeps it as a sampling. The test sequence collects samplings for time interval LP, and calculates the average value and the standard deviation from those samplings. The test sequence judges whether the IUT correctly implements the frame rate control mechanism with the specified QoS level, based on those calculated statistical values and *maximum tolerance acceptable* denoted by ϵ which the test examiner gives in advance (see Fig. 3).

For the sake of simplicity, we assume that the distribution of frame rates follows the normal distribution like in Fig. 3. Then the judgment process can be described as the following procedure.

1. calculate the average value μ and the standard deviation s derived from the samplings kept during time interval LP.
2. apply normalization expression $z = (x - \mu)/s$ to ϵ, and calculate area C that is an integral of $(-\infty, (\epsilon - \mu)/s]$ in the standard normal distribution (see Fig. 3).

3. when the value of C is smaller than the *reliance level r* which was given by the test examiner in advance, we conclude that the IUT passes the test.

In general, as shown in Fig. 3, we expect that the average frame rate μ measured during time interval LP may match neither the original frame rate fps nor the frame rate considering packet losses fps' explained in the next section, due to external and/or internal load factors.

2.2 Discussion about Playback Quality of Objects

The following factors are considered to give some influences to playback quality of objects.

- jitter in packet arrival time and packet loss ratio
- heavy load/low performance at a client computer

For example, suppose that a server transmits to a client a video file encoded in 30 frames/sec at a fixed transmission rate. A client computer receives packets from the server and tries to play back the video at an appropriate frame rate. In this case, the playback quality depends on the receiving rate, packet loss ratio, jitters in packet arrival time, and load of the client computer.

If the client receives packets at almost the same rate as the server transmits, if packet loss ratio is almost 0 %, and if its load is light enough, the frame rate to be achieved will be close to the originally encoded one (i.e., 30 frames/sec). On the other hand, if the packet loss ratio is high and/or the client's load is high, the frame rate will be less than 30 frames/sec. According to the above discussion, we define the ideally achievable frame rate as the following expression.

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

Here, fps and $Loss$ denote the originally encoded frame rate and the packet loss ratio, respectively. $f(x)$ $(0 \leq f(x) \leq 1)$ denotes the function representing ratio of how much each packet's loss causes the frame loss. Here, $f(x) = x$ when each frame is transmitted by exactly one packet. When each frame is transmitted by several packets and/or there is inter-frame dependency like MPEG movies, $f(x)$ will be larger than x. β $(0 < \beta \leq 1)$ is another factor such as CPU load other than flow characteristics at a client computer. For the sake of simplicity, we suppose that $\beta = 1$ in this paper.

3 Test Scenarios for Multi-media Systems

As explained before, each test scenario for a multi-media system consists of a multiple sub-scenarios. We specify these scenarios in a **timed EFSM**. In timed EFSM, variables and guard expressions with those variables (i.e., execution condition of transitions) used in EFSM can be used in addition to the basic functions of timed automata[1]. Moreover, synchronization and constraints among multiple timed EFSMs are specified in the form of synchronized parallel execution of those timed EFSMs using the multi-way synchronization of LOTOS [9]. That is, we specify behavior of the whole system by

making timed EFSMs execute synchronously and in parallel with the constraint oriented description style[13].

A timed EFSM is given as 6-tuple $M =< S, A, C, V, \delta, s_0 >$, where $S = \{s_0, s_1, \cdots, s_n\}$ is a finite set of states, A is a finite set of actions (events), C is a finite set of clock variables, V is a finite set of variables, $\delta : S \times A \times C \times V \rightarrow S \times V$ is a transition function, and s_0 is an initial state. Let G be a set of gates which represent interaction points to an external environment and IO be a set of inputs/outputs from/to a gate. Here, $A \subseteq G \times IO$. $g?x$ represents an action which inputs a value from gate g and stores it in variable x, and $g!E$ represents an action which outputs the value of expression E to gate g, respectively. A transition function δ is represented by $s \xrightarrow[Def]{g?x[Guard]} s'$ where s and s' are the current state and the next state just after an action is executed at state s, respectively. A transition condition for an action, denoted by $Guard$, is represented by a logical conjunction of linear inequality with clock variables in C, variables in V and constants. Def is a set of value assignments to variables including reset of clock variables, which is represented as $\{x := x + 1, clock := 0\}$ for example, and a value assignment is executed when a state transition occurs.

