A Survey of NAT Behavior Discovery in VoIP Applications

Shiang-Ming Hunag[1], *Quincy Wu*[2]
[1]*Department of Computer Science, National Chiao Tung University, Taiwan*
[2]*Graduate Institute of Communication Engineering, National Chi Nan University, Taiwan*
smhuang@cs.nctu.edu.tw, solomon@ipv6.club.tw

Abstract

Because of the foreseeing depletion of Internet Protocol (IP) addresses, Network Address Translation (NAT) is ubiquitously deployed to allow hosts to connect to the Internet through a single shared public IP address, which is a popular approach in deploying wireless local area network (WLAN). Although NAT proves to work well with traditional client/server applications, its existence and non-standard behaviors are the major problem which cripples voice over IP (VoIP) applications. In addition to some efforts which attempt to devise complicated protocols to tackle all NAT varieties, there are also efforts in Internet communities trying to standardize the behaviors of NAT. Therefore, it becomes crucial for a network device to discover the existence of NAT in its subnet and to determine the NAT behaviors, so that it can choose the optimal NAT traversal mechanisms to apply. In this paper, we surveyed the divergent NAT behaviors and then proposed a simplified NAT behavior discovery approach which is more suitable for VoIP applications. The proposed approach can reduce the call establishment time of VoIP applications, which is useful in scenarios where VoIP devices are administrated within a specific domain, e.g., 3G cellular networks.

Keywords: NAT, STUN, NAT behavior discovery.

1 Introduction

Internet Protocol (IP) address is a resource which is required by every device to connect to the Internet. Due to IP version 4 (IPv4) *address depletion* in the 1990s, Network Address Translation (NAT) [1] was proposed to allow a group of devices in an internal private network to hide behind a single server and to access the external Internet using a shared public IPv4 address. NAT was proposed to be a short-term solution, as the original NAT specification [2] described NAT to be: "If nothing else, this solution can serve to provide temporarily relief while other, more complex and far-reaching solutions are worked out". However, nowadays the deployment of NAT has reached an almost ubiquitous situation [3-4], although the "far-reaching solution" of the original NAT specification -- IP version 6 (IPv6) [5] has already been developed for a few years.

NAT translates IPv4 addresses and transport port numbers of the pass-through packets between private and public address realms. This operation invalidates normal behaviors of many protocols, especially those for voice over IP (VoIP) applications [6-9]. Many VoIP protocols, such as Session Initiation Protocol (SIP) [10] and Real-Time Streaming Protocol (RTSP) [11], are problematic when they interwork with NAT because many communication parameters carried within their application layer messages are IPv4 addresses and transport port numbers of the endpoints. Since these parameters are used for setting up end-to-end connections between endpoints, in case one endpoint is located in a private network behind NAT, the IPv4 addresses carried in these messages for that endpoint would be private and therefore not routable from other endpoints in public networks. Figure 1(a) illustrates a scenario where a SIP client and a SIP server are located in two different subnets separated by a NAT device. When the client sends a SIP message to the server, the NAT assigns a mapped address to the private address of the client and creates a binding between the two addresses. This binding is stored in the address mapping table of the NAT, and the NAT translates packets according to this mapping table. Figure 1(b) shows an example of the SIP message sent from the client to the server. In this message, fields used for setting up connections are shown in bold text. We can observe that many parameters carried in this message are not routable because they are the private address which the client detects from its local network interface.

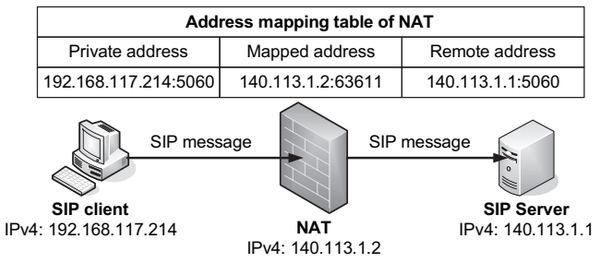
The address binding and the translation mechanism are the basis of NAT. They were documented in the NAT specifications [1-2] when it was proposed. However, these documents did not specify the rules for address binding and the principle for subsequent packet handling. Therefore, each vendor developed their NAT products based on their own understanding about NAT, and this results in divergent NAT behaviors in handling the pass-through packets, especially inbound packets (i.e., packets sent from the public realm to the private realm of a NAT) [12-16].

The divergent behaviors of NAT devices make NAT traversal difficult. It has been shown that a NAT traversal solution which is a good choice in one scenario may behave poorly in many other scenarios [17]. To reduce

*This work was sponsored in part by NSC under grant number NSC 96-2219-E-260-001 and 98-2221-E-260-020.

*Corresponding author: Quincy Wu; E-mail: solomon@ipv6.club.tw

the effort required for endpoints to traverse NAT, Internet Engineering Task Force (IETF) BEHAVE Working Group proposed NAT behavioral requirements to unify the NAT behaviors [18-21]. Moreover, it also proposed a NAT behavior discovery approach for endpoints to detect the behaviors of the NAT in its current network [13].



(a) NAT and Its Address Mapping Table

```
INVITE sip:callee@sip.ipv6.club.tw SIP/2.0
Via: SIP/2.0/UDP
192.168.117.214:5060;branch=z9hG4bK6608;rport
From: <sip:caller@sip.ipv6.club.tw>;tag=8653
To: <sip:callee@sip.ipv6.club.tw>
Call-ID: 7557@192.168.117.214
CSeq: 1 INVITE
Contact: <sip:caller@192.168.117.214:5060>
Content-Type: application/sdp
Content-Length: 144
v=0
o=userX 20000001 20000001 IN IP4 192.168.117.214
s=A call
c=IN IP4 192.168.117.214
t=0 0
m=audio 9000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

(b) The SIP Message Sent from a Client behind a NAT Device

Figure 1 Problematic SIP Messages under NAT

Prior to the NAT behavior discovery approach proposed by the IETF BEHAVE Working Group, there had been proposals from RFC 3489 [22], and Vovida [23] which also aim to discover the NAT behaviors. In this article, we describe these approaches and discuss their applicability. Furthermore, we propose a simplified NAT behavior discovery approach which is more suitable for VoIP applications.

