VoIP:
Skype architecture & complete call setup
Seminar 2

By:

Prateek Arora

Abstract

- Skype is a peer-to-peer VoIP client developed by KaZaa in 2003.
- The call-setup and routing is all handled behind the scene by Peer-to-Peer technology acquired from JoltID [http://www.joltid.com] – founded by Mr. Niklas Zennström who was one of the original founders of Kazaa and also is a founder of Skype.
- Skype claims that it can work almost seamlessly across NATs and firewalls and has better voice quality than the MSN and Yahoo IM applications.
- It encrypts calls end-to-end, and stores user information in a decentralized fashion. Skype also supports instant messaging and conferencing.

Skype Architecture

- 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 Architecture (contd.)

- An **ordinary host must connect** to a **super node** and **must register** itself with the **Skype login server** for a successful login.
- The Skype login server is an important entity in the Skype network. User names 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.

Skype Architecture (contd.)

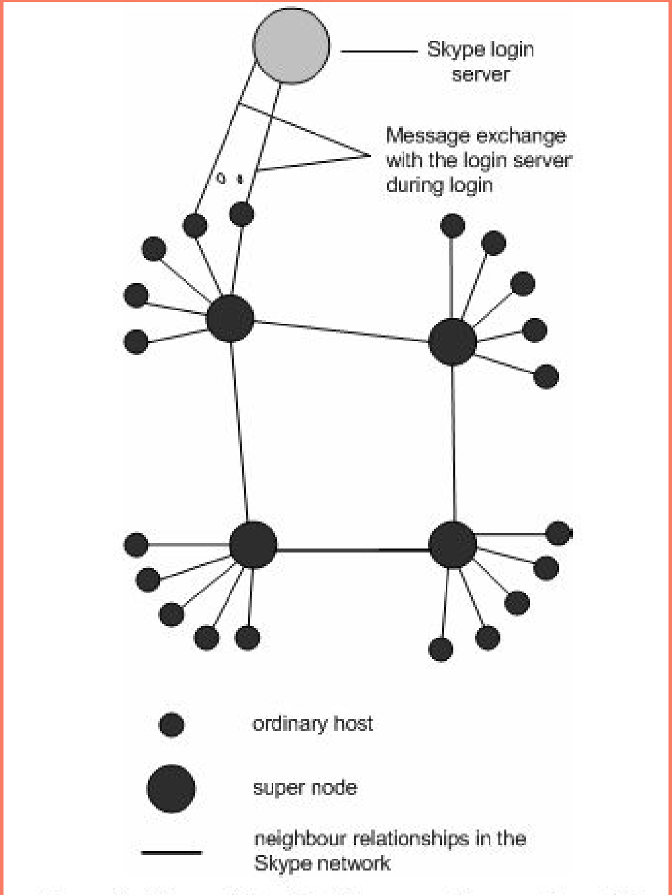

Skype login server
Message exchange with the login server during login
● ordinary host
● super node
— neighbour relationships in the Skype network

Figure 1. Skype Network. There are three main entities: supernodes, ordinary nodes, and the login server.

Apart from the login server, there is no central server in the Skype network. Online and offline user information is stored and propagated in a decentralized fashion and so are the user search queries.

Protocols & Traffic Traversals

- NAT and firewall traversal are important Skype functions.
- Each Skype node uses **a variant of STUN protocol** to determine the type of NAT and firewall it is behind.
- There is no global NAT and firewall traversal server because if there was one, the Skype node would have exchanged traffic with it during login and call establishment.
- 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**.

Technology & Codecs Used

- Skype claims to have implemented **a ‘3G P2P’ or ‘Global Index’ technology**, which is guaranteed to find a user if that user has logged in the Skype network in the last 72 hours.
- 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**.

Key Components

- A Skype client listens on particular **ports** for incoming calls, maintains a table of other Skype nodes called **host cache**, uses wideband **codecs**, maintains a **buddy list**, **encrypts** messages end-to-end, and determines if it is behind a NAT or a firewall.

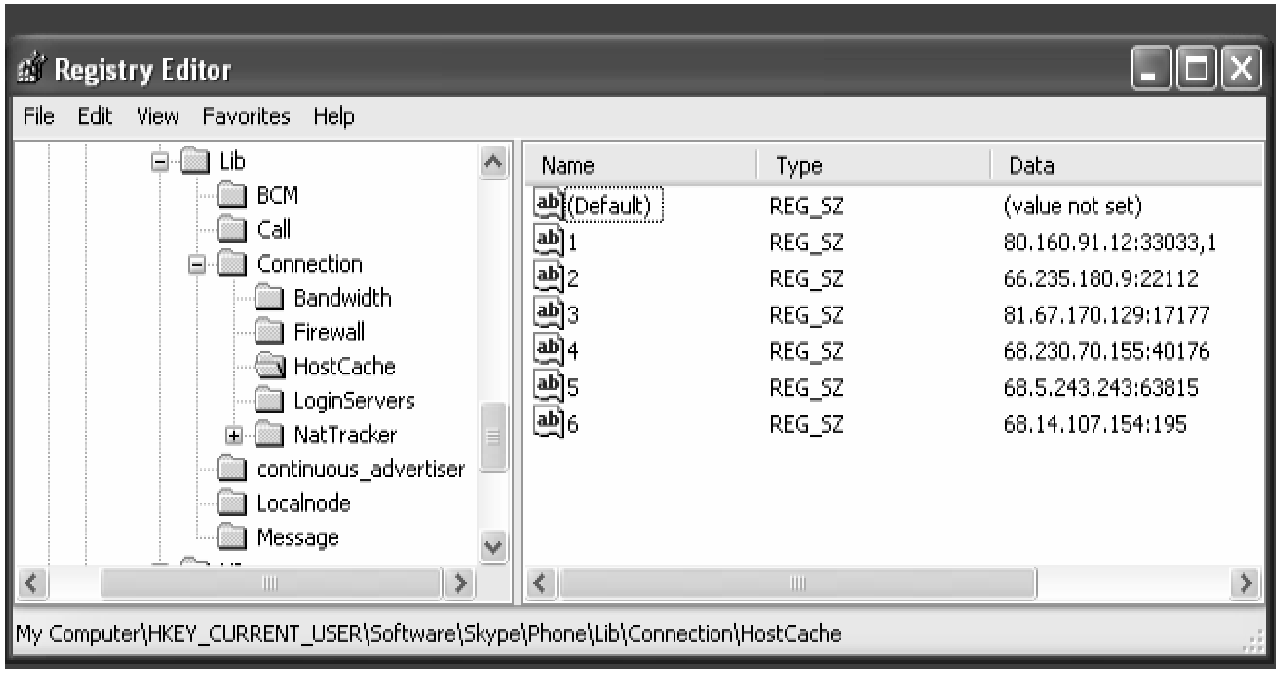
Ports

- 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.
- In addition, 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.

