17

CONFORMANCE TESTING OF MULTI-PROTOCOL IUTs

Yongbum Park[1]
Protocol Engineering Center, Electronics and Telecommunications Research Institute
161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Korea
E-mail: ybpark@pec.etri.re.kr

Myungchul Kim
Information and Communications University
58-4 Hwaam-Dong, Yusong-Gu, Taejon, 305-348, Korea
E-mail: mckim@icu.ac.kr

Sungwon Kang
Korea Telecom Research & Development Group
463-1 Junmin-Dong, Yusong-Gu, Taejon, 305-390, Korea
E-mail: kangsw@sava.kotel.co.kr

Abstract To declare conformance of multi-protocol Implementation Under Test (IUT), every layer of the multi-protocol IUT should be tested. According to ISO9646, single-layer test method is applied to testing the highest layer of multi-protocol IUT and single-layer embedded test method is used for the layers other than the highest layer because the interfaces between layers are not exposed. So far the conventional researches have focused on testing layer by layer all the protocols in a multi-protocol IUT.

This paper proposes a new method for testing a multi-protocol IUT. The proposed test method assumes that a multi-protocol IUT is under test and that the interfaces between the layers can not be controlled or observed by the tester. We apply the proposed test method to TCP/IP and compare the application results with those of the existing test methods in terms of various criteria such as the number of test cases[2], the number of test events and test

[1] He is also a Ph.D. student of Information and Communications University.
[2] Refer to ISO9646 for the definitions of test case, test event, and behavior line.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35567-2_25

G. Csopaki et al. (eds.), *Testing of Communicating Systems*

coverage. It turns out that the proposed test method significantly reduces the number of test cases as well as the number of test events while providing the same test coverage. In addition, the proposed test method shows the capability to locate the layer that is the source of failure in testing multi-protocol IUTs.

Keywords: Conformance testing, multi-protocol, embedded test, test in context

1. INTRODUCTION

Conformance testing methodology and framework has been published as an International Standard by ISO/IEC JTC1 [ISO9646]. This standard defines the concept of conformance testing, test methods, test specifications, test realization, and the requirements for the third party testing and also provides the Tree and Tabular Combined Notation (TTCN) for test scenario description. It is mainly for the case where a single-layer protocol is under test. It proposes a successive use of single-layer embedded test method for testing a multi-protocol IUT, i.e. an IUT that is a stack of protocols or profile. This method is referred to as incremental testing of a multi-protocol IUT.

According to [ISO 9646], testing is performed for a single specific layer and its lower layers play the role of a service provider. It is generally assumed that the lower layers are correct and all the causes of failures reside in the target layer. As a matter of fact, sometimes it is very difficult to determine which layer is the source of failure when it occurs during testing.

The single-layer embedded test method is applied to every layer of a multi-protocol IUT except the highest layer. In the method, test is specified for each single-layer and it need include specification of protocol activity above the one being tested but need not specify control or observation at layer boundaries within the multi-protocol IUT. Thus for a multi-protocol IUT consisting of protocol N_1 through upto protocol N_m, the test cases for protocol N_i,$1 <= i < m$, shall include the specifications of the PDUs of protocol N_{i+1} through upto to N_m as well as protocol N_i. The existing approaches to the test coverage and the test sequence generation for the embedded test method ([Petr96], [Petr97], [Zhu98], [Yevt98]) so far focused on testing one by one single-layer protocols within a multi-protocol IUT.

In the embedded test method (also called testing in context), System Under Test (SUT) is considered as a composition of two Finite State Machines (FSMs) where one FSM to be tested is called component and the other is called context. Context is an FSM for protocols that are not being tested and interacts with component through internal interfaces that are not

directly controllable or observable by the tester. In addition context is assumed error-free. [Petr96] and [Petr97] modeled testing in context as a communicating FSM and proposed a method to generate test sequences. Since in the embedded test method the interfaces between protocol layers are not directly controllable or observable, the test coverage of the method is limited. [Zhu98] proposed an approach and developed a tool to evaluate test coverage of the embedded test method. [Yevt98] proposed an approach to minimizing the test suite for testing in context. All these researches assume error-free context and focus on testing one by one single layer protocols of a multi-protocol IUT. Incremental testing starts from the lower layer and proceeds toward the higher layer. If errors were found at the lower layer, testing upper layers could be meaningless because the errors affect the behavior of the upper layer. In this case, the existing test method may assign an inconclusive verdict to the test case for the upper layer. This is the problem that is dealt with in this paper.

In this paper we propose a new test method for testing a multi-protocol IUT. The method yields one single test suite which has the same effect as applying both of the test suites; one for the single-layer test method and the other for the single-layer embedded test method. Comparing with the existing test methods, the proposed one has the following advantages:

* The upper layer and the embedded layers can be tested simultaneously with a single test suite.