We specify interaction and synchronization among timed EFSMs with the multi-way synchronization mechanism. The test scenario for the whole system S can be defined as follows.

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

Here, $EFSM$ is a name of a timed EFSM, and $|[gate\_list]|$ is the synchronous parallel operator where $gate\_list$ is a gate list of the events to be synchronized between its operator's both sides of timed EFSMs. The operator $|||$ is the asynchronous parallel operator, and it denotes that its operator's both sides of timed EFSMs can run in parallel without any synchronization. Those parallel operators can be used recursively.

3.1 Test Scenario for Object Playback Functions

We suppose an IUT which plays back media objects such as audio and video data. We specify a test scenario to test a playback function of the IUT with two timed EFSMs: S_T and S_Q, which correspond to the traffic testing scenario and the quality testing scenario, respectively. The whole test scenario $Player$ is given as follows.

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

S_T specifies traffic characteristics which the IUT receives, and S_Q specifies quality constraints about frame displaying time which the IUT should satisfy. Note that these scenarios test given IUTs from external environments.

In S_T, we explicitly specify an allowable time interval between subsequent packets which the IUT receives. Here, for applicability to various network environments, we use four parameters: (1) transmission rate $AvRT$ of a media object, (2) maximum burst length $Burst$, (3) maximum packet loss ratio $Loss$, and (4) maximum jitter in packet arrival time JT. For the sake of simplicity, we assume that all packets have the same size and denote this by $PctSz$. We show example description of S_T in Fig. 4 where a double circle means the initial state.

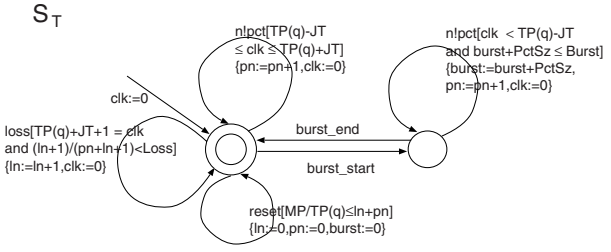


Fig. 4. Traffic Testing Scenario

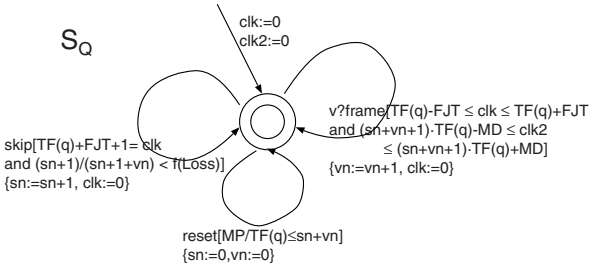


Fig. 5. Quality Testing Scenario

In Fig. 4, clk and $TP(q)$ represent a clock variable and a time interval by which a server transmits each packet to realize playback quality q of a media object, respectively. Here, $TP(q) = PctSz/AvRT$. ln and pn denote the numbers of lost packets and transmitted packets during a time interval MP, respectively. There are four branches in S_T : (1) normal mode: each packet is transmitted to gate $n(n!pct)$ in a time range of maximum jitter JT, that is, $TP(q) - JT \leq clk \leq TP(q) + JT$; (2) burst transmission: when going into burst transmission mode ($burst\_start$), each packet is transmitted in smaller interval, that is, $clk < TP(q) - JT$ while the totally transmitted data size does not exceed $Burst$. When burst transmission finishes, the current state returns to a normal mode ($burst\_end$); (3) packet loss: each packet can be lost if packet loss ratio $((ln + 1)/(pn + ln + 1))$ measured during time interval MP is less than $Loss$; and (4) initialization: pn, ln and $burst$ are initialized to 0 ($reset$) every time interval MP. As we will explain in Sect.4, by executing these choices repeatedly in an appropriate probability we can test the IUT in the environment including jitter, loss and burst to receive packets.

