NTP-SIOT: A Test Tool for Advanced Mobile Services

Yi-Bing Lin, National Chiao Tung University
Ching-Feng Liang, Kuei-Hui Chen, and Hsin-Yu Liao, ITRI

Abstract

This article describes system design for an Open Mobile Alliance Service Interoperability Test Platform called NTP-SIOT. Based on the TTCN-3 specifications, we show how 4G mobile applications such as OMA multimedia messaging service can be tested in this platform.

Existing mobile telecommunications networks mainly support traditional voice services. In the future, beyond third generation (B3G) and fourth generation (4G) mobile telecommunications networks will provision many advanced data applications. Before these mobile applications can be launched for service, it is essential to conduct testing in order to ensure that these applications are correctly implemented. Under the support of National Telecommunication Program (NTP) in Taiwan, this article describes system design for an Open Mobile Alliance (OMA) service interoperability test (SIOT) platform called NTP-SIOT. This platform supports test case development for advanced mobile services (i.e., 4G services) based on the TTCN-3 specifications [1, 2]. In NTP-SIOT, we have developed several OMA multimedia messaging service (MMS) [3, 4], and instant message and presence service (IMPS) test cases that have been approved by OMA. This article shows how applications such as MMS can be tested in this platform. Typically, the OMA test cases are split into two categories: *conformance* and *interoperability* test cases. The conformance test cases are used to verify the adherence to normative requirements described in the technical specifications. The interoperability test cases verify that implementations of the specifications work satisfactory. Our test cases support both conformance and interoperability tests.

This article describes how MMS conformance tests can be achieved in NTP- SIOT. We first briefly introduce the MMS service that utilizes the HTTP and short message service (SMS) technologies. Figure 1 illustrates the MMS architecture. In this architecture, the handset (see (1) in Fig. 1), connects to the MMS server (see (2) in Fig. 1) through the Wireless Application Protocol (WAP) gateway for content delivery, and through the Short Message Service Center (SMSC) for control message delivery.

The MMS service can be mobile originated (MO) or mobile terminated (MT). In the MO procedure, a handset (an MMS client; see (1) in Fig. 1) uses the HTTP `post` method to send a multimedia message to the MMS Server following path (a) in Fig. 1. In the MT procedure, the MMS server uses the SMS to notify the handset of a multimedia message arrival (path (b) in Fig. 1). The handset then uses the HTTP get method to retrieve the multimedia message (path (a) in Fig. 1). Figure 2 shows the NTP-SIOT environment for testing the above MMS procedures.

In this figure, the NTP- SIOT (see (2) in Fig. 2) acts as the MMS server in both the MO and the MT procedures. It connects to the tested handset (see (1) in Fig. 2) through the real mobile network or a network emulator such as Anritsu MD8470A [11]. In the MO test procedure, the NTP-SIOT waits for a multimedia message sending from the handset, and verifies if the multimedia message is correctly received. In the MT test procedure, the NTP-SIOT sends a multimedia message to the handset, and the tester (see (3) in Fig. 2) compares the received multimedia message and the original multimedia message to verify if the handset can receive and process the multimedia message correctly.

In this article we first describe the NTP-SIOT architecture. Then we show how the MO and the MT MMS test cases are implemented in this test platform.

TTCN-3 Test System

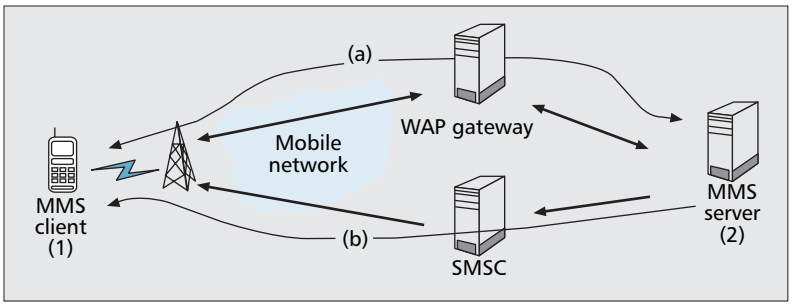
NTP-SIOT is a Testing and Test Control Notation version 3 (TTCN-3) test system. This system manages test execution, interprets or executes compiled TTCN-3 code, and realizes proper communication with the systems under test (SUT). As illustrated in Fig. 3, the TTCN-3 system consists of the following parts.