* Producing a smaller size of test suite and shorter lengths of test cases, it reduces the load of test suite description and test execution.

* The proposed method provides a means of testing multi-protocol IUTs with inaccessible internal interfaces.

The rest of the paper is organized as follows: Section 2 proposes a new test method for testing multi-protocol IUTs. Section 3 presents application of the proposed test method to TCP/IP and provides some results of comparing this method with the existing ones. In Section 4, we conclude this paper with some discussion on the future work.

2. CONFORMANCE TEST METHOD FOR MULTI-PROTOCOL IUTS

According to the conformance test framework of [ISO9646], testing of multi-protocol IUTs should be carried out by a successive use of the single-layer embedded test method to every layer of multi-protocol IUT. In this case, when a specific protocol is tested, it is assumed that every protocol except the target protocol is assumed to be error-free. However, it is possible

to determine conformance of a multi-protocol IUT by a single testing assessment of applying a single test suite to the multi-protocol IUT if we can locate which layer is the source of failure in testing of multi-protocol IUT. In general, two adjacent layers interact with each other and upper-layer protocol PDU is included in the user data field of its lower-layer PDU. With the knowledge of interaction and relationship of PDUs, we are able to observe and control both layers at once by observing and controlling only the lower layer. This is the main idea of this paper.

B-ISDN ATM Adaptation Layer (AAL) [ITU94] is an example of a multi-protocol. In AAL, Service Specific Connection Oriented Protocol (SSCOP) and Common Part Convergence Sublayer (CP-AAL) are below Service Specific Coordination Function (SSCF), which is the highest sublayer of AAL. The interfaces of SSCOP are not open and can be indirectly accessed only through SSCF and CP-AAL. Another example of a multi-protocol is Multiprotocol Label Switching (MPLS) of IETF, which is an integration of layers 2 and 3 to improve performance of routers [IETF97]. In MPLS implementations of commercial routers, the interface between layr 2 and layer 3 is not open.

For our test method for multi-protocol IUTs, we assume the following:

* protocol for a layer or a sublayer is represented as an FSM
* the highest and the lowest interfaces of a multi-protocol IUT are open

In spite of the second assumption, the behavior of multi-protocol can be inferred indirectly by controlling and observing the lowest interface. With this insight, we propose a new test method to test several protocols at the same time with a single test suite.

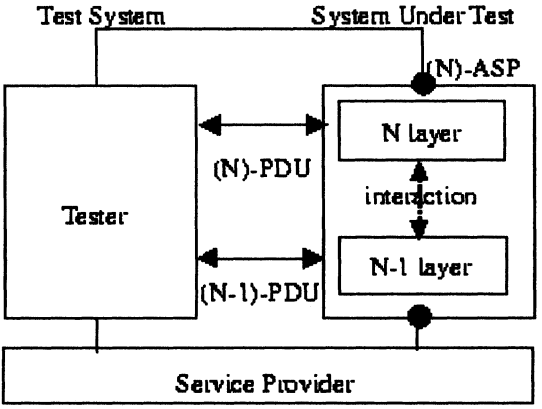

Figure 1. Configuration of a multi-protocol IUT

Using Figure 1 we explain our method. In Figure 1 the upper interface of N layer and the lower interface of N-1 layer are open but the interface between N layer and N-1 layer is not directly controllable or observable by the tester.

According to the existing test method in the standard [ISO9646], N layer and N-1 layer protocols of the System Under Tester (SUT) in Figure 1 should be tested separately. The single-layer test method is applied to N layer protocol and the single-layer embedded test method is applied to N-1 layer protocol. While the test event for N layer protocol testing is described with (N)-ASP and (N)-PDU, the test event for N-1 layer protocol testing is described with (N)-ASP and (N-1)-PDU.

Test suite from the multi-protocol test method uses test events used in the single-layer test method and the single-layer embedded test method. In the test configuration of Figure 1, the test event for multi-protocol test method is described with (N)-ASP, (N)-PDU and (N-1)-PDU. Note that (N)-PDU is included in the user data field of (N-1)-PDU.

In the existing test method of [ISO9646], the upper and the lower interfaces of IUT are controlled and observed by tester. These two interfaces are called Points of Control and Observation (PCOs). In the multi-protocol test method proposed in this paper, one input or output event at a lower PCO, say PCO3 in Figure 2, includes the behaviors of the two protocol layers. Let us explain the Figure 2, which shows the PCO structure of the multi-protocol test method for TCP/IP.

