WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2010; **10**:230–240
Published online 7 April 2009 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/wcm.757

ITRI-WiMAXT: A WiMAX conformance testing tool

Hsin-Yi Lee[1], Yi-Bing Lin[1*,†], Ching-Feng Liang[2] and Shiang-Ming Huang[1]

[1]*Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC*
[2]*Computer and Communications Research Laboratories, Industrial Technology Research Institute, Taiwan, ROC*

Summary

The WiMAX Forum certifies and promotes the compatibility and interoperability of broadband wireless products where many testcases need to be developed for WiMAX procedures. In this paper, we develop a conformance test tool called ITRI-WiMAXT based on the TTCN-3 specifications. Then we show how WiMAX procedures are tested in ITRI-WiMAXT. Based on ITRI-WiMAXT, we have proposed several test procedures accepted by the WiMAX Forum. Copyright © 2009 John Wiley & Sons, Ltd.

KEY WORDS: conformance test; testing and test control notation version 3 (TTCN-3); worldwide interoperability for microwave access (WiMAX)

1. Introduction

Worldwide interoperability for microwave access (WiMAX) is a broadband wireless system based on the IEEE 802.16 standard, which supports high-speed wireless access to provide fixed and mobile broadband services over large coverage areas [1,2].

A simplified WiMAX network reference model is shown in Figure 1. The model is composed of three parts: *mobile stations* (MS; Figure 1 (1)), *access service network* (ASN; Figure 1 (2)) and the *connectivity service network* (CSN; Figure 1 (3)). The ASN comprises *base stations* (BSs; Figure 1 (4)) and *ASN gateways* (ASN-GW; Figure 1 (5)). A BS communicates with the MSs through radio links. Each BS is connected to an ASN-GW through the R6 interface. The BSs also communicate with each other for handover of data connections. The ASN-GW performs user data management for an MS through the BS. The CSN connects to the ASN to offer internet protocol (IP) connectivity. The entities residing in the CSN include the *AAA server* (Figure 1 (6)) that performs authentication, authorization and accounting, the *home agent* (HA; Figure 1 (7)) that provides mobile IP functionality [3] and the *DHCP server* (Figure 1 (8)) that allocates the IP addresses.

The WiMAX Forum certifies and promotes the compatibility and interoperability of broadband wireless products where many testcases need to be developed for WiMAX procedures. With the support of National Telecommunications Program (NTP) and Industrial Technology Research Institute (ITRI), we have proposed several test procedures [4,5] accepted by the WiMAX forum.

Based on the Testing and Test Control Notation version 3 (TTCN-3), this paper develops a conformance test tool called ITRI-WiMAXT. TTCN-3 is a high-level testing language used by European Telecommunication Standards Institute (ETSI) and WiMAX Forum

*Correspondence to: Yi-Bing Lin, Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC.
†E-mail: liny@csie.nctu.edu.tw; jasonyblin@gmail.com

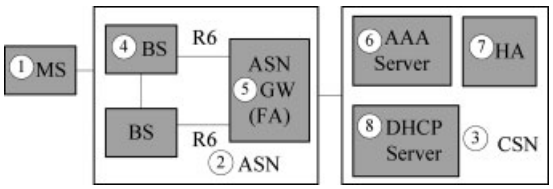

Fig. 1. A simplified WiMAX network reference model.

to create publicly available standardized test specifications. It precisely describes the test behaviour by defining the meaning of 'pass' and 'fail' for a testcase. This paper will show how WiMAX R6 message flows are tested in ITRI-WiMAXT. Note that based on the same tool, we have also developed conformance test tools for open mobile alliance (OMA) push-to-talk [6], download service [7], multimedia messaging service, instant message and presence service and browsing [8].

The paper is organized as follows. Section 2 describes the WiMAX message format and the initial network entry procedure. Section 3 introduces the TTCN-3 test system. Section 4 proposes the ITRI-WiMAXT test system. Section 5 provides our conclusions.

2. WiMAX Message Format and Initial Network Entry Procedure

The format of the WiMAX messages delivered in interface R6 is illustrated in Figure 2, which contains a message header and a message body consisting of several *Type-Length-Values* (TLVs). The TLVs encodes *Information Elements* (IEs) to specify the parameters of a WiMAX message. For example, the MS Info IE includes parameters such as the MAC address and the authenticator ID of the MS. The WiMAX message header consists of the following fields:

- The version field (8 bits) indicates the protocol version number. The current version number is 1.

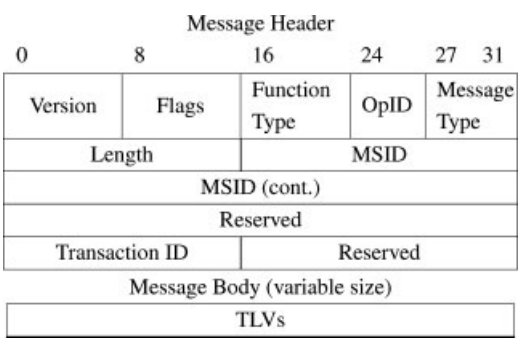

Fig. 2. WiMAX message format.

