

IP Tunneling in Peer-to-Peer SSL Connections

Reda Aouad Aline Chalhoub Tala Debs
Department of Electrical and Computer Engineering
Faculty of Engineering and Architecture
American University of Beirut
Beirut, Lebanon
{rma66, agc03, tmd07}@aub.edu.lb

Abstract

Peer-to-peer networks on the Internet provide multiple services for the connected peers, such as file sharing and real-time data transmission. The popularity of such networks is increasing worldwide due to the distributed, decentralized architecture which is scalable and relies on the peers' resources. This paper presents a method to create a secure point-to-point channel between two Internet hosts when both are part of an existing peer-to-peer network, but one of them is behind a restrictive firewall or proxy. The solution will permit the latter to seamlessly bypass its firewall limitations through the established channel.

1. Introduction

Internet users behind firewalls may be subject to restrictions that prevent them from gaining normal access to the Internet, beyond web browsing. This paper presents a technique to create a secure, point-to-point tunnel between two hosts over a peer-to-peer network that allows users to resume normal activity on the Internet as though they were directly connected. One of the connected hosts provides a seemingly transparent and secure Internet connection to the other, using NAT (Network Address Translation). Firewalls or proxies prohibit hosts by blocking ports through which TCP/UDP traffic pass, or by restricting access to certain web pages. The solution presented in this paper takes advantage of the fact that firewalls or proxies have at least one open port to provide secure web access for users on the inside network. Through this open port, a secure tunnel is established between the two end hosts. Since the hosts are part of a peer-to-peer network, the

firewall or proxy cannot block access to the hosts providing Internet access, since their addresses are not known a priori, and may change frequently. Security of the data originating at the host behind the firewall/proxy is ensured by keeping it private from the firewall/proxy, since the channel is established using an SSL (Secure Sockets Layer) tunnel similar to a VPN (Virtual Private Network). A prototype implementation of the system uses Skype as an already-established peer-to-peer network, since it provides several useful features such as security and firewall/proxy traversal.

2. Peer-to-Peer Networks

A peer-to-peer network connects nodes on the Internet without the need of a conventional centralized server to establish the connections. Peers on the network share their bandwidth among each other rather than connecting through a server that has a limited capacity. Although a native peer-to-peer network contains no centralized server, almost all peer-to-peer applications use few servers – called trackers – to keep track of the connected nodes and manage the connections between those behind NAT devices; ordinary nodes can also function as servers [1].

Typically, nodes in a peer-to-peer network are connected in an ad-hoc manner. Peers are responsible for hosting the shared resources. The distributed nature of the network implies that nodes provide some of their available resources in bandwidth, storage space and computing power[2]. Thus, the network is more scalable, and as more nodes connect to the network, more resources are accessible, which is a major drawback in the client-server architecture. Another main advantage of a peer-to-peer network is its resilience to node or link failures, a clear weakness in client-server topology, where a single point of failure stops the

provided services. Some peer-to-peer protocols also encrypt the connections between the hosts for increased privacy [3].

3. Skype Overview

Skype is the most popular peer-to-peer, Voice-over-IP (VoIP) software which was primarily designed to make PC-to-PC, PC-to-phone and phone-to-PC calls over the internet. Skype is also similar to other instant messaging (IM) applications in the way it provides the same services, but the underlying protocol is quite different; it runs on a peer-to-peer network which requires a very minimal administrative infrastructure. It provides three main services: Voice over IP, Instant Messaging, and file-transfer [4]. Despite its popularity, the Skype protocol remains proprietary and unrevealed to the world.

3.1. Network Overlay

The Skype peer-to-peer network defines two types of nodes (see Figure 1): Skype Client (SC) and Super-Node (SN). A SC is any host running the Skype application and using any of its services. Any node (host) having a public routable IP address and sufficient computational power and internet bandwidth can be promoted to a SN (a SC cannot prevent itself from becoming a SN). The network also contains a major entity, the Skype login servers, on which users' credentials are stored. The SN has the role of relaying the traffic between SCs and between a SC and the Skype login servers [5].

The login servers are the only servers found in the network; all communication and information transfer is done in a decentralized manner through SNs. This network infrastructure proves to be very scalable and adaptable to continuous increase in the number of Skype users. Therefore, the centralized part of the network remains very small and easy to manage and maintain.

Each SC in the network maintains a list of SN IP addresses and ports called the Host Cache (HC), which is maintained and refreshed regularly. This list helps the SC to connect to SNs after logging in through either UDP or TCP packets.

