マルチメディア通信と分散処理 30－2
(1986. 7. 25)

マルチレーヤ環境における OSIトランスポート、セション・プロトコルの製品検証

加藤聰彦　　鈴木健二

国際電信電話株式会社 研究所

異機種コンピュータ間通信では、標準プロトコルの役割が重要であるが、標準プロトコルが定義されていても、その異なった解釈等により、標準プロトコルを実装したつもりの通信システムが相互に通信できない場合がある。これを防ぐために、個々の通信システムと標準プロトコルの間の整合性を検査する製品検証(Conformance Testing)が重要となる。
本稿では、階層構造を持つOSIプロトコルを実装した通信システムで、複数のレーヤのプロトコルプログラムをひとまとまりとして検証するマルチレーヤ環境での製品検証において、個々のレーヤを順に検証する方式を提案し、OSIトランスポート・プロトコル・クラス2と、セション・プロトコルBSS(Basic Synchronized Subset)に適用した結果について述べる。

CONFORMANCE TESTING FOR OSI SESSION AND TRANSPORT PROTOCOLS IN THE MULTIPLE LAYER ENVIRONMENT

Toshihiko KATO　Kenji SUZUKI

KDD R & D Labs.　2-1-23, Nakameguro, Meguro-ku, Tokyo, 153

The conformance testing of communication products is essential in heterogeneous computer communications. In the confromance testing for OSI protocols several testing methods have been proposed in two categories. One is a test of single layer and the other is that of several adjacent layers together. The former is called testing in Single Layer Environment, and the latter is called testing in Multiple Layer Environment. In this paper, we propose the layer by layer testing method for Multiple Layer Environment based on the automaton model, and demonstrate the conformance testing for Implementation Under Test which consists of OSI Transport Protocol Class 2 and Session Protocol Basic Synchronized Subset.

1. Introduction

Current standardization activities on OSI (Open Systems Interconnection) within ISO and CCITT have been much progressed. Some of the layer protocols are now the stable Standards or Recommendations. In order to develop the OSI products, the powerful methodologies to check their conformance to OSI protocols, so called Conformance Testing[1], are indispensable. In the case of conformance testing, testing methods are highly dependent on the availability of access to the upper layer service primitives of the tested layer. For the products which have the program structure corresponding to OSI Reference Model, upper layer service primitives of the tested layer can be used in conformance testing (testing in the Single Layer Environment (SLE)). However for the other type of products which implement several adjacent layers in one program module, the upper layer service primitives of the tested layer cannot be used in testing directly (testing in the Multiple Layer Environment (MLE)).

The application of these two methods depends entirely on the internal program structure of the System Under Test (SUT), whether the service primitives of Implementation Under Test (IUT) can be manipulated and monitored or not for testing purposes. In practice, MLE testing is more applicable than SLE to a wide range of products, since many communication softwares are realized with several adjacent layer protocols together in one program due to the performance and resource management of the system.

For SLE testing, we have proposed[2-4] the testing method based on the automaton model and have already applied it to OSI Transport Protocol[5]. However MLE testing is more applicable to various types of products, and therefore it is important to establish MLE testing method.

In this paper, we describe our methodology and the results of MLE testing for Transport Protocol Class 2 and Session Protocol[6] BSS (Basic Synchronized Subset).

2. MLE Testing Methodology

In the general scheme for conformance testing, SUT is connected to the Protocol Tester (PT) via a communication line. PT sends the test sequence to SUT, and examines the response by comparing to the reference response prepared beforehand. It is also required to implement some additional testing aid in SUT in order to generate the service primitives through the upper layer boundary.

MLE testing methods are characterized by the following points, i.e.

① whether to test several adjacent layers as a whole, or layer by layer starting from the lower layer, and

② whether to use an additional testing aid in the SUT.

As for the point ①, we have adopted the layer by layer method. The reason is due to the layered protocol structure of OSI. Since the OSI protocols are specified layer by layer independently, it is too complicated to integrate several adjacent layer protocols into one model. In this method, the highest layer within IUT is examined by the same method as in single layer testing. Regarding the point ②, we have investigated both cases. That is, we use the layer by layer method both with and without the testing aid.

ISO has started the standardization of the conformance testing. In ISO terminology, layer by layer testing in MLE is called Embedded Single-layer Testing, and the case with or without the testing aid correspond to Distribute or Remote Testing respectively. Our former method therefore corresponds to DSE (Distributed Embedded Single-layer) testing method, and the latter to RS (Remote Single-layer) testing method.