2 NAT Traversal Solutions

NAT invalidates normal behaviors of many VoIP protocols. Several solutions have been proposed to mitigate this problem [3][17][22][24-33]. Among them, Simple Traversal of User Datagram Protocol (UDP) through NAT (STUN) [22] is a technique widely adopted in existing SIP devices. It is a UNilateral Self-Address Fixing (UNSAF) [34] compliant protocol because it enables endpoints behind NATs to determine and fix the transport addresses (i.e., IP addresses plus transport port numbers). In other words, STUN allows a client to discover its mapped public transport address (assigned by the NAT) which will be used in SIP communications, instead of inserting the private address (which is non-routable) found on its network interface. The operation of STUN is illustrated in Figure 2. For each of the local transport addresses, a STUN client (which is also a SIP user agent [UA]) learns from a STUN server the external mapped (reflexive) transport address seen from the public Internet. After the SIP UA obtains the mapped transport address (i.e., IPv4 address 140.113.1.2 and transport port number 63611) from the STUN server, it could include this transport address as parameters to replace the private address in Figure 1(b) and compose valid SIP messages for interaction with the SIP server.

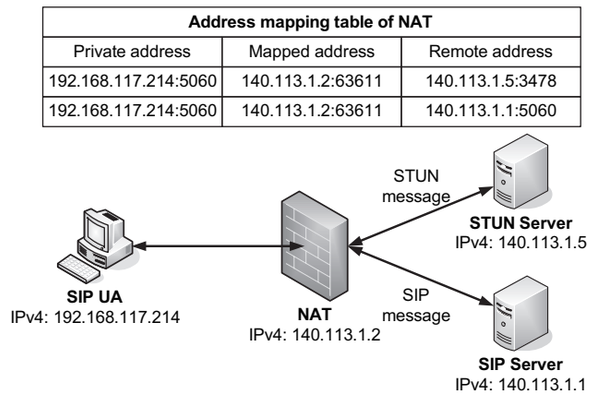


Figure 2 Discovering Mapped Addresses by the STUN Mechanism

Although STUN can solve NAT traversal problems for many NAT devices, it does not work with some NAT devices whose address mapping behavior is endpoint dependent (this kind of NAT is sometimes called a *symmetric NAT*) [32]. Figure 3 shows the address mapping table of an endpoint dependent mapping NAT. Note that this kind of NAT allocates two different mapped addresses when the SIP UA sends packets toward the STUN server and the SIP server (because they are two different endpoints). Consequently, the mapped address learned from the STUN server is not appropriate for the SIP UA to interact with the SIP server. In this circumstance, STUN

may activate an extension -- Traversal Using Relays around NAT (TURN) [33] in order to handle NAT traversal. TURN enables SIP UAs to utilize a STUN server as a traffic relay, which ensures that the SIP UA can always traverses the NAT. However, heavy loading is imposed on the STUN server which may potentially become the bottleneck, and it is very likely that the media packets follow an indirect path from the source to the STUN server and then to the destination, resulting in unacceptable long delay. For these reasons, IETF specifies STUN as a tool to be used as part of other NAT traversal solutions instead of a standalone solution, and this protocol is suggested to be renamed as "Session Traversal Utilities for NAT" [32]. Moreover, IETF extends STUN to run above Transmission Control Protocol (TCP) and Stream Control Transmission Protocol (SCTP), in addition to UDP. Detailed operation of STUN will be elaborated in the next section.

Address mapping table of NAT (endpoint dependent mapping)		
Private address	Mapped address	Remote address
192.168.117.214:5060	140.113.1.2:63611	140.113.1.5:3478
192.168.117.214:5060	140.113.1.2:63655	140.113.1.1:5060

Figure 3 Address Mapping Table of an Endpoint Dependent Mapping NAT

The IETF recommended NAT traversal solutions for VoIP are outbound mechanism [35] and Iterative Connectivity Establishment (ICE) [17][36]. The two mechanisms provide NAT traversal of SIP signaling and its associated media flows, respectively. Actually, they are universal NAT traversal solutions not limited to VoIP applications. The outbound mechanism preserves signaling connections between SIP entities (e.g., UAs, registrar servers, and proxy servers) to allow delivery of SIP messages across NATs. ICE is a generic methodology built upon existing UNSAF protocols for providing a unified NAT traversal solution: with additional extension attributes exchanged through Session Description Protocol (SDP) [37], ICE allows the communication peers to negotiate all possible connection options for setting up media flows; with a STUN server running on each media port of the communication peers, ICE allows connectivity checks between peers even though no public STUN server is available. However, running ICE requires both the two peers (e.g., SIP UAs) supporting the ICE mechanism. If one of them does not support it, the two peers will ignore the ICE extension attributes in SDP and the media flow negotiation falls back to utilize the default transport address in SDP (i.e., the connection address and media port specified in the m= and c= lines, respectively).

3 Behaviors of NAT

In contrast to ICE which is an ambitious protocol heading to tackle all kinds of NAT devices, RFC 4787 proposed basic requirements for a NAT device to minimize the complications it introduces to UDP applications [18]. It specifies detailed requirements for vendors to implement their NAT products, which compensate the lack of behavioral description of the NAT specifications in RFC 1631 [2]. NAT devices fulfilling the requirements in RFC 4787 are called BEHAVE-compliant NATs. Once all NATs follow these requirements, it will be easier for endpoints to traverse NATs because the NAT behaviors will be universally consistent and thus easier to handle. In the following paragraphs, we introduce the behaviors of NATs. The behavior requirements defined in RFC 4787 are also summarized in Table 1 (please refer to RFC 4787 [18] for further details).

3.1 Address Mapping Behaviors

An address binding is created for each unique source transport address carried in outbound packets (i.e., packets passing through the NAT from its private realm to the public realm). Some NATs create address bindings according to the source transport addresses of outbound packets (this behavior is called endpoint independent mapping), while some NATs may create address bindings depending on both the source transport addresses and the destination transport addresses of the outbound packets. This behavior is called endpoint dependent mapping (as earlier described in Section 2).

3.2 Port Assignment Behaviors

NATs have different policies for assigning a mapped transport port number to a private transport address. Some NATs attempt to preserve the same port number used in the private transport address as its mapped port number; some other NATs utilize ports in an *overloading* approach which enables a single public port number to be shared by multiple private transport addresses.

3.3 Address Mapping Refresh Behaviors

Each binding in the NAT address mapping table has a timer. It starts counting down when the binding is created. If a packet passing through the NAT matches the address binding, the timer of this binding will be reset; if there is no packet matching the address binding before the timer expires, this address binding will be deleted. Different NATs may adopt different initial values for the address binding timer, and they may reset the timer under different criteria: some reset it under outbound traffic, others reset it under inbound traffic (i.e., traffic in the reverse direction of

outbound), and still others reset it under bidirectional traffic (i.e., either outbound or inbound traffic).

3.4 Packet Filtering Behaviors

A NAT may maintain an access list for each of its bindings in its address mapping table, in order to allow that: only public endpoints which have received outbound traffic from this NAT can send inbound traffic back. This list may be based on the IPv4 addresses of public endpoints (resulting in address dependent filtering) or based on the transport addresses of public endpoints (resulting in address and port dependent filtering).

