On Distributed Multi-Point Concurrent Test System and Its Implementation

Hao Luo and Huaxin Zeng

School of Information Science and Technology,
Southwest Jiaotong University, Chengdu 610031, P.R. China
jackfrued@gmail.com, huaxinzeng1@yahoo.com.cn

Abstract. As the rapid expansion of the Internet, novel network applications are constantly emerging accompanied by the increasingly growing of the complexity for network protocols. As a result, IP routers are becoming more and more important in today's Internet. However, test methodology and test systems for IP routers often fall behind the state of the art. Based on the deficiency of current stand-alone test systems, this paper analyzes the necessity for distributed test architecture and introduces a new test system called Distributed Multi-point Concurrent Test System (DMC-TS) which can mirror real-world networks in the test experiments and conduct conformance testing, performance testing and interoperability testing for the routers under test. Key issues for the implementation of DMC-TS are discussed, especially on test synchronization problem. Consequently, two types of test synchronization problem are pointed out with the corresponding solutions. Some experiments are carried out to illustrate the feasibility and practicability of DMC-TS.

Keywords: distributed, concurrent test manager, test agent, synchronization.

1 Introduction

With world-wide applications of the Internet, IP routers are playing more and more important roles in computer networks. Whether these routers can reliably work or not has direct impact on today's Internet; also, the Quality of Service (QoS) as well as customer satisfaction with service provider networks are directly influenced by the performance of these routers. Consequently, product testing for IP routers becomes the prerequisite to ensure their ability to process routing information and to forward data and thus must be carried out before deployment in mission critical networks.

Since routing protocol is one of the most important factors for routers, testing of routing protocols has drawn great interest of academic and industrial communities worldwide since late 1970's. Pioneer work on conformance testing resulted in a framework on conformance testing as specified in ISO/IEC standard 9646 [1]. The document provided a framework on methodology for conformance testing of OSI (Open System Interconnection) protocols. As a result, four abstract test methods for end systems and two test methods for relay systems were proposed. In addition, a semi-formal test definition language called TTCN (Tree and Tabular Combined Notation) was also defined which put the history of conformance testing using non-formal methods such as natural languages to an end.

Two test methods for relay systems defined in ISO/IEC standard 9646 are Loopback Test Method (LTM) and Transverse Test Method (TTM). LTM executes test procedures by sending test data to the router under test and then receiving the loop back messages from the same port. It had been the most popular test method for relay systems until the emergence of TTM in early 1980's. TTM enables the testing of routers through a pair of ports and thus more applicable in practice and superior over LTM. At present, TTM is still the most popular test method for routers as it is simple to apply and capable to test the behavior of a router between a pair of ports with predefined test data. Generally, both of the two test methods can be implemented by stand-alone systems, however only one port or a pair of ports of the routers can be tested simultaneously.

The rapid development of network technology makes the port number and the forwarding rate of a single port increase constantly. Nowadays the core routers in the backbone of the Internet generally have dozens or even hundreds of ports and the throughput of each port have reached the magnitude of Gbps or more. Meanwhile the demands on security, robustness as well as QoS capability for IP routers are being put forward. By and large, traditional test systems based on LTM and TTM are no longer feasible and realistic as the routers under test have become increasingly complex and the related protocols that need to be tested are widely distributed. It's necessary to update the traditional test methodology and to develop new test system that can apply test control and observation to all ports of the routers under test so as to mirror real-world networks in the test experiments.

The deficiency of current stand-alone test systems as stated above stimulates the activity of research work on Distributed Multi-point Concurrent Test System (DMC-TS) and the relevant issues presented in this paper and the paper is organized in the following manner. The necessity for distributed test architecture and corresponding system is discussed in section 2. In section 3, the prototype of DMC-TS is presented with the detailed discussion on its implementation. In section 4, we deal with test synchronization problem which is very common in distributed test architecture. Some experiments are carried out to illustrate our current work as well as the feasibility and practicability of DMC-TS in section 5. Finally, conclusions are drawn in section 6, summarizing our contributions and discussing some future research issues.

2 The Necessity for Distributed Test System

IP routers have been traditionally perceived as a network device with a 3-layer architecture adopting the in-band signaling point of view. However, this point of view cannot explain why the routing information protocols such as RIP and BGP and the network management protocol such as SNMP in a router are actually implemented on the top of a transport protocol (TCP or UDP). This contradiction can be avoided by adopting out-band signaling concept [2]. At this point of view, a router is composed of two sets of protocol stacks, one for user data transfer (up to network layer, usually being referred to as User Plane, or U-Plane for short), and the other for constructing a network capable of transporting data traffic (up to application layer, usually being referred to as Control Plane, or C-Plane for short) [3]. However, current performance testing for IP routers often takes only U-Plane into account but ignores the influence of C-Plane. Indeed, the activities of C-Plane have crucial impact on the performance of U-Plane. As a result, performance testing that considers the interaction between U-Plane and C-Plane usually yields more accurate and reliable results than the ones that only take the performance of U-Plane into account, since the former provides the routers under test with the circumstances that is more like the real-world networks when putting these routers into practice.