Table I. Function and message types (a partial list).

Function type	Message type
8 (authentication relay)	2 (AuthRelay_EAP_Transfer)
9 (MS state)	1 (MS_PreAttachment_Req)
	2 (MS_PreAttachment_Rsp)
	3 (MS_PreAttachment_Ack)
	4 (MS_Attachment_Req)
	5 (MS_Attachment_Rsp)
	6 (MS_Attachment_Ack)
	7 (Key_Change_Directive)
	9 (Key_Change_Ack)

- The flag field (8 bits) consists of two flags represented by bits. Bit 8 is set if the next expected transaction ID is reset. Bit 7 is set if the source and the destination identifier TLVs are included in the message.
- The function type field (8 bits) indicates the function type (see Table I).
- The OpID field (3 bits) indicates the operation type, which can be 001 (request/initiation), 002 (response), 003 (Ack) or 004 (indication).
- The message type field (5 bits) indicates the message type (see Table I). A function type and a message type uniquely determine a WiMAX message. For example, function type 9 and message type 1 represent the MS_PreAttachment_Req message (this message informs the ASN-GW that an MS attempts to enter the WiMAX network).
- The length field (16 bits) indicates the size of the message in bytes including the header.
- The MSID field indicates the 6-byte MAC address of the MS.
- The reserved fields are set to 0.
- The transaction ID field (16 bits) identifies the messages that are delivered in the same transaction.

The TLV format is shown in Figure 3. The type field is a numeric code that represents an IE. The length field specifies the size of the value portion in octets. The value field contains data for this IE (e.g., the MS MAC address for the MS Info IE). The value field itself may contain nested TLVs. Therefore, a WiMAX message

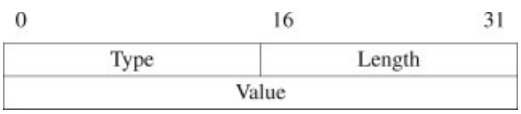

Fig. 3. WiMAX information element format.

DOI: 10.1002/wcm

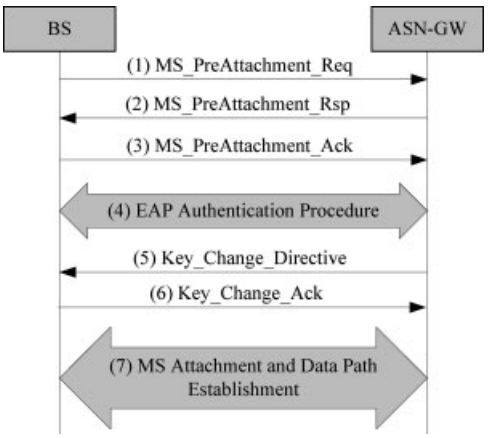

Fig. 4. The initial network entry procedure.

can be represented as a regular expression [9]:

$$W = HS^*, \quad S = T_S L V | T_N L S^* \tag{1}$$

where W denotes a WiMAX message, H and B denote WiMAX message header and body, T_S and T_N denote the types for single and nested IE, L denotes the length for this IE and V denotes the data for a single IE. This regular expression will be used to decode the WiMAX message in Section 4.

The WiMAX entities interact through a client–server model, where a WiMAX entity plays both server and client roles. We use the initial network entry procedure as an example to illustrated how the WiMAX messages are exchanged between the BS and the ASN-GW. This procedure is executed when an MS enters the WiMAX network [10]. Figure 4 shows the message flow, and the details are given below.

Step 1. When an MS enters the WiMAX network, the BS sends an MS_PreAttachment_Req message to the ASN-GW. In this message, the function type is 9, the message type is 1 and the OpID is 001. The flags are set to 0 since the next transcation ID will not be reset and the source/destination identifier TLVs are not included in this message. The IEs of the message are the MS Info (mandatory) and the BS Info (optional).

Step 2. The authenticator in the ASN-GW creates a new context block related to this MS and responds to the BS with an MS_PreAttachment_Rsp message. In this message, the function type is 9, the message type is 2, the OpID is 002 and the flags are set to 0. The IEs are the MS Info and the BS Info.

Step 3. The BS sends an MS_PreAttachment_Ack message to the authenticator to confirm that the BS has started the negotiation with the MS (the MS is not shown in this figure). In this message, the function type is 9, the message type is 3, the OpID is 003, the bit 8 of the flag is set to 1 and the bit 7 of the flag is set to 0. There is no IE in this message. Note that Steps 1–3 are a transaction initiated by the BS, where the BS is the client and the ASN-GW is the server.

Step 4. The extensible authentication protocol (EAP) authentication process is performed between the AAA server, the authenticator in the ASN-GW and the MS (via the BS).

Step 5. The authenticator sends a Key_Change_Directive message to the BS to indicate the completion of the EAP authentication process.