Similarly, we give S_Q on playback quality of frames. For S_Q, we use two parameters FJT and MD, where FJT specifies the maximum value of fluctuation on the time interval of subsequent frames while a media object is played back, and MD specifies the maximum time skew representing how long the current frame can be delayed or preceding from the expected displaying time of the frame (e.g., 1000th frame in 30 fps video is expected to be displayed at 1000×33ms=33sec after the video started). We show an example description of S_Q in Fig. 5, where $TF(q)$ indicates the standard time interval

between two subsequent frames with quality q, and vn and sn indicate the numbers of displayed frames and skipped frames measured during time interval MP, respectively.

There are three branches in S_Q of Fig. 5: (1) frame display: when the current time (clk) is within the appropriate time interval ($TF(q) - FJT \leq clk \leq TF(q) + FJT$) and the current frame ($sn + vn + 1$-th frame) is not too much delayed or preceding from the expected time ($(sn+vn+1) \cdot TF(q) - MD \leq clk2 \leq (sn+vn+1) \cdot TF(q) + MD$), a frame is allowed to be displayed ($v?frame$); (2) frame skipping: when the current time exceeds the allowable time interval without displaying any frames and the frame skipping rate $(sn+1)/(sn+vn+1)$ measured during MP is less than $f(Loss)$ defined in Sect. 2.2, a frame is allowed to be skipped ($skip[TF(q) + FJT + 1 = clk]$); and (3) initialization: we initializes variables sn and vn to 0 every time interval MP.

3.2 Scenario for Testing Lip-Synchronization among Multiple Objects

We describe a sub-scenario for testing lip-synchronization among multiple media objects as a constraint among multiple quality testing scenarios for those objects, using the constraint oriented description style[13].

For example, let $Player[n_v, v, skip_v]$ and $Player[n_a, a, skip_a]$ be quality testing scenarios for video playback and for audio playback, respectively, and we assume that these two scenarios are executed independently in parallel. We also denote n_v, n_a and v, a to be input gate names and output gate names of video and audio, respectively. $skip$ behaviors for video and audio are distinguished by $skip_v$ and $skip_a$. Moreover, we let c_v and c_a denote the sequence numbers of frames of video and audio frames, and $TF_v(q_v)$ and $TF_a(q_a)$ denote the playback interval of video and audio frames, respectively. Here, we describe the sub-scenario $Const_{sync}$ such that the maximum time skew between video and audio must be within T_{lip}msec as shown in Fig. 6. In general, T_{lip} should be less than 80msec. In Fig. 6, it is explicitly described that each output of a video frame $v?$ (output of an audio frame $a?$) is allowed only if the time skew between the expected time of the current frame and that of audio (video) is kept less than T_{lip} msec.

The test scenario for the whole system including the lip-synchronization constraint is given by the following constraint oriented description.

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

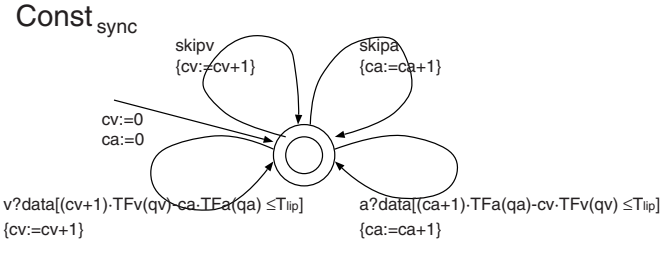


Fig. 6. Constraint for Lip Synchronization

4 Test Case Generation

We derive test sequences from the traffic testing scenario, the quality testing scenario, and the constraint for lip synchronization, explained in Sect. 3. Hereafter, we denote each test sequence as the following sequence $Tseq$.