As stated previously, the rapid development of network technology resulted in the growing complexity of the protocols that run inside IP routers. The running of some protocols often involves a number of communication ports simultaneously. For example, if test operator wants to duplicate real-world conditions for the router under test, routing simulation must be done beforehand. Therefore a large routing table must be created automatically by the test system through multiple communication ports in limited time. Also, normal routing information including routing requests and updates must be transmitted from the test system to the router under test periodically or randomly during the test process as well as flapping routes for simulation of abnormal conditions. In this case, traditional test system based on LTM and TTM that involves only one port or a pair of ports becomes impotent. With distributed test system and concurrent testing approach, all possible conditions can be simulated by means of injecting all kinds of test data including routing information, network management queries and so on into the router under test from a number of test components in a controllable manner. This is also the main benefit for developing a distributed test system.

Another important reason for developing a distributed test system is modeling and generation of reasonable workload. The performance of an IP router depends on the characteristics of the workload it must server. As a result, what kind of test traffic should be used becomes an important issue that must be considered, especially when carrying out performance testing or QoS testing. In order to acquire the reliable test results, it is ideal to replicate Internet traffic or to use a workload model that can reveal various characteristics of Internet's traffic. However, it is very difficult to replicate Internet traffic or to put forward a feasible workload model due to the complexity and uncertainty of the network system. Although some research work has pointed out the "self-similar" nature of network traffic [4], it is hard to make the best of this model in test practice.

However, by making full use of distributed test architecture, the problem of workload modeling and generation can be expediently resolved in a simple manner by generating packet streams with different source/destination addresses, different length, different types of service and different priorities from a number of test components. In addition, by superposition of the packet streams from different test components, workload that fits to a specific arriving process and a special traffic pattern can be generated and in consequence workload that a router would serve when being deployed in real-world networks can be simulated.

Last but not least, the scalability and extensibility of the test system are considered. Although some commercial test systems for IP routers based on stand-alone test systems can be configured with a number of test ports so that they can also carry out extensive testing for routers under test, the scalability and extensibility is inevitably limited. Employing a distributed test system is the key factor to surmount the problem of scalability and extensibility in testing. As new test components can be added up into existing test system dynamically as needed, the capability of the test system can therefore be promoted without making great changes to existing system. In addition, a distributed test system can make the best of all potential resources and thus provides the test system with high performance/price ratio.

At present, researchers have reached an agreement of developing test systems based on distributed architecture [5, 6], but the prerequisite is to design a reasonable architecture for distributed testing and actualize an efficient approach for concurrent test control. Also some other issues on the implementation must be solved beforehand. These issues are the major motivation of the research work presented in this paper.

3 The Prototype and Implementation of DMC-TS

In order to duplicate real-world conditions for routers under test and satisfy the requirements for scalability and extensibility of the test system itself, a new test system called Distributed Multi-point Concurrent Test System (DMC-TS) is introduced in this paper, which is composed of one Concurrent Test Manager (CTM) and several Test Agents (TAs). The CTM itself does not have the test ability but fulfills the specified tests by sending test requests and control requests to one or more TA(s). Each TA listens to the test and control requests and then executes the test procedures according the requests from CTM by injecting test data into the Router Under Test (RUT) as well as observing the external behavior of the RUT. A test verdict will be generated when the assigned test is accomplished and it will be sent back to CTM as a response to the former test request.

The test and control requests sending from CTM to TAs during the test process are non-blocking ones, which means that CTM is able to send requests successively to several TAs without blocking for the responses of the former requests. Also, there are some requests that need to be sent to several TAs at the same time and this can be done by connecting CTM and TAs with a fast switch and communicating through multicast or broadcast. As several TAs are involved, a number of points of control and observation (PCOs) can be established and the RUT can be controlled and observed exhaustively from various PCOs.

Both of the functional testing (such as conformance testing) and the non-functional testing (such as performance testing) can be carried out with DMC-TS using the test configuration as illustrated in Fig. 1. Under this configuration, each TA can represent an adjacent router or a subnet connected to the RUT and thus background traffic, routing information update messages and network management queries can be generated by these TAs through the test ports connected to the RUT during the test process. In this way, real-world conditions can be simulated for the RUT so that the test results of conformance testing and performance testing can be more accurate and reliable. Of course, more than one RUT can be involved in the test. These routers can form a system that can be treated as a whole and usually be referred to as the System Under Test (SUT).

Experience has also shown that two implementations of the same protocol are not necessarily interoperable even both of the implementations have previously passed the conformance testing. Therefore, sometimes there is a need for arbitration. A complete test system for IP routers should encompass interoperability testing as well as conformance testing and performance testing and this can be done with DMC-TS by employing a special TA (as the shadowed TA in Fig. 1) with the ability to fulfill the passive monitoring or active intervention of the communication process between two RUTs as well as tracing the potential faults.

