A Method for QoS Functional Testing in Distributed Multi-media Systems

Tao Sun, Keiichi Yasumoto, and Masaaki Mori

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 realized or not in a client side program according to the qualities designated in advance, and/or whether a time lag of playback among multiple media objects is controlled within the specified time interval. The proposed technique, we give specifications of an input flow and the playback quality to be realized (e.g., the range of fluctuation of frame rates) on playback behaviors in timed EFSMs. From the specifications, we generate test sequences to test whether a given IUT (implementation under test) realizes the QoS in accordance with the specifications. 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 using Java and JMF, and applied it to a video playback system.

Index Terms—functional testing, Java, multi-media systems, QoS

I. INTRODUCTION

With rapid popularization of multi-media communication systems, it is in urgent need to establish a method for QoS testing of real-time systems which treat various multi-media objects. Traditional software testing methods which have succeeded in protocol engineering, focus mainly on the correctness of input/output correspondences [9]. Therefore those methods could not be directly applied to QoS testing such as video/audio playback timings. Essentially important test problems of multi-media objects 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, 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 for multi-media test. Suppose that a specification for a video playback system describes that a video frame is drawn exactly every 33 msec plus/minus 5msec. In general, when a given IUT does not satisfy such a specification a little (e.g., only a frame were delayed 10msec), it is not 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.

There exist some works on multi-media testing such as testing of multi-media transmission system [4], the quality of multi-media contents [5] and interoperability and performance testing of distributed systems [12]. However they do not deal with testing on temporal relations of input/output actions. A few works, which deal with temporal relations in multi-media systems explicitly, have been reported [3], [10]. These works 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, [14] proposes a method for functional testing of media synchronization protocols using 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.

In this paper, we suppose a distributed multi-media system consisting of servers and clients connected on a network, and present a method for testing the QoS of the frame rates and lip-synchronization on playbacks of media objects. In this method, we specify the basic playback behaviors of media objects by timed EFSM (similar to timed automata [1]), and designate the traffic characteristics like jitter/packet loss ratio and the playback 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 specifications of objects by using the constraint oriented description style [11]. From the specifications, we generate test sequences to test whether a given IUT (implementation under test) realizes the QoS in accordance with the specifications.

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 samplings with low quality (e.g., frame rates less than a thresh-
old) below a specified threshold in a normal distribution of all samplings. We have implemented a test system for test sequence execution using Java and JMF, and applied it to a video playback system.

II. TARGET SYSTEM AND OUTLINE OF QoS FUNCTIONAL TESTING

In this paper, we propose a method for QoS functional testing in distributed multi-media systems as shown in Fig. 1 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 input flow.

In the proposed method, similarly to the existing test method, we suppose that when a specification and the implementation (IUT: implementation under test) of a system are given, we test whether the IUT works correctly or not with satisfying the specification.

A. Outline of Existing Test Methods

In the existing real-time testing methods as in [6], [7], testing is carried out by giving an input to the IUT at appropriate time, and by observing and testing if an output action is executed at appropriate time satisfying the constraints given in the specification. 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.

[3] has proposed a testing method for some relations of starting/ending time among multiple media objects specified in SMIL [13]. In this research, some statistical approach has been made by collecting actual time differences as samplings by executing objects in IUT many times, and by calculating a normal distribution. Testing result is produced by checking the ratio of an area below some threshold in the distribution.

In this paper, we apply the statistical technique as in [3] to testing of 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 multi-media system.

B. 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 specification 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. Hereafter, we call the number of frames output in a unit of time as frame rate.

In general, when we measure the frame rate 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, in IUT, we sample a frame rate every $SP$ time for a sufficiently long time interval (denoted as $MP$), calculate statistic information from distribution of those samplings, and judge from the information whether or not the IUT is correctly implemented.