Step 6. The BS acknowledges with a Key_Change_Ack message, and performs mutual authentication with the MS (not shown in this message flow) [11]. Note that Steps 5 and 6 are a transaction initiated by the ASN-GW, where the ASN-GW is the client and the BS is the server.

Step 7. The BS and the authenticator perform the MS attachment to complete the MS registration. Then the ASN-GW establishes the data path.

In this paper, we will show how the above WiMAX procedure can be tested in ITRI-WiMAXT.

3. TTCN-3 Test System

The TTCN-3 test system architecture is illustrated in Figure 5, where the system is conceptually divided into several entities that interact with each other during the test execution. This section briefly describes the TTCN-3 entities and interfaces related to ITRI-WiMAXT. Details of the abstract TTCN-3 architecture can be found in References [6,7,12].

3.1. TTCN-3 Test System Entities

The *test management* (TM; Figure 5 (a)) is responsible for overall management of the system. When the system is initialized, test execution starts with the TM. This entity interacts with the test system user (Figure 5 (i)), and provides test event logs to the user. The TM is also responsible for the invocation of TTCN-3 modules that define an executable test suite. The TM propagates the

DOI: 10.1002/wcm

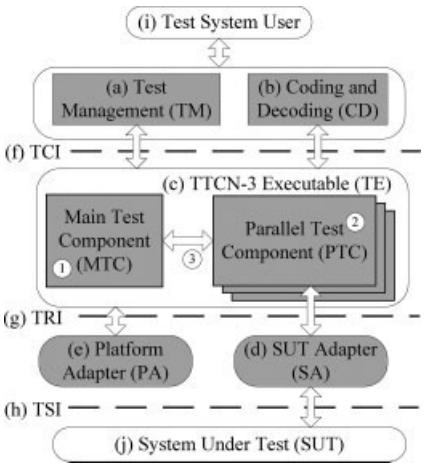

Fig. 5. The TTCN-3 test system architecture.

module parameters such as *implementation extra information for testing* (e.g., the IP address of the ASN-GW and the timer values) to the *TTCN-3 executable* (TE; see Figure 5 (c)). The *coding and decoding* (CD; see Figure 5 (b)) is responsible for encoding and decoding the data associated with message- or procedure-based communications. The codecs in the CD are implemented in JAVA or C. Therefore, they can be easily ported to different test systems.

The TE is responsible for the interpretation and execution of the test system modules. In the TE, the *main test component* (MTC; Figure 5 (1)) is responsible for the testcase execution. During the execution, the MTC may create the *parallel test components* (PTCs; see Figure 5 (2)), and control these PTCs through the test component commands, e.g., **create**, **start** and **stop** (Figure 5 (3)). Each of the PTCs handle one or more interfaces with the *system under test* (SUT; see Figure 5 (j)). The PTCs are executed concurrently. For example, the TTCN-3 test system may play both roles as the BS and the target ASN-GW to interact with the serving ASN-GW under the test (the SUT) [10]. In this case, the BS and the target ASN-GW are run on two individual PTCs, and the messages exchanged through different interfaces towards the serving ASN-GW are mapped and processed by the corresponding PTCs. The MTC collects the PTCs' verdicts and determines a global, final verdict for the whole test. Testcase execution ends when the MTC terminates.

The *SUT adapter* (SA; Figure 5 (d)) bridges the test components and the SUT through the *test system interface* (TSI; Figure 5 (h)). The SA is implemented in JAVA or C. It propagates the test component's requests, and then notifies the test component of any response messages from the SUT. The *platform adaptor* (PA; Figure 5 (e)) adapts the TE to a particular execution platform. It integrates external functionality such as ciphering function and provides the realization of timers (i.e., to start/stop a timer, and to notify the TE when a timeout occurs).

Table II. Examples of TTCN-3 communication operations.

Operation	Description
send	To send a message to the SUT through a port
receive	To receive a message from the SUT through a port
map	To set up a connection to the SUT or a test component by mapping the ports
unmap	To break a connection by releasing the corresponding port

3.2. TTCN-3 Test System Interfaces

The *TTCN-3 control interface* (TCI; Figure 5 (f)) [13] and the *TTCN-3 runtime interface* (TRI; Figure 5(g)) [14] are defined as sets of interface operations. These interface operations implement the TTCN-3 operations defined in References [12] and [15]. For example, the TTCN-3 **send** communication operation invokes the **triSend** interface operation to send a message to the SUT through a port. Ports provide connection between a test component and the SUT. Some of the TTCN-3 communication operations are listed in Table II. The TM/CD interact with the TE through the TCI. Some of the TCI interface operations invoked by the TE are described as follows:

- The **tciGetModulePar** operation accesses the values of the module parameters.
- The **encode** operation encodes a TTCN-3 value into, e.g., a WiMAX message to the SUT.
- The **decode** operation decodes a message into a TTCN-3 value.

The TRI defines a set of SA interface operations. Some of them invoked by the TE are described as follows:

- The **triExecuteTestcase** operation is called immediately before a testcase execution. The operation is used to set up, e.g., the parameters of the testcase.
- The **triMap** operation is invoked by the TTCN-3 **map** operation to establish a connection to the SUT.
- The **triUnmap** operation is invoked by the TTCN-3 **unmap** operation to close the SUT connection, which will release the network resources.