The Test Management and Control (TMC; see (1) in Fig. 3) is responsible for test execution control and test event logging. The TTCN-3 Executable (TE; see (2) in Fig. 3) is responsible for the interpretation or execution of the TTCN-3 modules (i.e., abstract test suites). The SUT Adapter (SA; see (3) in Fig. 3) adapts the TTCN-3 communication operations (of the TE) with the SUT (see (5) in Fig. 3). The Platform Adapter (PA; see (4) in Fig. 3) adapts the TE to a particular execution platform by creating a single notion of time for a TTCN-3 test system, and implementing external functions as well as timers.

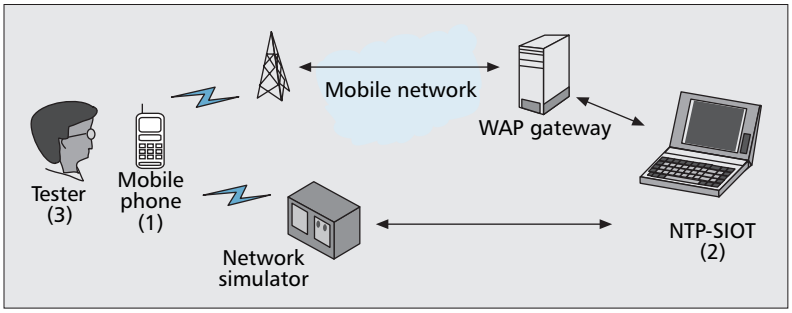
Test Management and Control (TMC)

The TMC consists of four entities [2]. The Test Management entity (TM; see (6) in Fig. 3) is responsible for overall management of the test system. After initiation, test execution starts with the TM that is responsible for invocation of TTCN-

0890-8044/07/$20.00 © 2007 IEEE



■ Figure 1. *The MMS architecture.*



■ Figure 2. *NTP-SIOT for MMS.*

3 modules (e.g., the MMS_Testcases_C_CT_Sending module consisting of a set of MMS sending test cases.) including propagating module parameters and/or extra testing information to the TE if necessary.

The Test Logging entity (TL; see (7) in Fig. 3) is responsible for maintaining the test log, for example, test component creation, start and termination, and data delivery to/from the SUT. The logging requests to the TL are posted externally from the TE or internally from the TM.

The External CoDecs (ECD; see (8) in Fig. 3) supports external codecs to the TE. The codecs are used for encoding and decoding of TTCN-3 values into bitstrings to be sent to the SUTs. The TE passes the TTCN-3 data to an appropriate encoder to produce the encoded data. The received data is passed to an appropriate decoder to translate the received data into TTCN-3 values. Note that if the TE does not use the built-in codecs (see EDS elaborated below), then the external codecs in the ECD entity are used. The external codecs can be used in parallel with, or instead of, the built-in codecs associated with the TE. A standardized interface is specified for porting the external codecs between different TTCN-3 systems and tools.

The Component Handling entity (CH; see (9) in Fig. 3) is responsible for distributing parallel test components. Specifically, the TE can be distributed among several test devices, and the CH implements communication among these test devices. Each *test device* includes the TE, SA, PA, CD, and TL entities. Both the CH and TM mediate the test management and test component handling between the TEs on different test devices.

TTCN-3 Executable (TE)

The TE is decomposed into three interacting entities (T3RTS, ETS, and EDS) to execute or interpret a TTCN-3 module [1]. These entities are described as follows.

The Executable Test Suite entity (ETS; Fig. 3 (10)) executes or interprets test cases, and handles the sequencing and matching of test events [5]. The ETS interacts with the T3RTS entity to send, receive, and log test events during test case execution, to create and remove TTCN-3 test components, as well as to handle external function calls, action operations, and timers. The ETS entity indirectly interacts with the SA through the T3RTS.