Supernodes & Host Cache

- 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.
- A SC stores host cache in the Windows registry at **HKEY_CURRENT_USER / SOFTWARE / SKYPE / PHONE / LIB / CONNECTION / HOSTCACHE** and in the file "**shared.xml**" for each Skype node.
- After running a SC for two days, it was observed that HC contained a maximum of 200 entries.
- Host and peer caches are not new to Skype. Chord, another peer-to-peer protocol has a finger table, which it uses to quickly find a node.

Supernodes & Host Cache (contd.)

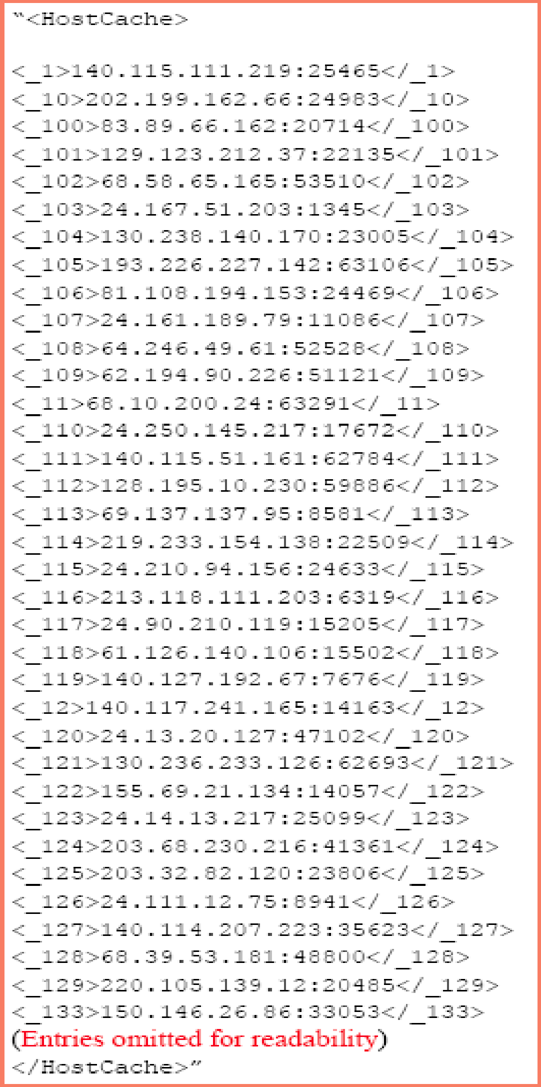


Name	Type	Data
(Default)	REG_SZ	(value not set)
1	REG_SZ	80.160.91.12:33033,1
2	REG_SZ	66.235.180.9:22112
3	REG_SZ	81.67.170.129:17177
4	REG_SZ	68.230.70.155:40176
5	REG_SZ	68.5.243.243:63815
6	REG_SZ	68.14.107.154:195

My Computer\HKEY_CURRENT_USER\Software\Skype\Phone\Lib\Connection\HostCache

Figure 17. Skype host cache list

Supernodes & Host Cache (contd.)



```
"<HostCache>

<_1>140.115.111.219:25465</_1>
<_10>202.199.162.66:24983</_10>
<_100>83.89.66.162:20714</_100>
<_101>129.123.212.37:22135</_101>
<_102>68.58.65.165:53510</_102>
<_103>24.167.51.203:1345</_103>
<_104>130.238.140.170:23005</_104>
<_105>193.226.227.142:63106</_105>
<_106>81.108.194.153:24469</_106>
<_107>24.161.189.79:11086</_107>
<_108>64.246.49.61:52528</_108>
<_109>62.194.90.226:51121</_109>
<_11>68.10.200.24:63291</_11>
<_110>24.250.145.217:17672</_110>
<_111>140.115.51.161:62784</_111>
<_112>128.195.10.230:59886</_112>
<_113>69.137.137.95:8581</_113>
<_114>219.233.154.138:22509</_114>
<_115>24.210.94.156:24633</_115>
<_116>213.118.111.203:6319</_116>
<_117>24.90.210.119:15205</_117>
<_118>61.126.140.106:15502</_118>
<_119>140.127.192.67:7676</_119>
<_12>140.117.241.165:14163</_12>
<_120>24.13.20.127:47102</_120>
<_121>130.236.233.126:62693</_121>
<_122>155.69.21.134:14057</_122>
<_123>24.14.13.217:25099</_123>
<_124>203.68.230.216:41361</_124>
<_125>203.32.82.120:23806</_125>
<_126>24.111.12.75:8941</_126>
<_127>140.114.207.223:35623</_127>
<_128>68.39.53.181:48800</_128>
<_129>220.105.139.12:20485</_129>
<_133>150.146.26.86:33053</_133>
(Entries omitted for readability)
</HostCache>"
```

A typical "shared.xml"

Supernodes & Host Cache (contd.)

- It was found that the Skype application only makes TCP-connections with a few of the hosts above, the rest are contacted with an UDP-connections.
- This could mean that **only some** of those contacted with TCP are **supernodes** and **the rest** possibly are **only peers** connected to the same supernodes as the Skype application client.