3.2. NAT/Firewall Traversal Feature

An important feature of the Skype application is its ability to bypass almost every NAT and/or firewall device; the protocol uses several

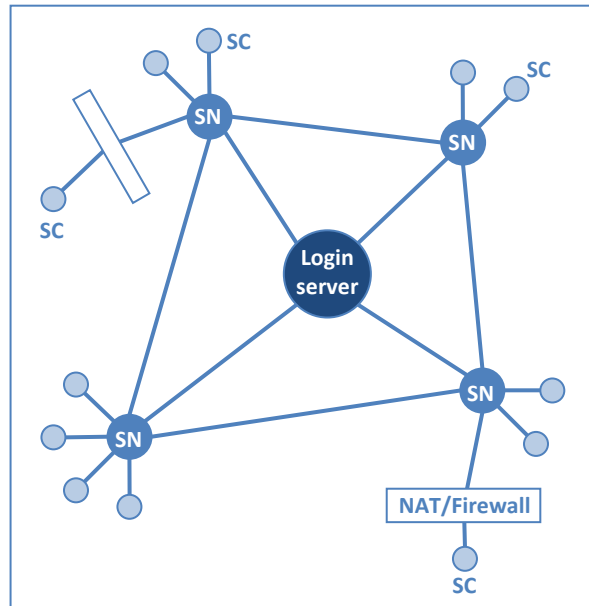


Figure 1: Skype Network Overlay [5]

techniques to guess the type of device the application is behind and try to bypass it [5].

Even if the two end users communicating are behind such devices, Skype protocol still works by routing traffic through any node that is not behind any of these devices. This is achieved by taking advantage of the fact that any firewall should allow at least one port open for outward connections, which often is a commonly known port, such as 80 or 443.

3.3. Security & Privacy

According to Garfinkel [6] and the latest Skype Security Evaluation Executive Summary [7], all traffic through the Skype network is end-to-end encrypted using several standard encryption techniques. This ensures the maximum security possible, eliminating the possibility of a man-in-the-middle attack, and preserves the user privacy, as well as it keeps the protocol hidden from the public and prevents anyone from guessing the type of information carried in the network (chat messages, audio streams, files, etc).

3.4. Application Programming Interface (API)

The Skype API allows passing commands between Skype and user-created applications in clear text. The user applications can be applications that extend the Skype functionality or add to it. The

API permits external applications to access and control several Skype functions. For privacy concerns, the applications should be granted user's permission to have the desired access rights. All actions performed using the Skype Access API are mirrored on the Skype application [8].

4. System Approach

The main concerns that are addressed in this work are the presence of the firewall that restricts the user's activity as well as his security and privacy. Several alternatives are possible, and summarized in the following section.

4.1. Alternatives

A simple approach is to use a client-server setup; the client will establish a connection to the server through a port known to be open on the firewall. The server will act as a proxy for the client host by forwarding its requests to the Internet on its behalf, then serving back the replies. The main advantage of this solution is the ease of implementation. However, such architecture is prone to a single point of failure and can be easily blocked by simple firewall rules restricting connections to the well-known server.

A better but more challenging solution is to establish connections between host pairs, one being behind a firewall (referred to as the client) and the other serving its requests (referred to as the server), over a peer-to-peer network. Virtual network interfaces are installed on the machines of both hosts in order to create a virtual, secure tunnel that emulates a VPN. On the client side, changes to the routing table should be made: all traffic, except that related to the peer-to-peer application, should be routed through the virtual interface. This allows all traffic generated by applications on the client (other than the peer-to-peer application) to be sent over the secure tunnel towards the server, thus bypassing the firewall. At the server, NAT is enabled; the public NAT interface is the physical interface through which the server is connected to the Internet, while the private interface is the virtual one. Thus, packets arriving to the server's virtual interface will be sent over the Internet through the physical one. Forwarded packets will find their way back to the client through NAT on the server.

4.2. Solution

The peer-to-peer solution described above was implemented using Skype as the peer-to-peer network. First, Skype offers a totally secure tunnel, which encrypts all the data transmitted through its network and saves the effort of installing and configuring tunnel software. Second, Skype has an advantage over other peer-to-peer protocols in bypassing firewalls and NAT devices. Third, the Skype API is relatively easy to use in several programming languages. [deleted sentence]

Once the Skype users (at the two ends of the tunnel) are both logged in to Skype, the application negotiates the IP addresses to be assigned to the virtual interfaces, without conflicting with any IP addresses on the client or the server, similar to what a VPN software would do. The client's routing table is modified to send all traffic not related to Skype through the virtual interface. When this is accomplished, the application has the role of capturing all the packets appearing on the client's virtual interface and encoding them into an ASCII stream for the purpose of sending them as strings through the Skype network using the Access API.

