

# IPV6 CONFORMANCE TESTING: THEORY AND PRACTICE

Yujun Zhang, Zhongcheng Li

Institute of Computing Technology, Chinese Academy of Sciences. Beijing, China

## Abstract

IPv6 is in its growing stage in which new protocols are being proposed and more and more IPv6 devices are being produced. Conformance testing is the most important method to improve the reliability of IPv6 implementations. With a view to provide test ability for IPv6, the features and the test requirements of IPv6 are analyzed. Two difficulties of the standard test framework applying to IPv6 conformance testing, test packets description and complicated algorithm implementation, are pointed out. IPv6 test framework is proposed to solve the two difficulties, in which a new IPv6 test suite specification language is defined. Two test methods, called the virtual test method and the low-layer congregating test method, are adopted to enhance single physical tester's test ability. IPv6 test suite is designed and four IPv6 implementations are tested. An example of test case is given to explain IPv6 test framework and IPv6 test suite specification language.

## 1. Introduction

Protocol is the basis of computer network. The protocol specification standard, specified by request for comments (RFCs) in natural language, is the basis of the implementation. For a protocol specification standard, there always are many different implementations. Testing of an implementation for conformance to its specification standard is necessary. This type of testing is called conformance testing. In conformance testing, test suite is generated from protocol specification standard and all products implementing the protocol are tested against the same test suite (Fig1). Conformance testing is not intended to be exhaustive, and an implementation successfully passing test suite does not imply the 100 percent guarantee. But it does ensure, with a reasonable degree of confidence, that the implementation is consistent with its protocol specification standard, and it does increase the probability that different implementations can interoperate. Compared to other testing, conformance testing has the advantages such as test result comparable and the less costing, which makes conformance testing the most basic and important testing

before a new protocol implementation tends towards application and market.

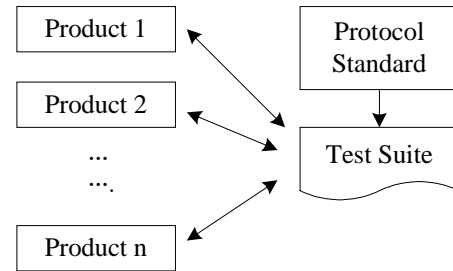


Figure 1 Conformance Testing Model

IPv6 (Internet Protocol version 6) is the next generation Internet protocol. More and more IPv6 devices are being produced. Different producers have different IPv6 implementations. In order to ensure that many IPv6 devices can interoperate properly, conformance testing is necessary. ISO/IEC 9646 defines the standard methodology and framework for conformance testing. But standard test framework is not appropriate for IPv6 testing. As a new protocol, IPv6 conformance testing hasn't been studied fully, in which many open problems to be solved exist.

This paper focuses on IPv6 conformance testing and is organized as follows. In section 2, IPv6 is introduced and analyzed from the point of testing and IPv6 test framework is proposed. In section 3, the self-defined test suite specification language is introduced. In section 4, two test methods are adopted during IPv6 testing. In section 5, test suite and test practice are described. Section 6 gives an example of a test case. Section 7 analyzes two existing IPv6 testing researches and concludes this paper.

## 2. IPv6 and Test Framework

### 2.1 IPv6 Introduction

IPv6 was designed to meet requirements that did not exist when IPv4 was first conceived. It takes into account many experiences accumulated using IPv4 in the last 20 years. IPv6 is designed to replace IPv4 in the future. As the next generation Internet protocol, IPv6 provides capabilities that go beyond larger addresses. The main technical benefits of IPv6 are specified as follows.

- Simplified header and flexible extension. IPv6 simplifies the header of the packet to reduce the cost of CPU and to save the network bandwidth. Optional IP-layer information is encoded in separate extension headers that may be placed between the IPv6 header and the upper-layer header in a packet. The main advantage is to diminish the cost of header processing.
- Unlimited addresses and more addressing hierarchies. By extending the size of the address field in the network layer header from 32 to 128 bits, IPv6 raised this theoretical limit to  $2^{128}$ . IPv6 allows for a much greater number of addressable nodes, more levels of addressing hierarchy, defining new types of addresses, and so on.
- Plug and play. IPv6 supports stateless address auto-configuration. A host can be connected to the network and be immediately ready for use without manual intervention. Plug and play greatly simplifies the work of network administration and control.
- Authentication and encryption in network layer. Security always is the most important issue on Internet. IPsec, which is developed by IETF from 1995, is optional for IPv4, and now is mandatory in IPv6. Based on IPsec, IPv6 provides authentication and encryption in network layer.
- Quality of service guarantees. A new capability is added to enable the labeling of packets belonging to particular traffic “flow” for which the sender has requested special handling, such as nondefault quality of service or “real-time” service.