- It is also unclear how the list above is compiled:

  An uninstall of Skype leaves the file

  C:\Documents And Settings\All Users\Application Data\Skype\shared.xml

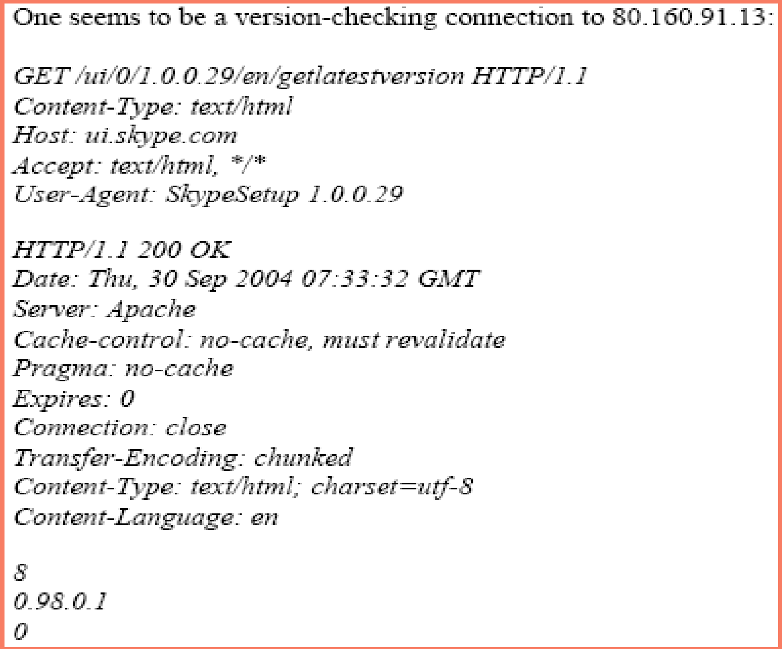
  seemingly untouched.

  When manually deleted it is reconstructed when a new copy of the Skype application is installed.

Supernodes & Host Cache (contd.)

- One could speculate whether this list is dynamically downloaded from the Skype servers or not.
- When sniffing the conversation from the Skype application with **Ethereal** there are only two brief initial connections to the Skype backend servers.

Supernodes & Host Cache (contd.)

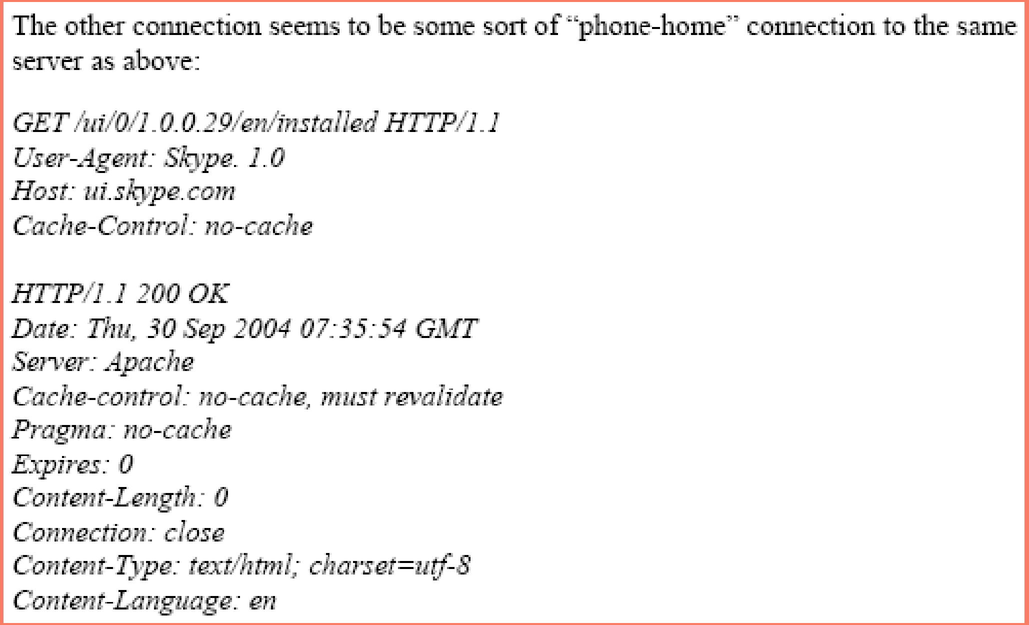


One seems to be a version-checking connection to 80.160.91.13:

```
GET /ui/0/1.0.0.29/en/getlatestversion HTTP/1.1
Content-Type: text/html
Host: ui.skype.com
Accept: text/html, */*
User-Agent: SkypeSetup 1.0.0.29

HTTP/1.1 200 OK
Date: Thu, 30 Sep 2004 07:33:32 GMT
Server: Apache
Cache-control: no-cache, must revalidate
Pragma: no-cache
Expires: 0
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
Content-Language: en

8
0.98.0.1
0
```



The other connection seems to be some sort of "phone-home" connection to the same server as above:

```
GET /ui/0/1.0.0.29/en/installed HTTP/1.1
User-Agent: Skype. 1.0
Host: ui.skype.com
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Thu, 30 Sep 2004 07:35:54 GMT
Server: Apache
Cache-control: no-cache, must revalidate
Pragma: no-cache
Expires: 0
Content-Length: 0
Connection: close
Content-Type: text/html; charset=utf-8
Content-Language: en
```

Nowhere in the trace could the data in the HostCache section of the shared.xml file be seen, but immediately after the connection above was finished, the locally installed Skype Application client started to send UDP-packets to hosts in that very list.
This seems to indicate that some potions of the list is hardcoded into the application itself, but this is unclear as there are no officially technical documentation that covers the internal workings of Skype.

Importance of Supernodes

- The importance of the supernodes seems quite clear though, since all call-handling is seemingly done by supernodes.
- Since the end-user have no way of influencing which supernode the Skype application connects to or even if the end-users computer itself is promoted to a supernode there are no easy steps for running the Skype application confined to a particular network say a corporate network.
- At some point the Skype application clients inside the network must connect to external entities, both the centrally managed Skype back-end servers, but also to a arbitrary supernode which in most cases is some other Skype application end-user.