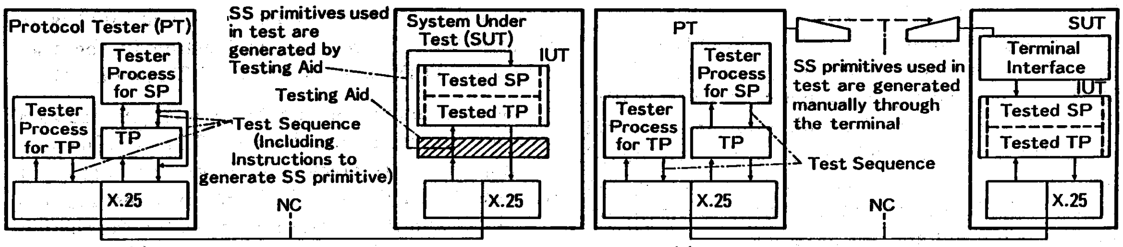
Another characteristic of our method is that our DSE testing method uses an additional program called Testing Aid implemented below IUT. This configuration is selected in order that Testing Aid can distinguish all states of the tested layer in IUT, including the states which cannot be detected by user of the tested layer such as the state "after receiving PR SPDU" in SP.

In the following discussion, we will assume that IUT consists of TP and SP. Fig.1 shows the testing methodologies studied here for TP and SP. In this figure, Testing Aid is implemented between the X.25 program and IUT for DSE testing method. In RS testing method, only the terminal interface program is located at the upper boundary of SP program, and the upper layer service primitives are applied from the terminal which is connected to SUT.

3. MLE Testing Procedure Based on Automaton Model

3.1 Specification of Automaton Model of TP and SP

The automaton model is used as a reference in conformance testing, and the accuracy of the test depends on how precisely it describes the behaviour of the tested protocol. In the case of MLE testing of TP and SP, the automaton model of TP should be specified taking into account the behaviour of SP, while the automaton model of SP can be specified independently. We will discuss the method for



(a) DSE testing method for TP and SP (b) RS testing method for TP and SP

Fig. 1 MLE Testing methodologies for TP and SP

specifying the precise automaton model of TP and SP for MLE testing.

3.1.1 Definition of Precise Automaton Model of TP and SP

The automaton models of TP and SP are defined in the state tables attached to the annexes of X.224 and X.225 respectively. However, these state tables are not accurate enough to be used for testing purposes for the following reasons :

①The state tables in X.224 do not describe the behaviour in a data transfer phase such as Explicit Flow Control.
②The state tables in X.225 do not describe different behaviours in an error state when Exceptions functional unit is selected.
③Predicates and actions using some variables which characterize the "state" of TP or SP, are introduced.

Therefore, in the precise automaton models, new states should be introduced to represent the behaviours not described in these state tables, and predicates and actions should be converted to the states.

Furthermore, OSI products have some pluralities depending on the implementations, such as what procedure is selected in the Treatment of Protocol Errors in TP and what functional units of SP are implemented. In our approach, a nondeterministic automaton is prepared which represents all the cases of the pluralities (we call it a general automaton), and the general automaton is tuned up for individual IUT (we call it a tuned up automaton). The tuned up automaton is used as an automaton model in the testing. In the general automaton, the pluralities are described by static conformance parameters. The tuned up automaton is derived by assigning true or false to each static conformance parameter, removing the states not used and merging the equivalent states.

In conformance testing, it is practical to validate IUT for one connection in one testing procedure. With respect to single connection IUT acts as either an initiator or a responder. So the automaton models are specified separately for initiator and responder.

3.1.2 Specification of Automaton Model of TP for MLE Testing

In the case of MLE testing for TP and SP, it is required that the behaviours of SP should be selected enough to generate the Transport Service (TS) primitives indispensable for testing. The automaton model of TP is specified by combining the tuned up automaton of TP and the automaton representing the selected behaviours of SP in the following steps :

① Specify the inputs of the model for MLE testing by enumerating selected Session Service (SS) primitives, Network Service (NS) primitives, TPDUs (DT TPDU containing different SPDU is defined as a different input), and timeouts in TP.
② Specify the states of the model of TP for MLE testing by enumerating possible tupples of the states of TP and SP.

3.2 Generation of Test Sequences for MLE Testing

The test sequences for MLE testing are generated using an identification technique for the finite state automaton by the following scheme.

(1) We assume that the IUT has the same number of states as the automaton model for MLE testing. Under this assumption, each state of IUT is identified by the Distinguishing Sequence (DS), which is the input sequence generating a characteristic output sequence for each state.

(2) The test sequence consists of the state identification part and the transition confirmation part. In the state identification part, it is confirmed that IUT has all the states corresponding to the states defined in the automaton model by applying the DS to each state. In the transition confirmation part, the following steps are repeated for all states and inputs :

① Transfer to a state to be tested using an already confirmed sequence.

② Apply one input and examine the output.

③ Check the state transition by applying DS.

DS is an identical input sequence which identifies all states. However, depending on the protocol tested, there are cases where the automaton model has no DS. As described later, the automaton models of TP Class2 and SP BSS have no DS, so we introduce different and plural sequences as DS.

In DSE testing method, the test sequences are generated by the above scheme. In RS testing method, the following limitations are also imposed :

① The upper layer service primitives should output PDUs or lower layer service primitives to inform PT of the trigger of the next test input.