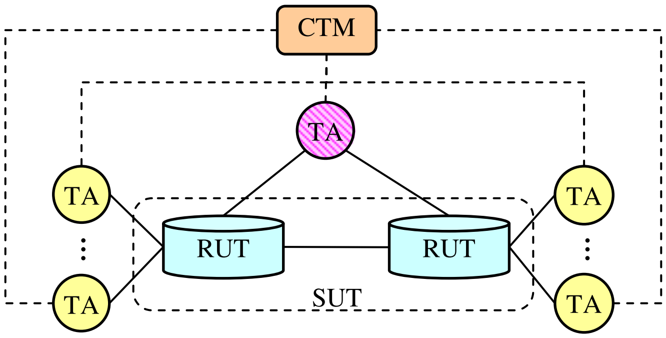

Fig. 1. Conformance and performance testing configuration of the DMC-TS

In the architecture of DMC-TS, TA is loosely coupled with CTM (using dotted lines in Fig. 1), i.e. each TA can perform some simple test without the participation of CTM. In this way, we can make TA into a stand-alone portable tester in the future. However, the most important feature of TA is to play the role of a test component in DMC-TS. Therefore, on one hand the number of TAs can be increased in order to promote the test ability of the system; on the other hand the connection between the RUTs and TAs can be modified for the purpose of establishing some possible network topologies for testing.

3.1 Design of Concurrent Test Manager

As show in Fig. 2, CTM is composed of the Graphical User Interface (GUI) and the Execution Engine (EE). The GUI module is mainly used to set test configuration through dialog boxes or forms, implement user control by buttons or menus and display the test results with line, pie, or bar graphs. Among these functionalities, test configuration setting is usually done by selecting and editing the Concurrent Test Control Script (CTCS). The CTCS is a simple script language dedicated to implement concurrent test control by providing test operator with the core language features including basic operators, loops and branch statements as well as some macros that are used to accomplish a series of complicated operations by a single line command. CTM keeps a mapping table to maintain the relationship between macros and their related implementation functions. As a matter of course, the table can be modified by test operators in order to make some extension to existing macros. Apparently, CTCS is very convenient to test operators as they want to add new functionalities to the script language according to the test requirements.

The EE module is the most important module to achieve multi-point concurrent test control. It interprets the CTCS by converting the code into test request or control request messages and then sending them to one or more TA(s). The EE module can also communicate with the GUI module through Data and Control Channel (DCC) and it converts the control operations initiated by test operator into control request messages and sends them to one or more TA(s). On receiving a response message from TA, the EE module will decide whether to handle the message itself or to forward it to the GUI module through DCC for further manipulation. This can be achieved by checking type and identifier field in the header of the received message. The format of these messages will be discussed later in section 3.3.

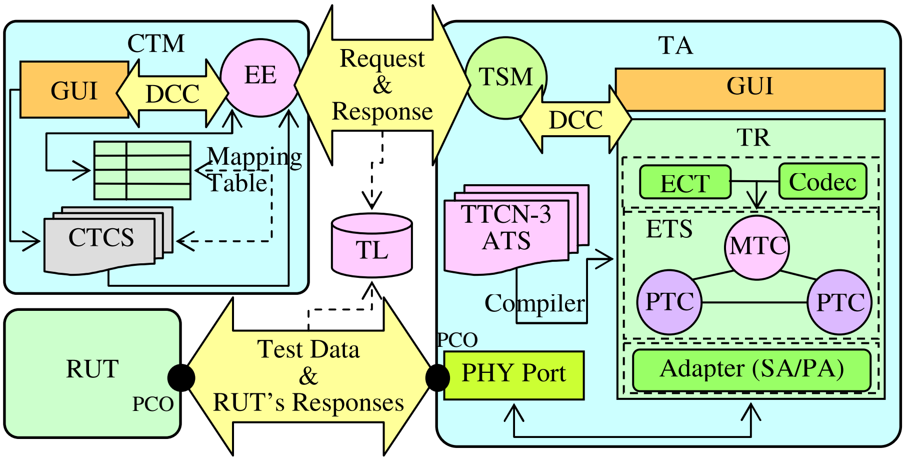

Fig. 2. Details on implementation of concurrent test manager and test agent

3.2 Design of Test Agent

As show in Fig. 2, TA is mainly composed of three parts, including the Test Service Module (TSM), the Test Runner (TR), and the GUI module. The TSM is responsible for listening to the test requests and control requests from CTM, scheduling these messages and then deciding what should be done next. The TR is the most important part in the implementation of TA. It is responsible for executing the actual test jobs by integrating the Execution Control Thread (ECT), the Codec (Encoder/Decoder), the Executable Test Suites (ETS), and the Adapter. When TA receives a request from CTM, TSM queues the message into its buffer and then do some scheduling for the message according to its priority. ECT is initiated by TSM on receiving the first test request message and it is responsible for controlling the execution of ETS. ECT will not stop its execution until the test is accomplished by normal termination, abnormal termination or user compulsory termination. For normal termination and abnormal termination, the ECT needs to send the test verdict or the exception status to TSM through DCC. TSM will encapsulate the information reported by ECT into a specific response message using pre-defined message format and after that send it back to CTM as the corresponding response to the former request. CTM will make further manipulation of the response message.