DOI: 10.1002/wcm

```
(1)  testcase mtc_InitialNetworkEntry () runs on WimaxMtc system WimaxInterface{
         var WimaxBS bs;
(2)      bs := WimaxBS.create;
(3)      map(bs:client, system:bs_client);
(4)      map(bs:server, system:bs_server);
(5)      bs.start(tcc_initialNetworkEntry());
(6)      timer t_initialNetworkEntry;
         t_initialNetworkEntry.start(1.0);
         alt{
(7)          [] all component.done {
                 t_initialNetworkEntry.stop;
             };
(8)          [] t_initialNetworkEntry.timeout {
(9)              all component.stop;
(10)             setverdict(inconc);
             };
         };
(11)     unmap(bs:client, system:bs_client);
(12)     unmap(bs:server, system:bs_server);
     }
(13) type component WimaxMtc {}
     type component WimaxInterface {
(14)     port WimaxPort bs_client;
(15)     port WimaxPort bs_server;
         ... //ports for the target ASN-GW, DHCP server, and so on
     };
     type component WimaxBS {
(16)     port WimaxPort client;
(17)     port WimaxPort server;
     }
```

Fig. 6. The *mtc_InitialNetworkEntry* testcase.

- The **triSend** operation is invoked by the TTCN-3 **send** operation to send an encoded message to the SUT.

An example of the TRI interface operations invoked by the SA is the **triEnqueueMsg** operation that buffers a message from the SUT. It is called immediately after the SA has received a message.

4. ITRI-WiMAXT Test System

Based on the TTCN-3 platform described in the previous section, this section shows how the ITRI-WiMAXT implements TE, TM, CD and SA to test the WiMAX-specific behaviour. We use the *mtc_InitialNetworkEntry* testcase in Figure 6 to illustrate how ITRI-WiMAXT works. This test case verifies the initial network entry procedure illustrated in Figure 4. Figure 7 shows the graphical test log for initial network entry procedure and the interaction between the MTC (Figure 7 (1)), PTC (component 1; Figure 7 (2)) and the SUT (system; Figure 7 (3)). Each match box in Figure 7 indicates that the received message matches the pass criteria. The final pass box indicates that the SUT passes this test case.

4.1. ITRI-WiMAXT Test Executable

Figure 8 shows the relationship among the components in ITRI-WiMAXT TE using the unified modelling language [16]. In Figure 6, the test system plays the role of a BS, and the SUT is the serving ASN-GW. The MTC executes on a test component of type WimaxMtc