The Encoding/Decoding System entity (EDS; see (11) in Fig. 3) is responsible for the encoding and decoding of test data, which includes data used in communication operations with the SUT. If no encoding has been specified for a TTCN-3 module, the encoding of data values is tool specific. The EDS entity indirectly interacts with the SA through the T3RTS.

The TTCN-3 Runtime System (T3RTS; see (12) in Fig. 3) entity interacts with TM, SA, and PA, and manages the ETS and EDS entities. The T3RTS starts the execution of a test case or function in the ETS entity. It queries the TM entity for module parameter values required by the ETS and sends logging information to the TL entity. It also collects and resolves associated verdicts returned by the ETS entity [5]. The T3RTS entity instructs the SA to send messages or procedure calls to the SUT or the PA. It notifies the ETS entity of incoming messages or procedure calls from the SUT as well as timeout events. Before sending/receiving messages and procedure calls to/from the SA, or handling function calls and action operations in the PA for the ETS entity, the T3RTS invokes the EDS entity for their encoding or decoding. The T3RTS entity is also responsible for storing events sent from the SA (or the PA) to the TE, but have not been processed. The T3RTS entity maintains several port queues for input test events. Timeout events, which are generated by TTCN-3 timers, call timers, or test case timers, are kept in a timeout list specified in [6].

SUT Adapter (SA)

The SA adapts the communication between the TE and the SUT. It maps the TTCN-3 test component ports to Test System Interface (TSI) Ports and implements the real TSI (Fig. 3 (c)) defined in [5]. The SA is responsible for propagating the TE's requests and SUT action operations from the TE to the SUT, and notifying the TE of any received test events form the SUT by buffering them in the TE's port queues.

Platform Adapter (PA)

The PA implements TTCN-3 external functions and provides a TTCN-3 test system with a single notion of time. External functions and timers are implemented in this entity. Timers that have been declared in the TTCN-3 modules are *explicitly* classified in the TE. Timers that are created by the TE for guarding TTCN-3 procedure calls or execute operations are known in the TE as implicit timers. Both explicit and implicit timers created within the TE are implemented by the PA. Every timer is assigned a unique Timer Identification (TID). The PA treats both explicit and implicit timers in the same manner. The interface with the TE invokes external functions. It also starts, reads, stops timers, and queries the status of timers using their TIDs. The PA is responsible for notifying the TE of expired timers.

Interfaces in a TTCN-3 Test System

Two interfaces are defined in a TTCN-3 test system: the TTCN-3 Control Interface (TCI; Fig. 3a) and the TTCN-3 Runtime Interface (TRI; Fig. 3b). The TCI specifies the interface between TMC and TE. The TRI defines the interface between TE and SA/PA. All operation definitions are specified using the Interface Definition Language (IDL) [7, 8]. A

TCI/TRI operation call is an atomic operation invoked by the calling entity. The called entity implements the TCI/TRI operation, and returns control to the calling entity after the operation is performed.

TTCN-3 Control Interface (TCI)

A TCI interface is bidirectional where calling and called parts reside in the TE and in the TMC. The TCI consists of four subinterfaces [2]. The TCI Test Management Interface (TCI-TM) supports operations for managing test execution, providing module parameters and external constants, and offering test event logs. For example, the `tciStartTestCase` operation called by the TM starts a test case in the currently selected module of the TE with the given parameters.

The TCI Test Logging Interface (TCI-TL) includes operations for retrieving test execution information. For example, the `tliMReceive_m` operation is called by the TE to log a received message for the communication with the SUT. The TL presents the information provided in the parameters of this operation to the user.

The TCI Component Handling Interface (TCI-CH) consists of operations that implement the management and communication between TTCN-3 test components in a centralized or distributed test system. For example, the `tciCreateTestComponent` operation is called by the CH at the local TE when a remote TE requests to create a test component. The local TE creates the required TTCN-3 test component and passes its reference pointer back to the remote TE through the CH.