Codecs

- The white paper "**Skype conferencing white paper by PowerModeling: http://www.powermodeling.com/files/whitepapers/Conference%20Test%20feb%2009.pdf**" observes that Skype uses **iLBC**, **iSAC**, or a third unknown codec.

- **GlobalIPSound** has implemented the iLBC and iSAC codecs and their website lists Skype as their partner.

- When measured it was found that the Skype codecs **allow frequencies** between **50-8,000 Hz** to pass through. This frequency range is the characteristic of a wideband codec.

Buddy List

- Skype stores its buddy information in the Windows registry.
- 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.

Encryption

- Skype uses AES (Advanced Encryption Standard), also known as Rijndel. It is also used by U.S. Government organizations to protect sensitive information.
- Skype uses 256-bit encryption, which has a total of 1.1 x 1077 possible keys, 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.

NAT and Firewall

- It is observed that SC uses a variation of the STUN, and TURN protocols to determine the type of NAT and firewall it is behind.
- It is also observed 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.

Experimental Setup

- The details about Skype functions were found by making an experimental setup by the author of the paper "**An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol**".
- All experiments were performed for Skype version 0.97.0.6. Skype was installed on two Windows 2000 machines. One machine was a Pentium II 200MHz with 128 MB RAM, and the other machine was a Pentium Pro 200 MHz with 128 MB RAM. Each machine had a 10/100 Mb/s Ethernet card and was connected to a 100 Mb/s network. The experiments were performed under three different network setups.
- In the first setup, both Skype users were on machines with public IP addresses; in the second setup, one Skype user was behind port-restricted NAT; in the third setup, both Skype users were behind a port-restricted NAT and UDP-restricted firewall. NAT and firewall machines ran Red Hat Linux 8.0 and were connected to 100 Mb/s Ethernet network.
- Ethereal and NetPeeker were used to monitor and control network traffic, respectively. NetPeeker was used to tune the bandwidth so as to analyze the Skype operation under network congestion.
- For each experiment, the Windows registry was cleared of any Skype entries and Skype was reinstalled on the machine. All experiments were performed between February and April, 2004.

Skype Functions

Skype functions can be classified into:

1. Startup
2. Login
3. User search
4. Call establishment and tear down
5. Media transfer
6. Presence messages

Startup

- When SC was run for the first time after installation, it sent a HTTP 1.1 GET request to the Skype server (skype.com).
- The first line of this request contains the keyword ‘installed’.
- During subsequent startups, a SC only sent a HTTP 1.1 GET request to the Skype server (skype.com) to determine if a new version is available.
- The first line of this request contains the keyword ‘getlatestversion’.

Login

- Login is perhaps the most critical function to the Skype operation.
- It is during this process a SC authenticates its user name and password with the login server, advertises its presence to other peers and its buddies, determines the type of NAT and firewall it is behind, and discovers online Skype nodes with public IP addresses.
- It was observed that these newly discovered nodes were used to maintain connection with the Skype network should the SN to which SC was connected became unavailable.

Login Process

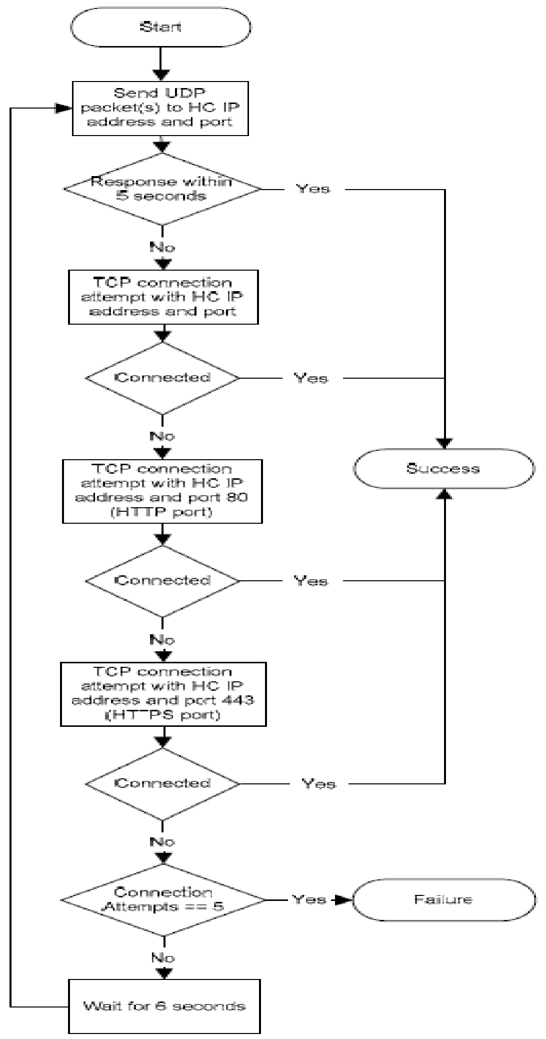

Start
Send UDP packet(s) to HC IP address and port
Response within 5 seconds
Yes
No
TCP connection attempt with HC IP address and port
Connected
Yes
No
Success
TCP connection attempt with HC IP address and port 80 (HTTP port)
Connected
Yes
No
TCP connection attempt with HC IP address and port 443 (HTTPS port)
Connected
Yes
No
Connection Attempts == 5
Yes
Failure
No
Wait for 6 seconds

Figure 2. Skype login algorithm. Only one entry is present in the HC. If there is more than one entry, SC sends UDP packets to them before attempting a TCP connection. Authentication with the login server is not shown.

Login Process (contd.)

