To Call or To Be Called Behind NATs is Sensitive in Solving the Direct Connection Problem

Cheng-Yuan Ho, *Member, IEEE,* Chien-Chao Tseng, *Member, IEEE,*
Fu-Yu Wang, Jui-Tang Wang, and Ying-Dar Lin, *Senior Member, IEEE*

***Abstract*—In this article, we first depict the call-role sensitivity problem in Network Address Translation (NAT) traversal, and then propose an approach to resolving the problem. The problem is whether a direct connection can be found between two peers across NATs mainly depends on the NAT type at the caller's side. We propose the extra-candidate connectivity check where both peers initiate a direct connectivity check to eliminate the effect of the call role. We have implemented the extra-candidate connectivity check and conducted experiments with 18 different NATs. Experimental results show that our approach can indeed resolve the call-role sensitivity problem, and maximize the direct connectivity rate (DCR) which is improved by 18.71% from the original scheme.**

***Index Terms*—NAT, NAT traversal, direct connection.**

I. Introduction

In this paper, we present the call-role sensitivity problem that may occur when two peers try to establish a peer-to-peer (P2P) direct connection across Network Address Translation (NAT) devices, and then propose a mechanism to resolve the problem. The call-role sensitivity problem is whether a P2P direct connection can be found between two peers behind different NATs mainly depends on the NAT at the *caller's* side.

NAT is a common solution to the IP address depletion problem of Internet Protocol version 4 (IPv4). The basic concept of NAT is to map an address of one realm to an address of another, while also providing transparent routing for the hosts behind an NAT, henceforth referred to as *internal hosts* [1]. Traditional NAT adopts Network Address Port Translation (NAPT) and can translate many internal network addresses and their transport ports into a single external network address and many transport ports, hereafter referred to as *mapped-addresses*. Therefore, with NAPT, an NAT can serve many internal hosts in a private network with a single external public IP address.

However, NAT also introduces the NAT traversal problem. A host *external* to an NAT can not originate connections to an internal host unless the NAT has established a mapped-address for the internal host. Furthermore, NAT may employ filtering rules to block unauthorized inbound traffics.

Manuscript received August 12, 2010. The associate editor coordinating the review of this letter and approving it for publication was M. Ma.

C.-Y. Ho is with the D-Link NCTU Joint Research Center, National Chiao Tung University, Taiwan (e-mail: cyho@csie.nctu.edu.tw).

F.-Y. Wang is with the Network Benchmarking Lab (NBL), NCTU, Taiwan (e-mail: sagual@nbl.org.tw).

J.-T. Wang is with the Industrial Technology Research Institute (ITRI), Taiwan (e-mail: rtwang@itri.org.tw).

C.-C. Tseng (corresponding author) and Y.-D. Lin are with the Department of Computer Science, NCTU, Taiwan (e-mail: {cctseng, ydlin}@cs.nctu.edu.tw).

This work was supported in part by D-Link Co., Taiwan.

Digital Object Identifier 10.1109/LCOMM.2010.102810.101471

Several such protocols have been proposed to facilitate NAT traversal. Session Traversal Utilities for NAT (STUN) [2], [3] is a protocol that allows an internal host to *discover* the presence and type of its NAT and mapped-address. In certain situations, it is necessary for two hosts behind NATs to use an intermediate node as a communication *relay*. Traversal Using Relays around NAT (TURN) [4] is a protocol that allows two hosts to control the operation of the relay and to exchange packets with its peers through the relay. Interactive Connectivity Establishment (ICE) [5] makes use of both STUN and TURN for any two peers to *discover* and *exchange* three *candidate* transport addresses including *host* (itself), *mapped* (on the NAT), and *relayed* (on the relay server) transport addresses. Moreover, it also provides a connectivity check algorithm for two hosts to determine and agree on which *candidate pair* to use for NAT traversal. Hence, ICE is currently the most promising NAT traversal technique.

However, our experiments show that the role of peers affects the ICE connectivity check result and makes ICE fail to discover a direct connection which indeed exists between two peers. Failure in direct connection tests suggests the use of a relay to forward packets on behalf of two peers. The use of relays may unnecessarily increase the bandwidth cost and processing load of the relay servers, and even worse the end-to-end packet delay, which is especially harmful to real-time P2P applications [6].