The TCI Coding/Decoding Interface (TCI-CD) provides operations to retrieve and access codecs. For MMS testing in NTP-SIOT, TCI-CD is implemented in JAVA with the following source files: `MMS_Codec.java` is the main codec program, `MMS_Constants.java` defines constants used in TCI-CD, `MMS_Encoder.java` implements encoding for MMS tests, and `MMS_Decoder.java` impelements decoding for MMS tests. The MMS encode operation is invoked by the TE to encode a value into a binary packet data unit (PDU) based on the encoding rules. Parts of the pseudo code are listed in Fig. 4.

In the MMS encode operation, if no encoding rule in Fig. 4 lines 1–4 are matched, then Fig. 4 line 5 is executed for exception handling.

The MMS decode operation invoked by the TE decodes a `message` according to the decoding rules and returns a TTCN-3 value. Parts of the pseudo code are listed in Fig. 5. In lines 1 and 2 of this operation, the message is decoded as a character string. In lines 3 and 4, the message is decoded as an MMS PDU. Line 5 determines the PDU type of `message` and its TTCN-3 parser. Line 6 invokes the function `decodePdu` to covert the MMS PDU into a TTCN-3 value.

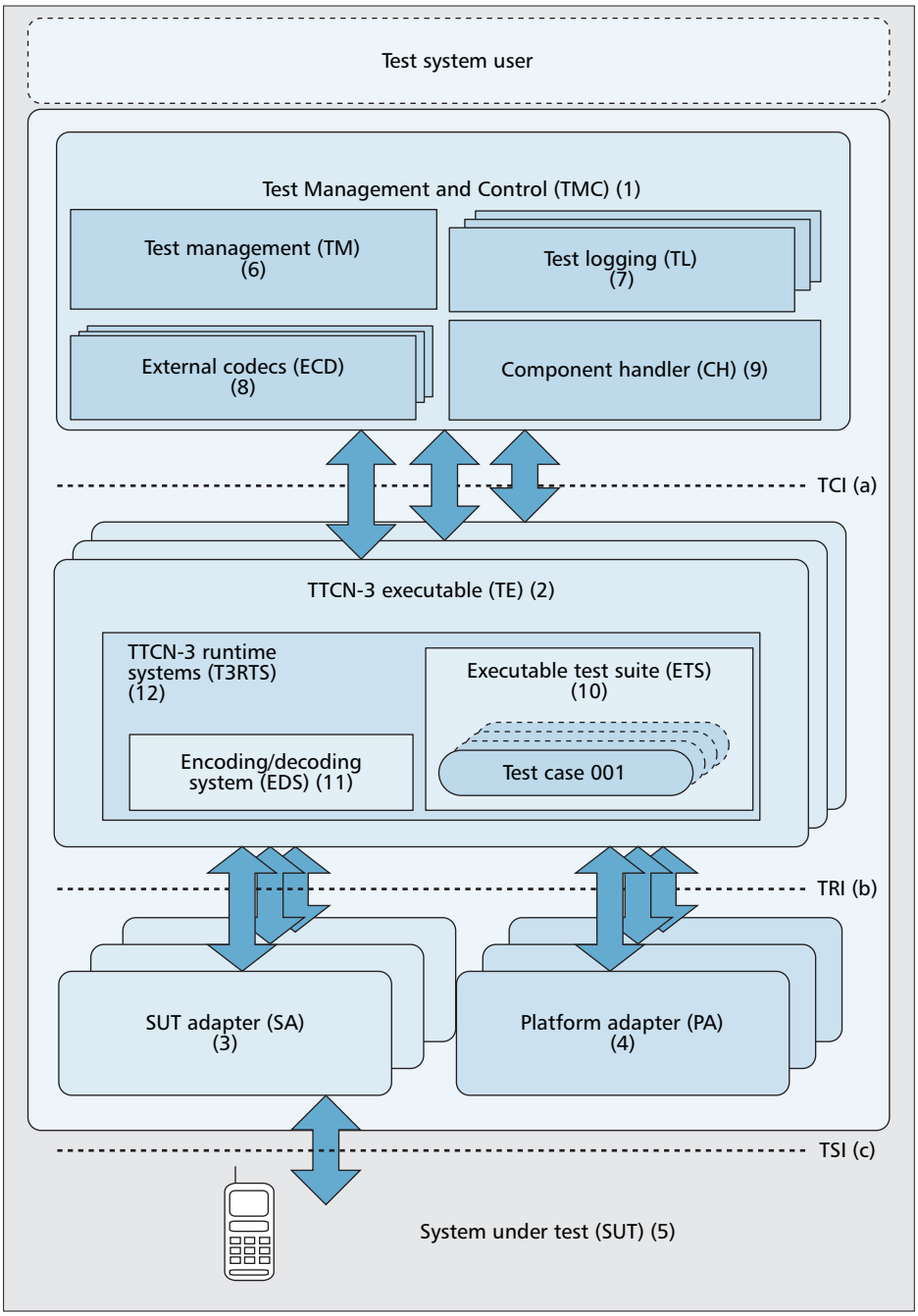


Figure 3. *The TTCN-3 system.*

TTCN-3 Runtime Interface (TRI)

The TRI consists of two subinterfaces, triCommunication and triPlatform. The triCommunication interface supports the communication of a TTCN-3 ETS with the SUT. This interface is implemented in the SA to initialize the TSI, establish connections to the SUT, and handle message or procedure based communication with the SUT. In addition, the triCommunication interface offers an operation to reset the SA.

The triPlatform interface represents a set of operations, which adapts an ETS to a particular execution platform. Specifically, it provides mechanisms to start, stop, read a timer, enquire its status, and add timeout

```
public TriMessage encode (Value value) {