DOI: 10.1002/wcm

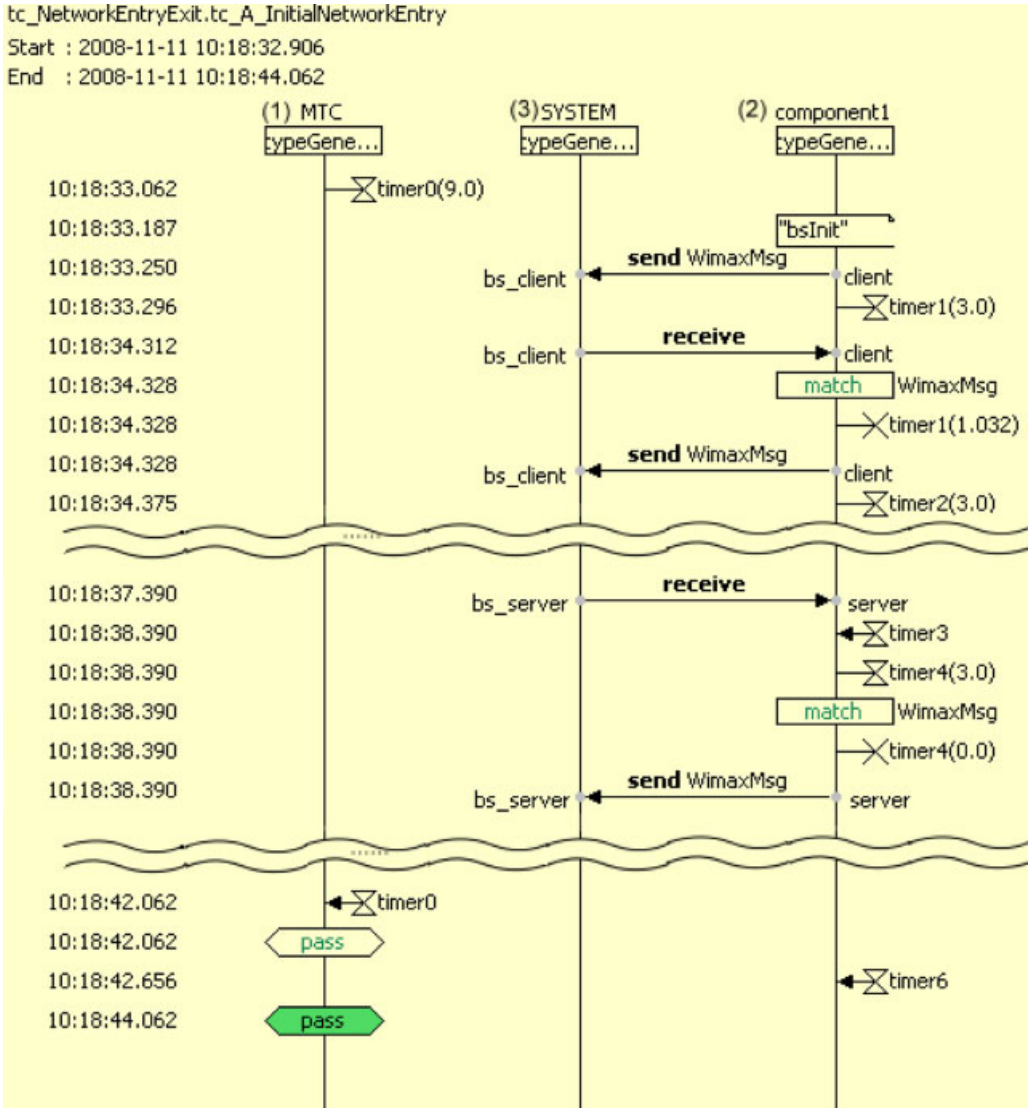

Fig. 7. Graphical test log for Initial Network Entry procedure.

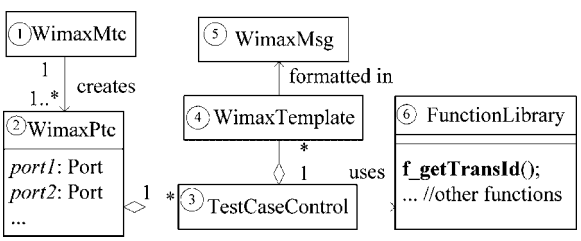

Fig. 8. The ITRI-WiMAXT test executable.

(Figure 8 (1)) with a test system interface of type WimaxInterface (Line 1, Figure 6). The WimaxMtc interacts with the PTCs throught the test component operation (see Section 3.1) to execute the testcases. The WimaxInterface includes all ports that are used to interact with the SUT in all WiMAX related testcases. For example, the BS PTC ports are defined in Lines 14 and 15 in Figure 6. The MTC may create one or more PTCs (WimaxPtc; see Figure 8 (2)). A WimaxPtc can be WimaxBS (for BS), WimaxASN (for ASN-GW), WimaxDHCP (for DHCP server) and so on. In the *mtc_InitialNetworkEntry* testcase, a WimaxBS called *bs* (Line 2, Figure 6) is created. This PTC plays the role as a WiMAX BS, and has two ports for the server and the client modes, respectively (see Lines 16 and 17, Figure 6). In the server mode, *bs* listens on the port *server* (i.e., WiMAX port 2231/UDP) to wait for transactions initiated from the ASN-GW (the SUT).

DOI: 10.1002/wcm

In the client mode, *bs* initiates a transaction with the port *client*, which connects to the WiMAX port of the ASN-GW (Lines 3 and 4, Figure 6). Then *bs* executes the designated TestCaseControl (Figure 8 (3); see also Line 5, Figure 6). The MTC will wait until either the executions of all PTCs are completed (Line 7, Figure 6), or the timer *t_initialNetworkEntry* is expired (Lines 8–10, Figure 6). When the testcase is complete, the **unmap** operation is invoked to release the ports (Lines 11 and 12, Figure 6).

To process the WiMAX messages, the ITRI-WiMAXT defines a TTCN-3 record type WimaxMsg (Figure 8 (5)) based on the WiMAX specification described in Figure 2. The WiMAX message header and the message body (i.e., the IEs) are implemented as the WimaxMsgHeader and the IEList (Lines 1 and 2, Figure 9). The IEList consists of a set of InfoElements (Line 3, Figure 9). At Line 4 of Figure 9, the InfoElement is a TTCN-3 record type representing the IE of the WiMAX message described in Figure 3. The InfoElement contains an integer *ieType* and an IeValue type *ieValue* (Lines 5 and 6, Figure 9), which represent the type and the value fields of the IE, respectively. The IeValue is a TTCN-3 union type, which can be single or nested. A single IeValue is a hex string (Line 9, Figure 9). A nested IeValue includes another IEList (Line 8, Figure 9). Note that the length fields of the WiMAX message and the IE are calculated in the CD, which are not presented in the WimaxMsg and the InfoElement.

```
    type record WimaxMsg {
(1)     WimaxMsgHeader header,
(2)     IEList ieList optional
    };
(3) type set of InfoElement IEList;
(4) type record InfoElement {
(5)     integer ieType,
(6)     IeValue ieValue
    };
(7) type union IeValue {
(8)     IEList ieNested,
(9)     hexstring ieSingle
    };
```

Fig. 9. The TTCN-3 WimaxMsg.