At the other end, the application receives the data from the client and reconstitutes the IP packets by decoding them back from the ASCII stream. NAT then performs the function described in the peer-to-peer approach as detailed in Section 4.1.

On the way back, the application reverses the process: it encodes the IP packets coming as replies from the Internet into an ASCII stream, and sends them over Skype to the client. The latter converts them into IP packets and injects them into the operating system TCP/IP stack.

5. System Architecture

As shown in Figure 2, the client should have a physical connection to the Internet, which is subject to firewall constraints. This physical interface is the route for all Skype traffic, since Skype has the capability to bypass almost all firewalls. The virtual interface at the client machine is the default gateway for all its generated traffic except that related to Skype. The application on the client side has the role of encoding the IP packets coming to its virtual interface into ASCII strings and sending them over the tunnel created by Skype to the other end (the server), where the application decodes the string back into IP packets.

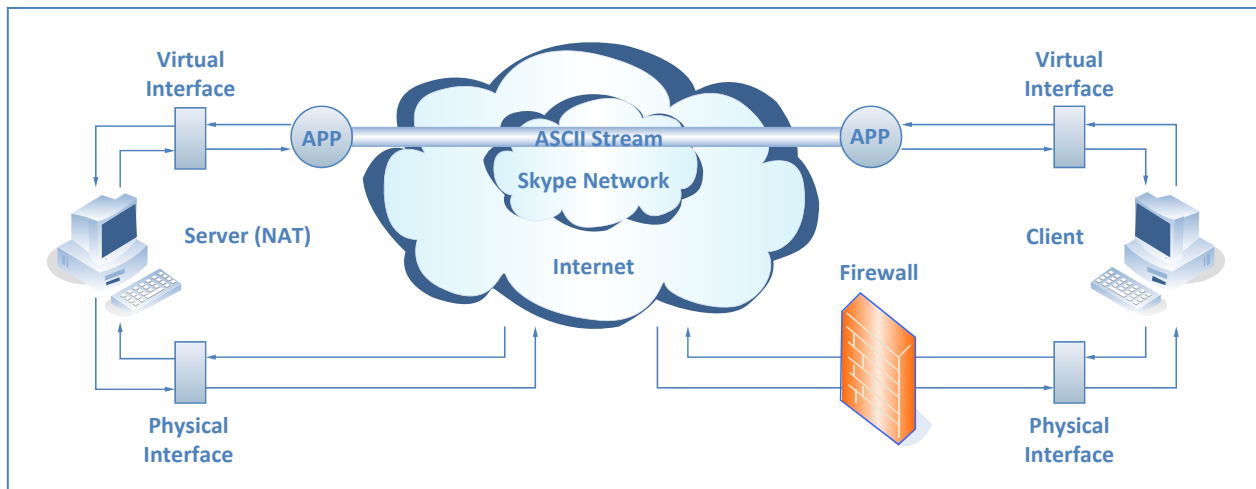


Figure 2: System Architecture

As described earlier, the virtual interface on the server is the private network of the NAT installed on the server, and the physical interface is the public one.

6. Client Packet Processing

The process of routing Skype packets through the physical interface and all remaining IP packets through the virtual interface is not a straightforward task to accomplish under Windows XP, since XP doesn't natively support it. The first approach relies on exploiting the client's Host Cache in order to add static entries to the client's routing table specifying that all traffic destined to the Skype SN be routed through the physical interface, while having the virtual interface as the default gateway for the remaining traffic. The problem with this solution is that in the recent versions of Skype, the HC is encrypted.

The second approach relates to the established TCP/UDP connections in Windows: the operating system can provide the list of connections that Skype established. Accordingly, the traffic destined to the IP addresses at the endpoints of these connections could be routed through the physical interface. The drawback of this solution is that it will prevent Skype from changing its connections to other nodes since new connections established by Skype would be routed through the virtual interface which is set as the default gateway.

Routing based on process ID has also been attempted; all packets tagged with the process ID of Skype could be routed through the physical

interface, leaving the remaining traffic to pass through the virtual one. This approach, however, has been unsuccessful under Windows XP.

The solution is to implement NAT on the client side. The process is described in Figure 3: the virtual interface is the default gateway for all traffic including Skype packets. After capturing all packets sent through this interface, only Skype packets are sent to the physical interface after being address-translated through NAT. This will ensure that Skype packets are forwarded to the Internet and that replies are received back. The remaining packets are encoded in Skype ASCII strings as described in Section 4.2.

