

Socially Enhanced Network Address Translation

Alexis Malozemoff and Muthucumaru Maheswaran

Advanced Networks Research Lab

School of Computer Science

McGill University

Montreal, Quebec, Canada

Email: {amaloz, maheswar}@cs.mcgill.ca

Abstract—The rapid evolution of the Internet has forced the use of Network Address Translation (NAT) to help slow the decline of publicly available IPv4 address space. While NAT provides the requisite address space as well as privacy and security to its users, it also eliminates the ability to establish incoming connections to devices within a private network. To address this issue, we propose combining social network topologies with traditional NAT architecture to better integrate peer-to-peer communication through NATed networks. This socially enhanced NAT allows incoming connections from trusted parties, resolving one of the central criticisms of the NAT approach.

I. INTRODUCTION

Since its humble beginning, the Internet has grown into a tool used by millions of people throughout the world. This rapid evolution has left some of the original design principles of the Internet [1] in the dust. Whether this is an issue or not depends on whom one talks to: the Internet community tends to decry any architectures that do not adhere to these principles, while consumers tend to choose the tool that works for them, ignorant and/or uninterested in the “inelegance” of the solution. The classic example of this situation is Network Address Translation (NAT). NAT is a functionality built into gateway routers, and was introduced as a stopgap for the lack of IPv4 address space caused by the Internet explosion in the 1990s [2]. NAT provides additional IPv4 address space by creating a network of private IP addresses. All outbound packets get their source IP address overwritten by the NATed router’s public IP address, and similarly, all incoming packets get their destination IP address overwritten by the private IP address of the destination host. NAT maintains a state table to associate given packet flows with the appropriate host within the private network. Besides providing additional IP address space, NAT also provides some security by only allowing incoming packets when an outgoing connection has already been established. The central issue researchers have with NAT is that, by giving a host a private IP address, that host is no longer addressable outside the private network. This global addressability was one of the goals of the Internet, and because NAT violated these goals, it was largely ignored by the IETF and other standards organizations for many years [3]. It was assumed that NAT would be in use temporarily to prevent the depletion of IPv4 address space while the Internet converted to the newer IPv6 [4], where the increased address space would make NAT obsolete. But the conversion

to IPv6 is taking much longer than anticipated, if it will ever fully happen at all. In addition, the switch to IPv6 actually *increases* the need for NAT, as NAT can be used as a bridge between IPv4 and IPv6 domains [5]. Finally, the perceived security and privacy benefits of NAT to unsophisticated users makes deployment of NAT likely even with full IPv6 integration [6].

During the early days of the Internet, when most connections were based on the client-server model, NAT did not greatly effect a casual user’s experiences. But with the rise of peer-to-peer communication and social networking among users, NAT gets in the way. Current NAT technology is unable to handle symmetric connections because of the packet rewriting techniques in use. Under the assumption that NATed devices are not going to disappear anytime soon, great effort has been spent on developing NAT traversal techniques. Some of the more popular are STUN-based approaches [7], TURN [8] and NATBLASTER [9], but many other techniques, both open and proprietary, exist. The large number of protocols has caused widely divergent NAT architectures, and no standardization has been established. Thus, even with a simple NAT traversal technique such as hole punching, the success rate of NAT traversal is only 84% for UDP connections, and falls to 64% for TCP [6]. In addition, much like how NAT was a stopgap for IPv4 address space depletion, NAT traversal techniques are a stopgap to the larger problem caused by NATs, that of end-to-end connectivity.

Rather than rely on NAT traversal methods that may or may not work depending on the given NAT at hand, we propose the development of a new type of NAT, called the SocialNAT, which alleviates this issue while retaining the benefits of legacy NAT devices. We argue that the main architectural flaw of NAT is eliminating the ability to establish incoming connections between trusted users, not necessarily the privatization of IP addresses as many people claim. By integrating social network topologies with traditional NAT design, users can establish connections between peers in private realms, allowing peer-to-peer communication without losing the benefits of NAT. The trust links and policy settings between two users determine whether an incoming session should be let through. With this design, the privacy and security of traditional NAT are retained while peer-to-peer applications can now work between trusted parties. Due to the built-in legacy support, SocialNATs can be installed

incrementally by end users, and traditional NATed devices can be upgraded with a software patch. The architecture is designed so that software changes made to hosts are minimal; only the installation of a light-weight application is needed for a host to take advantage of the SocialNAT architecture.