3.5 Hairpin Behaviors

Some NATs allow two endpoints in its private realm to communicate with each other using their public mapped transport addresses. This behavior is called hairpin. The operation of hairpin is illustrated in Figure 4(a) where the SIP UA 1 delivers a UDP packet to the mapped transport address of the SIP UA 2. Header translation performed by the NAT is illustrated in Figure 4(b).

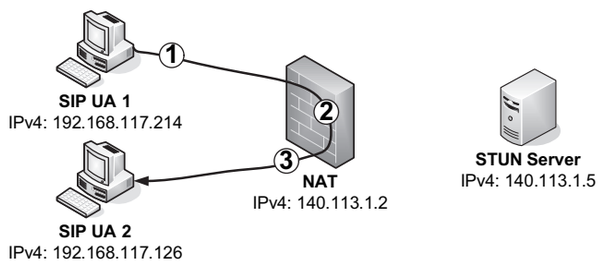
3.6 Packet Fragmentation Behaviors

Some NATs are only capable of handling fragmented packets which are received in order (i.e., a sequence of fragmented packets with the header in the first packet). These NATs simply drop the packets if their arrival sequence is not in order.

3.7 ALG Behaviors

Some NATs hunt for IPv4 addresses within application layer messages of pass-through packets, and translate them if they match a binding in the address mapping table.

Address mapping table of NAT		
Private address	Mapped address	Remote address
192.168.117.214:5060	140.113.1.2:63611	140.113.1.5:3478
192.168.117.126:5060	140.113.1.2:63632	140.113.1.5:3478



(a) Hairpin Operation

Source address	Destination address
192.168.117.214:5060	140.113.1.2:63632
Payload	

①

Source address	Destination address
140.113.1.2:63611	140.113.1.2:63632
Payload	

②

Source address	Destination address
140.113.1.2:63611	192.168.117.126:5060
Payload	

③

(b) Header Translation

Figure 4 Hairpin Behavior of NAT

3.8 ICMP Behaviors

Some NATs drop inbound ICMP messages; some other NATs even delete the corresponding address binding when receiving an ICMP error message.

There are different terminologies mentioned in different documents to classify NAT behaviors. In RFC 3489, NATs are classified into four different types: full cone NAT, restricted cone NAT, port-restricted cone NAT and symmetric NAT. On the other hand, STUN uses BEHAVE-compliant terminologies defined in RFC 4787 to classify NAT behaviors. The RFC 3489 terminologies and the corresponding BEHAVE-compliant terminologies are summarized in Table 2 where full cone NAT is named as a more meaningful terminology: "endpoint independent mapping, no filtering" NAT. Also, the restricted cone NAT, port-restricted cone NAT and symmetric NAT are named as "endpoint independent mapping, address dependent filtering" NAT, "endpoint independent mapping, address and port dependent filtering" NAT and "endpoint dependent mapping, address and port dependent filtering" NAT,

Table 1 NAT Behaviors Required in RFC 4787

NAT behaviors	RFC 4787 requirements
Address mapping	A NAT *must* have endpoint independent mapping behavior.
Port assignment	A NAT *must not* have port overloading behavior.
Address mapping refresh	A NAT's UDP mapping timer *must not* expire in less than two minutes, unless the destination port number is in the well-known port range (0-1023); the NAT mapping refresh direction *must* support outbound refresh behavior.
Packet filtering	If application transparency is important, it is *recommended* that a NAT should have endpoint independent filtering behavior; if a stringent filtering behavior is preferred, it is *recommended* that a NAT should have address dependent filtering behavior.
Hairpin	A NAT *must* support hairpinning.
Packet fragmentation	A NAT *must* support receiving in-order and out-of-order fragments.
ALG	NAT ALGs for UDP-based protocols *should* be turned off.
ICMP	Receipt of any sort of ICMP message *must not* terminate the NAT mapping.

Table 2 Terminologies for Classifying NAT Behaviors

RFC 3489 terminologies for network environments	BEHAVE-compliant terminologies
Open Internet	No address mapping, no filtering
Full cone NAT	Endpoint independent mapping, no filtering
Restricted cone NAT	Endpoint independent mapping, address dependent filtering
Port-restricted cone NAT	Endpoint independent mapping, address and port dependent filtering
Symmetric NAT	Endpoint dependent mapping, address and port dependent filtering
Symmetric UDP firewall	No address mapping, address and port dependent filtering
UDP blocked	Unreachable

respectively. In the remaining parts of this article, we use the BEHAVE-compliant terminologies to describe NAT behaviors.

Note that there are some vendor shipping NAT devices that do not behave consistently and may change their behaviors over time or under load. It is hard to discover the behaviors of these NATs because they are *nondeterministic*. We argue that there is no deterministic behavior of these NAT devices because the NAT itself is a nondeterministic machine.

4 NAT Behavior Discovery

Current methods of NAT behavior discovery include three major approaches: RFC 3489 approach, Vovida approach, and BEHAVE approach. All these approaches are based on the STUN protocol, and they all perform some sequence of tests between the STUN client and the STUN server to enable the STUN client discovering NAT behaviors. The STUN server possesses two IPv4 addresses (e.g., a_1 and a_2) and listens on four sockets that bind on four different transport addresses (e.g., $a_1:p_1$, $a_1:p_2$, $a_2:p_1$ and $a_2:p_2$). The STUN client sends a combination of test packets to different transport addresses of the STUN server. According to the results of these tests, the STUN client (SIP UA) determines the NAT behaviors.

There are three types of tests to be performed between STUN clients and STUN servers: TEST1, TEST2, and TEST3. Figure 5 illustrates these tests. When a STUN client sends a TEST1 request to a STUN server from a client transport address $a_c:p_c$ to a server transport address $a_n:p_n$ (n = 1 or 2), the server will send back a TEST1 response from the same server transport address (i.e., $a_n:p_n$) to the transport address where the TEST1 request was initiated (e.g., through path ① and ② in Figure 5); when the client sends a TEST2 request to the server transport address $a_n:p_n$, the server will send back a TEST2 response from its transport address $a_{(3-n)}:p_{(3-n)}$ to the transport address where the TEST2 was initiated (e.g., through path ① and ③ in Figure 5); when the client sends a TEST3 request to the server transport address $a_n:p_n$, the server will send back a TEST3 response from its transport address $a_n:p_{(3-n)}$ to the transport address where the TEST3 request was initiated (e.g., through path ① and ④ in Figure 5). Note that if these requests and responses are delivered above UDP, they may not successfully arrive at the destination because UDP is an unreliable transport protocol. For reliability, the client performs retransmission on each test: if a test fails (i.e., the client does not receive the response from the server after a request was sent for a specific time), the client considers that either the request sent by itself or the response sent by the server was lost, and the client sends the same request again to the server. The retransmission process will be repeated until the client receives a response from the server, or the process times out. The retransmission process has critical impact on the overall performance in NAT behavior discovery, as we shall see in Section 6.