Parameter values of a specific WimaxMsg are filled in the WimaxTemplate (Figure 8 (4)). Figure 10 shows how the MS_PreAttachment_Req message is parameterized in the WimaxTemplate called *msPreAttachmentReq*. According to the description in Section 2,

```
(1) template WimaxMsg msPreAttachmentReq (hexstring a_msId, integer a_transId,
                IEList a_ieList) := {
        header := {
(2)         version := 1,
(3)         flag := '00000000'B,
(4)         funcType := 9,
(5)         opID := 1,
(6)         msgType := 6,
(7)         msId := a_msId,
(8)         transId := a_transId,
        },
(9)     ieList := a_ieList
    };
```

Fig. 10. WimaxTemplate *msPreAttachmentReq*.

```
(1) client.send(msPreAttachmentReq(a_msId, f_getTransId(REQ), a_ieList) );
(2) client.receive(msPreAttachmentRsp(/*parameters of msPreAttachmentRsp*/);
(3) client.send(msPreAttachmentAck(/*parameters of msPreAttachmentAck*/);
    ....
(4) server.receive(keyChangeDirective(/*parameters of keyChangeDirective*/) );
(5) server.send(keyChangeAck(/*parameters of keyChangeAck*/));
    ..... //conclude the verdict
```

Fig. 11. TestCaseControl program segment of *bs* (for *mtc_InitialNetworkEntry*).

DOI: 10.1002/wcm

```
    modulepar {
(1)     charstring ASNGW_IP_ADDRESS := "140.113.214.113";
(2)     charstring BS_IP_ADDRESS := "140.113.214.114";
(3)     hexstring MSID := '001122334455'H;
        ... // other WiMAX parameters
    }
```

Fig. 12. Examples of the module parameters.

the values of the version, flag, funcType, opID and msgType fields for this message are set to 1, 0, 9, 1 and 6, respectively (Lines 2–6 in Figure 10). Fields MSID, transaction ID and TLVs are determined during the test-case execution, and are passed to the WimaxTemplate through the template parameters *a_msId*, *a_transId* and *a_ieList* (Lines 1, 7–9 in Figure 10).

A WiMAX procedure is executed by the TestCaseControl through sending and receiving a sequence of WimaxTemplates. For example, the **tcc_initialNetworkEntry** TestCaseControl for *mtc_InitialNetworkEntry* is illustrated in Figure 11. Lines 1–3 in Figure 11 correspond to Steps 1–3 in Figure 4, where the BS is in the client mode. Specifically, the TestCase sends an MS_PreAttachment_Req message (through WimaxTemplate *msPreAttachmentReq* in Figure 10) to the SUT at Line 1, where *client*.**send** invokes **f_getTransId** in the FunctionLibrary (Figure 8

```
    public TriMessage encode(final Value TTCNValue) {
        byte[] encMsg = new byte[65535];
(1)     RecordValue rv = (RecordValue) TTCNValue;
(2)     if(rv.getType().getName().equals("WimaxMsg")) { // is it a WimaxMsg?
            int msgLen;
            try {
(3)             encMsg[0] = (byte)((IntegerValue)rv.getField("version")).getInt();
(4)             ... // encodes the header
                List ieList = (List)rv.getField("ieList");
(5)             if( ! isEmpty(ieList) ) { // does the message has the body?
(6)                 msgLen = encodeIEList(encMsg, ieList, 20);
                }
(7)             encMsg[4] = (byte) (msgLen/256);
(8)             encMsg[5] = (byte) (msgLen%256);
(9)         } catch (Exception e) {
(10)            e.printStackTrace();
            }
(11)    } else ... // encode other message types such as DHCP
(12)    ... //trim the encMsg
(13)    return = new TriMessageImpl(encMsg);
    }
    private int encodeIEList(byte[] encMsg, List ieList, int ptr){
        int plen = ieList.getLength();
(14)    for(int i = 0; i < plen; i++){
(15)        Record ie = (Record)ieList.getDataItem(i);
(16)        int T = ((IntegerValue) ie.getField("ieType") ).getInt();
(17)        encMsg[ptr++] = (byte) (T / 256);
(18)        encMsg[ptr++] = (byte) (T % 256);
(19)        ... // encode the value, and then calculate the length
        }
        return ptr;
    }
```

Fig. 13. ITRI-WiMAXT **encode** operation.

DOI: 10.1002/wcm

(6)) to generate a new transaction ID. Line 2 of Figure 11 checks whether the received *msPreAttachmentRsp* is correct. Then Line 3 sends *msPreAttachmentAck* to the SUT (ASN-GW). Lines 4 and 5 in Figure 11 correspond to Steps 5 and 6 in Figure 4, where the BS is in the server mode. If all messages from the ASN-GW are successfully verified, the verdict of this testcase is set to pass. Otherwise, the verdict is set to fail.