The paper proceeds as follows. Section II discusses the necessary background information and terminology used throughout the paper. Section III covers the SocialNAT architecture. Sections IV and V discuss the benefits of the SocialNAT architecture over traditional NAT and future work to be done, respectively. We close with a survey of related works in Section VI and a conclusion in Section VII.

II. BACKGROUND INFORMATION

Before describing the SocialNAT architecture, we first present a brief overview of the requisite background information and associated terminology.

A. Online Social Network Terminology

SocialNATs are based on the concept of the Online Social Network (OSN) [10]. The key element of an OSN is the relationships between users, and the trust levels associated with these links. SocialNATs use the trust already established in OSNs and apply it to the physical topology of the Internet.

B. NAT Terminology

The other key concept at the heart of the SocialNAT architecture is NAT [11]. NAT comes in many different flavors and, due in part to the lack of standardization, it is difficult to determine exactly what features a given NAT implements. We will briefly discuss NAT in general, followed by some of important variants. All terminology is taken from RFC 2663 [11] and [12].

NAT is a functionality built into routers, often at the gateway of a network. NATed devices (*i.e.*, gateway routers with NAT functionality) will be denoted as NATs (Network Address Translators) in the discussion below. NATs are placed on the gateway between a private and public *address realm*, or simply *realm*. A realm is defined as a routable network. Private address realms are also referred to as *domains*. NATs route between two realms using *transparent routing*, that is, they rewrite the IP header to enable the packet to be routed in the required realm. This modification is done without the packet sender's/receiver's knowledge. NATs maintain state information on all *sessions* between the private and public realm. A session is defined as a packet flow and its associated initial direction. Sessions are marked by the tuple (*source IP address, source port, target IP address, target port*) for TCP/UDP connections, and similarly for other protocols. In our discussion we will only consider TCP/UDP sessions, but the architecture applies to all protocols.

While many different NAT variants exist, we will focus on the most common: *traditional/outbound* NATs. With traditional NATs, the source IP address of outbound packets is rewritten with the NAT's IP address, and similarly, the

destination IP address of incoming packets is rewritten with the private host's IP address. This is possible by tracking each session established across the NAT. With traditional NATs, only outbound sessions can be established, as any incoming packet not associated with an already established outbound session is dropped.

One variant of traditional NAT common in consumer circles is the Network Address Port Translator (NAPT). Here, a single public IP address is used for all outbound sessions, which is accomplished by also translating the port number as well as the private IP address. We will assume the use of NAPTs to simplify the discussion, but the techniques discussed below should work with most, if not all, NAT variants.

III. SOCIALNAT ARCHITECTURE

The SocialNAT architecture consists of hardware boxes called SocialNATs, which replace traditional NATs as gateway devices on a private network. Hosts run software called a Connection Manager (CM) to allow their device to connect with the SocialNAT and understand the protocols used. A centralized server, called the Policy Manager (PM), hosts the policies and IP addresses of the users. Users create their policies through an application running on the OSN, and the PM communicates with the OSN to manage the policies.

Figure 1 depicts the SocialNAT architecture. Here, SocialNATs separate two private networks from the public Internet. With traditional NATs, Alice is unable to initiate a connection with Bob because Bob's NAT blocks any new incoming sessions, even if Alice and Bob are friends and Bob wants the connection to take place. If determined and knowledgeable enough, Bob can use port-forwarding to bind a public port to the service running on his device. But if his device is a smartphone or laptop, which frequently changes its private IP address, this can become a tedious operation. Additionally, by port-binding his service to a public port, this service becomes available to all Internet users, making it vulnerable to attacks. With the SocialNAT architecture, Bob can allow incoming connections to his service for certain trusted users, such as Alice, and reject connections for any other users. A probing protocol (Section III-E) is used to verify Alice's authenticity; once the trust between Alice and Bob is established the connection is allowed to proceed.