## 2.2 Differences between IPv6 and IPv4

IPv6 and IPv4 are both network layer protocols and play the same role in network architecture. IPv6 is designed based on IPv4, and IPv6 adopts many new concepts that don't exist in IPv4. This section will narrate the differences between IPv6 and IPv4 from the point of testing.

IPv6 eliminates or makes optional some of the IPv4 header fields to reduce the packet-handling overhead, which provides some compensation for the larger addresses. Even with the addresses, which are four times as long, the IPv6 header is only 40 octets in length (Fig 2), compared with 20 octets for IPv4 (Fig 3). Less header fields can expedite processing rate of router. IPv6 options are placed in separate headers, located after IPv6 basic header, such that processing at every intermediate stop between source and destination may not be required.

IPv6 defines many extension headers, such as fragment header, hop by hop option header, destination option header, routing header, routing header type 0, ICMPv6 (Internet Control Message Protocol version 6) header,

authentication header, encrypted security payload header, mobile header. IPv6 basic header does not contain any optional element. This does not mean that we cannot express options for special-case packets. The capabilities that the variable-sized option field offered in IPv4 are now deployed by a chain of extension headers that follow IPv6 basic header.

Ver	TC	Flow Label	
Payload Length		Next Header	Hop Limit
Source Address			
Destination Address			

Figure 2 IPv6 Basic Header

Ver	IHL	TOS	Total Length	
Identification		Flags	Fragment Offset	
TTL	Protocol	Header Checksum		
Source Address				
Destination Address				
Options			Padding	

Figure 3 IPv4 Basic Header

Each IPv6 packet is composed of basic header and one or more extension headers and payload data, which makes IPv6 packets are provided with structural features (Fig 4). Each extension header has fixed length, accomplishes particular capability and is identified by “next header” field of the preceding header. However, Each IPv4 packet is composed of basic header and options and payload data (Fig 5). Option in IPv4 packet has variable length.

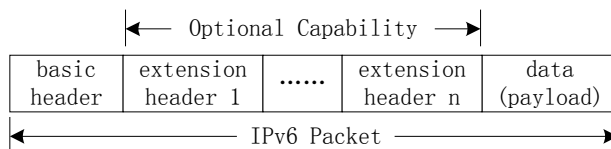


Figure 4 IPv6 Packet Structure

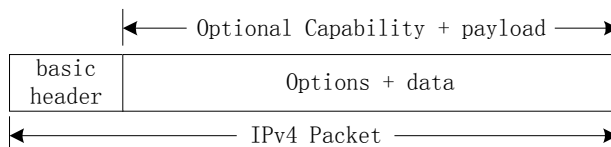


Figure 5 IPv4 Packet Structure

IPv4 packets have only one kind of structure, which is “fixed length basic header + variable length data”. IPv6 packets have countless kinds of structure, which are “fixed length basic header + fixed length extension header 1 + ... .. + fixed length extension header n + variable length data”. In IPv6 packet format, each

extension header may be one of ten kinds of possibility and the number of extension headers is variable, therefore the number of IPv6 packet structure is enormous.

Besides the packet format, IPsec is another difference between IPv6 and IPv4 that influences test implementation. IPsec provides security services at the IP layer. IPsec can be used to protect one or more “paths” between a pair of hosts, between a pair of security gateways, or between a security gateway and a host. IPsec can be used in transport mode and tunnel mode. Transport mode provides protection primarily for upper layer protocols, and tunnel mode is applied to tunneled IP packets. IPsec is optional in IPv4, while mandatory in IPv6. IPv6 defines two extension headers, authentication header (AH) and encapsulating security payload header (ESP), to support IPsec. AH provides connectionless integrity, data origin authentication and anti-replay service. ESP provides confidentiality and all the service that AH can provide. Encryption algorithms (such as aes-abc, des-abc and 3des-abc) and authentication algorithms (such as hmac-sha1-96 and hmac-md5-96) should be implemented. These algorithms may be used to encrypt a packet in tunnel packet or to calculate hash value from an packet. Computational procedure is very complicated and computational load is very huge.