② In the transition confirmation part, the transitions can be tested only for the states that are strongly connected with the initial state.

4. Experiment for MLE Testing of TP and SP

4.1 Experimental Scheme

We have performed the conformance testing experiment for TP and SP developed in VAX11 /780[2,3]. Fig.2 shows the experimental system configurations of DSE testing method for TP and SP.

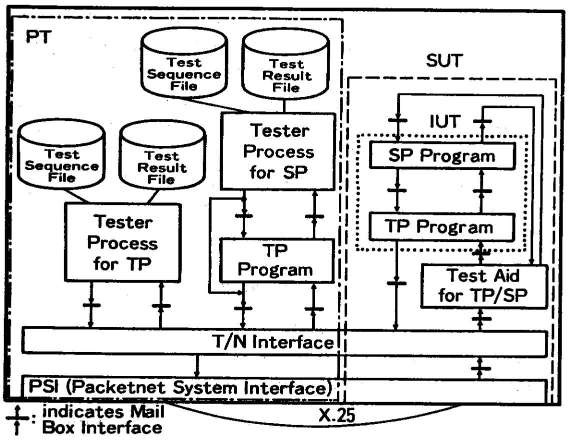


Fig. 2 Experimental System Configurations of DSE Testing Method for TP and SP in VAX11/780

As shown in the figure, TP and SP are executed as separate processes and connected through a mailbox interface provided by VAX/VMS. X.25 protocol is provided by PSI (Packetnet System Interface) of VAX/VMS, and T/N Interface is prepared to convert NS primitives to PSI commands and vice versa.

PT is also implemented in VAX 11/780 and comprises Tester Process, Test Sequence File and Test Result File. Tester Process reads one test input from the Test Sequence File, and applies it to SUT. It then waits for the response and stores the result in the Test Result File. These procedures are continued for all test inputs. If the test input is an SS primitive, PT sets the instruction in the user data of DT TPDU and sends it to SUT though NC.

In the experiment, Testing Aid is implemented as a separate process located between T/N Interface and TP program. It receives and analyzes the test input from the Tester Process. If the received test input is DT TPDU and its user data conveys the instruction to generate an SS primitive, Testing Aid generates the SS primitive with the requested synchronization and discards the DT TPDU. The other inputs are passed to the TP program as test inputs.

The RS testing method was examined using a terminal interface program interfaced to SP program instead of Testing Aid. In case an SS primitive is needed, it is generated through the terminal manually.

As an example of MLE testing, we give the results of MLE testing of TP Class2 and SP BSS below.

4.2 MLE Testing of TP Class2

According to the procedure described in 3.1, the general automaton of TP Class2 is defined first, and then the automaton model for MLE testing is created by combining the tuned up automaton of TP Class2 with the automaton of selected behaviours of SP.

In order to specify the general automaton, the following modifications are applied to the state tables in X.224.

① When a Transport Connection (TC) is released, the initiator may release the NC in the case that no other TCs are assigned. Therefore the general automaton of Class2 initiator should specify different states to examine whether other TCs are assigned to the same NC or not.

② If Explicit Flow Control is used, Class2 initiator and responder send DT TPDU when the transmit window is open. Different states should therefore be specified to examine whether the transmit window is open or not.

③ TP Class2 initiator and responder cannot send more than two ED TPDUs successively without receiving EA TPDU. Therefore different states should be specified to examine whether EA TPDU has been received or not.

④ Class2 initiator and responder may use the ER TPDU in the Treatment of Protocol Errors, and therefore the error state after sending a ER TPDU should be specified explicitly.

Table1 shows a part of the general automaton of Class2 initiator. The general automaton contains 22 states, 25 inputs, and the following six static

Table1 Part of General Automaton of TP Class2 Initiator

STATE / INPUT	CLOSED*1				Wait for NC (WFNC*1)		Wait for CC (WFCC*1)		Wait before releasing (WBCL*1)		CLOSING*1		ERROR	
	NC not assigned	NC assigning	NC assigned not mpx	NC assigned mpxed	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed
	N1	N2	N3	N4	A11	A12	A21	A22	A31	A32	C1	C2	D1	D2
TCONreq	A11 NCONreq	A12	A21 CR	A22 CR	A11	A12	A21	A22	A31	A32	C1	C2	D1	D2
TCONreq -other*2	N2 NCONreq	//	N4 NCONreq	//	A12	//	A22	//	A32	//	C2	//	D2	//
NCONcnf	//	N4	//	//	A21 CR	A22 CR	//	//	//	//	//	//	//	//
CC	//	//	N3 DR	N4 DR	//	//	B01 TCONcnf	B02 TCONcnf	C1 DR	C2 DR	C1	C2	D1	D2
DR	//	//	N3 DC	N4 DC	//	//	P1:N3 TDISind ¬P1:N1 NDISreq TDISind	N4	P1:N3 TDISind ¬P1:N1 NDISreq TDISind	N4	P1:N3 ¬P1:N1 NDISreq	N4	P1:N3 DC ¬P1:N1 DC NDISreq	N4 DC