It is important to note that SocialNATs are not designed to resolve all security issues already existent in NATs. For example, denial-of-service attacks can still work on SocialNATs. The main goal of SocialNATs is the ability to allow incoming sessions to be established from trusted sources, while maintaining all the strengths of traditional NATs. SocialNATs are not an all encompassing solution to security and end-to-end connectivity on the modern Internet, but provide a first step towards integrating online social links with traditional devices to provide additional networking capabilities.

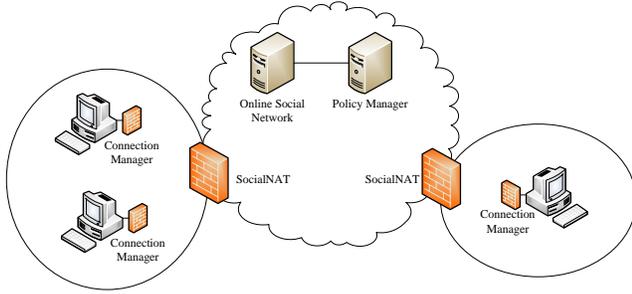


Fig. 1. The SocialNAT architecture

A. Connection Manager

The Connection Manager (CM) is software running on a user’s device that provides the protocols needed to connect with the SocialNAT framework. It functions similar to a personal firewall by intercepting and analyzing traffic coming and going from the host machine. A user sets her policies through the Policy Manager (Section III-B), which are subsequently downloaded onto the CM. The CM maintains a list of the user’s friends (who must also be connected to a SocialNAT) and those friends’ public IP addresses. The CM also has a list of all active sessions between the user and another device. Any new outgoing session must be checked against the list of public IP addresses to see if the destination IP address is behind a SocialNAT. If this is so, the probing protocol is used, and all the packets being sent to that (*destination IP, destination port*) tuple are buffered. Once the connection is established, the CM creates a session entry in its table and the packets in the buffer are sent out. If the connection fails (*e.g.*, due to a policy violation), the session entry is deleted and the CM notifies the user.

The CM is also used to handle incoming connections. When an incoming probe packet arrives, the CM checks the information against the user policy. If the policy matches, a session is established.

B. Policy Manager

The Policy Manager (PM) runs as an OSN application and server. The OSN application is used by users to set their policies and download those policies to their CM. The server maintains a list of SocialNAT-connected devices and their associated public IP addresses. These are updated by the SocialNAT whenever a change occurs within its private network. Either a SocialNAT or Connection Manager may request an update for a given user’s IP address. To allow for privacy, this request is only accepted if the requestor is already a friend of that given user. The SocialNAT-to-PM protocol is briefly discussed in Section III-F. The PM is assumed to be a trusted server or distributed system managed by a trusted source.

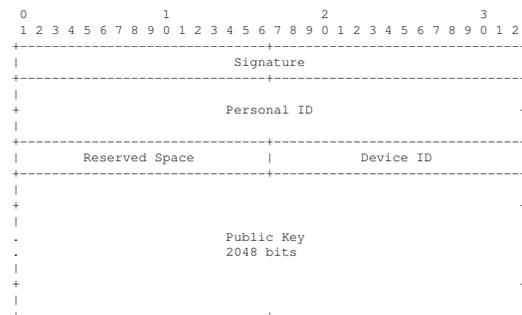
C. Social ID

The Social ID (SID) provides a certification of a user’s identity. The SIDs are created during the bootstrapping process (Section III-G), and are used throughout the probing

process (Section III-E) to validate the incoming user and compute the appropriate policy before a session is established.

Figure 2 shows the Social ID’s packet diagram. The SID is signed by the OSN and this 32-bit signature is stored in the signature field, allowing users to verify the integrity of the ID. Each SID contains 64 bits of space for the specific user’s personal ID number on the OSN, which is used by the OSN to uniquely identify each user. The current design only facilitates the use of a single OSN as an entry point into the SocialNAT architecture. The 16-bit device ID uniquely identifies a user’s various devices. For example, Alice can create an SID for her laptop and smartphone. Each SID would be differentiated by the device ID set by the OSN. 16 bits of reserved space are provided for future extensions. Finally, a 2048-bit public key is appended to the SID and used in the probing protocol. The private key is stored locally on a user’s device. Overall, the entire Social ID is 272 bytes long.