We suppose that a traffic specification which characterizes an input flow, and a quality specification which represents expected playback quality of an object are given. From those specifications, we generate test cases (a set of test sequences) in the following procedure:

- each test sequence transmits packets to IUT at time within the allowable time range considering packet arrival jitters and bursts specified in the traffic specification.
- each test sequence measures actual packet loss ratio in IUT for time interval $MP$, and calculates the ideal frame rate $fps'$ as explained in Sect. II-C.
each test sequence measures average frame rate every time interval $SP$ by observing output from IUT, and keeps it as a sampling. The test sequence collects samplings for time interval $MP$, and calculate the average value and the standard deviation from those samplings. The test sequence judges whether IUT is correctly implemented or not, based on those calculated statistical values and maximum tolerance acceptable denoted by $\epsilon$ 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 $\mu$ and the standard deviation $s$ derived from the samplings kept during time interval $MP$.

2) apply normalization expression $z = (x - \mu)/s$ to $\epsilon$, and calculate area $C$ that is an integral of $(-\infty, (x - \mu)/s]$ in the standard normal distribution table (see Fig. 3).

3) When the value of $C$ is small enough, we think that IUT passes the test since we can conclude that frame rates below $\epsilon$ are rare. On the other hand, as the value becomes large (closer to 0.5), we think that IUT does not pass the test since frame rates below $\epsilon$ appear rather frequently.

In general, as shown in Fig. 3, we expect that the average frame rate $\mu$ measured during time interval $MP$ 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. We have to consider how to correct such mismatching.

C. 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 playback 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.

$$fps' = fps \cdot (1 - \alpha \cdot LossRatio - \beta)$$

Here, $fps$ and $LossRatio$ denote the originally encoded frame rate and the packet loss ratio, respectively. $\alpha$ denotes the ratio of how much each packet’s loss causes the frame loss. Here, $alpha$ will be 1.0 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, $\alpha$ becomes more than 1.0. $\beta$ is another factor other than flow characteristics at a client computer such as CPU load. For the sake of simplicity, we suppose that $\beta = 0$ in this paper.

D. Specifications of Multi-Media Systems

In this paper, we describe behavior of playbacks of each object as a timed EFSM where variables and guard expressions with those variables (i.e., execution condition of transitions) can be used in timed automata [1].

A timed EFSM is given as $M = \langle S, A, C, V, \delta, s_0 \rangle$. Here, $S = \{s_0, s_1, \ldots, s_n\}$ is a finite set of states, $A$ is a set of actions, $C$ is a set of clock variables, $V$ is a set of variables, $\delta : S \times A \times Guard \times Def \rightarrow S \times V$ is a transition function, and $s_0$ is the initial state. Each action of $A$ belongs to the set $G \times IO$ where $G$ is a set of gates (interaction points to an external environment) and $IO$ is a set of inputs/outputs (from/to a gate) denoted by $?x$ and $!f(x)$. For example, $g ? x$ represents inputting a value from gate $g$ and storing it in variable $x$, and $g ! f(x)$ represents outputting the value of $f(x)$ to gate $g$. $Guard$ is a set of execution conditions where each condition must be a linear inequality such as $[1 \leq c \ and \ c \leq 5]$ with clock variables, variables and constants. $Def$ is a set of value assignments to variables (including reset of clock variables) denoted as $\langle c := 0 \rangle$.

Each action $a \in A$ is denoted by a tuple of input/output action on a gate, a guard expression and assignments like $gate ? x \ E [Guard(x, y, \ldots)] \{ x := 10, c := 0 \}$.

1) Specification of Object Playback Behavior: As object playback behavior, we have to specify the quality that the object should be played back for a given flow with some characteristics.

For the purpose, first we specify the traffic specification of each flow which the IUT receives so that some flexibility is allowed for the receiving time of each packet. Then, we use some parameters: average receiving rate $AVRT$, the maximum burst length $Burst$, the maximum packet loss ratio $Loss$, and the jitter in packet arrival time $JT$. We assume the size of all packets are the same and denoted as $PrtSz$. Traffic specification $Spec_T$ can be described as shown in Fig. 4.