### 2.3 Standard Test Framework for IPv6 Testing

Conformance testing is the process of verifying that an implementation performs in accordance with a particular standard, specification, or environment. Implementation under test (IUT) is seen as a black box. Test suite is defined in terms of input events send to an IUT and output events received from the IUT. Test system is not a complete implementation of protocol under test (PUT), but it must simulate most part capabilities of PUT. Test system should be able to generate the packets of PUT and validate the packets from IUT. During testing, test system simulates one protocol implementation communicating with IUT. In other words, conformance testing is exclusively concerned with the external behavior of an implementation.

ISO/IEC 9646 defines standard methodology and framework for conformance testing, in which the tree and tabular combination notation (TTCN) is recommended as the standard test suite specification language (Fig 6). The standard test framework has the main features as follows.

- Test suite is described by TTCN. All details of test suite, including complicated algorithms, are described by TTCN completely.
- Test system is the compiler and executer of TTCN. Test system is universal and test suite is independent of test system. The protocol under test is transparent to test system.

- In this test framework, any test suite for any protocol, if described by TTCN, can be loaded to the same test system and be executed.

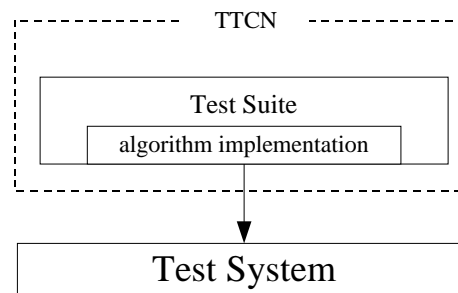


Figure 6 Standard Test Framework

The standard test framework is not appropriate for IPv6 testing in two aspects: test packets description and complicated algorithms implementation.

- TTCN is the common test suite description notation in the standard test framework. Before describing a test packet using TTCN, we must define a structure type to match the packet. This process is called type declaration. By assigning the specified values to every filed in the self-defined structure type, a material test packet is described successfully. This process is called constraint declaration. The number of IPv6 test packet structures, including valid and invalid, is enormous. It is low efficient to define a structure type for each possible IPv6 test packet.
- In the standard test framework, all details of test suite, including complicated algorithms, are described by TTCN completely. TTCN defines some simple syntax for calculation. TTCN is a script language, which makes it useless for some complicated algorithms. IPsec is mandatory in IPv6 and defines many complicated algorithms. To test IPsec implementations, we need to generate the valid packets and verify the packets from implementation under test (IUT). These complicated algorithms must be implemented by test suite or test system. It is difficult to describe the algorithms using TTCN.

### 2.4 IPv6 Test Framework

In order to perform IPv6 conformance testing efficiently, it is better not to adopt TTCN and the standard test framework. IPv6 test framework is proposed, in which a new formal test suite specification language (TSSL) is defined and test system is developed. In general, test system is implemented by high-level programming language. It is better to implement complicated algorithms by high-level programming language than by test suite specification language. In IPv6 test framework, test suite is described by TSSL. All the algorithms used in IPv6 testing are implemented in test system by high-level

programming language. During designing test suite, if some algorithm is needed, then the correlative algorithm property, including the algorithm name and the input parameters, is assigned in test suite by TSSL (Fig 7). When the test suite is compiled and executed in test system, test system will invoke the algorithm module.

IPv6 test framework reduces the complexity of test suite specification language. IPv6 test suite is not independent of test system. Test suite combines with test system to accomplish IPv6 testing.

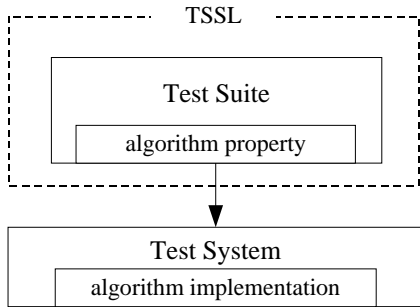


Figure 7 IPv6 Test Framework

### 3. Test Suite Specification Language

In IPv6 test framework, a new test suite specification language, called TSSL, is defined.

#### 3.1 Test Suite Structure

The structure of IPv6 test suite described by TSSL has the hierarchy like a tree (Fig 8). TSSL defines three types of files to describe IPv6 test suite. The three types of files are test suite file (named \*.tsf), test group file (named \*.tgf), test case file (named \*.tcf). Test suite file defines the test suite correlative information, the global parameter, the headers used in IPv6 with default values, the included test group filenames. Test group file defines the test group correlative information, the included test case filenames. Test case file defines the test case correlative information, test packets description, test procedures and test events. On designing IPv6 test suite using TSSL, besides the test packets, five parts are to be described:

- Test suite: corresponding to a protocol or a protocol stack. The test suite in this paper is for IPv6.
- Test group: corresponding to one standard specification. For instance, IPv6 general specification (RFC 2460), Path MTU discovery (RFC 1981), Neighbor discovery for IPv6 (RFC 2461).
- Test case: corresponding to a function specified in a RFC. For example, unrecognized next header specified in RFC 2460.