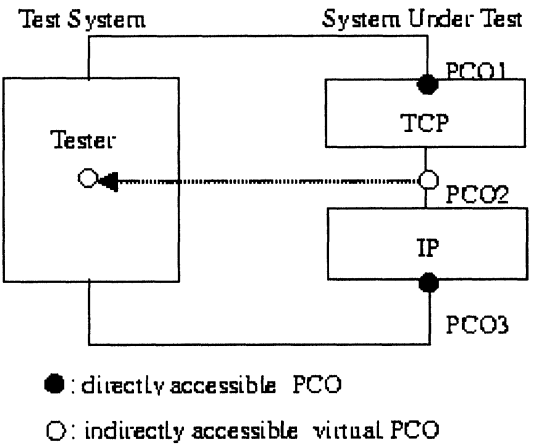

Figure 2. PCO structure example in the multi-protocol test method

In Figure 2, PCO1 and PCO3 are directly accessible by the tester whereas PCO2 is a virtual PCO not directly accessible by the tester. The test events at PCO1 are presented with TCP Abstract Service Primitives (ASPs) and the test events at PCO3 are presented with IP PDUs. Since IP PDU includes TCP PDU in its data (payload) field, it is possible to control and observe TCP PDU as well as IP PDU at PCO3. As this example demonstrates, the multi-protocol test method is able to test both TCP and IP protocols together at the same time.

3. APPLICATION OF THE MULTI-PROTOCOL TEST METHOD TO THE TCP/IP

This section presents an application of the proposed test method to TCP/IP and compares the results of this application with those of the existing test methods in terms of criteria such as the number of test cases, the number of test events and the test coverage.

3.1 FSM for the simplified TCP/IP

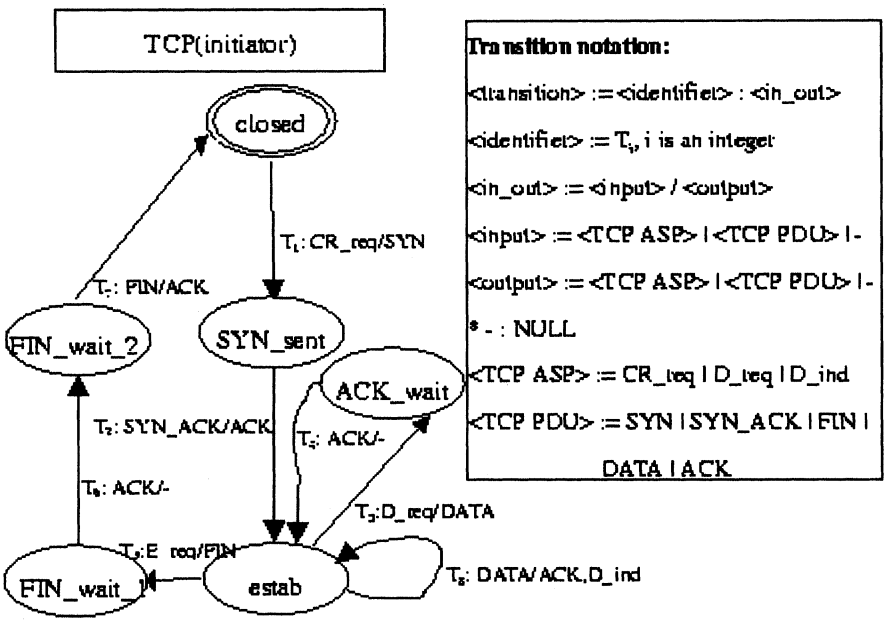

Figure 3. FSM for a simplified TCP

Figure 3 is a simplified FSM of TCP [Stev94]. TCP supports connection management, timer management, reliable data transfer through error control, and flow control functions. In the FSM of Figure 3, connection management and data transfer functions are represented but the states relating to timers in connection management phase and error control in data transfer phase are omitted. In this paper only the initiator role of TCP is used to explain the multi-protocol test method. Test cases for the responder role of TCP can be obtained with the same approach.