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

Here, $a.Tseq, Tseq +_p Tseq$ and $Tseq^{*K}$ denote sequential execution of actions, choice between two sequences and iterative execution of the sequence, respectively. $Tseq1 +_p Tseq2$ specifies that $Tseq1$ and $Tseq2$ are executed at probability $1 - p$ and p, respectively. In $Tseq^{*K}$, K denotes the number of iterations. If these values are not specified, default values such as $p = 0.5$ and $K = 100$ are used when executing test sequences.

4.1 Test Case Generation for a Single Object Playback

In the proposed method, the test system observes time at which each frame is displayed in the IUT, takes a sampling of a frame rate every time interval SP, and reports the test result by calculating ratio of samplings with low frame rate below a threshold in a normal distribution of all samplings as explained in Sect. 2.1.

Consequently, from test scenarios S_T and S_Q, we derive test sequences $Test_T$ and $Test_Q$ by adding new sequences for collecting a sampling every time interval SP and for test verdict computation when monitoring period LP expires, and by fixing the probability of choice "+" and the number of iteration "*" in the sequences. We show examples of $Test_T$ and $Test_Q$ in Table 1 and Table 2, respectively.

In Table 1, $Open()$ and $Read()$ denote some file operation primitives. $Packet()$ denotes a primitive to create a packet. In Table 2 $Sampling(x), CalcStatistics$ and $Judgesult$ are primitives which record x as a sampling, calculates statistical information, and calculates test verdict, respectively.

Table 1. Example Test Sequence $Test_T$ for Testing Input Traffic

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

Table 2. Example of Test Sequence $Test_Q$ for Testing Playback Quality

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

Derived test sequences $Test_T$ and $Test_Q$ must be executed in parallel for an IUT. Since each action in those sequences can be executed at any time instance within a specified time interval, the number of possible action sequences for their parallel composition will be so many. Therefore, we do not serialize those sequences. Instead, we make the test system capable of executing multiple sequences in parallel.

4.2 Test Case Generation for Lip-synchronization among Multiple Objects Playback

In the proposed method, a test of the lip-synchronization among multiple playbacks of different media objects is similarly carried out using the statistical method stated in Sect. 2.1, where a time lag of the latest frames between two objects are collected as samplings. From test scenario $Const_{sync}$ in Sect. 3.2, we can derive a test sequence $Test_{sync}$ as shown in Table 3.

Let us denote $Test_v$ and $Test_a$ be test sequences for testing playbacks of video and audio objects, respectively. Here, $Test_v := (Test_{T_v}|||Test_{Q_v})$. With $Test_v$, $Test_a$ and $Test_{sync}$, we finally obtain the following test sequence for testing a given IUT w.r.t. lip synchronization and playback qualities.

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

Table 3. Example Test Sequence $Test_{sync}$ for Lip Synchronization

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

Since the above test sequence contains parallel and synchronization behaviors, we have to implement a test system which provides parallel and synchronous execution of multiple test sequences. Details of a test system are given in Sect. 5.

5 Implementation of Test System

A test system consists of a given IUT and a program which executes test sequences derived in Sect. 4. We call the program as a *tester*. We would like to treat a given IUT as a black box. So, we make the IUT and its tester run in parallel as different processes, and make them communicate with each other.

5.1 Implementation of Tester Program

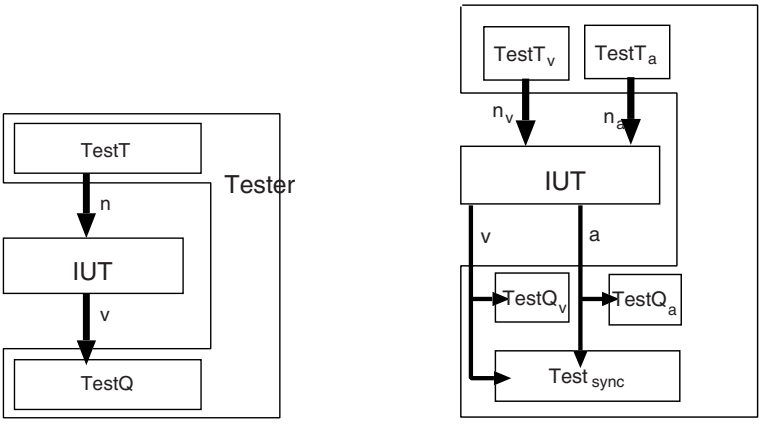
Test sequences which we derived in Sect. 4 have the following characteristics.