- Test procedure: A test case consists of many possible test procedures. For example, in order to test the header order specified in RFC 2460, we need to compose any possible header order to test the IUT. Each possible order can be tested in a test procedure.
- Test behavior: a test procedure consists of many test behaviors, including initialize, send, receive and verdict, etc. Test behavior is the minimal unit in test suite.

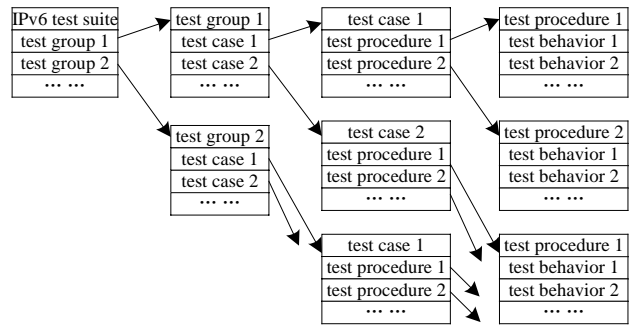


Figure 8 Test Suite Structure specified by TSSL

#### 3.2 Test Packet Specification

TSSL predefines the structure of headers including MAC header, the IPv6 basic header, all the extension headers, UDP header and TCP header. The fields in any header are all set default values. Compared to TTCN, it is not necessary to define the packet's structure before constructing a test packet. In TSSL, packet structure definition is combined with packet definition. We can construct any test packet simply by putting the needed headers in the correct order. At the same time, the fields in the headers can be assigned the appropriate values (Table 1). For some fields that need to be calculated further (such as checksum), the test system supporting TSSL can calculate and fill them in the fields automatically. Using TSSL, the step of defining packet structure is unnecessary, which simplifies test packets description.

Table 1 Header definition and packet definition

Header Definition Format	Packet Definition Format
<Header>	<Packet>
Type = {IPv6, AH, .....}	Name = string(packet_name)
Filed1_name =	Header1
Filed2_name =	Header2
Filed3_name =	Header3
.....	.....
</Header>	</Packet>

#### 3.3 Test Behavior Specification

Conformance testing is a process of control and observation of IUT. Test suite must describe the test behaviors. TSSL defines test behaviors needed in IPv6 testing. A test procedure defined in test case file may

consist of many test behaviors. In this section, the most used test behaviors defined in TSSL are introduced.

- Sending a test packet to IUT belongs to the control stage of conformance testing. TSSL defines the test behavior for sending a specified test packet to IUT. The syntax is as following, in which packet-name specifies the packet's name to be sent to IUT.

Send packet-name

- Receiving a test packet from IUT belongs to the observation stage of conformance testing. TSSL defines the test behavior for receiving a packet, in which TSSL can specify conditions that the received packet should match, receiving timer, the verdict result, the overtime processing, etc. The syntax is as follows.

```
Receive [- clear_buffer true | false];
        [- expect_data start, length, value;.....];
        [- lifetime number];
        [- verdict true | false];
        [- overtime pass | fail | inconclusive |
          continue];
```

- There may be an interval between two test behaviors. TSSL defines a test behavior to accomplish this function. The syntax is as follows.

Delay number-milliseconds

- In IPSec testing, we need to decrypt and verify the packets from IUT. TSSL defines a test behavior, called accept, to accomplish this function. In this behavior, TSSL can specify the algorithm properties, such as the key and the algorithm name used for decrypting and verifying the packets from IUT. The real algorithm implementation is embedded in test system by high-level programming language. TSSL can also assign specified conditions to match a decrypted packet. The syntax is as follows.

```
Accept [- timeout number];
        [- action pass | fail];
        [- filter name, startpos, length, value;
          name, startpos, length, value; .....]
        [- ckey crypt];
        [- calg algorithm];
        [- akey crypt];
        [- aalg algorithm];
        [- ahkey crypt];
        [- ahalg algorithm];
```

filter: the condition that the received packet or the received and decrypted packet should meet.

ckey and calg: decrypt the received packet using algorithm and crypt.

akey and aalg: authenticate the received packet using algorithm and crypt. This is valid for AH in ESP.

ahkey and ahalg: authenticate the received packet using algorithm and crypt. This is valid for AH or combination of AH and ESP.