The TTCN-3 [7] (Testing and Test Control Notation version 3) is selected as the test description language for DMC-TS. TTCN-3 has been put forward and maintained by ETSI as a general purpose test specification and implementation language that is dedicated to black-box testing of a wide range of computer and telecommunication systems. Typical areas of application are protocol testing, service testing, module testing, testing of CORBA based platform, APIs, etc. TTCN-3 introduces lots of advanced features such as template matching mechanism and dynamic test control which make the description of test data and test control very flexible and convenient. One of the most important reasons to select TTCN-3 is that it defines test cases on abstract level so that test developers can concentrate on the development of test logic instead of worrying about how to implement them on a given platform or operating system. A collection of the test cases on the abstract level is called Abstract Test Suites (ATS) which can be converted into ETS by specific TTCN-3 compiler or interpreter. Together with the Codec, System Adapter (SA) and Platform Adapter (PA), ETS can run on different test platform and Implementation Under Test (IUT) and thus the implementation independent test can be achieved.

The GUI module for TA has the similar functionality as it is in CTM. As stated above, TA can conduct some simple test that involves only one test port or a pair of test ports without the participation of CTM. In this case, the GUI module can provide a man-machine interface for the test operators. PHY Port in Fig. 2 is the physical port of TA for sending test data and receiving responses from the RUT. PCO stands for the point of control and observation during the test process.

3.3 Some Relevant Issues

Other issues relevant to the implementation of DMC-TS that haven't been discussed above are list below:

- **Test Log.** In order to carry out some off-line analysis and debugging for the test system itself, Test Log (TL) module is needed for DMC-TS. As illustrated in Fig. 2, TL provides the storage function for test data and the responses of RUT as well as request and response messages between CTM and TA.
- **Message Format.** In order to ensure unambiguous communication between CTM and TA, the format of the request and response message must be clearly defined. The message is composed of 1-byte header, 1-byte reserved field and message data as shown in Fig. 3. The detail of fields in message header is outlined in Table 1.

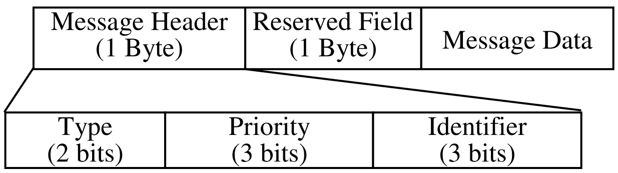

Fig. 3. Message Format

Table 1. Detail of message header fields

Message Type	Priority		Identifier
00 Test Request	000	High Priority (for control message)	000 – STATE CHECK
	001		001 – INITIALIZE
01 Control Request	010		010 – START
	011	Normal Priority (for test message)	011 – STOP
10 Test Response	100		100 – PAUSE
	101	Low Priority (for date transmission)	101 – RESUME
11 Control Response	110		110 – DATA
	111		111 – UPDATE

– **State Diagram of CTM and TA.** The working state of CTM and TA can be depicted by state diagram as illustrated in Fig. 4 and Fig. 5 respectively.