- The HC must contain a valid entry for a SC to be able to connect to the Skype network. If the HC was filled with only one invalid entry, SC could not connect to the Skype network and reported a login failure.
- To understand and gain useful insights in the Skype login process by observing the message flow between SC and this invalid HC entry. The experimental setup and observations for the login process are described below:
- First, we flushed the SC host cache and filled it with only one entry which was the IP address and port number of a machine on which no Skype client was running.
- The SC was then started and a login attempt was made. Since HC had an invalid entry, SC could not connect to the Skype network. We observed that the SC first sent a UDP packet to this entry. If there was no response after roughly five seconds, SC tried to establish a TCP connection with this entry. It then tried to establish a TCP connection to the HC IP address and port 80 (HTTP port). If still unsuccessful, it tried to connect to HC IP address and port 443 (HTTPS port). SC then waited for roughly 6 seconds. It repeated the whole process four more times after which it reported a login failure.

Login Process (contd.)

- It was observed that 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.

Login Server

- After a SC is connected to a SN, the SC must authenticate the user name and password with the Skype login server.
- The login server is the only central component in the Skype network. It stores Skype user names and passwords and ensures that Skype user names are unique across the Skype name space.
- SC must authenticate itself with login server for a successful login.
- It was observed during the experiments that SC always exchanged data over TCP with a node whose IP address was 80.160.91.11.
- It was since then believed 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.

Bootstrap Super Nodes

- After logging in for the first time after installation, HC was initialized with seven IP address and port pairs.
- It was observed that upon first login, HC was always initialized with these seven IP address and port pairs except for a rare random occurrence. In the case where HC was initialized with more than seven IP addresses and port pairs, it always contained those 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**.
- These IP address and port pairs and their corresponding host names obtained using a reverse lookup are:

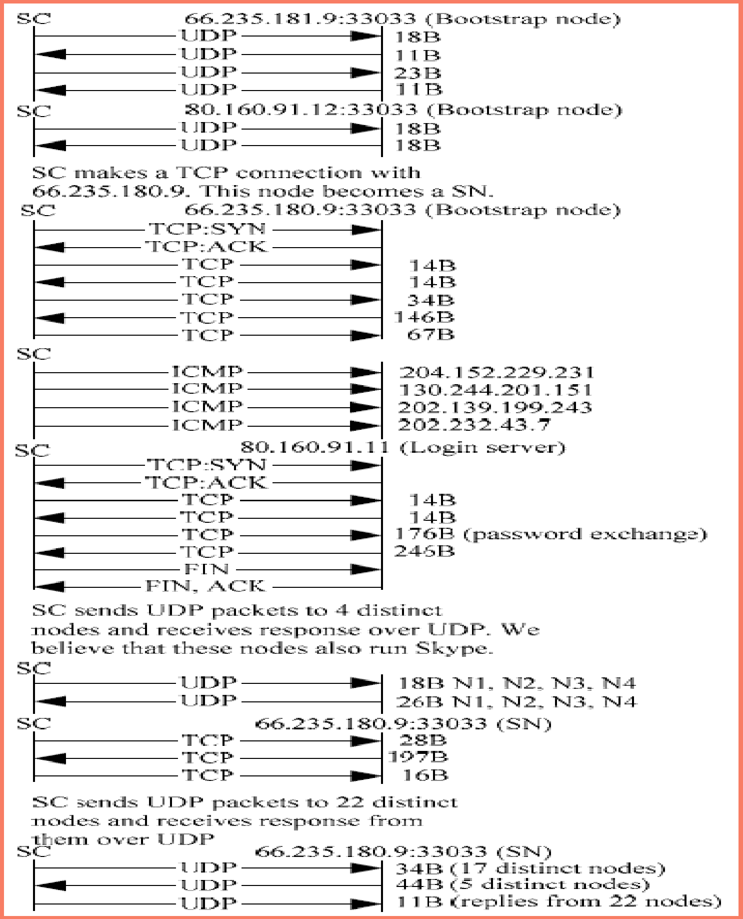
IP address : port	Reverse lookup result
66.235.180.9:33033	sls-cb10p6.dca2.superb.net
66.235.181.9:33033	ip9.181.susc.suscom.net
80.161.91.25:33033	0x50a15b19.boanxx15.adsl-dhcp.tele.dk
80.160.91.12:33033	0x50a15b0c.albnxx9.adsl-dhcp.tele.dk
64.246.49.60:33033	rs-64-246-49-60.ev1.net
64.246.49.61:33033	rs-64-246-49-61.ev1.net
64.246.48.23:33033	ns2.ev1.net

- From the reverse lookup, it appears that bootstrap SNs are connected to the Internet through four ISPs: Superb, Suscom, and ev1.net are US-based ISPs.

Bootstrap Super Nodes

- After installation and first time startup, it was 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.
- It was also observed that if the HC was flushed after the first login, SC was unable to connect to the Skype network.

First-time Login Process


SC
66.235.181.9:33033 (Bootstrap node)
UDP 18B
UDP 11B
UDP 23B
UDP 11B
SC
80.160.91.12:33033 (Bootstrap node)
UDP 18B
UDP 18B
SC makes a TCP connection with 66.235.180.9. This node becomes a SN.
SC
66.235.180.9:33033 (Bootstrap node)
TCP:SYN
TCP:ACK
TCP 14B
TCP 14B
TCP 34B
TCP 146B
TCP 67B
SC
ICMP 204.152.229.231
ICMP 130.244.201.151
ICMP 202.139.199.243
ICMP 202.232.43.7
SC
80.160.91.11 (Login server)
TCP:SYN
TCP:ACK
TCP 14B
TCP 14B
TCP 176B (password exchange)
TCP 245B
FIN
FIN, ACK
SC sends UDP packets to 4 distinct nodes and receives response over UDP. We believe that these nodes also run Skype.
SC
UDP 18B N1, N2, N3, N4
UDP 26B N1, N2, N3, N4
SC
66.235.180.9:33033 (SN)
TCP 28B
TCP 197B
TCP 16B
SC sends UDP packets to 22 distinct nodes and receives response from them over UDP
SC
66.235.180.9:33033 (SN)
UDP 34B (17 distinct nodes)
UDP 44B (5 distinct nodes)
UDP 11B (replies from 22 nodes)

Message flow for the first login after installation for SC on a public IP address. ‘B’ stands for bytes and ‘N’ stands for node. SYN and ACK packets are shown to indicate who initiated TCP connection. Message flows are not strictly according to time. Messages have been grouped together to provide a better picture. Message size corresponds to size of TCP or UDP payload. Not all messages are shown.