STATE / INPUT	OPEN*1							
	window open				window closed			
	ED not sent		ED sent		ED not sent		ED sent	
	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed
	B01	B02	B11	B12	B21	B22	B31	B32
TDTreq	B01 DT	B02 DT	B11 DT	B12 DT	P2:B21 retain*6	P2:B22 retain	P2:B31 retain	P2:B32 retain
TDTreq -last*3	P2:B21 DT ¬P2:B01 DT	P2:B22 DT ¬P2:B02 DT	P2:B31 DT ¬P2:B11 DT	P2:B32 DT ¬P2:B12 DT	//	//	//	//
TEXreq	P2:B11 ED ¬P2:B01	P2:B12 ED ¬P2:B01	P2:B11 retain ¬P2:B01	P2:B12 retain ¬P2:B01	P2:B31 ED	P2:B32 ED	P2:B31 retain	P2:B32 retain
TDISreq	C1 DR	C2 DR	C1 DR	C2 DR	C1 DR	C2 DR	C1 DR	C2 DR
DT=*4	B01 AK TDTind	B02 AK TDTind	B11 AK TDTind	B12 AK TDTind	B21 AK TDTind	B22 AK TDTind	B31 AK TDTind	B32 AK TDTind
AK-open*5	B01	B02	B11	B12	B01	B02	B11	B12

Note *1: These states correspond to those of state tables in X.224 *2: TCONreq for other TC *3: TDTreq which closes the transmit window *4: DT TPDU in normal sequence *5: AK TPDU which opens the transmit window *6: Retain TDTreq or TEXreq

Table2 Part of Minimum Behaviours of SP selected for Testing of TP

STATE / INPUT	Other SC is not connected					Other SC is connected				
	idle no TC	await TCONcnf	await AC	data transfer	await SRELrsp	idle no TC	await TCONcnf	await AC	dta transfer	await SRELrsp
	STA01-1	STA01B-1	STA02A-1	STA713-1	STA09-1	STA01-2	STA01B-2	STA02A-2	STA713-2	STA09-2
SCONreq	STA01B-1 TCONreq	*1				STA01B-2 TCONreq				
TCONcnf		STA02A-1 CN					STA02A-2 CN			
AC			STA713-1					STA713-2		
SDTreq				STA713-1 DT					STA713-2 DT	
SDTreq-last*2				STA713-1 DT					STA713-2 DT	
DT				STA713-1					STA713-2	
SEXreq				STA713-1 EX	STA09-1 EX				STA713-2 EX	STA09-2 EX
FN-nr				STA09-1					STA09-2	
SRELrsp+					STA01-1 DN TDISreq					STA01-2 DN TDISreq
TDISind		STA01-1	STA01-1	STA01-1	STA01-1		STA01-2	STA01-2	STA01-2	STA01-2
SCONreq-other*3	STA01-2 TCONreq-other	STA01B-2 TCONreq-other	STA02A-2 TCONreq-other	STA713-2 TCONreq-other	STA09-2 TCONreq-other					

Note *1: The entries which are left blank are not used. *2: SDTreq-last generate TDTreq-last. *3: SCONreq-other, SUABreq-other and ABnr-other are inputs for the other SC.

Table3 Part of Automaton Model for MLE Testing of Class2 Initiator

STATE / INPUT	N1	N2	N4	A11	A12	A21	A22	A31	A32	C1	C2
SCONreq	A11 NCONreq	A12	A22 CR	Ø*	Ø	Ø	Ø	Ø	Ø	Ø	Ø
SCONreq-other	N2 NCONreq	//	//	A12	//	A22	//	A32	//	C2	//
NCONcnf	//	N4	//	A21 CR	A22 CR	//	//	//	//	//	//
CC	//	//	N4 DR	//	//	B011 DT (CN)	B021 DT (CN)	C1 DR	C2 DR	C1	C2
DR	//	//	N4 DC	//	//	N1 NDISreq	N4	N1 NDISreq	N4	N1 NDISreq	N4