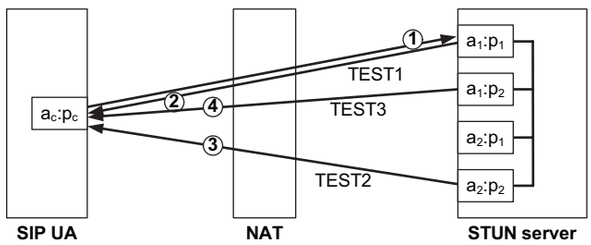


Figure 5 STUN Tests

Through a sequence of requests and responses exchanged between the client and server, STUN allows the client to discover the presence and behaviors of NAT on the path between the client and server. The following subsections describe how these STUN tests are utilized in different approaches.

4.1 RFC 3489 Approach

The flowchart of the RFC 3489 approach is shown in Figure 6 where three different tests are used. This approach detects the seven network environments listed in Table 2. We illustrate the RFC 3489 approach with an example. Suppose the STUN server is in the public network and the SIP UA (which runs a STUN client on its local transport address) is in a private network. In this example, we assume the SIP UA is located behind an endpoint independent

mapping, address and port dependent filtering NAT. The SIP UA determines its network environment with the following steps.

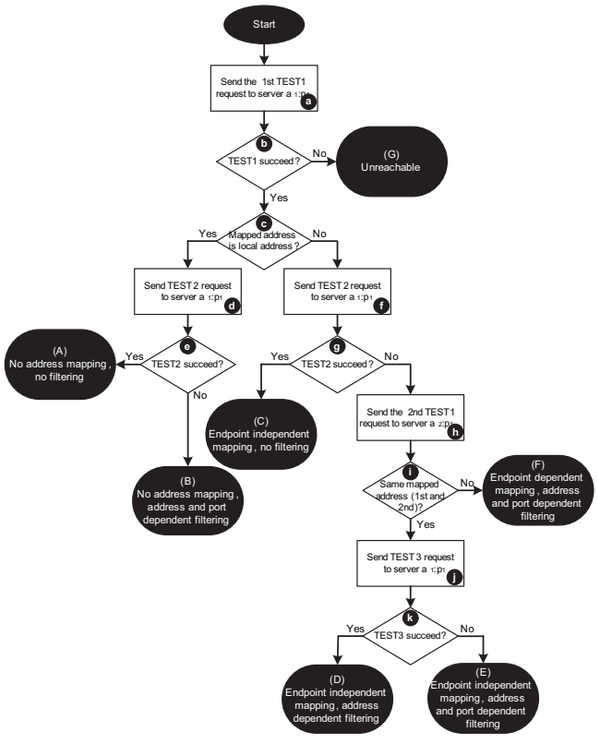


Figure 6 RFC 3489 Approach

Step 1: The client performs a TEST1 test by sending a TEST1 request through the NAT to the server transport address a_1:p_1 (Figure 6ⓐ). The server sends back a response from the transport address a_1:p_1 through the NAT to the client. This response carries the mapped transport address of the local transport address.

Step 2: Upon receipt of the TEST1 response (Figure 6ⓑ), the client discovers that the carried mapped transport address in the TEST1 response is not the local transport address in use (Figure 6ⓒ).

Step 3: The client performs a TEST2 test by sending a TEST2 request through the NAT to the server transport address a_1:p_1 (Figure 6ⓕ). The server sends back a response from the transport address a_2:p_2 which would be blocked by the NAT (because of the address and port dependent filtering behavior).

Step 4: After the retransmission procedure, the TEST2 test fails (Figure 6ⓖ). The client then performs a second TEST1 test by sending a TEST1 request through the NAT to another server transport address a_2:p_1 (Figure 6ⓗ). The server sends back a response from the transport address a_2:p_1 through the NAT to the client. This response also carries the mapped transport address of the local transport address.

Step 5: Upon receipt of the second TEST1 response (Figure 6ⓘ), the client discovers that the carried mapped transport address in this response is the same as that one obtained in the previous TEST1 response (in **Step 2**).

Step 6: The client performs a TEST3 test by sending a TEST3 request through the NAT to the server transport address a_1:p_1 (Figure 6ⓙ). The server sends back a response from the transport address a_1:p_2, which would be blocked by the NAT (because of the address and port dependent filtering behavior).

Step 7: After the retransmission procedure, the TEST3 test fails (Figure 6ⓚ). The client then discovers that it is in a private network behind an endpoint independent mapping, address and port dependent filtering NAT (Figure 6 [E]).

In the example above, the client needs to wait for the latency of two retransmission procedures (in **Step 4** and **Step 7**) which may cause unacceptable interruption for VoIP applications. Imagine a SIP UA running on a mobile device, when it hands over from its home network to a visited network behind an endpoint independent mapping, address and port dependent filtering NAT. If the SIP UA has to wait for a long time to discover its current network environment before it can resume the media transmission, it will cause an unacceptable long interruption.

4.2 Vovida Approach

To reduce the overall delay, Vovida improved the RFC 3489 approach by running some tests in parallel. Figure 7 shows the flowchart of the Vovida approach where several tests (i.e., the TEST1 test in Figure 6ⓐ, the TEST2 test in Figure 6ⓓ and ⓕ, and the TEST3 test in Figure 6ⓙ) all run in parallel (i.e., Figure 7ⓐ). Unlike the RFC 3489 approach that runs the tests sequentially, the Vovida approach runs all tests at the beginning and then discover NAT behaviors according to the results.

In this approach, the TEST1 test in Figure 7ⓒ is the same as the one in Figure 6ⓗ. This test (the second TEST1 test) can not be run at the beginning with the other three tests in Figure 7ⓐ because the STUN client only knows one transport address of the STUN server (e.g., a_1:p_1; obtained through domain name system or dynamic host configuration protocol). The other transport addresses of the STUN server is learned by the STUN client after receiving a response from the STUN server in Figure 7ⓑ. In Figure 7ⓒ, the client may also retransmit TEST2 requests and TEST3 requests to the server if these tests have not succeeded. If one of the TEST2 or TEST3 tests is not finished, the NAT behavior discovery logic will not advance to Figure 7ⓕ.

The flowchart of this approach is very similar to that of the RFC 3489 approach, so we do not elaborate the details here.