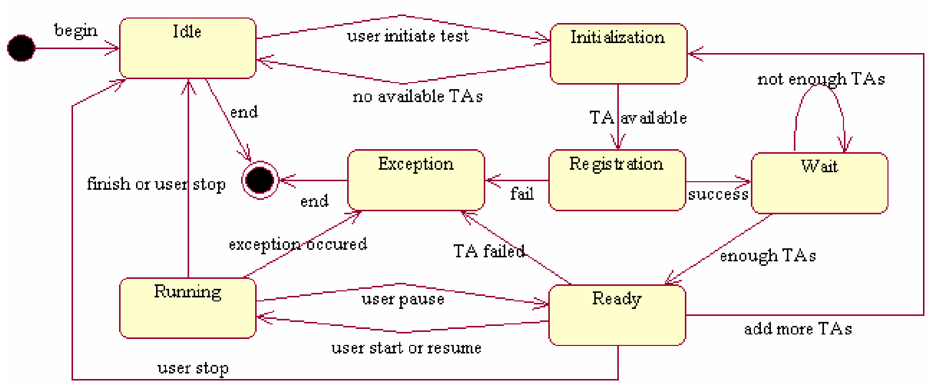

Fig. 4. State diagram of CTM

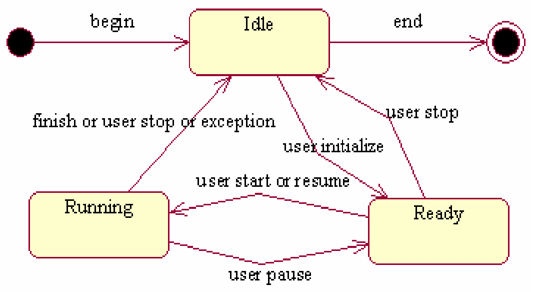

Fig. 5. State diagram of TA

4 Test Synchronization Problem

The significant difference between the DMC-TS and current test system is that the DMC-TS is constructed using a distributed architecture and thus may encounter the synchronization problem due to the lack of global state knowledge and global timer. The synchronization problem is referred to as the test sequences executed on multiple test components are not full ordered and thus has adverse impact on the test results, especially to conformance testing. So this problem must be solved before the DMC-TS can be used to conduct some test experiments. Typically, multi-port finite state machine is adopted to describe the synchronization problem [8].

A multi-port finite state machine with n ports (np-FSM) is a 6-tuple $M=(S, \Sigma, \Gamma, \delta, \lambda, s_0)$. S is a finite set of states and $s_0\in S$ is the initial state. $\Sigma=(\Sigma_1, \Sigma_2, \ldots, \Sigma_n)$, where Σ_k is the input alphabet of port k, and $\Sigma_i\cap\Sigma_j=\Phi$, for $i\neq j$. Let $I=\Sigma_1\cup\Sigma_2\cup\ldots\cup\Sigma_n$. $\Gamma=(\Gamma_1, \Gamma_2, \ldots, \Gamma_n)$, where Γ_k is the output alphabet of port k, and $\Gamma_i\cap\Gamma_j=\Phi$, for $i\neq j$. Let $O=(\Gamma_1\cup\{\varepsilon\})\times(\Gamma_2\cup\{\varepsilon\})\times\ldots\times(\Gamma_n\cup\{\varepsilon\})$, where ε stands for the null output. δ is the transition function $S\times I\rightarrow S$, and λ is the output function $S\times I\rightarrow O$.

A transition of an np-FSM M is a 4-tuple (s, σ, γ, s') where $s, s'\in S$, $\sigma\in I$, $\gamma\in O$, such that $\delta(s, \sigma)=s'$ and $\lambda(s, \sigma)=\gamma$. An np-FSM M can be represented by a directed graph $G=(V, E)$ where V represents the set S of states of M and E presents all specified transitions of M. An example of 3p-FSM is given in Fig. 6, where $S=\{S_0, S_1, S_2, S_3\}$, $\Sigma_1=\{X\}$, $\Sigma_2=\{Y\}$, $\Sigma_3=\{Z\}$, $\Gamma_1=\{a, b\}$, $\Gamma_2=\{c\}$, $\Gamma_3=\{d, e\}$. In this figure, the transition t_1 denotes that if S_0 is the current state and the input X is received, the state changes to S_1 and the output d is sent at ports 3. ε means no output occurs at port 1 and port 2.

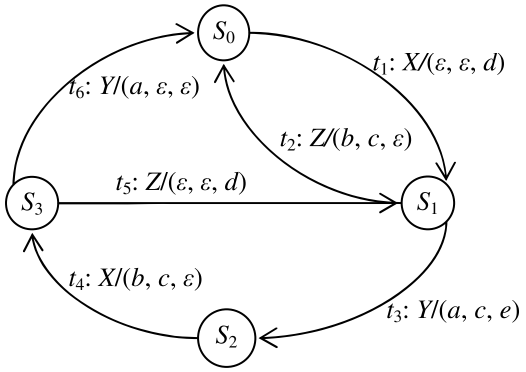
S_0
S_1
S_2
S_3
t_1: $X/(\varepsilon, \varepsilon, d)$
t_2: $Z/(b, c, \varepsilon)$
t_3: $Y/(a, c, e)$
t_4: $X/(b, c, \varepsilon)$
t_5: $Z/(\varepsilon, \varepsilon, d)$
t_6: $Y/(a, \varepsilon, \varepsilon)$

Fig. 6. An example of 3p-FSM

A global test sequence of np-FSM is a sequence in the form: $!x_1?y_1!x_2?y_2\ldots!x_t?y_t$ where, for $i=1,2,\ldots,t$, $x_i\in I$ and $y_i\in O$, for each port k, $|y_i\cap\Gamma_k|\leq 1$, i.e. y_i contains at most one symbol from the output alphabet of each port. '$!x_i$' means sending message x_i to the RUT and '$?y_i$' means receiving the messages belonging to y_i from the RUT.