## 4. Test Method

During conformance testing, any conclusion that we may draw about the conformance of the IUT will be made by observing and controlling the events that occur at interfaces of the upper tester and the lower tester. Since an IUT is generally a part of a real system, a variety of possibilities exist on how the behavior of an IUT can be controlled and observed during executing a test case. Each possibility is a test method. For end-systems, four abstract test methods are defined: the local test method, the distributed test method, the coordinated test method and the remote test method. For open relay systems, two abstract test methods are defined: the transverse test method and the loop-back test method.

Because the upper tester is difficult or impossible to be installed in some systems under test, the remote test method, which has no requisition for the upper tester, is widely used in practice. Based on IPv6 test framework and TSSL, we adopt two test methods, the virtual test method and the low-layer congregating test method, to enhance test ability of the remote test method. The two test methods make the concurrent test practical based on single physical tester.

### 4.1 The Virtual Test Method

IPv6 is a distributed protocol, which makes the concurrent testing necessary. In other words, more than one test components are necessary. In order to perform concurrent testing for IPv6 efficiently, the virtual test method is adopted. TSSL can construct test packets from MAC layer to transport layer. Because IUT identifies the test nodes only based on the received packet's source address, single tester can be used to simulate many testers by assigning different source address when constructing test packets. On designing test suite, TSSL can define many test nodes (such as TN1, TN2, .....), called logical testers, by assigning different addresses for test packets belonging to different test components. Each test behavior can be captioned to some test node.

Using the virtual test method, a lot of logical testers are available, but only one physical tester exists. This method extends concurrent test ability of the remote test method. The physical tester can easily harmonize and monitor the executing process, at the same time the more accurate test result can be obtained.

## 4.2 The Low-layer Congregating Test Method

In order to test the relay functions of IPv6 routing protocols and to reduce or avoid the synchronization problem among multi-testers, another test method, named the low-layer congregating test method, can be adopted based on the virtual test method. The ports of a relay system are congregated using a congregant (Fig 9). Because the logical testers can exchange messages through different ports on the relay system, this test method can verify all the relay functions. The lower-layer congregating test method can test all the real actions of the relay systems and has no extra requirement on the IUT. In addition, it simplifies the test coordination process and has the reliable and accurate test result. The low-layer congregating test method can also test the end system, in which the congregant works on transparent mode and connects to the lower layer service provider directly.

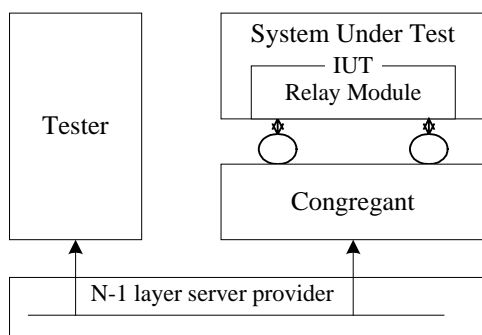


Figure 9 The Low-layer Congregating Test Method

Compared to the traditional loop-back test method and transverse test method, the low-layer congregating test method has the following advantages:

- The test method can test the relay system using single physical tester.
- The test method has no extra requirement for the relay system under test.
- The test method avoids the complicated synchronization problem among multi-physical-testers that exist in the transverse test method.

## 5. Test Practice and Result Analysis

Test suite is the kernel of conformance testing, which specifies what to test and how to test. Test suite is generated from protocol specification standard. Two approaches, formal test generation and manual test generation, are adopted to generate IPv6 test suite. We design IPv6 test suite, which covers IPv6 base specification, neighbor discovery, ICMPv6, security,

address auto-configuration, path MTU, transition, addressing, mobile IPv6, routing protocol, including 29 RFCs and one draft. IPv6 test suite is composed of 461 test cases. To design IPv6 test suite, more than four thousand test packets need to be described. If TTCN is used to describe these packets, four thousand structure types must be defined. Type declaration in TTCN is a troublesome work, but is not necessary in TSSL. Also, test suite will be bulky if the complicated algorithms is described by TTCN.

We performed the practical conformance testing against four IPv6 implementations on Linux, FreeBSD, a commercial operating system and a commercial IPv6 router. According to the PICS/PIXIT of the implementations, the numbers of executable test cases for the four implementations are not the same. IPv6 test suite and test results are shown in Table 2, in which the column of Exec- means the number of executable test cases for an implementation; the column of Passed means the number of test casts that are successful for an implementation; × means that no test case is executed against an implementation.