In this work, we first identify the call-role sensitivity problem, and explain why ICE suffers the problem. Then, we propose and implement a mechanism to resolve the problem and present the experimental results to justify our proposal. The remainder of this article is organized as follows. In Section II, we explain what a call-role sensitivity problem is and how it occurs. In Section III, we present the proposed method and the experiment results from our test bed. Finally, we summarize the findings and provide suggestions for further research in Section IV.

II. NAT Types and Call-role Sensitivity Problem

NATs can be classified into four NAT types: *full cone*, *address restricted cone*, *port restricted cone*, and *symmetric* [2], according to the mapping and filtering rules. All cone NATs have the mapping rule that *all* requests from the same internal transport address are mapped to the *same* mapped-address. Whereas, a symmetric NAT is one where each request from the same internal transport address to a specific remote transport address is mapped to a *unique* mapped-address. A full cone NAT allows any external host sends packets to an internal host if the NAT already has a mapped-address for the internal host. However, an address restricted cone NAT allows

Caller \ Callee	FC	AR	PR	SM
FC				
AR				
PR				
SM				

Successful
Failed
FC: Full Cone NAT
AR: Addr. Restricted NAT
PR: Port Restricted NAT
SM: Symmetric NAT

Fig. 1. Theoretical direct connectivity.

an external host with a particular IP address sends packets to a mapped-address only if the NAT had sent a packet with the mapped-address to *that* IP address before. A port restricted cone NAT is like a restricted cone NAT, but the restriction includes port numbers. A symmetric NAT adopts the same filtering rule as port restricted cone NAT.

Accordingly, whether two hosts behind different NATs can establish a direct connection depends on which host initiates the communication first. For example, suppose *host1* and *host2* are behind a full cone NAT and a port restricted NAT, respectively, and know each other's mapped-address by some signaling protocol such as Session Initiation Protocol (SIP) or STUN. The *host1* can not establish a direct connection to *host2* if *host1* initiates the communication first because the packets sent by *host1* will be dropped by the port restricted NAT of *host2*. However, because of the loose restriction rule of the full cone NAT, *host2* can establish a direct connection to *host1* when *host2* initiates the communication first [7]. ICE resolves the above problem by having *each* of the two hosts initiate a connectivity check for each candidate pair. Figure 1 summarizes the cases where ICE can establish a direct connection theoretically under various NAT combinations.

Among various NAT types, port restricted and symmetric NATs employ the most stringent filtering rule. However, when both peers are behind the port restricted NATs they still can establish a direct connection because both of them initiate a connectivity check for the *same* mapped-address pair, and the *latter* one or *both* (if they initiate at the same time) will pass through the other end's NAT. However, when one peer is behind a symmetric NAT and the other peer is behind a port restricted or a symmetric NAT, it is impossible for the two peers to establish a direct connection [8]. The reason is that symmetric NATs assign a *new* mapped-address for *each* outgoing connection of an internal host, and the host can not foresee the mapped-address and notify the corresponding host in advance via a third party signaling server. Thus, the request from new mapped-address will be blocked by the peer's NAT.

However, some NATs do not behave as Fig. 1 because of *connection tracking*, one feature built on Netfilter within the Linux kernel. In this article, if an NAT has the attribute of connection tracking, this NAT is called *Linux-based NAT*. Connection tracking allows Linux-based NATs to keep track of connections and enforce more intelligent filtering policies. Furthermore, if a Linux-based non-full cone NAT denies *incoming* packets destined for a mapped-address of a connection, it also blocks the mapped-address. Later, when the NAT receives *outgoing* packets of the same connection, it will assign a *new* mapped-address for this connection.

Therefore, if we consider Linux-based and non-Linux-based NATs, the 16 combinations in Fig. 1 extend to 64 combinations

Caller \ Callee		FC		AR		PR		SM	
		NL	L	NL	L	NL	L	NL	L
FC	NL	3	3	3	3	3	3	3	3
	L	3	3	3	3	3	3	3	3
AR	NL	3	3	3	3	3	3	3	3
	L	2	2	2	2	1	1	1	1
PR	NL	3	3	3	3	3	3	4	4
	L	2	2	2	2	1	1	4	4
SM	NL	3	3	3	3	4	4	4	4
	L	2	2	2	2	4	4	4	4

L: Linux-based NAT; NL: Non-Linux-based NAT; Number: Group

Fig. 2. Theoretical Linux-based direct connectivity.