After bootstrapping to a specific SocialNAT, the user’s SID and publicly accessible address is uploaded to the PM. Whenever changes occur on the SocialNAT (*e.g.*, a user leaves the private network), this information is updated. Thus, the PM maintains an up-to-date list of all users currently connected to a SocialNAT and their publicly accessible IP address. While some may raise privacy concerns over the collection of a user’s location by the PM, we argue that this is no different, and in fact more secure in many cases, than the techniques employed by NAT traversal, where users rely on the rendezvous server to not disclose their public IP address except to other peers on the network. These peers are not necessarily trusted and may have malicious intent. In fact, the server itself is not trusted either. Because there is little uniformity among NAT traversal techniques, a different and untrusted server potentially exists for each peer-to-peer application using NAT traversal. With SocialNATs, the IP address is stored by a single trusted server running within the trusted framework of an OSN. The user’s IP address is already known to the OSN, and the PM does not publicly disclose this information except to trusted parties. Thus, privacy shouldn’t be a concern for users of SocialNATs.



D. The SocialNAT Device

As has been mentioned earlier, the SocialNAT functions as an enhanced NAT. All the functionality built into traditional NATs is also available in SocialNATs. That means individual vendors can easily deploy SocialNATs by a simple firmware upgrade to their device.

When a user installs a SocialNAT, the SocialNAT connects with the OSN and downloads an SID for the given device. Since the SocialNAT is not a user on the OSN, no personal ID exists to distinguish SocialNATs; thus a separate unique ID must exist within the OSN application to handle SocialNATs. Some OSNs, such as Facebook, provide groups; we propose having a group for each SocialNAT, wherein the policies can be set. Users can configure domain policies for their SocialNAT through the PM and download the information locally to be used in session establishment. The SocialNAT maintains a list of SIDs and IP addresses for the associated domains in the policy list. Incoming sessions are checked against this list to determine whether the incoming session matches a valid policy.

The main benefit of domain policy lists is to allow incoming connections from certain domains that contain useful services for the user. For example, a software-based telephone setup behind a SocialNAT would likely only accept incoming calls from that user's friends. But the user may wish to accept calls from essential services as well, such as the fire department. By whitelisting the fire department's SocialNAT, these calls would go through without each user needing to specifically add the fire department to her personal policy. These domain policies go a long way in enhancing the Internet's ability to function similar to a public switched telephone network, something current NAT technology prevents [13].

The other feature-enhancement SocialNATs provide is the ability to define valid users within a domain. SocialNATs can be configured to only allow certain users access to the private network by uploading their SID to the SocialNAT. With traditional private realms using DHCP [14], any user can access the private network by locating an ethernet cable that connects to the network. Most home networks nowadays use wireless, providing intruders more avenues of entry. Even with wireless encryption, cracking WEP/WPA is fairly easy using one of many freely available tools. Thus, the security of a domain and the users within that domain can rarely be verified, which has been exploited by many hackers to gain access to private networks. SocialNATs provide an additional layer of protection to private networks by only allowing pre-authorized users access to the network, even without any wireless encryption. This is further described in Sections III-G and III-H.

E. Probing Protocol

When a CM detects that a new session is destined for a device behind a SocialNAT, it uses the probing protocol to establish a connection. The probing protocol is built on top of ICMP Echo Request/Replies to facilitate their use in current Internet environments.

Figure 3 demonstrates the probe protocol packet flow in a common networking scenario between Alice (A) and Bob (B). The probe, initially containing Alice's SID, Bob's SID and a nonce (a one-time random number) encrypted by Bob's public key, is sent out from Alice's CM. Alice's SocialNAT (S_1) captures this packet and appends its own SID onto the probe before sending it onwards. When the packet reaches Bob's SocialNAT (S_2) and passes the domain policy check, S_2 challenges the probe by encrypting a nonce with S_1 's public key and sending it back to S_1 . At the same time, S_2 forwards the packet to Bob, who can be determined from the SID provided by Alice. Bob also checks against his own policy, and, if this challenge succeeds, replies with a nonce encrypted by Alice's public key as well as Alice's nonce decrypted by Bob's private key. Until both of these nonces are returned and verified, no traffic between Alice and Bob is allowed through S_2 . If at any point verification fails, a failure message is sent.