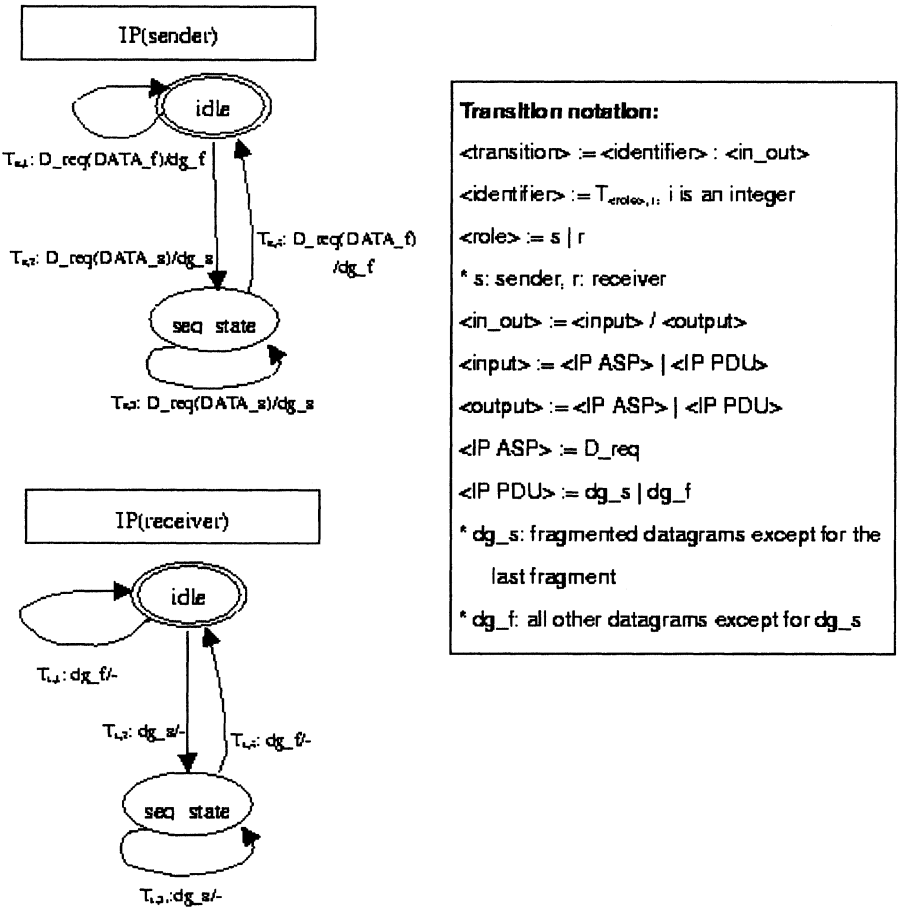

Figure 4. FSM for IP with fragmentation

IP can be represented with a single state FSM because it has no connection management function and supports only a simple datagram delivery service. However, because of fragmentation, it is useful to represent IP as an FSM with two states. When the IP transmits datagrams on demand of TCP and the size of requested data is greater than the size of IP datagram, IP does fragmentation to deliver the data with smaller datagrams. In this case the protocol behaviors for the last fragment and for the other fragments are different. Thus we use two states to distinguish these different behaviors. In this way, we come up with the FSM for IP as in Figure 4. In Figure 4 IP datagram is classified into dg_f and dg_s. dg_s stands for a datagram for a fragment except the last one and dg_f all the other datagrams except dg_s.

TCP and IP above it interact with each other and the interaction is dependent upon the size of TCP PDU. For example, TCP PDUs such as SYN, SYN_ACK, FIN and ACK in Figure 3 are delivered with dg_f PDU in Figure 4. DATA PDU of the TCP in Figure 3 is delivered with either dg_s or dg_f PDU depending on the size of the DATA PDU of TCP. Therefore, the transition T_3 in TCP interacts with either the transition $T_{s,1}$ or the transitions $T_{s,2}$, $T_{s,3}$ and $T_{s,4}$ in IP depending on the size of DATA.

3.2 Test cases for TCP in the multi-protocol test method

This section presents examples of test cases obtained by our multi-protocol test method. Naming convention for test cases is as follows:

<test case name> := <test method>_tc_<transition>[_<seq_no>]
<test method> := m | s | e
* m: multi-protocol test method
* s: single-layer test method
* e: single-layer embedded test method
<transition> := T_i | $T_{<role>, i}$, i is an integer
<role> := S | R
* S: sender, R: receiver
<seq_no> := integer

The interaction between T_3 in Figure 3 and the transitions in IP FSM of Figure 4 results in two varieties depending on the size of DATA. Thus, the number of test cases for T_3 in the multi-protocol test method becomes also two as m_tc_T_3_1 and m_tc_T_3_2. In this paper we use a simplified TTCN to describe test cases. Let us put line numbers in the left-most column.

m_tc_T_3_1: test case for T_3 - 1

1 +m_tc_T_2 /* preamble to put IUT to state estab */
2 PCO1 ! D_req(DATA) /* small DATA, no fragmentation */
3 PCO3 ? dg_f(DATA) /* PCO2 observation included */

In the above test case, the first behavior line is a test step to put IUT in "estab" state in order to prepare for execution of T_3. The second behavior line sends a small data not requiring fragmentation in IP and is input to the transition T_3. By feeding in a small data, not only the transition T_3 in the TCP FSM but also the transition $T_{s,1}$ in the IP FSM of Figure 4 is executed by the interaction between TCP and IP. These transition executions are presented in the third behavior line. Here receiving of dg_f is observed at PCO3 and thus the tester is able to observe PCO2 indirectly through observing the data (payload) field included in dg_f. As the result, this single test case covers two different transitions in two different protocols

respectively (namely, T_3 of TCP and $T_{s,1}$ of IP). This is summarized in Table 1.