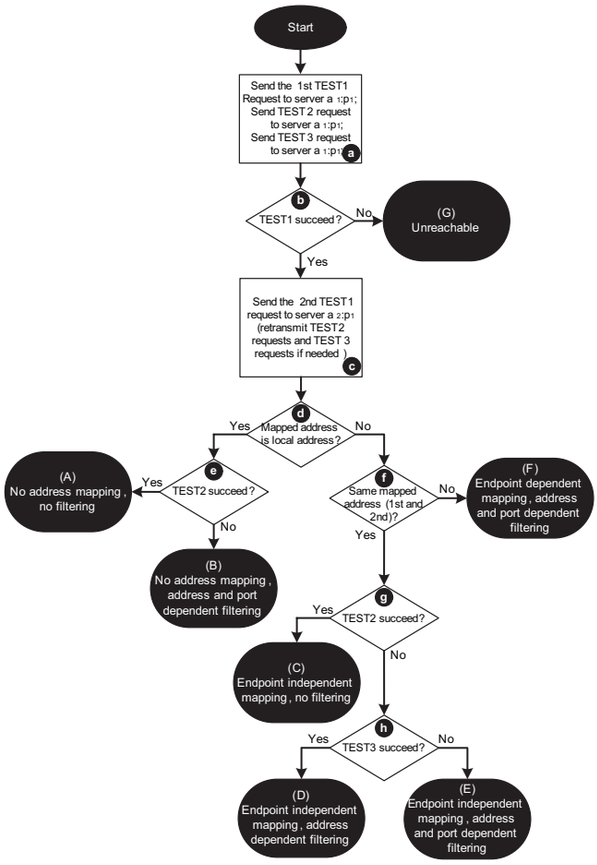
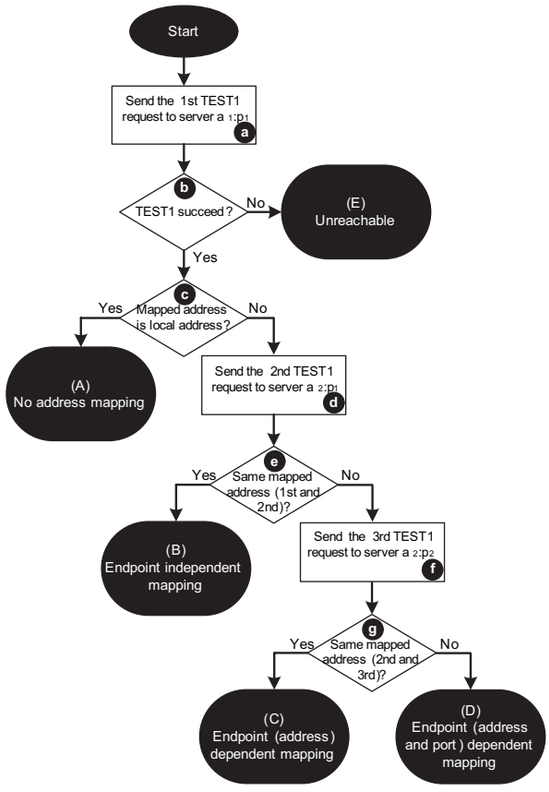


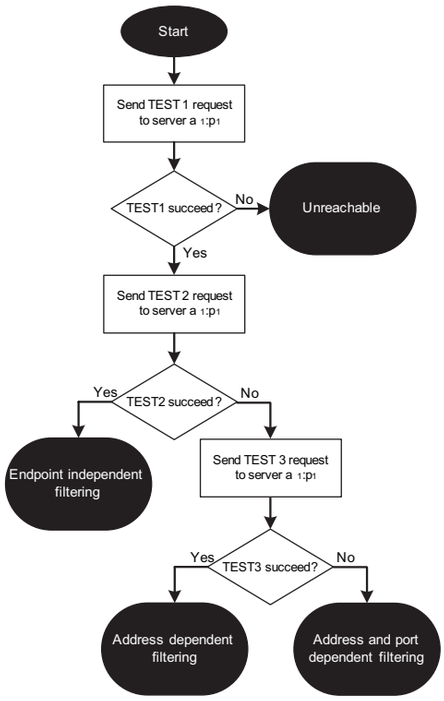
Figure 7 Vovida Approach

4.3 BEHAVE Approach

The logic of NAT behavior determination in the BEHAVE approach is similar to the previous two approaches (which are based on TEST1, TEST2 and TEST3). However, unlike the RFC 3489 approach and the Vovida approach that discover the address mapping behaviors of NATs along with the packet filtering behaviors, the BEHAVE approach decouples the discovery of these two behaviors into two procedures. The flowchart of the BEHAVE approach is shown in Figure 8 where the discovery of address mapping and packet filtering behaviors are illustrated in Figure 8(a) and Figure 8(b), respectively. The separation of these two procedures enables STUN clients to discovery the behaviors of address mapping or packet filtering independently, without unrequited overhead. For example, when a STUN client only wants to know the address mapping behavior, it can simply execute the address mapping behavior discovery procedure (Figure 8[a]) to avoid the overhead of packet filtering behavior discovery (Figure 8[b]). If a STUN client intends to use the BEHAVE approach to discover both the address mapping and packet filtering behaviors, it could combine the two procedures in Figure 8(a) and Figure 8(b) to reduce the overhead [13]. Details are not elaborated here.



(a) Discovery of Address Mapping Behavior



(b) Discovery of Packet Filtering Behavior

Figure 8 BEHAVE Approach

5 NAT Behavior Discover in VoIP Applications

For VoIP applications (e.g., SIP UAs) to traverse NAT, discovery of the packet filtering behaviors in their current network environments is unnecessary. Because SIP UAs are required to keep alive the address binding on NATs for receiving inbound SIP signaling messages, they need to send packets periodically to their peers to refresh the address binding timer. This behavior forces SIP UAs to send outbound packets prior to inbound packets, and helps to conquer the obstacles of firewalls and the packet filtering behaviors of NATs.

Besides the address mapping behaviors and packet filtering behaviors, a NAT has several other behaviors like address mapping refresh behaviors, hairpin behaviors, packet fragmentation behaviors and so on. The discovery of these behaviors is minor for VoIP applications compared with the discovery of the address mapping behaviors. For a SIP UA, the most important information is a suitable transport address to be inserted into its application level messages for connection setup.

By using STUN to discover the presence and behaviors of NAT, SIP devices (especially those that have implemented STUN but have not implemented ICE) are able to traverse NAT. The SIP UAs can utilize this information along with STUN and its extension (TURN) to handle NAT traversal. As shown in Table 1, the BEHAVE-compliant NATs support hairpinning and their address mapping behaviors are endpoint independent. These behaviors can be handled by STUN without relays. Although the STUN document specifies it to be merely a tool as part of other NAT traversal solutions, STUN is useful enough to handle the traversal of BEHAVE-compliant NATs without bottleneck effect or non-optimal routing path problem suffered by TURN. Since more and more NAT products are going to be (or have already been) BEHAVE-compliant NATs, if SIP devices are administrated within a specific domain (e.g., 3G cellular networks) and the NAT devices in this domain are all BEHAVE-compliant, then these SIP devices can use STUN as a standalone NAT traversal solution without any problems. Moreover, many existing SIP devices have implemented STUN, vendors may choose to upgrade their SIP devices with NAT behavior discovery to lower down the cost, rather than straightforwardly implementing the complicated ICE solution on their SIP devices.