Generally, the global test sequence sare generated from the specification of the IUT and characterized by their fault coverage (output faults and transfer faults) [9, 10]. A possible global test sequence for 3p-FSM of the example illustrated in Fig. 6 is:

$$!X?\{d\}!Y?\{a, c, e\}!X?\{b, c\}!Z?\{d\}!Z?\{b, c\} \tag{1}$$

In the DMC-TS, each TA executes a local test sequence constructed from the complete test sequence of the IUT. We can get the local test sequence for each TA from the global test sequence (1), and recorded as Seq_1, Seq_2 and Seq_3 with the corresponding TA recorded as TA_1, TA_2 and TA_3.

$$\text{Seq}_1 = !X?a!X?b?b,\ \text{Seq}_2 = !Y?c?c?c,\ \text{Seq}_3 = ?d?e!Z?d!Z \qquad (2)$$

Apparently, the conform execution of local test sequences Seq_1, Seq_2 and Seq_3 must give the result shown in Fig. 7(a), but if TA_1, TA_2 and TA_3 execute their local test sequences separately, the execution gives the failed result shown in Fig. 7(b). This kind of failure is called Synchronization Problem of Type-I (short for SPT-I) which is formally defined as follows.

Definition 1. For any two consecutive transitions $t_i=(s_i, \sigma_i, \gamma_i, s_i')$ and $t_j=(s_j, \sigma_j, \gamma_j, s_j')$, a SPT-I occurs if $port(\sigma_j) \notin ports(\gamma_i) \cup \{port(\sigma_i)\}$, where $port(\sigma_j)$ and port(σ_i) denotes the port associated with input σ_j and input σ_i respectively; $ports(\gamma_i)$ denotes the set of ports associated with values from γ_i that are not null.

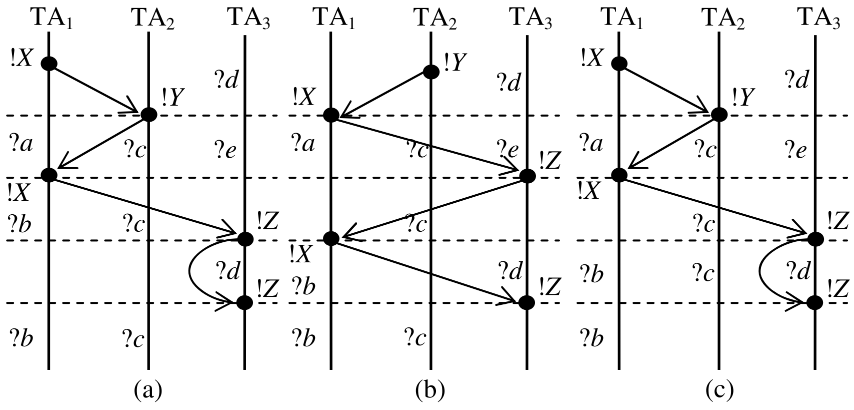

Fig. 7. Possible execution of local test sequences

The SPT-I can be easily solved by exchanging coordination messages among TAs. In [11], an algorithm is introduced to generate send and receive operations for coordination messages and insert them into the proper position of the local test sequences. Applying the proposed algorithm to the global test sequence (1), we get the following local test sequences, where '$!C_i$' means sending a coordination message to port i and '$?C_j$' means receiving a coordination message from port j.

$$\text{Seq}_1 = !X?a!C_2!X?b!C_3?b,\ \text{Seq}_2 = ?C_1!Y?c?c?c,\ \text{Seq}_3 = ?d?e?C_1!Z?d!Z \qquad (3)$$

However, the proposed algorithm still cannot overcome the errors as illustrated in Fig. 7(c). Although the input operations on each TA conform to (1) and the output operations on each TA conform to (2), the position of some output operations are not the same as show in Fig. 7(a). This kind of failure is called the Synchronization Problem of Type-II (short for SPT-II) which is formally defined as follows.

Definition 2. Given two consecutive transitions $t_i=(s_i, \sigma_i, \gamma_i, s_i')$ and $t_j=(s_j, \sigma_j, \gamma_j, s_j')$, a SPT-II occurs at port k if $(k \neq port(\sigma_j) \wedge (\gamma_i^k \neq \varepsilon \text{ XOR } \gamma_j^k \neq \varepsilon)$, where $port(\sigma_j)$ denotes the port associated with input σ_j; γ_i^k and γ_j^k denotes the output at port k on t_i and t_j respectively.

The existing of the SPT-II will probably cause the test system fails to detect the potential errors of the IUT. In [12], some algorithms for overcoming the SPT-II have been proposed but made a simplifying assumption that the time required for a coordination message to travel from a tester to another is greater than the reaction time of the IUT. Apparently, this assumption cannot be satisfied at all times. For DMC-TS, the SPT-II can be thoroughly overcome if and only if the position of output operations of the IUT can be checked without the consideration of time constraints and we follow the steps outlined below:

1. Record the position value of all output operations for each TA according to the global test sequence and input/output alphabets, where the position value means at which transition the output operation has occurred.
2. Add '!*T*' operation before all input operations in local test sequences, where '!*T*' stands for sending a special message to CTM. When CTM receives *T* message from TA, it broadcasts the message to all TAs including the one that initiates the message.
3. Define two variables *ot* and *nt* with the initial value of 0 for each TA respectively. Block all input operations in the local test sequence until the condition $nt > ot$ can be satisfied and after that, let $ot := nt$. These operations can be easily actualized in the test cases written in TTCN-3.
4. When TA receives the *T* message from CTM, let $nt := nt + 1$. For all output operations, record the current value of *nt* as its position value on each TA respectively.