4.2. ITRI-WiMAXT TM and CD Entities

The TM and the CD entities interact with the TE through the TCI interface. In ITRI-WiMAXT, some module parameters passed from the TM to the TE are illustrated in Figure 12, which include the IP addresses for the ASN-GW and the BS (Lines 1 and 2 in Figure 12), and an MSID (Line 3 in Reference 12). These module parameters can be dynamically set by the test system user, and are retrieved by the **triExecuteTestcase** operation for the testcase execution.

As implemented in JAVA, the ITRI-WiMAXT CD encodes and decodes the WiMAX messages. When *client*.**send** in Figure 11 is executed, the TE invokes the **encode** operation in the CD, which encodes a TTCN value *TTCNValue* (e.g., *msPreAttachmentReq* in Figure 10) into an encoded byte array *encMsg*. Line 1 in Figure 13 stores *TTCNValue* into a variable *rv*. Line 2 checks if a WiMAX message should be created. If so, Lines 3–10 encode *rv* based on the WiMAX format. Specifically, Lines 3 and 4 in Figure 13 encode the WiMAX message

```
    public Value decode(final TriMessage encTriMsg, final Type decodingHypothesis) {
(1)     if (decodingHypothesis.getName().equals("WimaxMsg")) { // is it a WimaxMsg ?
(2)         final RecordValue TTCNValue = (RecordValue) decodingHypothesis.newInstance();
(3)         encMsg = encTriMsg.getEncodedMessage();
(4)         return createWimaxMsg(TTCNValue, encMsg);
        } else ... // decode other message types such as DHCPMsg
    }
    private RecordValue createWimaxMsg(final RecordValue TTCNValue, final byte[] encMsg) {
(5)     ... // decode the message header from the first 20 bytes
(6)     if( hasIEList(TTCNValue) ){ // decode the message body
(7)         Value value = decodeIEList((RecordOfValue)TTCNValue.getField("ieList"),
(8)             encMsg, 20, encMsg.length);
(9)         setField(TTCNValue, "ieList", value);
        }
        return TTCNValue;
    }
    private Value decodeIEList(RecordOfValue ieList, final byte[]encMsg, int ptr, int lptr) {
        ieList.setLength(0);
(10)    while( ptr != lptr ) {
(11)        ... // decode and obtain the type T and length L of the IE from the first four bytes
(12)        if( matches(T, TS) ) { // is a single IE type matched?
(13)            ... // obtain the value V
            }else{ // is a nested IE type matched?
(14)            ... // recursively decode the information element
(15)            V = decodeIEList(...);
            }
(16)        ... // insert the T, L and V as an IE into the ieList
        }
(17)    return ieList;
    }
```

Fig. 14. ITRI-WiMAXT **decode** operation.

DOI: 10.1002/wcm

header of *rv*. Line 5 checks whether the WiMAX message has a body. If so, Line 6 encodes the message body by calling the function **encodeIEList** with parameters such as *encMsg*, *ieList* and the encoding position *ptr*. This operation repeatedly encodes the IEs (Lines 14–19). In each loop, an IE *ie* is retrieved from *ieList* (Line 15). The operation encodes the type and value fields, and then calculate the length of the IE (Lines 16–19). The encoded results are saved in *encMsg*. Then *ptr* is returned to **encode** for calculating the message length *msgLen*. Lines 7 and 8 in Figure 13 encode *msgLen* and fill it into *encMsg*. If an exception occurs (e.g, the array has been accessed with an illegal index), the debug information is printed (Lines 9 and 10). Lines 12 and 13 trim the unused tailing spaces of the byte array and return it.

When *client*.**receive** in Figure 11 is executed, the CD **decode** operation in Figure 14 is invoked, where an encoded TriMessage *encTriMsg* is decoded based on the *decodingHypothesis* type that can be WimaxMsg, DHCPMsg and so on. At Lines 1 and 2 in Figure 14, if *decodingHypothesis* indicates the type 'WimaxMsg', the **newInstance** operation creates an empty TTCN value *TTCNValue*. Line 3 retrieves the encoded byte array *encMsg* from *encTriMsg*. In Line 4, the **createWimaxMsg** operation is called to fill the fields in *TTCNValue*. Specifically, this operation decodes the WiMAX message header from the first 20 bytes of the *encMsg* (Line 5), and then decodes the remaining bytes as the WiMAX IEs (Lines 6–9) by calling the **decodeIEList** operation with parameters such as *ieList* (the WiMAX message body at Line 2 in Figure 9), the *encMsg* and the range of the remaining bytes (i.e, from currently decoding position to the end). This operation uses the regular expression (1) to repeatedly decode the IEs (Lines 10–16) until all bytes in *encMsg* are traversed. In each loop, the type field T and the length field L are decoded from the first 4 bytes (Line 11). If T matches the T_S type described in Section 2 (i.e., it is a single IE), the L-bytes value field is filled (Lines 12 and 13). Otherwise, it is a nested IE, and is recursively decoded into a value V (Lines 14 and 15). Finally, the T, L and V are inserted into the *ieList* as an IE (Line 16). When all bytes are decoded, the value *ieList* is returned (Line 17).