STATE / INPUT	B01			B02	B11	B12	B21		B22	B31	B32	
	STA02A	STA713	STA09	STA713	STA713	STA09	STA713	STA09	STA09	STA713	STA713	STA09
	B011	B012	B013	B022	B111	B122	B211	B212	B222	B311	B321	B322
SDTreq	Ø	B012 DT (DT)	Ø	B022 DT (DT)	B111 DT (DT)	Ø	B211 retain	Ø	Ø	B311 retain	B321 retain	Ø
SDTreq-last	Ø	B211 DT (DT)	Ø	B221 DT (DT)	B311 DT (DT)	Ø	//	Ø	Ø	//	//	Ø
SEXreq	Ø	B111 ED (EX)	B112 ED (EX)	B121 ED (EX)	B111 retain	B122 retain	B311 ED (EX)	B312 ED (EX)	B322 ED (EX)	B311 retain	B321 retain	B322 retain
SRELrsp+	Ø	Ø	C1 DT (DN) DR	Ø	Ø	C2 DT (DN) DR	Ø	C1 DR	C2 DR	Ø	Ø	C2 DR
DT=(DT)	Ø	B012 AK	Ø	B022 AK	B111 AK	Ø	B211 AK	Ø	Ø	B311 AK	B321 AK	Ø
DT=(FN)	Ø	B013 AK	Ø	B023 AK	B112 AK	Ø	B212 AK	Ø	Ø	B312 AK	B322 AK	Ø
AD-open	B011	B012	B013	B022	B111	B122	B012	B013	B023	B111	B121	B122

Note *: Ø denotes an entry which is not used in the MLE Testing.

Table4 DSs for TP Class2 Initiator

STATE	DS
N1	SCONreq
N2, A11, A12	NCONcnf, DR
N4, A21, A22, A31, A32	DC, CC, DR
B011, B021 (states corresponding to B01 and B02 of TP and to STA02A of SP)	DT=(AC), SEXreq, SDTreq, DR
B012, B022, B111, B121 (states corresponding to B01, B02, B11 and B12 of TP and to STA713 of SP)	SEXreq, SDTreq, DR
B211, B221 (states corresponding to B21 and B22 of TP and to STA713 of SP)	SDTreq, SEXreq, DR
B311, B321 (states corresponding to B31 and B32 of TP and to STA713 of SP)	DT=(FNnr), SEXreq, SRELrsp+, DR
B013, B023, B112, B122, B212, B222, B312, B322 (states corresponding to all states in data transfer phase of TP and to STA09)	SEXreq, SRELrsp+, DR
C1, C2	CC, DR

conformance parameters. P1 indicates whether NC is reused or not, P2 indicates whether Explicit Flow Control is used or not, and P3 through P6 indicate the procedure used in the Treatment of Protocol Errors.

In our IUT, NC is not reused, Explicit Flow Control is used, and Normal Release is used in the Treatment of Protocol Errors. As a result, the tuned up automaton uses 19 states other than N3, D1 and D2.

Table 2 shows part of the behaviours of SP initiator selected to test Class2 initiator. These behaviours use only 41 entries of BSS full duplex. The automaton model of Class2 initiator for MLE Testing is specified by combining these two automata. It has 29 states and 29 inputs. A part of the automaton model is described in Table3.

Table4 shows the DSs for Class2 initiator used in the experiment. These DSs are used in both DSE

```
Identification of State N1
Input : SCONreq
Refout: NCONreq
Output: NCONreq

Input : SUABreq          SCONreq-other
Refout: NDISreq          NCONreq
Output: NDISreq          NCONreq
Identification of State N2
Input : NCONcnf         DR
Refout:                 DC
Output:                 DC
Identification of State N3
Input : DC               CC               DR
Refout:                  DR               DC
Output:                  DR               DC

Input : NRSTind                           SCONreq
Refout: NRSTrsp          NDISreq          NCONreq
Output: NRSTrsp          NDISreq          NCONreq
Identification of State A11
Input : NCONcnf          DR
Refout: CR               NDISreq
Output: CR               NDISreq

Input : SCONreq          SCONreq-other
Refout: NCONreq
Output: NCONreq
Identification of State A12
Input : NCONcnf          DR
Refout: CR
Output: CR

Input : SCONreq          A3nr-other
Refout: CR
Output: ____                              An error is detected.
Identification of State A21
Input : DC               CC               DR
Refout:                  DT(CN)           DC                    NDISreq
Output:                  DR               DC                    _______
                         __

Identification of State 3011
Input : DT=(AC)          SEXreq           SDTreq          DR              NDISreq
Refout: AK               ED(EX)           DT(DT)          DC              NDISreq
Output: AK               ED(EX)           DT(DT)          DC

Input : SCONreq          NCONcnf          CC              DT=(AC)
Refout: NCONreq          CR               DT(CN)          AK
Output: NCONreq          CR               DT(CN)          AK
```

Fig3 Results in DSE Testing for TP Class2 Initiator

and RS testing methods. Fig.3 shows a part of the results in DSE testing method. The length of the test sequence is 4818 inputs. In RS testing method, the state of IUT cannot be transferred to the states 'Waiting before releasing' (A31 and A32), the reason being that no outputs are generated by TDISreq which causes the only possible state transitions to these states. The length of the test sequence in the RS testing method is 3488 inputs, and 1376 of them are SS primitives generated manually from the terminal.

4.3 MLE Testing of SP BSS

The general automaton for SP is specified, based on state tables in X.225, by the followings :

① The state tables in X.225 do not describe the behaviours that the error condition is recovered by AD or AI SPDU after ED SPDU is received while these SPDUs are discarded after ER SPDU is received. Different states should be specified to examine whether ED or ER SPDU is received.