We assume that the sending and receiving of *T* message is based on the reliable transmission and this assumption can be easily satisfied if taking TCP as transport layer protocol. For conformance testing, if the output of the IUT is not the expected one, the corresponding TA will yield a test verdict of "fail". If all TAs finished the execution of their local test sequences, the position value of output operations will be checked by comparing the values recorded by step 1 and step 4 to verify whether the SPT-II has occurred. If the SPT-II has indeed taken place, the fault of the IUT will be revealed by the unmatched position value.

5 Test Experiments

Conformance testing, performance testing and interoperability testing can be carried out for IP routers with DMC-TS. The following is an experiment of route flap testing. Route flap testing is a procedure that repeatedly changes routing tables with route withdrawals and updates. Core Internet routers must be able to handle many gigabits or terabits of traffic flawlessly during constant changes to route tables of 100,000 entries or more. Obviously, route flap testing involves both the U-Plane testing and the C-Plane testing of the RUT simultaneously. Following is the definition of U-Plane testing and C-Plane testing:

– **U-Plane Testing.** U-Plane testing helps determine how rapidly and accurately routers can manipulate very large volumes of data traffic. This type of testing includes generating realistic Internet traffic, including multiple types and classes of service, and stressing a router's forwarding ability and QoS performance.

– **C-Plane Testing.** As C-Plane uses various routing and signaling protocols to establish routes or path for U-Plane traffic, C-Plane testing usually includes route and link flapping, path establishment and destruction, failover and policy testing.

5.1 Testing of BGP Route Flapping

The objective of BGP route flapping test is to record the data performance through the RUT in the presence of an unstable BGP peer. This test is important to characterize the stability of the RUT and verify its ability to continue to provide data transport while being subjected to a chronic network failure resulting in an unstable BGP peer.

The test will establish two BGP peers on two different interfaces. As illustrated in Fig. 8. TA_1 plays the role of a BGP peer that advertises routes and forwards the UDP traffic from the RUT and TA_2 plays the role of another BGP peer that advertises and withdraws its routes (flapped routes) at a set rate for a set time. The rest of the TAs plays the role of ordinary routers that send UDP traffic to the RUT at specific rate during the test process.

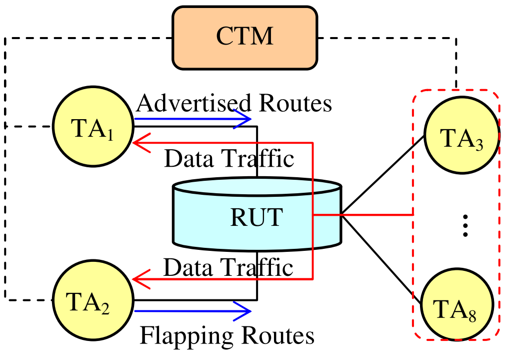

Fig. 8. Test BGP route flapping using the DMC-TS

The CTM will execute a pre-written CTCS to control the execution of the test and we follow the steps outlined below:

1. Configure BGP on the RUT.
2. Advertise a set number of routes to the RUT from TA_1.
3. Send UDP traffic from TA_3 – TA_8 to the RUT and record traffic statistics.
4. Perform route flapping on a set number of flapped routes from TA_2 for a set amount of time and record traffic statistics at a set period.
5. To characterize router behavior, vary one of the following and continue at step 3.
   – Number of traffic routes.
   – Number of flapped routes.
   – Data packet length.
   – Data packet distribution.

5.2 Test Results

In our experiment, two routers from different manufacturers marked as RUT-1 and RUT-2 are selected to make some comparison. 50,000 of BGP routes have been established and 10,000 of them are unstable ones. We use UDP packets with six different lengths from 64 bytes to 1518 bytes as the forwarding traffic and record the throughput and the latency statistics of the RUT-1 and the RUT-2 as shown in Fig. 9 and Fig. 10 respectively.