      // Build binary packet data unit (PDU) from a TTCN-3 "value"
1.    if (the TTCN-3 value indicates "mSendReq") {
            create a PDU representing the MMS request message;
2.    } else if (the TTCN-3 value indicates "mSendConf") {
            create a PDU representing the OK message for confirmation;
3.    } else if (the TTCN-3 value indicates "mRetrieveConf") {
            create a PDU representing the retrieving info. request;
4. ...
      }
5.    if (an exception occurs during encoding) {
      }
}
```

Figure 4. *The MMS encode operation.*

```
// Decode message based on the decoding hypothesis
public Value decode (TriMessage message, Type hypothesis) {
1.  if (the hypothesis indicates that message is a char string) {
2.     directly decode the message as a character string;
    }
3.  if (the hypothesis indicates that message is an MMS PDU) {
4.     retrieve the encoded MMS PDU "pdu" from the message;
    }
5.  determine the type of pdu (e.g., send MMS request) and the
    corresponding parser "parser" (e.g., parser for send MMS request);
// invoke the decodePdu function to convert pdu into a TTCN-3 value
6.  decodePdu (parser, pdu, value);
7.  return value;
  }
```

Figure 5. *The MMS decode operation.*

```
testcase MMS_1_2_con_102 ( ) {
//runs on MMS _Server system MMS_Server
1.  map (s_p);
2.  start the TWait timer;
3.  if (s_p.receive ( ) == "MMS_sendreq_102_r") {
4.     stop the TWait timer;
5.     set verdict to "pass";
6.     s_p.send (MMS_sendconf_s);
7.  } else if (s_p.receive ( ) == "MMS_sendreq_any") {
8.     stop the TWait timer;
9.     set verdict to "fail";
10.    s_p.send (MMS_sendconf_s);
11.}  else if (s_p.receive) {
12.    stop the TWait timer;
13.    set verdict to "fail";
14.    s_p.send (MMS_sendconf_err_s);
15.} else if (TWait times out) {
16.    set verdict to "inconc";
    }
}
```

Figure 6. *MO MMS test case.*

events to the expired timer list. In addition, it offers operations to call TTCN-3 external functions and to reset the PA.