Alice uses a nonce to verify that Bob's IP address is in fact Bob. This prevents a user posing as Bob from establishing a session with Alice. No check is done at the SocialNAT level by S_1 , as domain policies only apply to incoming sessions, not outgoing sessions. If everything succeeds, S_2 allows the incoming session to be established from Alice. Once the session is established, everything functions as a normal NAT.

Bob's policy checks against Alice's personal ID, and does not consider the device ID. This allows Alice to use any of her registered devices to access Bob. Thus, if Alice gets a new device, all she must do is register with the OSN and that device is fully integrated into the SocialNAT architecture.

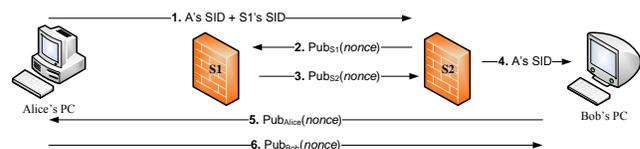


Fig. 3. Probe protocol packet flow

F. Policy Manager Protocol

The Policy Manager Protocol (PMP) is used to communicate information between the SocialNATs or CMs and the PM. PMP is built on top of UDP and uses the public keys provided within the SIDs as well as the known public key of the PM to secure and validate communication. Sessions are always initiated by a SocialNAT or CM and are used to locate users/SocialNATs or update location information.

The PMP packet contains a 16-bit magic number and 16-bit type field, followed by the payload, which varies depending on the type field. There are five main session types (*add*, *remove*, *request user locations*, *request domain locations*, *heartbeat*), and 8 bits for possible subtypes, which are used to minimize the number and size of updates. The *add* message uploads a user's SID and publicly accessible IP address to the PM. *Remove* deletes a user's information from the PM, for example, when a user leaves a SocialNAT-protected network. *Request user locations* retrieves the IP

addresses of a given user's friends, *Request domain locations* similarly retrieves the locations of the SocialNATs on a given SocialNAT's domain policy list. Finally, the *heartbeat* message notifies the PM that a user is still active. Due to lack of space, the details are not discussed here.

G. Bootstrapping Process

SocialNATs provide the same basic functionality as traditional NATs; thus the installation of the hardware is similar. In addition to installing the SocialNAT, users must install the Connection Manager on their individual machines. During the installation phase, the Connection Manager requests to access the OSN compatible with the SocialNAT framework to retrieve the user's SID. The user is required to login to the OSN and run an application on that OSN to download her SID. Any social network that permits the use of applications (e.g., Facebook) should be capable of providing this service. Upon receiving the SID, a user has bootstrapped with the system, and is capable of taking advantage of the SocialNAT framework.

H. DHCP-Extended Protocol

In order for hosts to register with a given network, we use DHCP [14] with the 'options' field to provide the requisite information. By using DHCP, only a minimal number of changes will be needed to register a host with a SocialNAT, and no new protocols need be developed.

The CM running on the client's side intercepts normal DHCP packets sent out by the client and modifies them to include the information required by the SocialNAT. If the CM is not installed on a client's device, it will proceed to use traditional DHCP to obtain an IP address. To maintain the security of the network as well as permit new users to install the software and retrieve a SID, a VLAN can be setup for all users without a CM installed to isolate them from the rest of the network. All outgoing traffic will be forwarded to the PM, where the CM can be installed and the SID can be retrieved.

IV. APPLICATIONS AND BENEFITS OF SOCIALNATS

SocialNATs provide many benefits to traditional NATs. Most importantly, the ability to establish incoming connections between trusted parties is restored. Connections between trusted parties can be established without any intermediary (and not necessarily trusted) servers relaying traffic. The one situation where SocialNATs are less successful is when a user attempts to communicate with a non-trusted host in a peer-to-peer manner. This can arise in the use of an application such as BitTorrent, where a user contacts unknown peers to download a file.