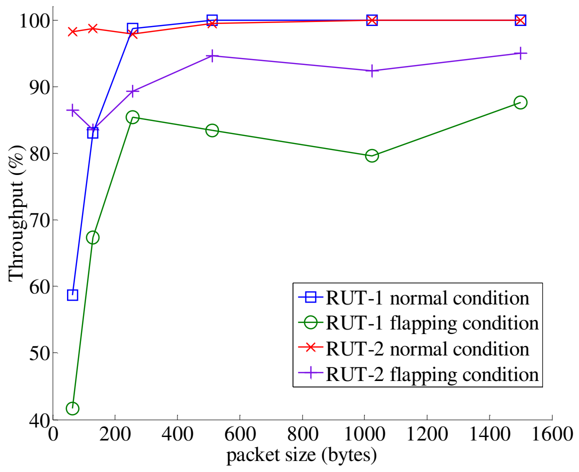

Fig. 9. Throughput under normal and flapping condition

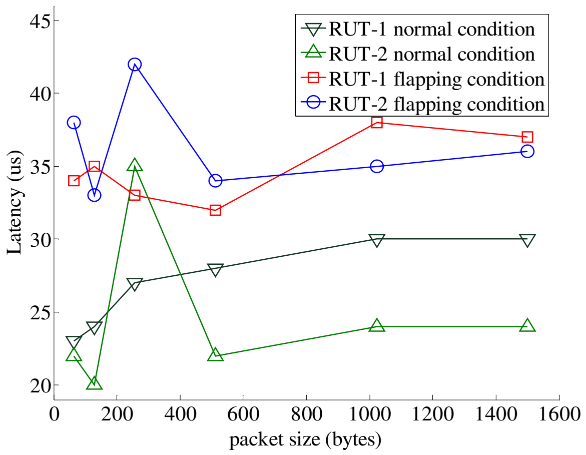

Fig. 10. Latency under normal and flapping condition

6 Conclusion and Future Works

The work presented in this paper is dedicated to construct a reasonable distributed architecture for concurrent testing and resolve the difficulty of test synchronization problem. Preliminary experiments have confirmed the feasibility and practicality of DMC-TS. However, using *T* message for checking the position of output operations will bring about additional overhead, and therefore how to reduce the overhead for sending and receiving of *T* message as well as coordination message is the next step of our research work. In addition, the current implementation of TA is based on the PC that runs Windows or Linux operating system and is configured with two or more network cards. In this way, the cost for the implementation of DMC-TS can be significantly reduced but the test ability is also limited by the operating systems and network cards of the PCs. Such limitations are more prominent when carrying out performance testing for IP routers. At this point, future test system should consider taking advantage of specialized hardware such as FPGA to implement sending and receiving operations for test data so as to improve the performance of the test system.

Acknowledgments. The research work presented in the paper has sponsored by Sichuan Network Communication Technology Key Laboratory as part of the research work on a new Internet architecture called SUPA (Single-layer User-data switching Platform Architecture), with financial support from the National Science Foundation of China (NSFC). The authors wish to acknowledge the financial support from NSFC. Our thanks are also given to our colleagues in the laboratory for fruitful cooperation in the past.

References

1. ISO/IEC JTC1/SC21.: Information Technology – Open Systems Interconnection – Conformance Testing Methodology and Framework (part 1-7) (1994)
2. Song, B., Zeng, H.X., Yue, L.Q.: On Concurrent Multi-port Test System for Routers and Its Support Tools. In: Liew, K.M., Shen, H., See, S., Cai, W., Fan, P., Horiguchi, S. (eds.) PDCAT 2004. LNCS, vol. 3320, pp. 469–483. Springer, Heidelberg (2004)
3. Zeng, H.X., Zhou, X., Song, B.: On Testing of IP Routers. In: Proceedings of PDCAT 2003, pp. 61–65. IEEE press, Los Alamitos (2003)
4. Leland, W.E., Taqqu, M.S., Willinger, W., et al.: On the Self-Similar Nature of Ethernet traffic (extended version). IEEE/ACM Transaction on Networking 2(1), 1–5 (1994)
5. Viho, C.: Test distribution: a solution for complex network system testing. Int. J. Softw. Tools Technol. Transfer 7, 316–325 (2005)
6. Wu, J.P., Li, Z.J., Yin, X.: Towards Modeling and Testing of IP Routing Protocols. In: Hogrefe, D., Wiles, A. (eds.) TestCom 2003. LNCS, vol. 2644, pp. 49–62. Springer, Heidelberg (2003)
7. ETSI ES201873 1-7.: Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3 (2008)
8. Khoumsi, A.: A Temporal Approach for Testing Distributed Systems. IEEE Transactions on Software Engineering 28(11), 1085–1103 (2002)

9. Petrenko, A., Bochmann, G., Yayo, M.: On Fault Coverage of Tests for Finite State Specification. Computer Network and ISDN Systems 29, 81–106 (1996)
10. Pap, Z., Subramaniam, M., Kovacs, G., Nemeth, G.A.: A Bounded Incremental Test Generation Algorithm for Finite State Machines. In: Petrenko, A., Veanes, M., Tretmans, J., Grieskamp, W. (eds.) TestCom/FATES 2007. LNCS, vol. 4581, pp. 244–259. Springer, Heidelberg (2007)
11. Rafiq, O., Cacciari, L., Benattou, M.: Coordination Issues in Distributed Testing. In: Proceeding of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 1999), Las Vegas (Nevada), USA. CSREA Press (1999)
12. Ural, H., Whittier, D.: Distributed testing without encountering controllability and observability Problems. Inf. Processing Lett. 88(3), 133–141 (2003)