In the TRI operations, the TE is responsible for encoding test data to be sent and decoding received test data. Explicit error handling is specified only for TRI operations called by the TE. The SA or PA reports the status of a TRI operation in the return value of a TRI operation. The TE may react to an error that occurred either within the SA or PA and issue a test case error. For TRI operations called by the SA or PA, no explicit error handling is required, since these operations are implemented in the TE. For MMS testing in NTP-SIOT, the TRI is implemented in a JAVA program `MMS_TestAdapter.java`.

In this program, the *connection* operations are implemented to:

- Resolve TRI communication operations on TSI ports that have mappings to multiple test component ports
- Provide multicast communication in TCI
- Pass information about the TSI and connections from the TE to the SA
- Support initialization after the invocation of a TTCN-3 test case

An example is triMap, which is called by the TE when it executes a TTCN-3 map operation. This operation instructs the SA to establish a dynamic connection to the SUT for the referenced TSI port. For MMS testing in NTP-SIOT, two ports are defined in the TE. The server port `s_p` is used when the test system is viewed as an HTTP server. The client port `c_p` is used when the test system is viewed as an HTTP client.

The *communication* operations are used to:

- Define encoded test data
- Indicate a source or destination address within the SUT or multicast communication in TRI
- Support procedure-based TRI communication operations

An example is `triEnqueueMsg` called by the SA after it has received a message from the SUT. This operation passes the message to the TE, indicating the component where the TSI port is mapped. Decoding of received message is done in the TE. Another example is `triSend`, called by the TE when it executes a TTCN-3 unicast send operation on a component port, which has been mapped to a TSI port. This operation instructs the SA to send a message to the SUT. For MMS testing in NTP-SIOT, two messages are sent by `triSend`: response message such as `m-send-conf` (to confirm the multimedia message delivery), and push message such as `m-notication-ind` (to notify the arrival of multimedia message; this push message is sent to the handset through the SMS) [9].

The *timer* operations are used to specify timer identifiers, and timer durations in seconds. The *miscellaneous* operations are used to specify the names of a test case or an external function in a TTCN-3 module, and indicate the success or failure of a TRI operation. An external function example is `viewSmilFile`, which allows viewing the SMIL file with provided SMIL player:

```
public void viewSmilFile(CharstringValue playerLocation, CharstringValue smilFileLocation)
```

where `playerLocation` indicates the location of the SMIL player program, and `smilFileLocation` indicates the location of the SMIL file to be played. Another external function is `getCurrentTime` that allows a test case to retrieve the current system time.

MMS Conformance Test Scenarios

We use mobile-originated (MO) MMS and mobile-terminated (MT) MMS to show how MMS test suits are implemented in NTP-SIOT.

An MMS MO test case is illustrated in Fig. 6. This test case verifies that messages with the SMIL layout portrait (a text followed by an image) is correctly sent from the client (the handset under test; i.e., the SUT). The TE first maps the main test component port to the system server port `s_p` (Fig. 6, line 1); the `triMap` operation at the TRI is invoked). Then it pops up an action window that instructs the tester (see (3) in Fig. 2) to send a multimedia message from the client (see (1) in

```
testcase MMS_1_2_con_201 ( ) {
//   runs on MMS_Mixed system MMS_Mixed
1.   map (c_p);
2.   c_p.send (MMS_notification_201_s);
3.   unmap (c_p_);
4.   map (s_p);
5.   start the TWait timer;
6.   if (s_p.receive ( ) ) {
7.      stop the TWait times;
8.      s_p.send (MMS_retrieveconf_201_s);
9.   } else if (TWait times out) {
10.     set verdict to "inconc";
11.        return;
     }
12.  if (the received message matches the sent message) {
13.     set verdict to pass;
     } else {
14.     set verdict to "fail");
     }
}
```

■ Figure 7. *MT MMS test case.*

Fig. 2). Immediately after the action window is pooped up, the `TWait` timer starts at the PA (Fig. 6, line 2). The TE receives a message from the client. One of the following four situations occurs.