② To avoid complex representation, the state tables in X.225 use predicates and actions with variables and boolean functions such as Vsc, II(t) etc. The behaviour to update the position of tokens is described by the actions. These descriptions with predicates and actions should be converted to states. However, due to the possible combinations of the values of the variables and positions of the tokens, the general automaton becomes too complex. Especially the possible combinations of the positions of four tokens bring severe complexity. In our approach, therefore, the general automaton of SP is specified under the certain limitation that all available tokens are owned by one side.

By this assumption, the general automaton of SP initiator has 110 states and 103 inputs. In this automaton, for example, STA713 (data transfer

Table 5 Part of Automaton Model of SP BSS Initiator

STATE	idle no TC*1 STA01	await TCONcnf*1 STA01B	await AC*1 STA02A	await TDISind*1 STA16	Does not have the right to issue SSYNmrsp (Vsc=false)							
					Owns all available tokens							
					await DN*1 STA03	await PR or MAA*1 STA04A	await PR or RA*1 STA05A			await RA after collision*1 STA06		
							abandon	restart	set	abandon	restart	set
INPUT	S01	S01B	S02A	S16	S031	S04A1	S05Aa1	S05Ar1	S05As1	S06a1	S06r1	S06s1
AC	//	//	S7131	S16	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr
SCONreq	S01B TCONreq	S01B	S02A	S16	S031	S04A1	S05Aa1	S05Ar1	S05As1	S06a1	A06r1	S06s1
TCONcnf	//	S02A CN	//	//	//	//	//	//	//	//	//	//
PRrs	//	//	S02A retain*3	S16	S15B1	S15B1	S06a1	S06r1	S06s1	S06a1 retain	S06r1 retain	S06s1 retain
RSs	//	//	S16 ABnr	S16	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S05Aa1	S11As1	S05As1
SGTreq	S01	S01B	S02A	S16	S031	S04A2 GT	S05Aa1	S05Ar1	S05As1	S06a1	S06r1	S06s1
DN	//	//	S16 ABnr	S16	S01 TDISreq	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr

STATE	Does not have the right to issue SSYNmrsp											
	Owns all available tokens								Owns no available tokens			
	await SSYNM rsp*1 STA10A	await SRSYNrsp*1 STA11A			await MAA after PR*1 STA15A	wait after PRrs*1 STA15B	wait after PRra*1 STA15C	data transfer*1 STA713	await RA after collision*1 STA06			await SREL rsp*1 STA09
		abandon	restart	set					abandon	restart	set	
INPUT	S10A1	S11Aa1	S11Ar1	S11As1	S15A1	S15B1	S15C1	S7131	S06a2	S06r2	S06s2	S092
SDTreq	S10A1 DT	S11Aa1	S11Ar1	S11As1	S15A1	S15B1	S15C1	S7131 DT	S06a2	S06r2	S06s2	S092 DT
MAA=*2	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S7131	S15B1	S15C1	S16 ABnr	S06a2	S06r2	S06s2	S16 ABnr
SSYNMreq=*2	S10A1	S11Aa1	S11Ar1	S11As1	A15A1	S15B1	S15C1	S04A1 MAP	S06a2	S06r2	S06s2	S092
RA=*2	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S7131	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr
SRSYNreq(s)	S05As1 PRrs RSs	S11Aa1	S05As1 PRrs RSs	S05As1 PRrs RSs	S05As1 PRrs RSs	S06s1 PRrs RSs	S15C1	S05As1 PRrs RSs	S06a2	S06r2	S06s2	S05As2 PRrs RSs
SGTreq	S10A2 GT	S11Aa1	S11Ar1	S11As1	S15A2 GT	S15B1	S15C1	S7132 GT	S06a2	S06r2	S06s2	S092

STATE	Does not have the right to issue SSYNmrsp				Has the right to issue SSYNmrsp (Vsc=true)							
	Owns no available tokens				Owns all available tokens				Owns no available tokens			
	await SSYNM rsp*1 STA10A	await MAA after PR*1 STA15A	wait after PRrs*1 STA15B	data transfer*1 STA713	await DN*1 STA03	await PR or MAA*1 STA04A	await SSYNM rsp*1 STA10A	data transfer*1 STA713	await PR or MAA*1 STA04A	await SREL rsp*1 STA09	await SSYNM rsp*1 STA10A	data transfer*1 STA713
INPUT	S10A2	S15A2	S15B2	S7132	S033	S04A3	S10A3	S7133	S04A4	S094	S10A4	S7134
MIP=*2	S16 ABnr	S16 ABnr	S15B2	S7134	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S7134
SSYNmreq=*2	S10A2	S15A2	S15B2	S7132	S033	S04A3	S10A3	S7131 MIP	S04A4	S094	S10A4	S7134
SSYNmrsp=*2	S10A2	S15A2	S15B2	S7132	S033	S04A3	S10A3MIA	S7133 M1A	S04A4	S094MIA	S10A4MIA	S7134MIA
GT	S10A1	S15A1	S15B2	S7131	S16 ABnr	S16 ABnr	S16 ABnr	S16 ABnr	S04A3	S16 ABnr	S10A3	S7133
SGTreq	S10A2	S15A2	S15B2	S7132	S033	S04A3	S10A4 GT	S7134 GT	S04A4	S094	S10A4	S7134