In addition to the obvious benefit of incoming connection establishment, SocialNATs also provide more security to wireless network setups. By only allowing certain users the ability to connect to the network, wireless hijackers are unable to access even unencrypted networks. While more advanced methods exist to secure wireless access points, SocialNATs provide security to the consumer, who is often

not knowledgeable enough to take advantage of these more advanced security measures.

For services running on a private network, such as an SSH server or a printer, only valid users have access to those services. Take an SSH server for instance. With traditional NAT, a user would need to portbind port 22 to the SSH server's private IP address. Any user would have access to this port, even a malicious user. In addition, if that private IP address ever changed (e.g., because of a server reboot), the user would have to login to the NAT box and reconfigure the portbinding. With SocialNATs, this portbinding would be automatic, and only trusted users would have access to the server. While such mechanisms as access control lists and other security measures do exist in enterprise settings, SocialNATs bring that security to the consumer level.

V. FUTURE WORK AND EXISTING ISSUES

Before developing a proof-of-concept implementation, a theoretical analysis is needed to determine the overhead of the architecture, providing an opportunity to tighten up the protocols and reduce the complexity of the architecture. We must also more carefully consider the integration of the SocialNAT and the OSN.

In terms of outstanding issues that must be resolved, one topic we haven't mentioned is the situation of multi-layered NATs, where the ISP, in addition to the home user, uses NAT. This is an important consideration that must be tackled before SocialNAT can be used throughout the Internet.

VI. RELATED WORK

SocialNATs function as a replacement for NAT traversal methods, of which there are many variants. The problem with NAT traversal is that there is no standardization between NAT devices; so a given technique may fail on one NAT device but work on another. Also, NAT traversal is application dependent, making the deployment of a peer-to-peer application more costly and time consuming. SocialNATs provide an integrated method that any application can use without changes to the application source code.

There has been some research focused on extending NATs to better interoperate with Internet design principles, most notably IPNL [3]. IPNL is an architecture that attempts to solve IPv4 address depletion. Thus, IPNL is in a sense an alternative approach to the issues tackled by IPv6. The benefits of IPNL is that only host machines and NAT boxes must be modified. Routing and identification are based on Fully Qualified Domain Names (FQDN) as well as IPNL addresses, which contain both a globally addressable IP address as well as the private IP address of a host. By introducing two identifiers (and a third random ID to prevent spoofing), the authors argue that this approach solves the location/identity problem found in IPv4 addresses, *i.e.*, where the location of a host also acts as the identity for that host. The NAT extensions proposed for the IPNL architecture do not address the connection establishment problem inherent in NAT. Thus, while solving IPv4's address depletion problem, IPNL does not address the end-to-end connectability of

hosts. As these are two of the current issues with IPv4, combining SocialNATs with IPNL would potentially solve both these problems without the overhead of implementing IPv6 on all network devices.

Middleboxes, such as NATs and firewalls, are often derided for disobeying the Internet's original design principles. [15] argues that the two main principles of the Internet are the ability to establish an end-to-end connection with any host through globally unique identifiers, and that intermediate network elements should not process packets. NATs violate both principles by disallowing incoming session establishment and privatizing internal IP addresses. The authors propose a delegation-oriented architecture to integrate middleboxes with these principles, which only requires host and intermediary software to be modified. [16] tackles a similar issue, that of the problem with end-to-end communication in the modern Internet. The authors propose an architecture that combines both on-path and off-path communication to create an end-middle-end architecture which still holds true to the Internet's original design goals. Both [15] and [16] are complex protocols, and [16] especially requires additional system administration to maintain, making it much less consumer friendly. The authors go to great length to try and satisfy the original Internet principles, and this results in intricate and heavy architectures, making it unlikely they will ever be deployed in a wide setting. SocialNATs propose a much simpler solution by accepting the original Internet principles as no longer applicable; the Internet has evolved, and so should the protocols and architectures running on it. We accept the limitations of end-to-end connectivity introduced by NATs, and work with this constraint rather than against it.