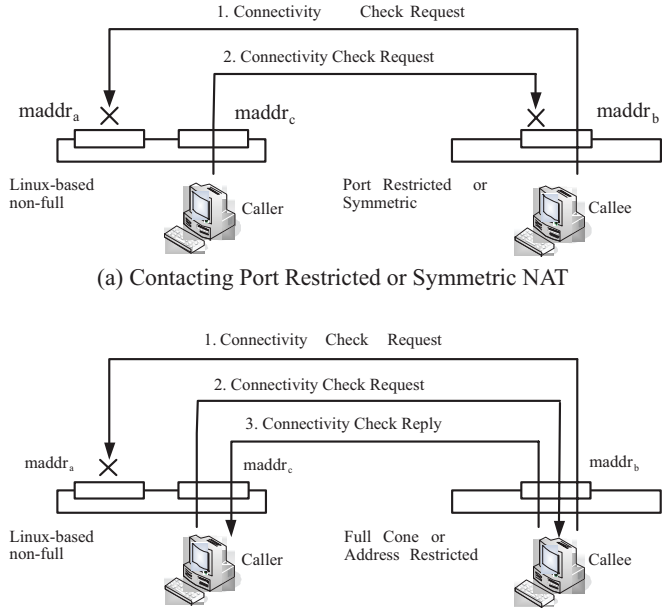

Fig. 3. Connectivity check for Linux-based non-full cone.

as shown in Fig. 2. Moreover, we can classify and label these 64 combinations into four groups: 1) from successful to failed, 2) successful with a different mapped-address, 3) successful as before, and 4) failed as before.

Figure 3(a) illustrates why NATs of *group 1* does have direct connectivity. Note that, in ICE, a *callee* collects all candidates of both peers *before* a caller does because the caller encapsulates three candidate transport addresses in the request. Therefore, the callee will send a connectivity check request for the mapped-address pair *earlier* than the caller. Suppose the mapped-addresses of the caller and callee are $maddr_a$ and $maddr_b$, respectively. With connection tracking, a Linux-based non-full cone NAT not only drops the connectivity check request sent by the callee, but also blocks $maddr_a$ and assigns a new mapped-address $maddr_c$ for the request sent by the caller. This request originated from $maddr_c$ will also be dropped by the port restricted cone or symmetric NAT at the callee's side because the NAT did not send packets from $maddr_b$ to $maddr_c$ before. This is the call-role sensitivity problem. However, NATs of *group 2* has direct connectivity with a *new* mapped-address if the callee is behind a full cone or address restricted cone NAT as shown in Fig. 3(b).

III. Proposed Mechanism and Experiment Results

In order to solve the call-role sensitivity problem, i.e., turning *group 1* into *group 2*, for Linux-based NATs, two hosts could exchange more NAT information and the host behind a non-Linux-based NAT could send connectivity check requests first. However, this approach requires more sophisticated NAT type detection mechanism and modifications on both STUN servers and clients.

A much simpler approach is to let the caller initiate a connectivity check to the callee with an *extra* candidate pair. In this section, we first detail the extra-candidate connectivity check mechanism, and then show testbed experiment results.

A. Extra-candidate Connectivity Check

With the extra-candidate connectivity check, each of two hosts can acquire, from the STUN server, and exchange *one more* mapped-address of NATs, in addition to the original mapped-address. For the extra mapped-address pair, the *caller* starts the connectivity check *first*, unlike the connectivity check for the normal pair. Because the caller and callee each starts a connectivity check for one pair, at least one request will pass through a Linux-based non-full cone NAT. Accordingly, two hosts behind NATs of group 1 now can establish a direct connection unless both are behind Linux-based port restricted NATs. The extra-candidate connectivity check could be done *after* or *in parallel* with the other connectivity checks. Sequential checks will prolong the time required to find a direct connection for the Linux-based NATs, whereas parallel ones need more sophisticated software efforts and thus more computation power.

The implementation of extra-candidate connectivity check is simple because many functions used in ICE can be re-called. For example, a host acquires one mapped-address of NAT and the callee starts the connectivity checks that are standard operating procedures in ICE. Therefore, we can re-call these functions to get one more mapped-address and let the caller start the connectivity check with the extra-candidate pair. As for the sequential or parallel check depends on when the caller starts the extra-candidate connectivity check.

