Skype & Network Management

Taken from class reference:

An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol
Salman A. Baset and Henning Schulzrinne
(see class web site)

Skype & Network Management

Like its file sharing predecessor KaZaa, Skype is an overlay peer-to-peer network.

There are two types of nodes in this overlay network, ordinary hosts and super nodes (SN).

An ordinary host is a Skype application that can be used to place voice calls and send text messages.

A super node is an ordinary host’s end-point on the Skype network.

Any node with a public IP address having sufficient CPU, memory, and network bandwidth is a candidate to become a super node.

Skype & Network Management

Although not a Skype node itself, the Skype login server is an important entity in the Skype network.

Usernames and passwords are stored at the login server. User authentication at login is also done at this server.

This server also ensures that Skype login names are unique across the Skype name space.

NAT and firewall traversal are important Skype functions. We believe that each Skype node uses a variant of STUN [1] protocol to determine the type of NAT and firewall it is behind.

We also believe that there is no global NAT and firewall traversal server.

Skype & Network Management

The Skype network is an overlay network and thus each Skype client (SC) should build and refresh a table of reachable nodes.

In Skype, this table is called host cache (HC) and it contains IP address and port number of super nodes. It is stored in the Windows registry for each Skype node.

Skype uses wideband codecs which allows it to maintain reasonable call quality at an available bandwidth of 32 kb/s.

It uses TCP for signaling, and both UDP and TCP for transporting media traffic. Signaling and media traffic are not sent on the same ports.

Skype & Network Management

Skype stores its buddy information in the Windows registry.

The Buddy list is digitally signed and encrypted.

The buddy list is local to one machine and is not stored on a central server.

If a user uses SC on a different machine to log onto the Skype network, that user has to reconstruct the buddy list.
A Skype client (SC) opens a TCP and a UDP listening port at the port number configured in its connection dialog box. SC randomly chooses the port number upon installation. SC also opens TCP listening ports at port number 80 (HTTP port), and port number 443 (HTTPS port). Unlike many Internet protocols, like SIP and HTTP, there is no default TCP or UDP listening port.

Skype uses AES (Advanced Encryption Standard) which is also used by U.S. Government organizations to protect sensitive information. Skype uses 256-bit encryption, in order to actively encrypt the data in each Skype call or instant message. Skype uses 1536 to 2048 bit RSA to negotiate symmetric AES keys. User public keys are certified by Skype server at login.

We conjecture that SC uses a variation of the STUN and TURN protocols to determine the type of NAT and firewall it is behind. We also conjecture that SC refreshes this information periodically. This information is also stored in the Windows registry. Unlike its file sharing counter part KaZaa, a SC cannot prevent itself from becoming a super node.

STUN (Simple Traversal of UDP over NATs) is a network protocol allowing clients behind NAT (or multiple NATs) to find out its public address, the type of NAT it is behind and the internet side port associated by the NAT with a particular local port. This information is used to set up UDP communication between two hosts that are both behind NAT routers. The protocol is defined in RFC 3489.

The host cache (HC) is a list of super node IP address and port pairs that SC builds and refreshes regularly. It is the most critical part to the Skype operation. At least one valid entry must be present in the HC. A valid entry is an IP address and port number of an online Skype node.

Skype functions can be classified into startup, login, user search, call establishment and tear down, media transfer, and presence messages. Login is perhaps the most critical function to the Skype operation. A SC must establish a TCP connection with a SN in order to connect to the Skype network. If it cannot connect to a super node, it will report a login failure.
Most firewalls are configured to allow outgoing TCP traffic to port 80 (HTTP port) and port 443 (HTTPS port).

A SC behind a firewall, which blocks UDP traffic and permits selective TCP traffic, takes advantage of this fact.

At login, it establishes a TCP connection with another Skype node with a public IP address and port 80 or port 443.

During the experiments we observed that SC always exchanged data over TCP with a node whose IP address was 80.160.91.11.

We believe that this node is the login server.

A reverse lookup of this IP address retrieved NS records whose values are ns14.inet.tele.dk and ns15.inet.tele.dk.

It thus appears from the reverse lookup that the login server is hosted by an ISP based in Denmark.

After logging in for the first time after installation, HC was initialized with seven IP address and port pairs. We observed that upon first login, HC was always initialized with these seven IP address and port pairs.

It was with one of these IP address and port entries a SC established a TCP connection when a user used that SC to log onto the Skype network for the first time after installation.

We call these IP address and port pairs bootstrap super nodes.

After installation and first time startup, we observed that the HC was empty. However upon first login, the SC sent UDP packets to at least four nodes in the bootstrap node list.

Thus, either bootstrap IP address and port information is hard coded in the SC, or it is encrypted and not directly visible in the Skype Windows registry, or this is a one-time process to contact bootstrap nodes.

The TCP connection with the SN persisted as long as SN was alive. When the SN became unavailable, SC establishes a TCP connection with another SN.

A SC behind a port-restricted NAT and a UDP-restricted firewall was unable to receive any UDP packets from machines outside the firewall.

It therefore could send and receive only TCP traffic. It had a TCP connection with a SN and the login server, and it exchanged information with them over TCP.

On average, it exchanged 8.5 kilobytes of data with SN, login server, and other Skype nodes.
Skype is a p2p client and p2p networks are very dynamic.
SC, therefore, must keep track of online nodes in the Skype network so that it can connect to one of them if its SN becomes unavailable. SC sends UDP packets to 22 distinct nodes at the end of login process and possibly receives a response from them if it is not behind a UDP-restricted firewall.

We conjecture that SC uses those messages to advertise its arrival on the network.

We also conjecture that upon receiving a response from them, SC builds a table of online nodes. We call this table alternate node table. It is with these nodes a SC can connect to, if its SN becomes unavailable.
The subsequent exchange of information with some of these nodes during call establishment confirms that such a table is maintained. SC sends ICMP messages to some nodes in the Skype network. The reason for sending these messages is not clear. (any guesses?)

If both Skype clients are on public IP address, then media traffic flowed directly between them over UDP. The media traffic flowed to and from the UDP port configured in the options dialog box.
The size of voice packet was 67 bytes, which is the size of UDP payload.
For two users connected to Internet over 100 Mb/s Ethernet with almost no congestion in the network, roughly 140 voice packets were exchanged both ways in one second. Thus, the total uplink and downlink bandwidth used for voice traffic is 5 kilobytes/s.

If either caller or callee or both were behind port-restricted NAT, they sent voice traffic to another online Skype node over UDP.
That node acted as a media proxy and forwarded the voice traffic from caller to callee and vice versa.

No silence suppression is supported in Skype.
We observed that when neither caller nor callee was speaking, voice packets still flowed between them.
Transmitting these silence packets has two advantages. First, it maintains the UDP bindings at NAT and second, these packets can be used to play some background noise at the peer.
In the case where media traffic flowed over TCP between caller and callee, silence packets were still sent. The purpose is to avoid the drop in TCP congestion window size, which takes some RTT to reach the maximum level again.