First-time Login Process (contd.)

- The SC host cache was empty upon installation. Thus, a SC must connect to well known Skype nodes in order to log on to the Skype network. It does so by sending UDP packets to some bootstrap super nodes and then waits for their response over UDP for some time. It is not clear how SC selects among bootstrap SNs to send UDP packets to.
- SC then established a TCP connection with the bootstrap super node that responded. Since more than one node could respond, a SC could establish a TCP connection with more than one bootstrap node. A SC, however, maintains a TCP connection with at least one bootstrap node and may close TCP connections with other nodes.
- After exchanging some packets with SN over TCP, it then perhaps acquired the address of the login server (80.160.91.11). SC then establishes a TCP connection with the login server, exchanges authentication information with it, and finally closes the TCP connection.
- The initial TCP data exchange with the bootstrap SN and the login server shows the existence of a challenge-response mechanism. 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.

NAT and Firewall Determination

- It is observed that a SC is able to determine at login if it is behind a NAT and firewall. It is assumed that there are at least two ways in which a SC can determine this information.
- One possibility is that it can determine this information by exchanging messages with its SN using a variant of the STUN protocol.
- The other possibility is that during login, a SC sends and possibly receives data from some nodes after it has made a TCP connection with the SN.
- It is observed that at this point, SC uses its variation of STUN protocol to determine the type of NAT or firewall it is behind. Once determined, the SC stores this information in the Windows registry. It is also observed that SC refreshes this information periodically.
- However, it is not clear on how often a SC refreshes this information since Skype messages are encrypted.

Alternate Node Table

- 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.
- From the experiments done, it can be seen that 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.
- It is assumed that SC uses those messages to advertise its arrival on the network. It is also assumed that upon receiving a response from them, SC builds a table of online nodes. The author of this paper calls 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.

Subsequent Login Process

- The subsequent login process was quite similar to the first-time login process. The SC built a HC after a user logged in for the first time after installation. The HC got periodically updated with the IP address and port number of new peers.
- During subsequent logins, SC used the login algorithm to determine at least one available peer out of the nodes present in the HC.
- It then established a TCP connection with that node.
- It was also observed that during subsequent logins, SC did not send any ICMP packets.

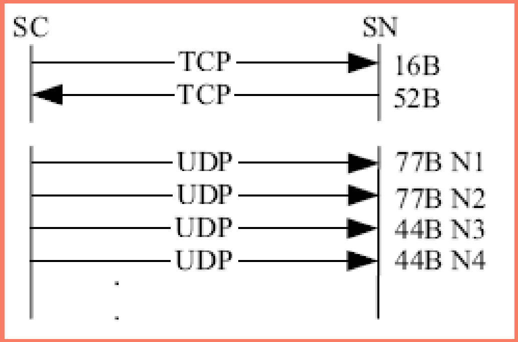
Login Process Time

- As per the experiments performed, the time to login was measured on the Skype network for the three different network setups as described in Experimental Setup earlier.
- For this experiment, the HC already contained the maximum of two hundred entries.
- The **SC with a public IP address** and the **SC behind a port-restricted NAT** took about **3-7 seconds** to complete the login procedures.
- The **SC behind a UDP-restricted firewall** took about **34 seconds** to complete the login process.
- For SC behind a UDP-restricted firewall, it was observed that it sent UDP packets to its thirty HC entries. At that point it concluded that it is behind UDP-restricted firewall. It then tried to establish a TCP connection with the HC entries and was ultimately able to connect to a SN.

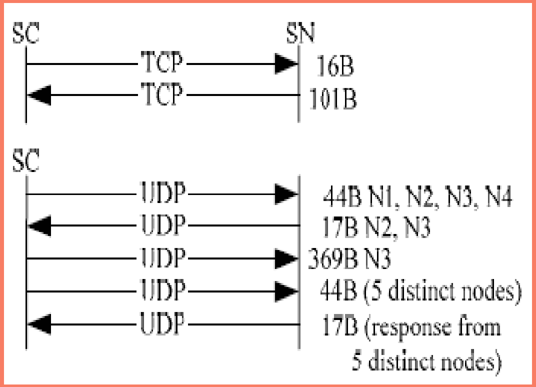
User Search

- Skype uses its Global Index (GI) technology to search for a user. Skype claims that search is distributed and is guaranteed to find a user if it exists and has logged in during the last 72 hours.
- Extensive testing suggests that Skype was always able to locate users who logged in using public or private IP address in the last 72 hours.
- Skype is a not an open protocol and its messages are encrypted. Whereas in login we were able to form a reasonably precise opinion about different entities involved, it is not possible to do so in search, since we cannot trace the Skype messages beyond a SN.
- Also, we were unable to force a SC to connect to a particular SN. Nevertheless, it was observed and following are search message flows for the three different experimental network setups.

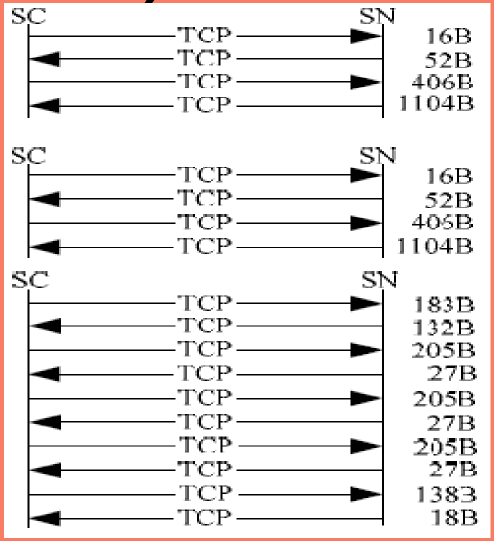
User Search (contd.)