B. Testbed and Experiment Result

We have implemented the extra-candidate connectivity check as mentioned above and conducted experiments with the NATs shown in Table I. Among the eighteen NATs, NATs 10 to 13 are Linux-based port restricted NATs. Furthermore, although NAT 2 is a Linux-based full cone NAT, it acts like an address restricted one for the connections originated by an internal host if the internal host attempts to create several connections spontaneously within a short period. Hence, as shown in Fig. 4, ICE can successfully find direct connections in only 203 NAT combinations out of the 324 combinations. However, with the extra-candidate connectivity check, we can establish 219 direct connections, which is 18.71% improvement over ICE. Furthermore, as mentioned previously, when two hosts are both behind Linux-based port restricted NATs, or when one peer is behind a symmetric NAT and the other is behind a port restricted or a symmetric NAT, it is impossible for the two peers to establish a direct connection. Therefore, for the selected NATs, 219 combinations are the maximum achievable direct connectivity.

TABLE I
NAT Brands and Models in Experiments

No	Brand	Model	No	Brand	Model
1	Dlink	Di-604	10	Edimax	Br-6204wg
2	Smc	Smcwbr14-g2	11	Linux	Iptables
3	Corega	Cg-barmx2	12	Linksys	Befsr41
4	Planex	Blw-54mr	13	Linksys	WRT150N
5	Smc	SmcWGBR-14n	14	Abocom	Fsm410
6	3com	3crwer 100-75	15	Asus	Rx3041
7	Belkin	F5d8231tw4	16	Netgear	Pr614
8	Draytek	vigor2104p	17	Zyxel	P334
9	Lemel	Lm-wlg6400	18	Freebsd	Pf

		FC		AR			PR								SM				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
FC	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	2	3	3	3	3	3	1	1	1	1	1	1	1	1	1	1	1	1	1
AR	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
PR	6	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4
	7	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4
	8	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4
	9	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4
	10	2	2	2	2	2	1	1	1	1	1	1	1	1	4	4	4	4	4
	11	2	2	2	2	2	1	1	1	1	1	1	1	1	4	4	4	4	4
	12	2	2	2	2	2	1	1	1	1	1	1	1	1	4	4	4	4	4
	13	2	2	2	2	2	1	1	1	1	1	1	1	1	4	4	4	4	4
SM	14	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4
	15	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4
	16	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4
	17	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4
	18	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4

Successful in both methods
Failed in both methods
Successful in Extra-Connectivity Check only (turning group 1 into group 2)

Fig. 4. Experimental results of extra-connectivity check.

IV. Conclusions

In this article, we have investigated the call-role sensitivity problem and explained why two hosts behind NATs cannot communicate with each other directly in certain situations where they ought to have a direct connection. Furthermore, we have proposed an extra-connectivity check to resolve this problem. Experimental results show that the proposed scheme improves 18.71% DCR over ICE and discovers the maximum number of direct connections across NATs. Accordingly, the proposed scheme reduces the processing loads of relay servers.

References

[1] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," IETF RFC 1631, May 1994.

[2] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) through network address translators (NATs)," IETF RFC 3489, Mar. 2003.

[3] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," IETF RFC 5389, Oct. 2008.

[4] J. Rosenberg, R. Mahy, and P. Matthews, "Traversal Using Relays Around NAT (TURN): relay extensions to Session Traversal Utilities for NAT (STUN)," IETF RFC 5766, Apr. 2010.

[5] J. Rosenberg, "Interactive Connectivity Establishment (ICE): a protocol for Network Address Translator (NAT) traversal for offer/answer protocols," IETF RFC 5245, Apr. 2010.

[6] H. Khlifi, J.-C. Gregoire, and J. Phillips, "VoIP and NAT/firewalls: issues, traversal techniques, and a real-world solution," *IEEE Commun. Mag.*, vol. 44, no. 7, pp. 93-99, July 2006.

[7] Y. Takeda, "Symmetric NAT traversal using STUN," IETF draft, June 2003.

[8] Y. Wang, Z. Lu, and J. Gu, "Research on symmetric NAT traversal in P2P applications," in *Proc. International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, pg. 59, 2006.