We propose a simplified approach for NAT behavior discovery in this scenario. This approach is based on the BEHAVE approach, and its flowchart is shown in Figure 9. It is simpler than the previous three approaches because it only considers the address mapping behaviors and its outputs are more meaningful and more suitable for VoIP applications. Because of the simplification, the proposed approach can reduce the call establishment time of VoIP applications, and therefore the proposed approach is more suitable for VoIP applications compared with the other three approaches. The differences between this approach and the previous approaches are described below.

1. This approach does not intend to discover the filtering behaviors in the network.
2. This approach combines equivalent outputs of previous approaches, i.e., the three outputs in Figure 6 (C), (D), and (E) (or Figure 7[C], [D], and [E]) are combined into a single output: endpoint independent mapping (Figure 9[B]); and the two outputs in Figure 8(a), (C) and (D) are combined into a single output: endpoint dependent mapping (Figure 9 [C]).
3. This approach reduces the complexity of the previous three approaches. It uses only two TEST1 tests (Figure 9 a and d) to determine the address mapping behaviors. Compared with the previous two approaches that need at most four tests to determine the behaviors of a NAT (i.e., two TEST1 tests plus a TEST2 and a TEST3 test for the outputs in Figure 6 [D] and [E]), this approach is simpler.

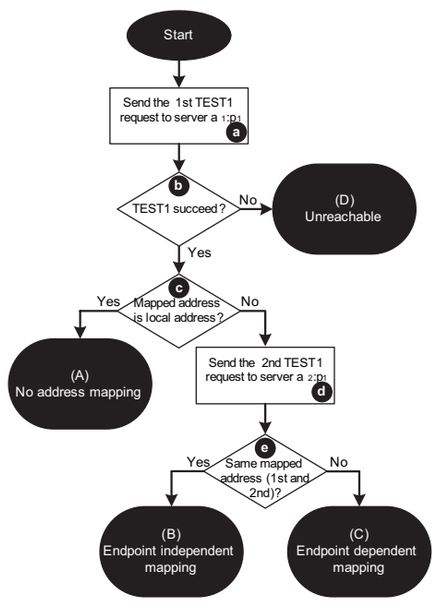


Figure 9 NAT Behavior Discovery in VoIP Applications

To prevent unnecessary using STUN servers as traffic relays, a SIP UA needs to determine whether to activate TURN or not. In other words, the SIP UA has to detect the address mapping behavior of the NAT to see if it is endpoint dependent mapping. If the SIP UA discovers an endpoint independent mapping NAT, then it can simply use STUN to traverse the NAT. Otherwise it must use TURN. Table 3 summarizes possible network environments for a SIP UA (only network translation behaviors are highlighted,

while firewalls and the filtering behaviors of NATs are not considered here because they are not significant as described in the previous paragraphs).

Table 3 Network Environments and the Corresponding NAT Traversal Techniques

Network environments	NAT traversal techniques
No address mapping	None
Endpoint independent mapping	STUN
Endpoint dependent mapping	TURN

6 Cost Analyses

We analyze the cost of these approaches in discovering the NAT address mapping behavior. The analysis environment is a simple STUN configuration as shown in Figure 5. These approaches are independent of the RTP traffic because they are executed before the establishment of VoIP calls, and therefore the performance metrics used here are *total execution time* and *count of generated packets*.

6.1 Parameters and Basic Derivations for Analytical Modeling

In VoIP applications, STUN tests are transmitted above UDP which does not support reliable packet transportation. If the STUN client does not receive a response from the STUN server for its transmitted test packet, it retransmits the test packet when the retransmission timer expires. The average delay for a failed test, $T_{timeout}$, needs to wait until the retransmission procedure is completed, so it will be equal to the total latency of the retransmission procedure.

$$T_{timeout} = \sum_{i=1}^{N} T_i \ ,$$

where N is the number of trials for a test, and T_i is the retransmission timer value of the i-th transmission trial. In the STUN protocol, T_i is an exponentially grown value:

$$T_i = \begin{cases} \gamma^{\text{i-1}} \cdot \Delta, & 1 < i \le m, \\ \gamma^{\text{m-2}} \cdot \Delta, & m < i < n, \end{cases}$$

where Δ is the initial value, γ is the growth factor, and m is the boundary of growth. The average delay for a successful test, $T_{response}$, is

$$T_{response} = T_{RTT} + \sum_{i=1}^{N} p_i T_{i-1} \ ,$$

where T_{RTT} is the round-trip time between then STUN client and STUN server, p_i is the probability that the STUN client successfully gets the response from the STUN server in the i-th transmission trial. p_i can be computed by:

$$\begin{aligned} p_i &= \Pr\left[\textit{all the previous } i\text{-}1 \textit{ transmission trials fail}\right] \cdot \\ &\quad \Pr\left[i\text{-}\textit{th transmission trial succeed}\right] \\ &= [1-(1-p)^2]^{i-1} \cdot (1-p)^2, \end{aligned}$$

where p is the packet loss rate.

Similarly, the count of generated packets of a failed test, $C_{timeout}$, suffers the whole retransmission procedure, and its average value is

$$\begin{aligned} Ctimeout &= \sum_{i=1}^{N} \left[\left(2 \cdot (1-p)^2\right) + \left(2 \cdot p \cdot (1-p)\right)\left(1 \cdot p\right) \right] \\ &= (2-p) \cdot N. \end{aligned}$$

The average count of generated packets for a successful test, $C_{response}$, is

$$C_{response} = \sum_{i=1}^{N} p_i [2 + (i-1)(2-p)] \ .$$

In the STUN protocol, the default parameters are: N = 7, Δ = 500ms, γ = 2, and m = 6 (which implies T_1 = 500ms, T_2 = 1000ms, T_3 = 2000ms, T_4 = 4000ms, T_5 = 8000ms, T_6 = 16000ms, T_7 = 8000ms). We can foresee that $T_{timeout}$ = 39.5 seconds which is unacceptable for VoIP applications.