From Table 2, we can know none of the four IPv6 implementations implement IPv6 RFCs completely. For those RFCs implemented, some inconformances are found. The inconformances are fed back the relevant implementers and accepted by them. We will give two examples:

- In mobile IPv6, when a node moves to another link, it should register its care-of address on its home agent. Home agent should perform duplicate address detection for the mobile node that wants to register on it. If the duplicate address detection fails, the home agent should reject the mobile node's registration and return a binding acknowledge message to the mobile node. We design a test case for this function. When performing this test case against the commercial router acting as home agent, the test result shows that even router detects duplicate address detection failing, it does not reject a mobile node's registration, but accepts the registration.
- RFC 2462 describes the stateless address auto-configuration mechanism. An IPv6 host should auto-configure an address on receiving a router discovery message containing the different prefix option. The lifetime of the auto-configured address is decided by the prefer lifetime field and the valid lifetime field in the received prefix option. But when testing IPv6 implementation on Linux, we found out that the auto-configured address always has the fixed valid lifetime regardless of the value in the prefix option.

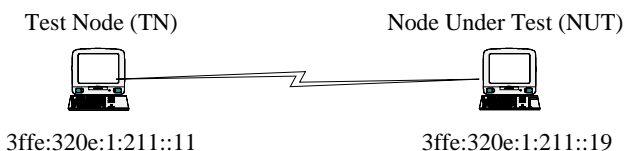
**Table 2 IPv6 test suite and test result against four implementations**

	Covered RFCs and Drafts	Total Number	Commercial OS		FreeBSD		Linux		Commercial Router	
			Exec-	Passed	Exec-	Passed	Exec-	Passed	Exec-	Passed
Base Specification	RFC2460, RFC2711	48	22	22	22	22	22	21	22	22
ICMPv6	RFC 2463	32	16	16	16	16	16	16	19	19
Neighbor Discovery	RFC2461, RFC3122	44	24	23	23	16	23	14	15	14
Path MTU Discovery	RFC1981	23	12	12	12	8	12	10	×	×
Addressing	RFC3513, RFC1887 RFC2374, RFC2375 RFC2526	22	11	8	11	9	11	9	19	15
Security	RFC2401 – 2406 RFC1828, RFC1829 RFC3217	88	×	×	88	82	×	×	×	×
Autoconfiguration	RFC2462, RFC3041	30	×	×	15	15	15	13	×	×
Mobile IPv6	draft-ietf-mobileip- ip6-24.txt	112	×	×	×	×	110	95	28	22
Transition	RFC2893, RFC3056 RFC3068	22	7	4	11	6	7	5	14	10
Routing Protocol	RFC2080, RFC2740 RFC2283, RFC2545	40	×	×	×	×	×	×	40	36

## 6. An Example of Test Case

To explain IPv6 test framework and TSSL, a test case for IPsec, named AH inbound with hmac-md5 in host transport mode, is given in this section.

RFC 2402 prescribes that a compliant AH implementation MUST support the mandatory-to-implement algorithm hmac-md5. Supporting an algorithm can be explained that an IPsec implementation can authenticate the packets with AH header and the specified algorithm destined to and originated from itself. We can design a test case to verify whether an IPv6 node, if configured the correlate security policy (SP) and security association (SA), can authenticate the packets with AH header and hmac-md5 from other nodes. The test environment is shown in Fig 10.



**Figure 10 Test Environment**

We configure SP and SA on NUT (Node Under Test), which requires that NUT must authenticate the packets from TN, but has no additional requirements for packets originated from itself. If a packet from TN passes the authentication, NUT will process the packet further; otherwise NUT should ignore the packet.