(1) each action in a sequence specifies a time range during which it can be executed. The exact execution time is not specified.
(2) a probability is specified in each choice "+" between two sub-sequences. The number of iterative execution of a sub-sequence is also specified.
(3) for testing an object playback, two sequences are specified to be executed in parallel.
(4) for lip-synchronization among multiple object playbacks, a constraint sequence is also specified to be executed synchronously with multiple test sequences for those objects.
(5) statistical calculation primitives such as Sampling(), CalcStatistics(), JudgeResult() are contained in test sequences.

We have implemented a tester program which satisfies the above requirements in Java language. Since test sequences are given in the syntax defined in Sect. 4, first we have developed a parser program using JavaCC. Based on some techniques used in our real-time LOTOS compiler [17], we have implemented parallel execution and synchronization mechanisms for multiple test sequences.

For the above (1), it is desirable to test if the IUT works correctly for all time instances within the specified time range of each action in a given test sequence. However, it is impossible since combination of time instances in multiple actions will be infinite (in case of dense time). So, we have just implemented our tester only to select a time instance at random within the specified range if the next action in the test sequence is an output to the IUT. If the next action is an input from the IUT, the tester measures the clock value based on the current system time and checks whether or not the action has been executed within the specified time range. If so, the tester continues. Otherwise, it stops to report that the test has failed. For the above (2), the tester just selects one of branching sub-sequences based on random numbers, and repeats the specified sub-sequence the specified times, respectively. To improve the validity of the test result in (1), we can increase the number of iterations[1]. For the above (5), we have implemented statistical calculation primitives in Java, according to techniques explained in Sect. 2.1.

[1] In [4, 12], time instances near the borders of time ranges are intensively selected for tests. We can also adopt the similar technique to improve the test validity.



(a) Single Object Playback (b) Lip Synchronization between Two Objects

Fig. 7. Test Environment

5.2 Construction of Test Environment

The tester must be able to observe the time when each frame is output in the IUT. Here, we assume that a certain event occurs and the time is notified to the tester via gate v when a frame is output in the IUT. The test sequence for a single object playback derived in Sect. 4 contains parallel execution of two sub test sequences: $Test_T$ and $Test_Q$. In this case, the whole test system is executed as shown in Fig. 7 (a). The test sequence for lip synchronization between two objects contains four sub-test sequences (two for each) and one constraint $Test_{sync}$. These sequences are executed as shown in Fig. 7 (b). In order to reduce the influence coming from the test environment such as additional delay and jitter, we execute both the tester and the IUT in the same computer or in the separate computers in the LAN where the influence of delay is smaller.

6 Experimental Results

With the test system explained in Sect. 5, we have executed test sequences and a given IUT and measured distribution of actual frame rates.

Our test purpose is to investigate whether a given IUT works satisfying QoS functions (specified by ideal frame rate fps', maximum tolerance acceptable ϵ, and reliance level r of area below ϵ in the standard normal distribution) when the IUT receives packets according to the specified traffic pattern (given by packet loss ratio $Loss$, maximum jitter JT in packet arrival time, and maximum burst length $Burst$).

Two kinds of programs have been used as IUT in our experiments: IUT_a which decodes and displays frames immediately after receiving packets (not regulate frame rates); and IUT_b which receives packets and stores them in a given buffer for absorbing jitters of the packet arrival time in order to display frames at the specified frame rate. We have implemented those IUTs in JMF2.1.1c(Java Media Framework) [18].