1. The received message matches the `MMS_sendreq_102_r` template (lines 3–6, Fig. 6; the NTP-SIOT receives a correct multimedia message): When the `s_p.receive` function is executed (Fig. 6, line 3), the TE invokes the decode operation in Fig. 5. After the message is decoded (through the `decodePdu` fuction in Fig. 5, line 6), the `MMS_sendreq_102_r` template is matched. This template declares the matching condition and requested value of each field in a SMIL file description command

```
smil    :=    {{smil_top_left_r("text,"0,0),
smil_left_r("img,"0)}}
```

   which represents a text followed by an image. The test case stops the TWait timer at the PA (Fig. 6, line 4), and sets verdict to pass (Fig. 6, line 5). It then sends the client a message encoded using the `MMS_sendconf_s` template (Fig. 6, line 6). This template is used to encode an "OK" message. Note that when function `s_p.send` is invoked, line 2 in Fig. 4 is executed to encode this message.
2. The received message matches the `MMS_sendreq_any` template (Lines 7–10, Fig. 6). This template indicates thtat NTP-SIOT receives a multimedia message with unexpected content. The test case stops the `TWait` timer, and delivers a message encoded by the `MMS_sendconf_s` template (the "OK" message) to the client. It then sets verdict to `fail`.
3. The TE receives other type of PDU from the client (lines 11–14, Fig. 6): the test case stops the `TWait` timer, and delivers an error message "unknown message format" encoded by the `MMS_sendconf_err_s` template to the client. It then sets verdict to `fail`.
4. The TE receives nothing from the client (lines 15 and 16): the PA notifies the TE that `TWait` is expired (Fig. 6, line 15). The test case sets verdict to `inconc` (Fig. 6, line 16); indicating that an inconclusive exception occurs).

In situations 1–3, the SA receives the test event (the multimedia message) from the client, and passes it to the TE through the `triEnqueueMsg` operation. This test event is buffered in the TE's port queue. The TE then invokes the `decode` operation at the CD to obtain the decoded message. After the decoded message is processed, the TE invokes the encode operation at the CD to encode the return value, and then invokes the `triSend` operation to deliver the encoded message to the client through the SA.

Figure 7 illustrates the MMS MT test case, which verifies that a multimedia message is correctly received by the client (the handset under test), and that the received message is reasonably presented without error. The TE first maps the main test component port to system client port `c_p` (Fig. 7, line 1), and then sends an SMS message encoded using the `MMS_notification_201_s` template (Fig. 7, line 2) to the client through path (b) in Fig. 1. The test case pops up an action window that instructs the tester to retrieve the message at the client. The system client port is unmapped (Fig. 7, line 3) and then remapped to system server port `s_p` (Fig. 7, line 4). The `TWait` timer is started (Fig. 7, line 5). If NTP-SIOT receives a response from the client (Fig. 7, line 6), then `TWait` is stopped (Fig. 7, line 7), and a return value encoded by the `MMS_retrieveconf_201_s` template is sent back to the client (Fig. 7, line 8); note that when the function `s_p.send` is called, line 3 of the encode operation in Fig. 4 are executed. If `TWait` is expired, then set verdict to `inconc` and the test case is stopped (Fig. 7, lines 9–11). Figure 7, lines 12–14 check if the multimedia message is correctly delivered to the handset.

Conclusions

This article has described the architecture and operations of NTP-SIOT, an MMS test system that was developed based on the TTCN-3 specifications. This system has been jointly developed by the National Telecommunication Program (NTP) and the Industrial Technology Research Institute (ITRI) in Taiwan. We used the MO and MT MMS procedures to illustrate how conformance tests can be implemented and conducted in NTP-SIOT. Currently, six IMPS tests cases developed in NTP-SIOT have been approved by OMA (for IMPS 1.2 and 1.3), including

`OMA-IOP-IMPS-2005-0052-Group-Max-Active-User,`
`OMA-IOP-IMPS-2005-0053-PRES-Mood`
`OMA-IOP-IMPS-2005-0054-PRES-Language`
`OMA-IOP-IMPS-2005-0055-PRES-Address`
`OMA-IOP-IMPS-2005-0056-PRES-Location`
`OMA-IOP-IMPS-2005-0051-Group-history-functionality [10]`

Acknowledgment

This work was sponsored in part by NSC Excellence projects NSC 94-2752-E-009-005-PAE, NSC 94-2219-E-009-001, and NSC 94-2213-E-009-104, NTP VoIP Project under grant number NSC 94-2219-E-009-002, NTP Service IOT Project under grant number NSC 94-2219-E-009-024, Intel, Chung Hwa Telecom, IIS/Academia Sinica, ITRI/NCTU Joint Research Center, and MoE ATU.