In order to accomplish the task of translating only Skype packets, a simple, basic NAT code was written, since Windows XP NAT doesn't have the ability to translate selective packets among the generated IP traffic.

7. Testing and Performance

The test environment is composed of two laptop PCs running Windows XP Professional, and connected to the Internet. On each, a Skype user is logged in and the applications are running under an administrator account.

The first step in setting up the client machine, which involves changing the routing table, takes no more than two seconds to accomplish. The next step is to establish the Skype tunnel, which may take up to 30 seconds, but does not normally exceed a few seconds, depending on the state of the Skype network – something which cannot be controlled. After creating the channel, packets

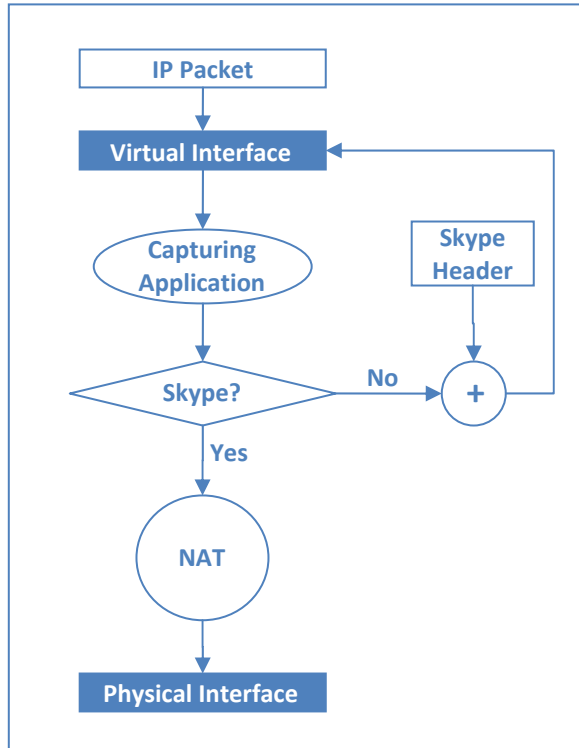


Figure 3: Client Packet Processing

tunnel to the server host through the NAT application on the client side.

The NAT application remains a bottleneck in the system, since it involves substantial processing on each IP packet to determine the process that sent it and then translate it and route it properly. On average, and after conducting several Internet bandwidth speed tests, results show that the bandwidth of the client is reduced by approximately 24% (from 156 Kbps to 119 Kbps).

8. Future Work

The first system prototype is implemented using Skype as the underlying peer-to-peer application. One adjustment to the system can be to increase the speed at which it translates, encodes and streams packets. Another major task for the future is to replace Skype by an open-source peer-to-peer system, thus removing the dependency of users on Skype and allowing greater flexibility with open-source code. This

implies re-implementing the special features Skype provides, such as the security and privacy of peers, as well as finding a new way of establishing a secure tunnel between them.

Acknowledgments

This research was supervised by Dr. Ayman Kayssi.

References

- [1] Mitchell, Bradley. Overview of P2P Applications and Networks. About.com. [Online] 2008. [Cited: November 7, 2008] http://compnetworking.about.com/od/p2ppeertopeer/a/p2pintroduction_2.htm
- [2] CompuMentor. Peer-to-Peer Networks. TechSoup. [Online] April 22, 2003. [Cited: November 7, 2007] <http://www.techsoup.org/learningcenter/networks/page4772.cfm>
- [3] Wikipedia. Peer-to-Peer. Wikipedia, the Free Encyclopedia. [Online] January 7, 2008. [Cited: November 7, 2007] <http://en.wikipedia.org/wiki/Peer-to-peer>
- [4] VoIP System. Saikat Guha, Cornell. [Online] 2006. [Cited: November 5, 2007] <http://saikat.guha.cc/pub/iptps06-skype/>
- [5] Baset, Salman and Schulzrinne, Henning. An Analysis of the Skype Peer-to-Peer Internet Telephony Network. September 15, 2004.
- [6] Garfinkel. VoIP and Skype Security. January 26, 2005.
- [7] Skype. Skype Security Evaluation. Skype. [Online] 2006. [Cited: November 5, 2007] <http://share.skype.com/images/stories/images/blog/products/2005-031%20security%20evaluation%20execsum.pdf>
- [8] Skype. API Reference for Skype. Skype. [Online] 2006. [Cited: November 5, 2007] https://share.skype.com/sites/devzone/2006/01/api_reference_for_skype_20_bet.html