**SA Entry Configured on NUT**

Source Address	Destination Address	Security Protocol	SPI	Authentication Algorithm	Authentication Key	Encryption Algorithm	Encryption Key
3ffe:320e:1:211::11	3ffe:320e:1:211::19	AH	4096	hmac-md5	0x0123456789abcdef	-	-

**SP Entry Configured on NUT**

Source Address Range	Destination Address Range	Upper Protocol	Policy	Direction	SA Parameters			
					Protocol	Mode	Address	Application
3ffe:320e:1:211::11	3ffe:320e:1:211::19	any	IPsec	Inbound	AH	Transport	3ffe:320e:1:211::11 -> 3ffe:320e:1:211::19	Require

The SP and SA entries configured on NUT make NUT firstly authenticate all the packets with algorithm hmac-md5 from TN and only further process those passed packets. Three steps are used to test this function. Firstly, TN sends an Echo\_Request message to NUT with AH header and hmac-md5, which requires TN to return an Echo\_Reply message. Secondly, TN sends a neighbor\_advertisement message to NUT with AH header

and hmac-md5, which advertises TN's MAC address to NUT. Thirdly, after receiving the two messages, NUT will authenticate them with the configured SA and SP entries. If the authentication is passed, NUT will generate an Echo\_Reply message without IPsec protection to TN; otherwise, no message is sent to TN. The test case described by TSSL is shown in Table 3.

**Table 3 A Test Case Described by TSSL**

Common Data Defined in Test Suite File	Test Case File	
<pre>[Default Address] TN_MAC = &lt;00 10 4b 0d 52 91&gt; NUT_MAC = &lt;00 90 27 58 9f 55&gt; TN_Global = 3ffe:320e:1:211::11 NUT_Global = 3ffe:320e:1:211::19  [Default Header] &lt;Header&gt; Type = DLC Source_Address = TN_MAC Destination_Address = NUT_MAC &lt;/Header&gt; &lt;Header&gt; Type = IPv6 Version = 6 Traffic_Class = 0 Flow_Label = 0 Payload_Length = 0 Next_Header = No_Next_Header Hop_Limit = 255 Source_Address = TN_Global Destination_Address = NUT_Global &lt;/Header&gt;</pre>	<pre>[Test Case Common Information] Case_Name = AH Inbound with \           hmac-md5 in Host Transport Mode Standards_Reference = RFC 2402 Purpose = Verify NUT implements \           algorithm hmac-md5.  [Test Packet/Setup] &lt;Packet&gt; Name = Echo_Request &lt;Header&gt; Type = IPv6 Next_Header = AH &lt;/Header&gt; &lt;Header&gt; Type = AH Next_Header = ICMP SPI = 4096 Algorithm = HMAC-MD5 Sequence_Number = 1 Crypt = 0123456789ABCDEF &lt;/Header&gt; &lt;Header&gt; Type = ICMP ICMP_Type = 128 Code = 0 Message_Body = 00 11 00 01 &lt;/Header&gt; &lt;/Packet&gt;</pre>	<pre>&lt;Packet&gt; Name = Neighbor_Advertisement &lt;Header&gt; Type = IPv6 Next_Header = AH &lt;/Header&gt; &lt;Header&gt; Type = AH Next_Header = ICMP SPI = 4096 Algorithm = HMAC-MD5 Sequence_Number = 2 Crypt = 0123456789ABCDEF &lt;/Header&gt; &lt;Header&gt; Type = ICMP ICMP_Type = 136 Code = 0 Message_Body = 60 00 00 00 \                %TN_Global 02 01 %TN_MAC &lt;/Header&gt; &lt;/Packet&gt; [Test Procedure] &lt;Procedure&gt; Send Echo_Request Send Neighbor_Advertisement Receive - expect_data 54,1,129;         - lifetime 2000ms;         - verdict true;         - overtime fail; &lt;/Procedure&gt;</pre>

On performing this test case, if NUT returns a general Echo\_Reply message to TN in 2000 milliseconds, we can infer that NUT implements this function correctly. Test system is the compiler and executer of TSSL and is installed on TN. We have performed the test case against

IPv6 implementation on FreeBSD. During the testing, two packets are generated from TN by test system and one packet is captured by test system. The test result is pass. The test executing procedure is logged by test system, which is shown in Table 4.