6.2 Total Execution Time

The total execution time needed for these approaches is divergent because each of them utilizes a different algorithm to discover a network environment. For example, when the NAT behavior is endpoint independent mapping, address and port dependent filtering, the RFC 3489 approach needs to perform four tests (Figure 6ⓐ, ⓕ, ⓗ and ⓙ) and it requires the latency of two failed tests (Figure 6ⓕ and ⓙ) plus the latency of two successful tests (Figure 6ⓐ and ⓗ). However, in the same environment, the total execution time required for the Vovida approach is the latency of one failed test or two successful tests (Figure 7ⓐ and ⓒ; the reason will be elaborated in the next paragraph), while the total execution time required for the BEHAVE approach and the proposed approach is the latency of two successful tests (Figure 8(a) ⓐ and ⓓ, and Figure 9ⓐ and ⓓ, respectively).

Suppose T_{RTT} = 50ms and loss rate p = 0 (which means that all requests/responses arrive at the destination successfully), the total execution time of these approaches is shown in Figure 10. Note that the Vovida approach

performs tests in parallel, and therefore its total execution time is

$$\begin{aligned} &max\{latency\ of\ TEST2,\ latency\ of\ TEST3,\\ &latency\ of\ TEST1+latency\ of\ another\ TEST1\}\\ &= max\{T_{timeout},\ 2T_{response}\} \end{aligned}$$

In other words, its latency is either $2T_{response}$ if the three tests in Figure 7ⓐ are successful, or $T_{timeout}$ if any one of the three tests in Figure 7ⓐ fails. Among these approaches, the proposed approach is faster than the other approaches in any case. The Vovida approach has similar execution latency for every network environments (which is caused by the parallel process execution), and the RFC 3489 approach needs longest execution time for almost all network environments.

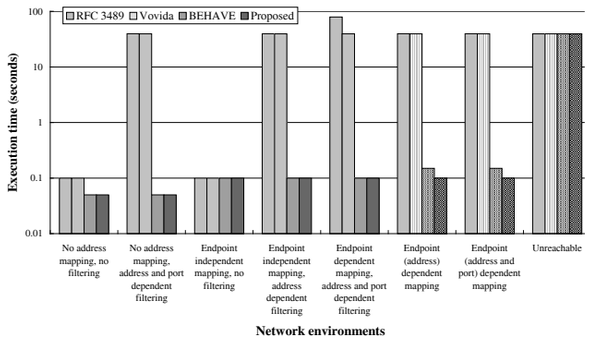


Figure 10 Total Execution Time Comparison ($p = 0$)

6.3 Count of Generated Packets

Suppose $p = 0$, two packets will be delivered between the STUN client and the STUN server (one for request and one for response) for a successful test and 14 packets are generated for a failed test. The packet count comparison between these approaches is shown in Figure 11. Among these approaches, the proposed approach generates fewer packets than the other approaches in any case. The Vovida approach generates the most packets because of its parallel characteristic.

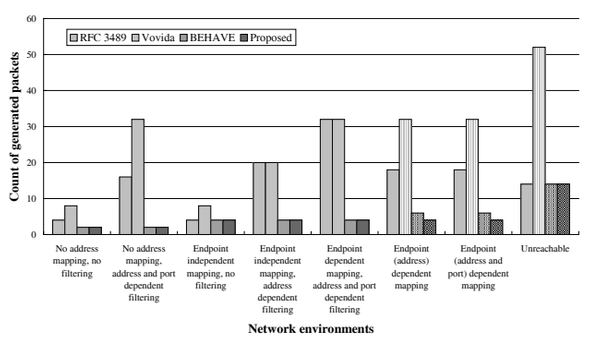


Figure 11 Packet Count Comparison ($p = 0$)

From the analyses of total execution time (Figure 10) and count of generated packets (Figure 11), we observed that the RFC 3489 approach has longer execution time and it needs to generate a lot of packets to discover the network environments, and the Vovida approach has to generate a lot of test packets because it launches parallel tests in Figure 7ⓐ and ⓒ. Compared with these approaches, the proposed approach is more efficient in most cases.

We further run these approaches in an experimental environment to measure their performance. The experimental results are very close to the analytical results above so we omit them here.

From the analyses in Sections 6, we know that the total execution time and count of generated packet of a NAT behavior discovery approach are dominated by $T_{timeout}$ and $C_{timeout}$, respectively. The proposed approach is superior to the other three approaches in any case whatever the loss rate p is, because it requires less $C_{timeout}$ or $T_{timeout}$.

7 Conclusion

In wireless network deployment, NAT is commonly adopted to allow multiple devices accessing the Internet with only a public IP address. This paper presents the NAT issues of VoIP applications, and introduces the current solutions for user applications to traverse NATs. We surveyed the divergent NAT behaviors and the IETF requirements for NAT devices. Moreover, we described how to utilize STUN to discover the behavior of the NATs, especially the BEHAVE-compliant NATs. We also proposed a simplified NAT behavior discovery approach for VoIP applications. The proposed approach is useful in scenarios where VoIP devices are administrated within a specific domain, e.g., 3G cellular networks.

References

[1] Pyda Srisuresh and Kjeld Borch Egevang, *Traditional IP Network Address Translator (Traditional NAT)*, January, 2001. IETF RFC 3022.

[2] Kjeld Borch Egevang and Pyda Francis, *The IP Network Address Translator (NAT)*, May, 1994. IETF RFC 1631.

[3] Whai-En Chen, Ya-Lin Huang and Han-Chieh Chao, *NAT Traversing Solutions for SIP Applications*, *Journal on Wireless Communications and Networking*, Vol.2008, No.4, 2008, pp.1-9.

[4] Wen-Sung Chen and Wen-Kang Jia, *An IP Shared Device Based on the Network Port Translation*, *Journal of Internet Technology*, Vol.7, No.1, 2006, pp.85-93.

[5] Stephen E. Deering and Robert M. Hinden, *Internet Protocol, Version 6 (IPv6) specification*, December, 1998. IETF RFC 2460.

[6] Victor Paulsamy and Samir Chatterjee, *Network Convergence and the NAT/Firewall problems*, *Proc. 36th Hawaii International Conference on System Sciences*, Waikoloa, HI, January, 2003, p.125c.

[7] Samir Chatterjee, Tarun Abhichandani, Bengisu Tulu and Haiqing Li, *SIP-Based Enterprise Converged Networks for Voice/Video-Over-IP: Implementation and Evaluation of Components*, *IEEE Journal on Selected Area in Communications*, Vol.23, No.10, 2005, pp.1921-1933.

[8] Hechmi Khlifi, Jean-Charles Grégoire and James Phillips, *VoIP and NAT/Firewalls: Issues, Traversal Techniques, and a Real-World Solution*, *IEEE Communications Magazine*, Vol.44, No.7, 2006. pp.93-99.