Table 4. Parameters used in Experiments

	JT	$Loss$	$Burst$(bit)
Exp1	30ms	0.0	0
Exp2	10ms	0.0	0
Exp3	30ms	0.1	0
Exp4	30ms	0.1	58800
Exp5	30ms	0.2	58800
Exp6	30ms	0.0	0
Exp7	30ms	0.0	58800
Exp8	30ms	0.1	0
Exp9	30ms	0.2	0
Exp10	30ms	0.2	58800

Table 5. Test Results

	average	standard deviation	ϵ	ratio under ϵ	test result
Exp1	24.6	2.65	25.0	4.2%	failed
Exp2	24.8	1.71	25.0	0.0%	passed
Exp3	22.0	3.50	22.5	12.3%	failed
Exp4	19.6	4.36	22.5	21.0%	failed
Exp5	20.1	5.03	20.0	20.3%	failed
Exp6	25.0	—	25.0	0.0%	passed
Exp7	25.0	—	22.5	0.0%	passed
Exp8	22.5	4.23	18.0	1.8%	passed
Exp9	20.0	4.86	16.0	16.6%	failed
Exp10	19.7	3.80	16.0	20.3%	failed

With the test system explained in Sect.5, by changing the above parameters given to IUT_a and IUT_b, we have examined the quality of playback mechanism in those IUTs (Exp1 to Exp 10). The list of parameters is shown in Table 4.

We have used a motion JPEG stream (320×240pixels, 25fps, the size of each frame is 5.88kbit) in all of our experiments and have applied SP=1s, MP=60s, LP=1800s and TP=40ms (i.e, each frame is transmitted by one packet) to the IUT. In the experiments we have measured the distribution of frame rates, by which we can decide whether the IUT passes the test or not.

First, we have executed Exp1 to Exp5 for the IUT_a. In the Exp1 and Exp2 we have investigated the difference in the distribution of the frame rate by changing the value of JT. From Exp3 to Exp5 we have executed test sequences which contains jitters, packet losses (Exp3, Exp4) and burst transmission (Exp5) to the IUT_a. The distribution of samplings for these experiments are shown in Fig. 8 and Fig. 9. These figures show that samplings are distributed in the wide range where the center is around $fps' = 25 \times (1 - Loss)$, and the distribution range of samplings is influenced by each of jitters, packet losses, and burst transmission.