**Table 4 The Test Executing Log**

<pre>Test Case [AH Inbound with HMAC-MD5 in Host Transport Mode] starts at 2003-10-29 09:37:06 Sent Packet Echo_Request at 2003-10-29 09:37:06 with size 86 bytes     00 90 27 58 9f 55 00 10 4b 0d 52 91 86 dd 60 00     00 00 00 20 33 ff 3f fe 32 0e 00 01 02 11 00 00     00 00 00 00 00 11 3f fe 32 0e 00 01 02 11 00 00     00 00 00 00 00 19 3a 04 00 00 00 00 10 00 00 00     00 01 87 5f b0 f6 7a c3 2e fd a1 ce 76 d5 80 00     97 44 00 11 00 01 Sent Packet Neighbor_Advertisement at 2003-10-29 09:37:06 with size 110 bytes     00 90 27 58 9f 55 00 10 4b 0d 52 91 86 dd 60 00     00 00 00 38 33 ff 3f fe 32 0e 00 01 02 11 00 00     00 00 00 00 00 11 3f fe 32 0e 00 01 02 11 00 00     00 00 00 00 00 19 3a 04 00 00 00 00 10 00 00 00     00 02 d0 e2 70 2e d5 41 c6 61 0c f4 f7 6a 88 00     1b 5f 60 00 00 00 3f fe 32 0e 00 01 02 11 00 00     00 00 00 00 00 11 02 01 00 10 4b 0d 52 91 Received the Expected Packet at 2003-10-29 09:37:06     00 10 4b 0d 52 91 00 90 27 58 9f 55 86 dd 60 00     00 00 00 08 3a 40 3f fe 32 0e 00 01 02 11 00 00     00 00 00 00 00 19 3f fe 32 0e 00 01 02 11 00 00     00 00 00 00 00 11 81 00 96 44 00 11 00 01 The Verdict Result: Pass Test Case [AH Inbound with HMAC-MD5 in Host Transport Mode] ends at 2003-10-29 09:37:06</pre>
--

## 7. Conclusions

IPv6 is the next generation Internet protocol. More and more IPv6 devices are being produced. This paper has attempted to show why it is necessary to study and practice IPv6 conformance testing. There are two famous organizations being engaged in IPv6 testing:

- The Interoperability Laboratory (IOL) of University of New Hampshire has done many works in IPv6 testing. IPv6 test suite in IOL is described by natural language. The most part of their test suite and test software has not been published yet.
- Another organization being engaged in IPv6 testing is TAHI project in Japan. The script language PERL is used to describe test case. Test suite is executed based on interpretation. Test environment is very complicated. More than one physical tester are necessary. There are some restrictions for IPv6 implementations to be tested.

Also some test device producers, such as Spirent, Agilent and IXIA, have developed IPv6 test devices. Compared to the current researches, this paper identifies why the standard test framework is not appropriate for IPv6 conformance testing. IPv6 test framework is proposed and a new test suite specification language is defined to describe test suite. IPv6 test framework solves two difficulties in IPv6 testing: test packets description and complicated algorithm implementation. Two test methods are adopted, based on which single physical tester is enough for concurrent testing and relay function testing. Test suite is executed based on compilation, therefore the testing efficiency is guaranteed. IPv6 test suite is designed and can be used to evaluate IPv6 implementations. IPv6 test framework and TSSL can also be used to test other computer network protocols without or with few modifications.

## 8. Acknowledgements

The authors would like to acknowledge William Eklow and Jim Webster for their comments and assistance in improving this paper.

## 9. References

- [1] ISO/IEC 9646 1-5: IT-OSI-Conformance testing methodology and framework, 1996
- [2] J.Tian & Z.Li, "The Next Generation Internet Protocol and Its Test", *Proceeding of IEEE International Conference of Communication*, 2001, pp. 210-215
- [3] Zhang Yujun & Li Zhongcheng, "A new formal test suite specification language for IPv6 conformance testing", *Proceedings of International Conference on Communication Technology*, 2003, pp. 174-177
- [4] Csondes, Sarolta Dibuz & Peter Kremer, "Experiments on IPv6 Testing", *Proceedings of IFIP the 13th International Conference on Testing of Communication Systems*, 2000, pp. 113-126
- [5] S.Deering & R.Hinden, "Internet Protocol, Version 6 (IPv6) Specification", IETF RFC 2460, 1998
- [6] Mark A. Miller, P.E. "Implementing IPv6, Second Edition: Supporting the Next Generation Protocols", Published by M&T Books, 1999
- [7] S.Kent & R.Atkinson, "Security Architecture for the Internet Protocol", IETF RFC 2401, 1998
- [8] S.Kent & R.Atkinson, "IP Authentication Header", IETF RFC 2402, 1998
- [9] D.Johnson, C.Perkins and J.Arkko, "Mobility Support in IPv6", IETF Mobile IP Working Group. (working in progress)
- [10] <http://www.tahi.org>
- [11] <http://www.iol.unh.edu>
- [12] <http://www.spirentcom.com/documents/674.pdf>
- [13] [http://www.ixiacom.com/library/white\\_papers/](http://www.ixiacom.com/library/white_papers/)
- [14] <http://advanced.comms.agilent.com/n2x/docs/whitepapers/ipv6-tran.htm>