Table 1. The relationship between transitions and behavior lines in the test case m_tc_T3_1

Behavior	transition of TCP	transition of IP
PCO1 ! D_req(DATA)		
PCO3 ? dg_f(DATA)	T_3	$T_{s,1}$

m_tc_T3_2: test case for T_3 - 2

```
1 +m_tc_T2
2    PCO1 ! D_req(DATA) /* large DATA, fragmentation */
3       PCO3 ? dg_s(DATA_s) /* PCO2 observation included */
4          PCO3 ? dg_s(DATA_s) /* PCO2 observation included */
5             PCO3 ? dg_f(DATA_f) /* PCO2 observation included */
```

In the test case m_tc_T_3_2, the second behavior line sends a large data requiring fragmentation in IP. By interacting with this behavior line, transitions such as $T_{s,2}$, $T_{s,3}$ and $T_{s,4}$ are executed and these interactions can be observed indirectly through the observation at PCO3 of TCP's DATA PDUs included in IP datagrams. As shown in this test case, multiple observations at the lower layer protocol may be required for single stimulus at the upper layer protocol in test case of multi-protocol test method. This test case covers transition T_3 of TCP and the transitions $T_{s,2}$, $T_{s,3}$, and $T_{s,4}$ of IP as shown in Table 2.

Table 2. The relationship between transitions and behavior lines in the test case m_tc_T3_2

Behavior	transition of TCP	transition of IP
PCO1 ! D_req(DATA)		
PCO3 ? dg_f(DATA_s)		$T_{s,2}$
PCO3 ? dg_f(DATA_s)		$T_{s,3}$
PCO3 ? dg_f(DATA_f)	T_3	$T_{s,4}$

Two test cases shown above are for testing behavior of the initiator role of the TCP. Testing behavior of the responder role of the TCP can be explained with the test case for transition T_8 in Figure 3. The following test case is one for receiving large TCP data requiring fragmentation in IP.

m_tc_T_8_2: test case for T_8
1 +m_tc_T_2
2 PCO3 ! dg_s(DATA_s)
3 PCO3 ! dg_s(DATA_s)
4 PCO3 ! dg_f(DATA_f) /* large DATA, fragmentation, PCO2 control included */
5 PCO3 ? dg_f(ACK)

In the test case m_tc_T_8_2, stimuli for transition T_8 of TCP FSM corresponds to the three behavior lines (lines 2, 3 and 4) describing the behavior of IP protocol. And the stimuli of these three behavior lines result in observing ACK PDU of the TCP. This test case provides an example demonstrating multiple stimuli at the lower layer protocol result in a single observation at the upper layer protocol. This test case covers the transition T_8 of TCP and the transitions $T_{r,2}$, $T_{r,3}$, $T_{r,4}$ and $T_{s,1}$ of IP.

Appendix 1 shows all the test cases for TCP FSM in Figure 3 obtained as the result of applying our multi-protocol test method.

3.3 Test case by the single-layer test method for the transition in TCP FSM

This section presents an example test case by the single-layer test method. In the test configuration of Figure 2, the highest layer protocol, TCP, can be tested by the single-layer test method because its upper and lower interfaces can be accessed by the tester [ISO9646]. In general, in the single-layer test method there are two PCOs that are located at the upper and lower interfaces respectively. These PCOs are PCO1 and PCO2 in the configuration of Figure 2 for testing the TCP. Following is the test case for transition T_3 in Figure 3.

s_tc_T_3: test case for T_3
1 +s_tc_T_2
2 PCO1 ! D_req(DATA)
3 PCO2 ? DATA

In the single-layer test method we are able to control and observe only the behavior of the TCP but IP. Therefore, IP PDU such as dg_f and dg_s used in test case m_tc_T_3_2 in the section 3.2 is not used in test case s_tc_T_3. In test case s_tc_T_3, the second behavior line may invoke the fragmentation in sender IP depending on the size of the DATA. In turn receiver IP delivers DATA to TCP after reassembling the fragmented data. Because the single-layer test method on TCP does not handle the behavior of IP, test case

$s\_tc\_T_3$ does not include transitions $T_{s,1}$ and transitions $T_{s,2}$, $T_{s,3}$, $T_{s,4}$ of IP FSM. That result is different from the multi-protocol test method shown in the previous section.

Appendix 2 shows all the test cases for all transitions of TCP FSM in Figure 3 as a result of applying the single-layer test method presented above.

3.4 Test case by the single-layer embedded method for the transition in IP FSM

This section presents an example test case for IP FSM by the single-layer embedded test method. PCOs used in the embedded test method are PCO1 and PCO3 in the configuration in Figure 2.

$e\_tc\_T_{s,2}$: test case for transition $T_{s,2}$

```
1 +e_tc_Ts,1
2    PCO3 ! dg_f(SYN_ACK)
3       PCO3 ? dg_f(ACK)
4          PCO1 ! D_req(DATA) /* large DATA, fragmentation */
5             PCO3 ? dg_s(DATA_s)
```

Test case above is to test IP. The fourth behavior line is a stimulus for transition $T_{s,2}$ of the IP. To send a large DATA PDU of TCP requiring fragmentation in the IP, TCP connection is to be established. The behavior related with TCP connection establishment is described in the second and third behavior lines.