![Fig. 4. Traffic Specification](image-url)
In Fig. 4, \( clk \) is a clock variable, and \( TP(q) \) is a time interval at which the server transmits each packet to realize playback quality \( q \) (note that \( TP(q) = Pct Sz / AvRT \)).

There are three branches in the traffic specification of Fig. 4. The sequence in the center of Fig. 4 represents receiving each packet every interval plus minus jitter (\( TP(q) \pm JT \)). Here, \( ofst \) is the variable keeping the offset of \( clk \) from the \( ±0 \) point of the last interval. The sequence in the left side of Fig. 4 represents a packet loss when no packet arrives before the clock exceeds \( TP(q) + JT \). Here, we specify that packets can be lost as long as the current packet loss ratio is less than \( Loss \), where variables \( ln \) and \( pn \) are used as the numbers of lost packets and arrived packets, respectively. The sequence in the right side of Fig. 4 represents the burst transmission of packets where multiple packets arrive in a short time interval as long as the total size of packets is less than \( Burst \).

Similarly, we specify the quality specification of the IUT to give allowable behavior in displaying frames for the given traffic specification. Quality specification \( Spec_Q \) can be described as shown in Fig. 5.

Here, \( TF(q) \) denotes the time interval between subsequent frames when the object is played back with quality \( q \). \( FJT \) is an allowable jitter for the time interval. In the quality specification in Fig. 5, the sequence in the left side represents displaying of each frame in time interval \( TP(q) \pm FJT \). The other sequence (right side) represents skipping to display a frame when a frame is delayed or lost. We restrict that skipping action can be executed as long as the current ratio of skipped frames (denoted as \( (sn + 1)/(sn + vn + 1) \)) is less than \( \alpha \cdot Loss \) as we explained in Sect. II-C, where \( sn \) and \( vn \) denote the numbers of skipped frames and displayed frames, respectively.

2) Specification of Lip-Synchronization among Objects: We specify lip-synchronization among objects as a constraint between object playback behaviors using the constraint oriented description style [11]. For example, let us denote \( Spec_V \) and \( Spec_A \) as video playback behavior and audio playback behavior, respectively. Here, those behaviors are the same as \( Spec_Q \) in Fig. 5 except that they use gates \( v \) and \( a \), respectively, instead of \( v \) in \( Spec_Q \). If the maximum time skew between video and audio must be within 80msec, we describe constraint \( Const_s \) as shown in Fig. 6.

Here, we denote \( c_v \) and \( c_a \) to be the sequence numbers of frames, and \( P_v \) and \( P_a \) to be the time intervals for displaying each frame, respectively.

III. GENERATION OF TEST SEQUENCES

Test sequences are generated from the traffic specification and the quality specification stated in Sect. II-D.

Hereafter, we denote each test sequence as the following sequence \( Tseq \).

\[
Tseq := a.Tseq[Tseq + Tseq][Tseq][Tseq^*]
\]

Here, \( a.Tseq, Tseq + Tseq \) and \( Tseq^* \) denote sequential execution of actions, choice between two sequences and iterative execution of the sequence, respectively.

To represent test sequences more concretely, in each choice of test sequences, we can specify a probability like \( Tseq1 \cdot p \) \( Tseq2 \), which means that \( Tseq1 \) and \( Tseq2 \) are executed at probability \( 1 - p \) and \( p \), respectively. Similarly, in each iteration, we specify the number of iterations like \( Tseq^*(K) \).

A. Test Case Generation for a Single Object

From a given specification \( Spec_T \), we derive a sequence \( Test_T \) that transmits packets to an IUT. In this \( Test_T \), we vary input timing of packets to IUTs and loss ratio of packets within a range represented in \( Spec_T \), and also designate enough iteration times to increase the reliability of test.

Test sequence \( Test_Q \) is derived from specification \( Spec_Q \) which specifies playback quality of an object. \( Test_Q \) observes time at which each frame is drawn in IUT, and takes a sampling of frame rate every time interval \( SP \). Finally it 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. II-B.

The followings are a procedure for generation of actual test sequences.