Note *1: These states correspond to the states of state tables in X.225. *2: = indicates SPDU or S primitive with correct serial number. *3: Retain the received SPDU.

state) of the state tables in X.225 is represented by 8 states according to the criteria such as whether Activity is in progress or not (i.e. Vact is true or false), whether there is the right to issue SSYNmrsp or not (i.e. Vsc is true or false), and whether there are the available tokens or not. The general automaton contains the following 12 static conformance parameters :

①10 parameters which indicate whether each functional unit is selected or not

②a parameter which indicates whether TC is reused or not

③a parameter which indicates whether Use of Transport Expedited Service is selected or not.

The tuned up automaton of SP initiator implementing BSS full duplex contains 68 states and 73 inputs. Table5 shows a part of the tuned up automaton.

The DSs for BSS initiator used in the experiment are described in Table6. Fig.4 shows a part of the results of DSE testing. The length of the test sequence for DSE testing is 20221 inputs. The length of test sequence for RS testing is 16032 inputs, and 1325 of them are SS primitives generated manually from the terminal.

Table 6 DSs for SP BSS Initiator

STATE	DS
S01	SCONreq
S01B	TCONcnf
S02A	RFnr
S031	SGTreq, DN
S033	GT, DN
S04A1	PRmaa, MAA=, MIA=, SGTreq, SDTreq
S04A3, S04A4	PRmaa, MAA=, SDTreq, SGTreq, SSYNmrsp=
S05Aa1	PRrs, RSa=, PRra, RA=, MIA=, SGTreq, SDTreq
S05Ar1	PRrs, RSr=, SRSYNrsp=, MIA=, SGTreq
S05As1	PRrs, RSs, RPrs, RSa=, SRSYNrsp=, MIA=, SGTreq
S06a1, S06a2	SRSYNreq(a), RSa=, PRra, RA=, MIA=, SGTreq, SDTreq
S06r1	SRSYNreq(a), RSr=, SRSYNrsp=, MIA=, SGTreq
S06r2	SRSYNreq(a), RSr=, SRSYNrsp=, MIA=, SGTreq, SDTreq
S06s1	SRSYNreq(a), RSs, PRrs, RSa=, SRSYNrsp=, MIA=, SGTreq
S06s2	SRSYNreq(a), RSs, PRrs, RSa=, SRSYNrsp=, MIA=, SGTreq, SDTreq
S092	SRELrsp-, MIA=
S094	SRELrsp-, SSYNmrsp=
S10A1	SSYNMrsp=, MIA=, SGTreq
S10A2	SSYNMrsp=, MIA=, SGTreq, SDTreq
S10A3, S10A4	SSYNMrsp=, SGTreq, SSYNmrsp=
S11Aa1	SRSYNreq(s), SRSYNrsp=, MIA=, SGTreq
S11Ar1	SRSYNreq(r)=, PRra, RA=, MIA=, SGTreq
S11As1	SRSYNreq(r)=, SRSYNreq(s), PRra, RA=, MIA=, SGTreq
S15A1, S15A2	MAA=, MIA=, SGTreq, SDTreq
S15B1	SRSYNreq(a), RSa=, PRra, RA=, MIA=, SGTreq
S15B2	SRSYNreq(a), RSa=, PRra, RA=, MIA=, SGTreq, SDTreq
S15C1	RA=, MIA=, SGTreq, SDTreq
S16	TIM
S7131, S7132	DT, MIA=, SGTreq, SDTreq
S7133, S7134	DT, SDTreq, SGTreq, SSYNmrsp=

4.4 Results and Discussions of the Experiments

(1) The TP and SP programs implemented in VAX11 /780 were tested using our testing method. As a result, some errors were detected related to Multiplexing in state CLOSED for TP Class2, and related to the incorrect serial number in SRSYNreq(r) primitive for SP BSS. These errors were detected because the model is specified in detail.