References

[1] ETSI, "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)," ES 201 873-5 V3.1.1, 2005.
[2] ETSI, "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 6: TTCN-3 Control Interface (TCI)," ES 201 873-6, V3.1.1, 2005.
[3] OMA, "Enabler Test Specification for MMS 1.2," OMA-IOP-ETS-MMS-V1_2-20040409-A, 2004.
[4] Y.-B. Lin and A.-C. Pang, *Wireless and Mobile All-IP Networks*, Wiley, 2005.
[5] ETSI, "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language," ES 201 873-1, V3.1.1, 2005.
[6] ISO/IEC, "Information Technology — Open Systems Interconnection — Conformance Testing Methodology and Framework — Part 3: The Tree and Tabular Combined Notation (TTCN)," ISO/IEC 9646-3, 1998.
[7] C.-M. Chou *et al.*, "CCL OSA: A CORBA-based Open Service Access System," *Int'l. J. Wireless and Mobile Comp.*, 2005.

[8] I. Pyarali and D. C. Schmidt, "An Overview of the CORBA Portable Object Adapter," *ACM Std. View Mag.*, vol. 6, no. 1, Mar. 1998.
[9] OMA, "Multimedia Messaging Service Encapsulation Protocol," OMA-MMS-ENC-V1_2-20050301-A, 2005.
[10] Open Mobile Alliance, Enabler Test Specification for OMA IMPS CSP, OMA-ETS-IMPS_CSP-V1_2_1-20051115-A, 2005.
[11] Anritsu Corp., MD8470A Signaling Tester Product Introduction, http://www.us.anritsu.com/products/ARO/North/Eng/showProd.aspx?ID=659

Biographies

YI-BING LIN [M'96, SM'96, F'04] (liny@csie.nctue.du.tw) is chair professor and vice president (dean) of Research and Development, National Chiao Tung University, Taiwan. He is also with the Institute of Information Science, Academia Sinica, Nankang, Taipei, Taiwan. His current research interests include mobile computing and cellular telecommunications services. He has published more than 200 journal articles and more than 200 conference papers. He is a co-author of the books *Wireless and Mobile Network Architecture* (with Imrich Chlamtac; Wiley, 2001) and *Wireless and Mobile All-IP Networks* (with Ai-Chun Pang; Wiley, 2005). He is an ACM Fellow, AAAS Fellow, and IET/IEE Fellow.

CHING-FENG LIANG received M.S. degree in electronic engineering from National Taiwan University of Science and Technology (NTUST) in 1993 and joined the Information and Communication Laboratory (ICL) of Industrial Technology Research Institute (ITRI) as an engineer. He has led more than 10 projects of Taiwan Ministry of Economic Affairs (MoEA) to study and develop the technologies of mobile network and services including GPRS/3G core network, WLAN/Cellular interworking and number portability service. He received the ITRI Award in 2005 and the Outstanding Project Award of Taiwan MoEA in 2003. He is currently the manager of the Core Network Department of ICL/ITRI.

KUEI-HUI CHEN is a senior engineer at ICL, ITRI since 2000. She received B.S. and M.S. degrees from the Department of Computer Science and Engineering, Yuan Ze University in 1998 and 2000. Her research interests include mobile communication, wireless network, embedded system, and distributed computing. She is leading the project of service interoperability testing (SIOT) platform development.

HSIN-YU LIAO is currently a senior engineer at ICL, ITRI. She received her bachelor degree from Computer Science and Engineering Department at Yuan-Ze University in 1995. Her research interests include embedded system, network communication, and multimedia software technology.