Message flow for user search when SC has a public IP address. ‘B’ stands for bytes and ‘N’ stands for node. Message sizes correspond to payload size of TCP or UDP packets.



Message flow for user search when SC is behind a port-restricted NAT. ‘B’ stands for bytes and ‘N’ stands for node. UDP packets were sent to N1, N2, N3, and N4 during login process and responses were received from them. Message size corresponds to payload size of TCP or UDP packets.



User search by a SC behind a UDP-restricted firewall. ‘B’ stands for bytes. Data is exchanged with SN only. Message size corresponds to payload size of TCP/UDP packets.

User Search (contd.)

- A SC has a search dialog box. After entering the Skype user id and pressing the find button, SC starts its search for a particular user.
- For **SC on a public IP address**, SC sent a TCP packet to its SN. It appears that SN gave SC the IP address and port number of four nodes to query, since after that exchange with SN, SC sent UDP packets to four nodes.
- It was also observed that SC had not exchanged any information with these four nodes during login. SC then sent UDP packets to those nodes. If it could not find the user, it informed the SN over TCP. It appears that the SN now asked it to contact eight different nodes, since SC then sent UDP packets to eight different nodes.
- This process continued until the SC found the user or it determined that the user did not exist.
- On average, **SC contacted eight nodes**. The **search took three to four seconds**. We are not clear on how SC terminates the search if it is unable to find a user.

User Search (contd.)

- A **SC behind a port-restricted NAT** exchanged data between SN, and some of the nodes which responded to its UDP request during login process as shown in the corresponding message flow figure before.
- A **SC behind a port-restricted NAT** and **UDP-restricted firewall** sent the search request over TCP to its SN. It is believed that SN then performed the search query and informed SC of the search results.
- Unlike user search by SC on a public IP address, SC did not contact any other nodes. This suggests that SC knew that it was behind a UDP-restricted firewall. The message flow figure for UDP-restricted firewall is given earlier.

Call Establishment

- We consider call establishment for the three experimental network setups. Further, for each setup, we consider call establishment for users that are in the buddy list of caller and for users that are not present in the buddy list. It is important to note that call signaling is always carried over TCP.
- For users that are not present in the buddy list, call placement is equal to user search plus call signaling. Thus, we discuss call establishment for the case where callee is in the buddy list of caller.
- If both users were on public IP addresses, online and were in the buddy list of each other, then upon pressing the call button, the caller SC established a TCP connection with the callee SC.
- Signaling information was exchanged over TCP. The initial exchange of messages between caller and callee indicates the existence of a challenge-response mechanism.
- The caller also sent some messages (not shown in Figure) over UDP to alternate Skype nodes, which are online Skype nodes discovered during login. For this scenario, three kilobytes of data was exchanged.

Call Establishment (contd.)

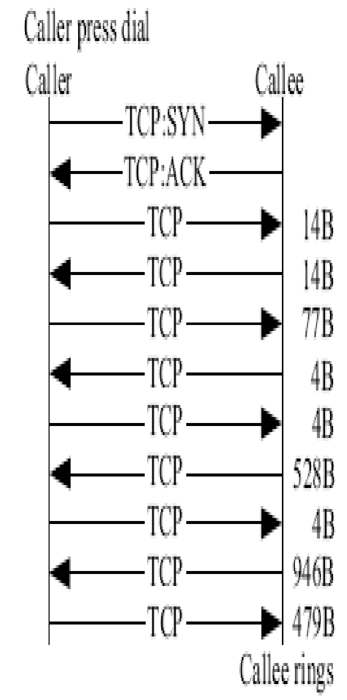


Figure 9. Message flow for call establishment when caller and callee SC are on machines with public IP addresses and callee is present in the buddy lists of caller. 'B' stands for bytes. Not all messages are shown.

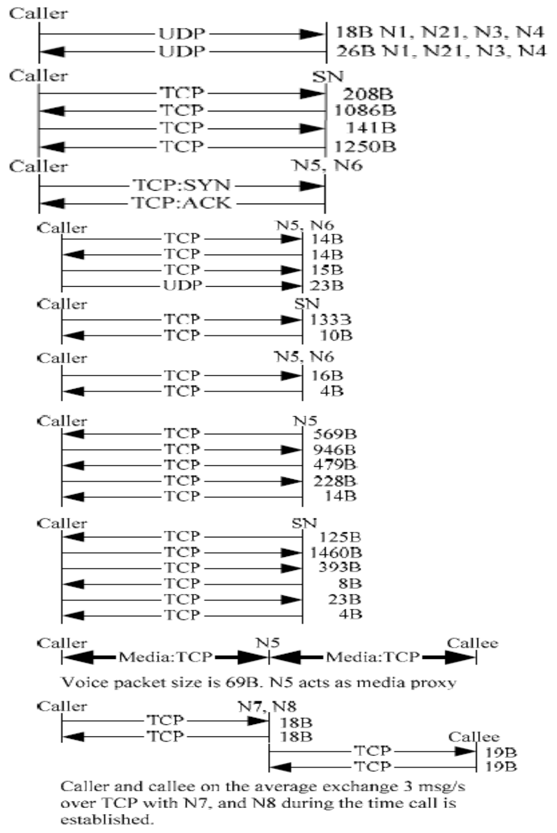


Figure 10. Message flow for call establishment when caller SC is behind a port-restricted NAT and callee SC is on public IP address. 'B' stands for bytes and 'N' stands for node. Not all messages are shown. Caller SC sent UDP messages to nodes 5, 6, 7, and 8 during login and received responses from them. We thus believe caller SC stored the IP address and port of these nodes in its internal tables, which we call the alternate node table.