(2) The length of the test sequences in the Embedded Single-layer Testing method for TP Class2 is similar to that in SLE Testing, which is 5471[4]. The

```
Identification of State S01

Input : SCONreq
Refout: TCONreq
Output: TCONreq

Identification of State S013

Input : TCONcnf
Refout: CN
Output: CN

Identification of State S02A

Input : RFnr
Refout: TDISreq
Output: TDISreq

Input : SCONreq       TCONcnf        AC              SRELreq
Refout: TCONreq       CN                             FNnr
Output: TCONreq       CN                             FNnr

Identification of State S031

Input : SGTreq        DN
Refout:               TDISreq
Output:               TDISreq

Input : SCONreq       TCONcnf        AC              SSYNMreq=
Refout: TCONreq       CN                             MAP
Output: TCONreq       CN                             MAP

Identification of State S04A1

Input : PRmaa         MAA=           MIA=            SGTreq          SDTreq
Refout:                                              GT              GT
Output:                                              GT              GT

Input : SRSYNreq(a)
Refout: PRrs          RSa
Output: PRrs          RSa

Identification of State S05Aa1

Input : PRrs          RSa=           PRra            RA=             MIA=
Refout:
Output:

Input : SGTreq        SDTreq
Refout: GT            DT
Output: GT            DT

Input : GT            SRSYNreq(r)=
Refout:               PRrs           RSr
Output:               PRrs           RSr

Identification of State S05Ar1

Input : PRrs          RSr=           SRSYNrsp=                       MIA=
Refout:                              PRra            RA
Output:                              PRra            RA

Input : SGTreq
Refout: GT
Output: GT
```

Fig4 Results in DSE Testing for SP BSS Initiator

Embedded Single-layer Testing method can test all the behaviours except the behaviours for incorrect TS primitives. However, the test for incorrect TS primitives is not necessary if it is confirmed that SP does not generate incorrect TS primitives in the testing of SP. The Embedded Single-layer Testing method for TP therefore has a similar testing capability to SLE testing in practice.

(3) The two testing methods of DSE and RS have merits and demerits. Since RS is not required to implement Testing Aid in SUT, it is more applicable than DSE. However the coverage of test items is limited. On the other hand, the testing accuracy of DSE can be increased depending on the functionality of Testing Aid. Testing Aid with small program size and enough functionality is therefore required. In our experiments, the size of Testing Aid is 0.4 Kstep for TP and 1 Kstep for SP. We believe that these sizes are reasonable to implement in SUT.

(4) We introduced different and plural Distinguishing Sequences to identify each state of the automaton model. At this stage, we believe it possible to use these DSs in conformance testing.

5. Conclusions

In this paper, we have proposed the layer by layer conformance testing method for a multiple layer environment based on the automaton model, and demonstrated a conformance testing experiment for IUT which consists of OSI Transport Protocol Class2 and Session Protocol BSS. According to our conformance testing of TP in a multiple layer environment, the coverage of test items was not so different from that of the testing items in a single layer environment, and the test sequence is almost the same length. These results show the effectiveness of our layer by layer conformance testing method in MLE.

The key remarks for this testing are the followings :
(1) The automaton models are precisely defined by introducing new states, which are not currently described or which are indicated by predicates and actions, in the state tables of TP standard (X.224) and SP standard (X.225). By this method, the coverage of the testing is extended significantly

(2) The functionalities of our testing aid include the monitoring of PDUs and lower layer service primitives as well as the generation of the upper layer service primitives. It therefore becomes easier to manage the timing of the generation of upper layer service primitives in synchronization with Protocol Tester.

(3) Besides the check of the behaviours defined by state table, the checking of various parameters in PDU and service primitive is indispensable in conformance testing. Generally speaking, the model based on an automaton is not effective for parameter checking. However, our method performs some parts of parameter checking by specifying Multiplexing or Explicit Flow Control as separate states, and PDUs or primitives with incorrect sequence numbers as separate inputs in the automaton model.

(4) In order to cope with the pluralities for each implementation, we introduced the General and Tuned Up concepts in the automaton model. It is possible to define the general automaton in the case of TP Class2, while it is difficult to define for SP BSS, the main problem being inclusion of token positioning. We should therefore polish this concept properly.

Acknowledgments

The authors wish to thank Dr. H. Kaji, Former Director, Dr. K. Nosaka, Vice Director, Dr. K. Ono, Deputy Director, and Dr. Y. Urano, Manager of Information Processing Lab. of KDD R&D Laboratories, for their continuous support for this study.

References

[1] ISO, "Working Draft for OSI Conformance Testing Methodology and Framework," ISO TC97/SC21 N410, Feb.,1985.
[2] K. Suzuki, T. Kato and K. Ono "Implementation and Testing of Transport Protocol," the Second International Conference on the Introduction of Open Systems Interconnection Standards, Ottawa, Canada, May 1984.
[3] K. Suzuki et. al., "Implementation and Testing of OSI Protocols," the Third International Conference on the Introduction of Open Systems Interconnection Standards, Cambridge, United Kingdom, Sept. 1985.
[4] T. Kato and K. Suzuki, "Testing of OSI Transport Protocol based on Automaton Model", Technical Report of WGDPS Meeting, 24-5, IPSJ, Nov. 1984 (in Japanese).
[5] CCITT, Rec. X.214, X.224, Oct. 1984.
[6] CCITT, Rec. X.215, X.225, Oct. 1984.