[9] Chris Boulton, Jonathan Rosenberg, Gonzalo Camarillo and Francois Audet, *Best current practices for NAT traversal for client-server SIP*, February, 2011. IETF Internet Draft (Work in Progress), draft-ietf-sipping-nat-scenarios-15.

[10] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley and Eve Schooler, *SIP: Session Initiation Protocol*, June, 2002. IETF RFC 3261.

[11] Henning Schulzrinne, Anup Rao and Robert Lanphier, *Real Time Streaming Protocol (RTSP)*, April, 1998. IETF RFC 2326.

[12] Shiang-Ming Huang, Quincy Wu and Yi-Bing Lin, *Enhancing teredo IPv6 tunneling to traverse the symmetric NAT*, *IEEE Communications Letters*, Vol.10, No.5, 2006, pp.408-410.

[13] Derek C. MacDonald and Bruce B. Lowekamp, *NAT behavior discovery using STUN*, May, 2010. IETF RFC 5780.

[14] Bryan Ford, Pyda Srisuresh and Dan Kegel, *Peer-to-Peer Communication Across Network Address Translators*, *Proc. 2005 USENIX Annual Technical Conference*, Anaheim, CA, April, 2005, pp.179-192.

[15] Pyda Srisuresh, Bryan Ford and Dan Kegel, *State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)*, March, 2008. IETF RFC 5128.

[16] Geoff Huston, *Anatomy: A Look Inside Network Address Translators*, *The Internet Protocol Journal*, Vol.7, No.3, 2004, pp.2-32.

[17] Jonathan Rosenberg, *Interactive Connectivity Establishment (ICE): A protocol for Network Address Translator (NAT) traversal for offer/answer protocols*, April, 2010. IETF RFC 5245.

[18] Francois Audet and Cullen Jennings, *Network Address Translation (NAT) behavioral requirements for unicast UDP*, January, 2007. IETF RFC 4787.

[19] Saikat Guha, Kaushik Biswas, Bryan Ford, Senthil Sivakumar and Pyda Srisuresh, *NAT behavioral requirements for TCP*, October, 2008. IETF RFC 5382.

[20] Pyda Srisuresh, Bryan Ford, Senthil Sivakumar and Saikat Guha, *NAT behavioral requirements for ICMP*, April, 2009. IETF RFC 5508.

[21] Dan Wing and Toerless Eckert, *IP multicast requirements for a Network Address Translator (NAT) and a Network Address Port Translator (NAPT)*, February, 2008. IETF RFC 5135.

[22] Jonathan Rosenberg, Joel Weinberger, Christian Huitema and Rohan Mahy, *STUN -- Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs)*, March, 2003. IETF RFC 3489.

[23] Vovida's STUN Client and Server Library, http://www.vovida.org/applications/downloads/stun/

[24] Universal Plug and Play (UPnP), http://www.upnp.org

[25] Henrik Levkowetz and Sami Vaarala, *Mobile IP traversal of Network Address Translation (NAT) devices*, April, 2003. IETF RFC 3519.

[26] Jani Hautakorpi, Gonzalo Camarillo, Robert F. Penfield, Alan Hawrylyshen and Medhavi Bhatia, *Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) deployments*, April, 2010. IETF RFC 5853.

[27] Whai-En Chen, Quincy Wu, Yi-Bing Lin and Yung-Chieh Lo, *Design of SIP Application Level Gateway for IPv6 Translation*, *Journal of Internet Technology*, Vol.5, No.2, 2004, pp.147-154.

[28] Michael Borella, David Grabelsky, Jeffrey Lo and Kunihiro Taniguchi, *Realm specific IP: Protocol specification*, October, 2001. IETF RFC 3103.

[29] Pyda Srisuresh, Jiri Kuthan, Jonathan Rosenberg, Andrew Molitor and Abdallah Rayhan, *Middlebox communication architecture and framework*, August, 2002. IETF RFC 3303.

[30] Marcus Leech et al, *SOCKS protocol version 5*, March, 1996. IETF RFC 1928.

[31] Jonathan Rosenberg and Henning Schulzrinne, *An extension to the Session Initiation Protocol (SIP) for symmetric response routing*, August, 2003. IETF RFC 3581.

[32] Jonathan Rosenberg, Rohan Mahy, Philip Matthews and Dan Wing, *Session Traversal Utilities for NAT (STUN)*, October, 2008. IETF RFC 5389.

[33] Rohan Mahy, Philip Matthews and Jonathan Rosenberg, *Traversal Using Relays around NAT (TURN): Relay extensions to Session Traversal Utilities for NAT (STUN)*, April, 2010. IETF RFC 5766.

[34] Leslie Daigle (Ed.), *IAB considerations for UNilateral Self-Address Fixing (UNSAF) across network address translation*, November, 2002. IETF RFC 3424.

[35] Cullen Jennings, Rohan Mahy and Francois Audet (Eds.), *Managing client initiated connections in the Session Initiation Protocol (SIP)*, October, 2009, IETF RFC 5626.

[36] Jonathan Rosenberg, Ari Keranen, Bruce B. Lowekamp and Adam Roach, *TCP candidates with Interactive Connectivity Establishment (ICE)*, February, 2011. IETF Internet Draft (Work in Progress), draft-ietf-mmusic-ice-tcp-12.

[37] Mark Handley and Van Jacobson, *SDP: Session description protocol*, April, 1998. IETF RFC 2327.

Biographies

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

Quincy Wu received his BS degree in Mathematics from National Tsing Hua University in 1992, and his PhD in Computer Science and Information Engineering from National Tsing Hua University in 2000. He joined National Center for High-Performance Computing with the NBEN (National Broadband Experimental Network) project, where he successfully designed and established the first island-wide IPv6 network among universities in Taiwan. In 2003, he began serving as a research assistant professor with National Chiao Tung University, and helped National Telecommunications Program Office to deploy a SIP-based VoIP Platform across several universities. Since 2004, he co-chairs the SIP-H323 Working Group of Asia-Pacific Advanced Network (APAN) and helped Taiwan Academic Network (TANet) to design and deploy VoIP services. He was appointed as an assistant professor of Graduate Institute of Communication Engineering, National Chi Nan University (NCNU) in 2005 and helped initiating the VoIP over WiMAX project in NCNU. In 2007, he was elected as the chairman of the SIP-H323 Working Group of APAN and helps coordinating the VoIP activities in Asia-Pacific academic networks. In 2009, he was promoted as an associate professor in NCNU and elected as the division head of the Network Division of Nantou Regional Center in TANet. His current research interests include session initiation protocol (SIP), open service architecture, Internet protocol version 6 (IPv6), design and analysis of approximation algorithms, wireless mesh network, and advanced metering infrastructure (AMI) in Smart Grid.