Appendix 3 shows all the test cases for all transitions of IP FSM in Figure 4 as a result of applying the single-layer embedded test method.

3.5 Comparison the multi-protocol test method with the existing methods

As shown in the previous sections, the test cases for the multi-protocol test method include the test cases for the single-layer test method and for the single-layer embedded test method. And the relationship of those test cases is as follows:

$m\_tc\_T_1 = s\_tc\_T_1 + e\_tc\_T_{s,1}$

$m\_tc\_T_2 = s\_tc\_T_2 + e\_tc\_T_{s,1} + e\_tc\_T_{r,1}$

$m\_tc\_T_3\_1 = s\_tc\_T_3 + e\_tc\_T_{s,1}$

$m\_tc\_T_3\_2 = s\_tc\_T_3 + e\_tc\_T_{s,2} + e\_tc\_T_{s,3} + e\_tc\_T_{s,4}$

$m\_tc\_T_4 = s\_tc\_T_4 + e\_tc\_T_{s,1}$

$m\_tc\_T_5 = s\_tc\_T_5 + e\_tc\_T_{s,1}$

$m\_tc\_T_6 = s\_tc\_T_6 + e\_tc\_T_{r,1}$

$m\_tc\_T_7 = s\_tc\_T_7 + e\_tc\_T_{r,1} + e\_tc\_T_{s,1}$
$m\_tc\_T_8\_1 = s\_tc\_T_8 + e\_tc\_T_{r,1} + e\_tc\_T_{s,1}$
$m\_tc\_T_8\_2 = s\_tc\_T_8 + e\_tc\_T_{r,2} + e\_tc\_T_{r,3} + e\_tc\_T_{r,4} + e\_tc\_T_{s,1}$

For example, the test case $m\_tc\_T_3\_2$ by the multi-protocol test method includes the test case $s\_tc\_T_3$ for testing TCP by the single-layer test method and the test cases $e\_tc\_T_{s,2}$, $e\_tc\_T_{s,4}$ and $e\_tc\_T_{s,4}$ for testing IP by the single-layer embedded test method. Table 3 summarizes this relationship and other comparison results in terms of the number of test cases, the number of test events and test coverage.

Table 3. Comparison of test cases for each test method

test method / *measure*	proposed multi-protocol test method	existing test methods		
		single-layer test method for TCP	single-layer embedded test method for IP	Combination of two existing test methods
no. of test cases	10	8	8	16
no. of behavior lines	32	22	23	45
no. of events	64	48	47	95
test coverage	all transitions			all transitions

As shown in the Table 3, the multi-protocol test method significantly reduces the number of test cases compared with the existing method which combines single-layer test method and single-layer embedded test method while providing the same test coverage. The size of the test case is decreased and the load for test suite description is reduced because the number of behavior lines is reduced by 29%. And the test execution time is reduced because the number of test events in all test cases is reduced by 33%.

This overhead of the existing test method is caused by duplicated execution of the transitions of the lower layer protocol. The reason is that the transitions of the lower layer protocol executed in the single-layer test method for testing the upper layer protocol is also executed in the single-layer embedded test method for testing the lower layer protocol. However, in the multi-protocol test method, we are able to test two-layer protocols at once and thus avoid duplicated execution of transitions. As the result, the number of behavior lines and the number of test events are reduced.

Another advantage of the multi-protocol test method is the ability to locate the cause of failure in a multi-protocol IUT. In the existing test method it is assumed that every layer below the target protocol is correct and all the cause of failure is from the target protocol. However, real causes of the failure may be from the lower layers other than the target protocol. In this case, the multi-protocol test method provides a capability to locate the exact source of the failure.

The multi-protocol test method has a limitation in testing thebehavior of the lower layer protocol on errors. This is a problem caused from that the tester can not directly access the internal interfaces but control and observe those interfaces indirectly via the upper layer protocols. This problem also exists in the embedded test method.

4. CONCLUSION AND FUTURE WORK

In this paper we proposed a new test method for testing the multi-protocol IUT. This test method is able to test multiple protocols with a single test suite and the test suite of the proposed test method is described with combination of test events that are used in the single-layer test method and the single-layer embedded test method. Application of the proposed test method to the simplified TCP/IP is presented as an example and some results of comparing this test method with the existing test methods are shown. By applying the multi-protocol test method, the number of test events, the number of test cases and the number of behavior lines are significantly reduced comparing to the existing test method by the single-layer test method and the single-layer embedded test method. Also the proposed test method is able to locate the exact source of failure in a specific layer in testing multi-protocol IUT.