5. Conclusions

This paper described the architecture and implementation of ITRI-WiMAXT, a WiMAX test system 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 showed how to encode and decode the WiMAX messages in ITRI-WiMAXT, and used the initial network entry procedure as an example to illustrate how conformance tests can be implemented and conducted for WiMAX. A major advantage of ITRI-WiMAXT is its modular implementation. The implemented CDs can be reused in the new testcases.

References

1. IEEE standard for local and metropolitan area networks, part 16: air interface for fixed broadband wireless access systems. *IEEE 802.16-2004*, June 2004.
2. IEEE standard for local and metropolitan area networks, part 16: air interface for fixed broadband wireless access systems, amendment 2: physical and medium access control layers for combined fixed and mobile operation in licensed bands and corrigendum 1. *IEEE 802.16e-2005*, February 2006.
3. Perkins C. IP Mobility support for IPv4. *RFC 2002*, August 2002.
4. Chen H-F, Hsu Y-C, Hsiao H-C, Lee H-Y, Tsai P-F. Test procedures for PMIPv4 IIOT renewal, WiMAX Forum Interoperability Task Group, August 2007. http://www.wimaxforum.org/apps/org/workgroup/nwiot/download.php/23218/01014_r003_ITRI_NWIOT_IIOT_PMIPv4_Renewal_TP.doc
5. Chen H-F, Hsu Y-C, Hsiao H-C, Tsai P-F, Lee H-Y. Test procedures for IIOT PMIPv4 network exit MS trigger, WiMAX Forum Interoperability Task Group, September 2007. http://www.wimaxforum.org/apps/org/workgroup/nwiot/download.php/23221/01020_r003_01016_r002_ITRI_NWIOT_IIOT_PMIPv4_Network_Exit_MS_Tirgger_TP.doc
6. Lin Y-B, Wang C-C, Lu C-H, Hsu M-R. NTP-PoCT: a conformance test tool for push-to-talk over cellular network. *Wireless Communications and Mobile Computing* 2007; **8**(5): 673–686.
7. Lin Y-B, Liang C-F, Chen K-H, Liao H-Y. NTP-SIOT: A test tool for advanced mobile services. *IEEE Network* 2007; **21**(1): 21–26.
8. Leong C-W, Yang S-R. NTP-BrowsingT: a conformance test tool for oma browsing capability for mobile devices. Presented at the *13th Mobile Computing Workshop*, 2007, Taiwan, R.O.C.
9. Hopcroft JE, Motwani R, Ullman JD. *Introduction to Automata Theory, Languages, and Computation* (3rd edn). Addison-Wesley: Reading, MA, USA, 2006.
10. WiMAX Forum. *WiMAX Forum Network Architecture (Stage 3: Detailed Protocols and Procedures)*, Release 1, V1.2, January 2008.
11. Mandin J. 802.16e Privacy key management (PKM) version 2. *IEEE C802.16e-04/131r1*, 2004.
12. 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.
13. 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.
14. 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.
15. ETSI. Methods for testing and specification (MTS). *The Testing and Test Control Notation*, version 3, Part 4, TTCN-3 Operational Semantics, ES 201 873-4 V3.3.1, 2008.
16. Object Management Group. *Unified Modeling Language (OMG UML) Infrastructure Specification*, V2.1.2, 2007.

DOI: 10.1002/wcm

Authors' Biographies

Hsin-Yi Lee received her B.S.C.S.I.E degree from National Chiao-Tung University in 2003. She is currently a Ph.D. student of the Department of Computer Science, National Chiao-Tung University. Her current research interests include wireless and mobile computing systems, and performance modelling.

Yi-Bing Lin (M'95-SM'95-F'03) is Chair Professor of Computer Science, National Chiao Tung University. His current research interests include wireless communications and mobile computing. Dr. Lin has published over 220 journal papers and more than 200 conference papers. Lin is the author of the book Wireless and Mobile Network Architecture (co-author with Imrich Chlamtac; published by John Wiley & Sons), the book Wireless and Mobile All-IP Networks (co-author with Ai-Chun Pang; published by John Wiley & Sons) and the book Charging for Mobile All-IP Telecommunications (co-author with Sok-Ian Sou; published by John Wiley & Sons). Lin is an IEEE Fellow, an ACM Fellow, an AAAS Fellow and an IET(IEE) Fellow.

Ching-Feng Liang received M.S. degree in electronic engineering from National Taiwan University of Science & Technology (NTUST) in 1993 and joined the Information & Communication Laboratory (ICL) of Industrial Technology Research Institute (ITRI) as an engineer. Liang 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. Liang received the ITRI Award in 2005 and the Outstanding Project Award of Taiwan MoEA in 2003. Liang is currently the manager of the Core Network Department of ICL/ITRI.

Shiang-Ming Huang was born in Taiwan. He is currently working towards his Ph.D. degree in Computer Science at National Chiao Tung University, Taiwan.

DOI: 10.1002/wcm