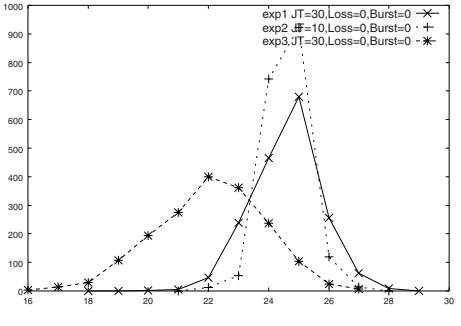


Fig. 8. Distribution for Experiments 1,2,3

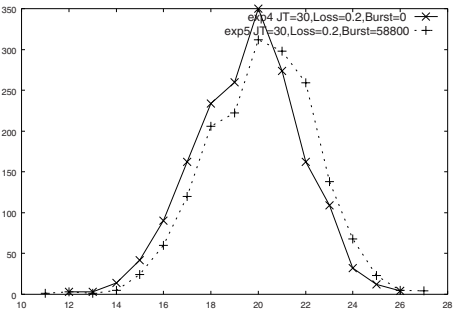


Fig. 9. Distribution for Experiments 4,5

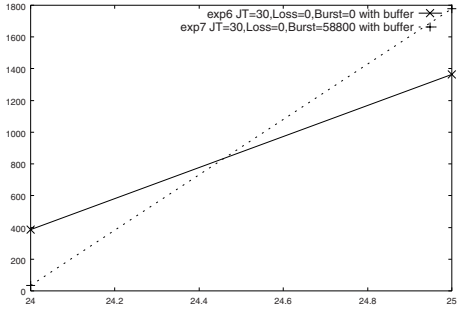


Fig. 10. Distribution for Experiments 6,7

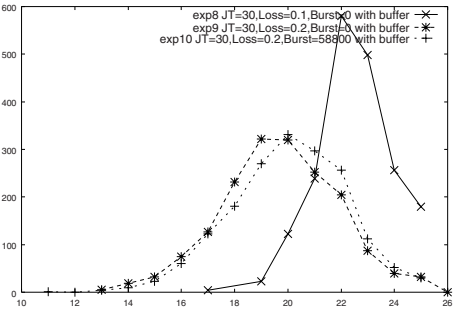


Fig. 11. Distribution for Experiments 8,9,10

Then, we have executed Exp6 to Exp10 for the IUT_b which has a buffer and can control the frame rate. In Exp6 and Exp7, we have executed the test sequence only containing jitters, and the test sequence containing both jitters and burst transmission, respectively. In Exp8 and Exp9, we have executed the test sequence containing jitters and packet losses, where the value of $Loss$ is bigger in Exp9 than in Exp8. In the last experiment Exp10, the test sequence contain all factors: jitters, packet losses and burst transmission. The distribution of these experiments are shown in Fig. 10 and Fig. 11. In Fig. 10, we see that all of samplings concentrate in 25(fps') or 24($fps' - 1$) because Jitters were absorbed. On the other hand, if we compare Exp8 with Exp9 in Fig. 11, we see that the distributed range of samplings is wider when the value of $Loss$ is higher. We think that this is because we specify $Loss$ as the maximum packet loss ratio for period MP=60s, but the actual packet loss ratio measured during short period (e.g., SP) may be more or less than the value of $Loss$. In Fig. 11 comparing Exp8 with Exp9, we see that the influence due to burst transmission is small.

Next, with the method explained in Sect.2.1 , we have decided whether tests pass or not. We calculated the average and standard deviation of samplings for each experiment by supposing their normal distribution. Here, for example we set $maximum\ tolerance\ acceptable\ \epsilon$ as $fps' \pm 20\%$ and $reliance\ level\ r$ as 2%. The list of the average, the standard deviation, the ratio of area under ϵ and the test result for all experiments are shown in table 5. According to the table, we see that the IUT_a passes the test in the condition which JT was within 10ms and $Loss$ was close to 0 (Exp2). About IUT_b, it passes the test when JT was within 30ms and Loss was within 0.1 (Exp6, 7, 8).

In order to evaluate the validity of our assumption which the distribution of frame rates follow the normal distribution, we calculated the proportion of the samplings under (or upper when $\epsilon > fps'$) ϵ to the whole of samplings and the proportion of $C\,[-\infty, (\epsilon - \mu)/s]$ to the whole area in the normal distribution. We show the result in Table 6. Comparing the normal distribution and the distribution of the actual samplings, we see that the proportion in the normal distribution is in most cases bigger than in the actual samplings for each ϵ. If the test passes on the proposed method, then the actual proportion under ϵ was not bigger than reliance level r. So we think our assumption is valid enough.

Table 6. Comparison between Normal Distribution and Distribution of Actual Samplings

	proportion under(upper) ϵ	
frame rate	normal distribution	actual samplings
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

7 Conclusion

In this paper, we proposed a test method for QoS functions in distributed multi-media systems. In the proposed method, using timed EFSM model, we describe a test scenario which specifies input flow characteristics and play-back quality to be realized for the flow, and generate test sequences from the scenario. Using the generated test sequences, we can statistically test whether a given IUT realizes certain quality for a given input flow. Through experiments with our test system and sample IUTs, it is confirmed that the proposed test method can efficiently test given IUTs depending on target network environments and required playback quality.

As part of future work, we would like to test various IUTs where some QoS control mechanisms (lip synchronization, prioritized media scaling, and so on) are implemented, assuming various types of network traffic. Through such experiments, applicability of our method should be clarified.

References

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. of 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. of 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. of the 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. of 1998 Int'l. Symp. on Software Engineering for Parallel and Distributed Systems (PDSE98)* (1998).
13. 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).
14. 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).
15. W3C: "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", http://www.w3c.org/TR/REC-smil/
16. Yamada, M., Mori. T., Fukada, A., Nakata, A. and Higashino, T.: "A Method for Functional Testing of Media Synchronization Protocols", *Proc. of the 16th Int'l. Conf. on Information Networking (ICOIN-16)* (2002).
17. 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).
18. Sun Microsystems: "Java Media Framework Home Page", http://java.sun.com/products/java-media/jmf/