In this paper we showed an example of application of the proposed test method to the multi-protocol IUT with two protocols in stack. It is future work to generalize the proposed test method to be applicable to the multi-protocol IUT consisted of more than two protocols in stack. Another further work could be applying the proposed test method to the real protocols such as ATM AAL and MPLS.

REFERENCES

[ISO9646] ISO 9646, "Information Technology - OSI - Conformance Testing Methodology and Framework," 1992.

[ITU94] ITU-T Recommendation Q.2110, "B-ISDN ATM Adaptation Layer - Service Specific Connection-Oriented Protocol(SSCOP)," 1994.

[IETF97] IETF draft-ietf-mpls-framework-02.txt, "A Framework for Multiprotocol Label Switching," 1997.

[Stev94] W. Richard Stevens, TCP/IP Illustrated, Addison-Wesley, 1994.

[Petr96] Alexandre Petrenko, Nina Yevtushenko, Gregor v. Bochmann and Rachida Dssouli, "Testing in context: framework and test derivation," Computer Communications 19, pp. 1236-1249, 1996.

[Petr97] Alexandre Petrenko and Nina Yevtushenko, "Fault detection in embedded components," IWTCS'97, pp. 272-287, Cheju Island, Korea, 1997.

[Zhu98] Jinsong Zhu, Son T. Voung and Samuel T. Chanson, "Evaluation of test coverage for embedded system testing," IWTCS'98, pp. 111-126, Tomsk, Russia, 1998.

[Yevt98] Nina Yevtushenko and Ana Cavali, "Test suite minimization for testing in context," IWTCS'98, pp. 127-145, Tomsk, Russia, 1998.

APPENDIX 1. TEST CASES FOR TCP BY THE MULTI-PROTOCOL TEST METHOD

m_tc_T1: test case for T_1
PCO1 ! CR_req
PCO3 ? dg_f(SYN) /* PCO2 observation included */
- covered transitions:T_1, $T_{s,1}$

m_tc_T_2: test case for T_2
+m_tc_T_1
PCO3 ! dg_f(SYN_ACK) /* PCO2 control included */
PCO3 ? dg_f(ACK) /* PCO2 observation included */
- covered transitions: T_2, $T_{r,1}$, $T_{s,1}$

m_tc_T_3_1: test case for T_3_1
+m_tc_T_2
PCO1 ! D_req(DATA) /* small DATA, no fragmentation */
PCO3 ? dg_f(DATA) /* PCO2 observation included */
- covered transitions: T_3, $T_{s,1}$

m_tc_T_3_2: test case for T_3_2
+m_tc_T_2
PCO1 ! D_req(DATA) /* large DATA, fragmentation */
PCO3 ? dg_s(DATA_s) /* PCO2 observation included */
PCO3 ? dg_s(DATA_s) /* PCO2 observation included */
PCO3 ? dg_f(DATA_f) /* PCO2 observation included */
- covered transitions: T_3, $T_{s,2}$, $T_{s,3}$, $T_{s,4}$

m_tc_T4: test case for T4
+m_tc_T3
PCO1 ! D_req(ACK)
PCO3 ? dg_f(ACK) /* PCO2 observation included */
- covered transitions: T_4, $T_{s,1}$

m_tc_T_5: test case for T_5

+m_tc_T_2
PCO1 ! E_req
PCO3 ? dg_f(FIN) /* PCO2 observation included */
- covered transitions: T_5, $T_{s,1}$

m_tc_T_6: test case for T_6
+m_tc_T_5
PCO3 ! dg_f(ACK) /* PCO2 control included */
- covered transitions: T_6, $T_{r,1}$

m_tc_T_7: test case for T_7
+m_tc_T_6
PCO3 ! dg_f(FIN) /* PCO2 control included */
PCO3 ? dg_f(ACK) /* PCO2 observation included */
- covered transitions: T_7, $T_{r,1}$, $T_{s,1}$

m_tc_T_8_1: test case for T_8_1
+m_tc_T_2
PCO3 ! dg_f(DATA) /* small DATA, no fragmentation, PCO2 control included */
PCO3 ? dg_f(ACK)
- covered transitions: T_8, $T_{r,1}$, $T_{s,1}$

m_tc_T_8_2: test case for T_8_2
+m_tc_T2
PCO3 ! dg_s(DATA_s)
PCO3 ! dg_s(DATA_s)
PCO3 ! dg_f(DATA_f) /* large DATA, fragmentation, PCO2 control included */
PCO3 ? dg_f(ACK)
- covered transitions: T_8, $T_{r,2}$, $T_{r,3}$, $T_{r,4}$, $T_{s,1}$