[17] introduces Social VPNs, which are VPNs built around trusted relationships on OSNs. Social VPN is a network overlay built on top of IP over P2P (IPOP) [18], an architecture for tying together virtual networks over a P2P overlay. Social VPNs and grid computing environments [19] both integrate OSNs with IPOP to create a collaborative networking environment between trusted parties. This differs from our design in that the underlying structure of these architectures, IPOP, uses NAT traversal techniques. SocialNATs provide the low-level architecture which avoids the use of these unreliable schemes.

VII. CONCLUSION

Discussion of the related work in Section VI gives a glimpse at the large number of architectures developed to tackle the issues generated by both the original Internet design and the rapid, and seemingly uncontrollable, evolution of the Internet. Rather than take a drastic approach to solving the Internet's problems, SocialNAT is a small addition to the current Internet architecture which facilitates the ability to establish sessions between trusted parties within separate private realms. SocialNAT also has advantages in protecting private networks, as a user's identity acts as a form of authentication to access a domain. While the SocialNAT architecture is not an end-all solution to the many disparate

problems with the modern Internet, it solves one problem that effects countless applications. By using SocialNATs, peer-to-peer applications should be easier to deploy and can be more trusted. Additionally, SocialNATs provide the low-level groundwork on which more complex architectures can be developed, without having to worry about getting around establishing incoming connections through NATs.

REFERENCES

- [1] D. Clark, "The design philosophy of the DARPA Internet protocols," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, pp. 106–114, 1988.
- [2] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," RFC 1631, May 2006.
- [3] P. Francis and R. Gummadi, "IPNL: A NAT-extended internet architecture," in *ACM SIGCOMM*, San Diego, CA, August 2001.
- [4] S. Deering, R. Hinden *et al.*, "Internet protocol, version 6 (IPv6) specification," RFC 2460, December 1998.
- [5] G. Tsirtsis and P. Srisuresh, "Network Address Translation-Protocol Translation (NAT-PT)," RFC 2766, February 2000.
- [6] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators," in *USENIX '05*, April 2005.
- [7] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389, October 2008.
- [8] J. Rosenberg, R. Mahy, and C. Huitema, "Traversal using relay NAT (TURN)," *Internet-Draft (work in progress)*, 2004.
- [9] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig, "NATBLASTER: Establishing TCP connections between hosts behind NATs," in *ACM SIGCOMM Asia Workshop*, 2005.
- [10] A. Misllove, M. Marcon, K. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM New York, NY, USA, 2007, pp. 29–42.
- [11] P. Srisuresh and M. Holdrege, "IP network address translator (NAT) terminology and considerations," RFC 2663, August 1999.
- [12] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer (P2P) communication across middleboxes," *Work in Progress (draft-fordmidcom-p2p-03)*, 2004.
- [13] "Solving the Firewall and NAT Traversal Issues for MoIP," Newport Networks Ltd, white paper, 2008.
- [14] R. Droms *et al.*, "Dynamic host configuration protocol," RFC 2131, March 1997.
- [15] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker, "Middleboxes no longer considered harmful," in *Proc. OSDI 2004*, San Francisco, CA, December 2004.
- [16] S. Guha and P. Francis, "An end-middle-end approach to connection establishment," in *Proceedings of the 2007 conference on applications, technologies, architectures, and protocols for computer communications*. ACM New York, NY, USA, 2007, pp. 193–204.
- [17] R. Figueiredo, P. Boykin, P. Juste, and D. Wolinsky, "Social VPNs: Integrating overlay and social networks for seamless P2P networking," in *Workshop on Collaborative Peer-to-Peer Systems*, Rome, Italy, 2008.
- [18] A. Ganguly, A. Agrawal, P. Boykin, and R. Figueiredo, "IP over P2P: enabling self-configuring virtual IP networks for grid computing," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium*, 2006.
- [19] P. Juste, D. Wolinsky, J. Xu, M. Covington, and R. Figueiredo, "On the Use of Social Networking Groups for Automatic Configuration of Virtual Grid Environments," in *Grid Computing Environments Workshop*, November 2008, pp. 1–10.