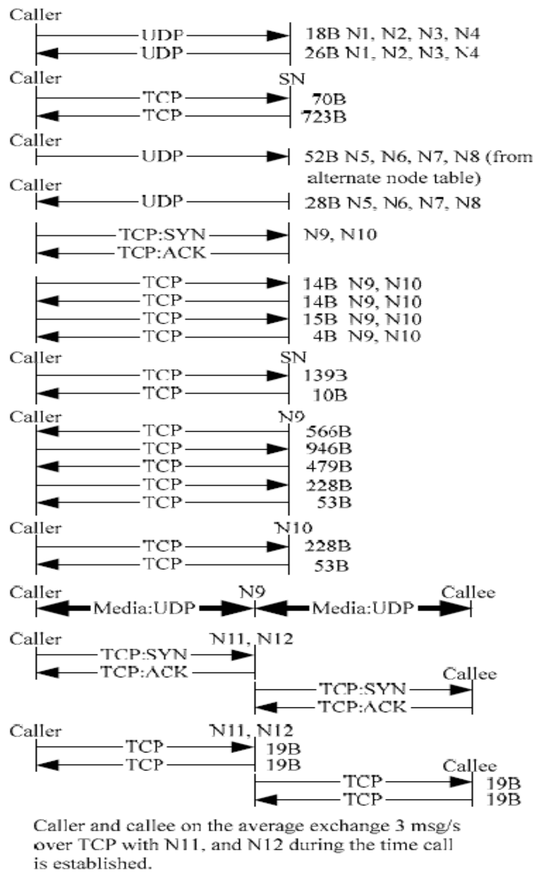


Figure 11. Message flow for call establishment when caller and callee SC are behind a port-restricted NAT and UDP-restricted firewall. 'B' stands for bytes and 'N' stands for a node. Not all messages are shown. Voice traffic flows over TCP.

Call Establishment (contd.)

- In the second network setup, where the caller was behind port-restricted NAT and callee was on public IP address, signaling and media traffic did not flow directly between caller and callee. Instead, the caller sent signaling information over TCP to an online Skype node which forwarded it to callee over TCP. This online node also routed voice packets from caller to callee over UDP and vice versa. The message flow is shown in the corresponding Figure earlier.

- For the third setup, in which both users were behind port-restricted NAT and UDP-restricted firewall, both caller and callee SC exchanged signaling information over TCP with another online Skype node. Caller SC sent media over TCP to an online node, which forwarded it to callee SC over TCP and vice versa. The message flow is shown in the corresponding Figure earlier.

Tear Down

- During call tear-down, signaling information is exchanged over TCP between caller and callee if they are both on public IP addresses, or between caller, callee and their respective SNs. The messages observed for call tear down between caller and callee on public IP addresses are shown in following figure.

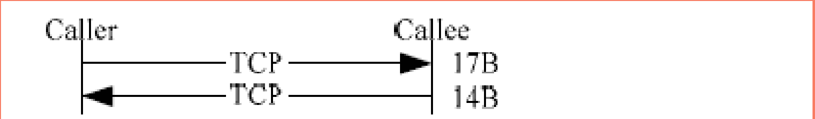


Figure 12. Call tear down message flow for caller and callee with public IP addresses

- For the second, and third network setups, call tear down signaling is also sent over TCP. We, however, do not present these message flows, as they do not provide any interesting information.

Media Transfer and Codecs

- 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. This bandwidth usage corresponds with the Skype claim of 3-16 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. The voice packet size was 67 bytes, which is the size of UDP payload. The bandwidth used was 5 kilobytes/s.

- If both users were behind port-restricted NAT and UDP-restricted firewall, then caller and callee sent and received voice traffic over TCP from another online Skype node. The TCP packet payload size for voice traffic was 69 bytes. The total uplink and downlink bandwidth used for voice traffic is about 5 kilobytes/s.For media traffic, SC used TCP with retransmissions.

- The Skype protocol seems to prefer the use of UDP for voice transmission as much as possible. The SC will use UDP for voice transmission if it is behind a NAT or firewall that allows UDP packets to flow across.

Congestion Metric

- Skype call quality was checked in a low bandwidth environment by using Net Peeker to tune the upload and download bandwidth available for a call.
- It was observed that uplink and downlink bandwidth of 2 kilobytes/s each was necessary for reasonable call quality.
- The voice was almost unintelligible at an uplink and downlink bandwidth of 1.5 kilobytes/s.

Keep-alive Messages

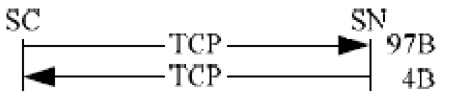


Figure 13. Skype refresh message to SN

It was observed in for three different network setups that the SC sent a refresh message to its SN over TCP every 60s.

Pros & Cons of Node Route

There are many advantages of having a node route the voice packets from caller to callee and vice versa, as follows:

Pros:

1) It provides a mechanism for users behind NAT and firewall to talk to each other.
2) If users behind NAT or firewall want to participate in a conference, and some users on public IP address also want to join the conference, this node serves as a mixer and broadcasts the conferencing traffic to the participants.

Cons:

1) There will be a lot of traffic flowing across this node.
2) Users generally do not want that arbitrary traffic should flow across their machines.

Conclusion

- Skype is the first VoIP client based on peer-to-peer technology. There are three factors are responsible for its increasing popularity.
- First, it provides better voice quality than MSN and Yahoo IM clients; second, it can work almost seamlessly behind NATs and firewalls; and third, it is extremely easy to install and use.
- It is believed that Skype client uses its version of STUN protocol to determine the type of NAT or firewall it is behind. The NAT and firewall traversal techniques of Skype are similar to many existing applications such as network games.
- It is by the random selection of sender and listener ports, the use of TCP as voice streaming protocol, and the peer-to-peer nature of the Skype network, that not only a SC traverses NATs and firewalls but it does so withhout any explicit NAT or firewall traversal server.
- Skype uses TCP for signaling. It uses wide band codecs and has probably licensed them from GlobalIPSound. Skype communication is encrypted end-to-end.

References

- www.google.co.in
- www.skype.com
- http://arxiv.org/ftp/cs/papers/0412/0412017.pd
- http://www.mathaba.net/MNN/www.skype-news.com
- http ://voipsa.org/pipermail/voipsec_voipsa.org/2005-Octo
- http://www1.cs.columbia.edu/~salman/skype/index.h