- derive action sequences by replacing each input action (output action) with the corresponding output action (input action) in specifications \( Spec_T \) and \( Spec_Q \).
- add new sequences for collecting a sample every time interval \( SP \) and for test verdict computation at the expiration time of monitor period \( MP \) to the sequence derived from \( Spec_Q \). Also add some complementary actions and/or assignments to the derived sequences.
- derive Test sequences \( Test_T \) and \( Test_Q \) by fixing the probability of choice “+” and the number of iteration “*” in the above sequences.
- specify relations between \( Test_T \) and \( Test_Q \) (e.g., the duration time from the time of receiving the first packet to the beginning of playback of the first frame) if necessary.
For example, the following test sequences are derived from the specification of Sect. II-D.

\[
\text{Test}_{T} := \{ fp := \text{Open}(\text{file}) \}.
\]

\[
\text{clk} := 0, \text{Lost} := 0.0, ln := pn := csn := \text{burst} := 0.0, \{ \{ \text{pt} := \text{Packet}(\text{Read}(fp), csn) \} \}
\]

\[
\text{n} \times \text{pt} \{ \text{TP}(q) - \text{JT} \leq \text{clk} + \text{ofst} \leq \text{TP}(q) + \text{JT} \}
\]

\[
\{ \text{pn} := \text{pn} + 1, \text{ofst} := \text{ofst} + \text{clk} - \text{TP}(q), \text{clk} := 0 \}
\]

\[
+ 0.2
\]

\[
\{ \{ \text{TP}(q) + \text{JT} < \text{clk} + \text{ofst} \text{ and } (\text{ln} + 1)/(\text{pn} + \text{ln} + 1) \leq \text{Loss} \}
\]

\[
\{ \text{ln} := \text{ln} + 1, \text{ofst} := \text{ofst} + \text{clk} - \text{TP}(q), \text{clk} := 0 \}
\]

\[
+ 0.5
\]

\[
\{ \text{ofst} := \text{ofst} + \text{clk} - \text{TP}(q), \text{clk} := 0 \}
\]

\[
* (\text{MP}/\text{TP}(q))
\]

\[
\text{Test}_{Q} := \{ \{ \text{clk} := 0, \text{ofst} := 0, \text{vn} := 0, \text{sn} := 0 \}. \}
\]

\[
(\{ \text{vn} := \text{vn} + 1, \text{ofst} := \text{ofst} + \text{clk} - \text{TF}(q), \text{clk} := 0 \}
\]

\[
+ \text{skip}(\{ \text{ofst} < \text{ofst} + \text{clk} - \text{TF}(q), \text{clk} := 0 \}) \times (\text{SN}/\text{TF}(q))
\]

\[
\text{Sampling}(\{ \text{vn} := \text{vn} + 1, \text{sn} := \text{sn} + 1, \text{ofst} := \text{ofst} + \text{clk} - \text{TF}(q), \text{clk} := 0 \}) \times (\text{SN}/\text{TF}(q))
\]

\[
* (\text{MP}/\text{SN})
\]

\[
\text{CalcStatistics}(\text{clk}, \text{JudgeResult})
\]

In the above specification, \(\text{Open}()\) and \(\text{Read}()\) denote some file operation primitives. \(\text{Packet}()\) denotes a primitive to create a packet. \(\text{vn}\) and \(\text{Sampling}(\text{x})\) denote the number of frames displayed in the current interval of \(\text{SN}\), and the primitive function to record \(\text{x}\) as a sampling, respectively. \(\text{CalcStatistics}\) and \(\text{JudgeResult}\) are sub-test sequences for statistical calculation of samples and for test verdict computation from statistical information, respectively, and are executed by procedures implemented as libraries.

Due to space limitation, we omit details about automatic derivation algorithm for test sequences, however it is relatively easy to implement.