APPENDIX 2. TEST CASES FOR TCP BY THE SINGLE-LAYER TEST METHOD

s_tc_T_1: test case for T_1
PCO1 ! CR_req
PCO2 ? SYN
- covered transitions: T_1

s_tc_T_2: test case for T_2
+s_tc_T_1
PCO2 ! SYN_ACK
PCO2 ? ACK
- covered transitions: T_2

s_tc_T_3: test case for T_3
+s_tc_T_2
PCO1 ! D_req(DATA)
PCO2 ? DATA
- covered transitions: T_3

s_tc_T_4: test case for T_4
+s_tc_T_3
PCO1 ! D_req(ACK)
PCO2 ? ACK
- covered transitions: T_4

s_tc_T_5: test case for T_5
+s_tc_T_2
PCO1 ! E_req
PCO2 ? FIN
- covered transitions: T_5

s_tc_T6: test case for T_6
+s_tc_T_5
PCO2 ! ACK
- covered transitions: T_6

s_tc_T_7: test case for T_7
+m_tc_T_6
PCO2 ! FIN
PCO2 ? ACK
- covered transitions: T_7

s_tc_T_8: test case for T_8
+s_tc_T_2
PCO2 ! DATA
PCO2 ? ACK
- covered transitions: T_8

APPENDIX 3. TEST CASES FOR IP BY THE SINGLE-LAYER EMBEDDED TEST METHOD

e_tc_$T_{s,1}$: test case for $T_{s,1}$
PCO1 ! CR_req
PCO3 ? dg_f(SYN)

e_tc_$T_{s,2}$: test case for $T_{s,2}$
+e_tc_$T_{s,1}$
PCO3 ! dg_f(SYN_ACK)
PCO3 ? dg_f(ACK)
PCO1 ! D_req(DATA) /*
large DATA, fragmentation */
PCO3 ? dg_s(DATA_s)

e_tc_$T_{s,3}$: test case for $T_{s,3}$
+e_tc_$T_{s,2}$
PCO3 ? dg_s(DATA_s)

e_tc_$T_{s,4}$: test case for $T_{s,4}$
+e_tc_$T_{s,3}$
PCO3 ? dg_f(DATA_f)

e_tc_$T_{r,1}$: test case for $Tr_{,1}$
PCO3 ! dg_f(SYN)
PCO1 ? CR_ind

e_tc_$T_{r,2}$: test case for $T_{r,2}$
+e_tc_$T_{r,1}$
PCO1 ! CR_resp
PCO3 ? dg_f(SYN_ACK)
PCO3 ! dg_f(ACK)
PCO3 ! dg_s(DATA_s)

e_tc_$T_{r,3}$: test case for $T_{r,3}$
+e_tc_$T_{r,2}$
PCO3 ! dg_s(DATA_s)

e_tc_$T_{r,4}$: test case for $T_{r,4}$
+e_tc_$T_{r,3}$
PCO3 ! dg_f(DATA_f)
PCO1 ? DATA_ind

BIOGRAPHY

Mr. Yongbum Park received B.S. in electronic engineering from Kyungbook National University in Korea in 1986 and received M.S. degree in computer science from Korea Advanced Institute of Science and Technology in 1989. Since 1989, he has been a senior research staff of Electronics and Telecommunications Research Institute. In 1995-1996, he was a guest researcher at National Institute of Standards and Technology of U.S.A. His research interests include communication protocol testing, Internet, wireless communication and mobile computing.

Dr. Myungchul Kim received B.A. in Electronics Engineering from Ajou University in 1982, M.S. in Computer Science from the Korea Advanced Institute of Science and Technology in 1984, and Ph. D in Computer Science from the University of British Columbia, Vancouver, Canada, in 1993. Currently he is with the faculty of the Information and Communications University, Taejon, Korea. Before joining the university, he was a managing director in Korea Telecom Research and Development Group for 1984 - 1997 where he was in charge of research and development of protocol and QoS testing on ATM/B-ISDN, IN, PCS and Internet. He has also served as a member of Program Committees for IFIP International Workshop on Testing of Communicating Systems, IEEE International Conference on Distributed Computing Systems, and IFIP International Conference on Formal Description Technique / Protocol Specification, Testing and Verification, the chairman of Profile Test Specifications - Special Interest Group of Asia-Oceania Workshop (for 1994 - 1997), and co-chair of the 10th IWTCS'97. His research interests include Internet, protocol engineering, telecommunications, and mobile computing.

Dr. Sungwon Kang received B.A. from Seoul National University in Korea in 1982 and received M.S. and Ph.D. in computer science from the University of Iowa in U.S.A. in 1989 and 1992, respectively. Since 1993, he has been a senior researcher at Korean Telecom. In 1995-1996, he was a guest researcher at National Institute of Standards and Technology of U.S.A. In 1997, he was the co-chair of the 10th International Workshop on Testing of Communication Systems. Currently he is the head of the Protocol Engineering Team at Korean Telecom R&D Group. His research interests include communication protocol testing, program optimization and programming languages.