Derived test sequences \(\text{Test}_{T}\) and \(\text{Test}_{Q}\) must be executed in parallel for an IUT. There are many kinds of action combinations among them since each action can be executed at any time instance within a specified time interval. So, test sequences would become quite large if we make product machines from them. In this paper, we represent test sequences in short as a compact form as a parallel composition of test sequences with operator \(\big|\big|\) of LOTOS[8], and adopt a method such that the execution order of actions and their execution time are decided dynamically when they are executed. That is, we let our test system (a tester, in short) have a facility of parallel processing.

Hereafter, we denote a test sequence to test an object \(\text{obj}_{ij}\) as \(\text{Test}_{i} := (\text{Test}_{T} \big|\big| \text{Test}_{Q})\).
implementation only supports a simple synchronization like 
\[T \in T_{\text{IUT}} \times \ldots \times T_{\text{IUT}} \times T_{\text{IUT}} \times \ldots \times T_{\text{IUT}}\] 

For the above (1), we have to test that 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 speci-
ified range if the next action in the test sequence is an output to 
IUT. If the next action is an input from 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 speci-
ified 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 test 
coverage in (1), we can increase the number of iterations.

We have carried out some experiment using a prototype of 
IUT implemented with JMF and several MPEG movies en-
coded at 30 frames/sec whose transmission rate is about 800 
Kbps. As a result, we have confirmed that the tester has enough 
performance to test such media objects in real-time w.r.t. frame 
rates and lip-synchronization quality.

B. Construction of Test Environment

When we want to test the playback quality of a single ob-
ject, the tester executes two threads corresponding to \(T_{\text{IUT}}\) 
and \(T_{\text{Q}}\) respectively in parallel. Here, the tester can use a 
different protocol such as UDP and RTP for each gate to com-
municate with IUT.

When the tester executes \(T_{\text{IUT}}\), it reads data from 
video/audio files with JMF facilities, and sends those data to 
IUT as packets. At the same time, the thread for \(T_{\text{Q}}\) receives 
the event indicating that a frame has been displayed (played) at 
IUT, and keeps the received time. Using this time, the tester 
tests the playback quality in steps explained in Sect. II-B, and 
reports the test result. Due to space restriction, we omit to ex-
plain the case of lip-synchronization among multiple objects, 
but it can be similarly tested.

In general, IUT outputs frames to display devices. So, in or-
der to make the tester know the time at which each frame is 
output, we may need to modify the IUT program slightly. We 
assume that our proposed method will be useful to adjust pa-
rameters such as the buffer size, media codecs, etc in a client 
program of a multi-media distributed system. In such a con-
text, we believe that slight modification to IUT may not be a 
problem.

V. Conclusions

In this paper, we proposed a test method for QoS functions 
in distributed multi-media systems. In the proposed method, 
we give a specification describing input flow characteristics and 
play-back quality to be realized for the flow, and generate test 
sequences automatically from the specification. Using the gen-
erated test cases, we can statistically test whether a given IUT 
realizes certain quality for a given input flow in the specifi-
cation.

For a case of parallel playback of objects among which inter-
media synchronization is specified, we describe a constraint be-
havior as a timed EFMS and generate the whole test sequence 
as a synchronous composition of test sequences for object play-
backs and the constraint for inter-media synchronization.

We have implemented a tester in Java language that executes 
test sequences in real-time. The tester makes it easy to test 
whether playback mechanisms of media objects are correctly 
realized in IUTs.

Another important and interesting problem is to develop a 
test method for playback qualities in various metrics other than 
frame rates, and for dependencies among objects other than 
inter-media synchronization such as priorities among objects.

REFERENCES

Multimedia Applications”, Proc. of 2nd Workshop on Protocols for Multi-
Test Cases for a Timed I/O Automaton Model”, Proc. 12th IFIP Workshop 
LOTOS - A Formal Description Technique Based on the Temporal Or-
Intl. Symp. on Software Engineering for Parallel and Distributed Systems 
Distributed Systems Design and Verification", Theoretical Computer Sci-
Distributed Systems - State of the Art and Beyond (Invited Paper)”, Test-
Specification”, http://www.w3c.org/TR/REC-smil/ 
(2002).