

Peer-to-Peer IP Telephony

GEIR EGELAND AND PAAL ENGELSTAD



Paal E. Engelstad
is Research
Scientist at
Telenor R&D



Geir Egeland
is Research
Scientist at
Telenor R&D

The recent marriage between IP telephony and Peer-to-Peer communication, e.g. in applications such as Skype, has made a tremendous impact on the telecommunications industry. Although the original Internet design was based on a peer-to-peer architecture, applications such as Skype must rely on an overlay Peer-to-Peer network in order to cope with the underlying client-server paradigm imposed by NATs and firewalls. Initially peer-to-peer communication and IP telephony evolved independently. This article takes a closer look at the exciting development seen in the intersection between IP telephony and Peer-to-Peer communication. NATs and firewalls are identified as show-stoppers for further growth, while IPv6 might represent a solution.

Introduction

In August 2003 a small application called Skype was introduced on the Internet. It could be downloaded for free, and it enabled its users to make VoIP calls to each other free of charge. The most compelling feature of this application, in addition to being free, was that the voice quality was comparable to that of traditional fixed line telephony. Other applications such as MSN and Yahoo messenger had for some time offered its users voice communication, but with poor voice quality and with one major obstacle – they could not operate when the users were located behind a Network Address Translator. Skype on the other hand, introduced advanced traversal techniques enabling the users to communicate peer-to-peer even when both the caller and the callee are located behind a Network Address Translator.

After a general overview of peer-to-peer communication, we present IP telephony and how it can be deployed in peer-to-peer settings. The term “peer-to-peer” is not used as a euphemism for file sharing or other related activities, but in its original architectural sense, that all hosts on the network are logically equals. This was indeed the case in the early days of the Internet, but as the usage of the Internet changed, and the dominating communication form on the Internet became client/server, the end-to-end model of the Internet was broken, disabling peer-to-peer communication.

We also describe some of the technology used to traverse Network Address Translators to enable peer-to-peer communication, as well as some of the problems associated with these solutions.

Peer-to-Peer communication

The original design of the ARPANET, which forms the basis of the Internet architecture, was inherently peer-to-peer. Certainly, Internet connections differed

in bandwidth, latency, and reliability, but apart from those physical properties, any host could communicate on equal terms with any other host on the network. Any Internet host could provide any service to any other and access any service provided by them. This is opposed to the client-server paradigm, where communication only occurs between a client and a server. A peer-to-peer paradigm is more general, since it allows for communication between not only a client and a server, but between any client or any peers.

The first peer-to-peer applications that appeared were chatting applications on Unix operating systems. It allowed users to send instant messages to each other, similar to what Yahoo, AOL and MSN offer today. Since they did not need a centralized server to do this, unlike today’s *Instant Messaging* (IM) applications, the applications were truly peer-to-peer. They simply used Unix user names and the computer’s fully qualified domain name, i.e. by typing, “talk john_doe@hostname.somedomain.com” on the command line.

When the technology matured, the first peer-to-peer applications also offering voice appeared. One of these was *Speak Freely*, an application that allowed two or more people to conduct a real-time voice conference over the Internet or any other TCP/IP network.

Emerging problems with middleboxes

Due to a shortage of IPv4 addresses, Internet Service Providers were looking for techniques that preserved their IPv4 address space. At the same time, home users wanted to enable multiple machines to share the Internet connection. This was solved with the introduction of *Network Address Translation* (NAT) [1] [2], which permits multiple computers to share a common internet connection with a single IP address or with a limited pool of IP addresses.

At the time NATs were introduced, the vast majority of Internet communication was client-server-based, such as web-browsing, e-mail and file transfers. NATs were therefore designed according to this paradigm, and were embraced by the market because the deployment of peer-to-peer applications that required a strict peer-to-peer architecture was not considered significant. A network element that implements NAT is normally referred to as a “NAT-box” or simply as a “NAT” (Network Address Translator). The NAT has an “inside” encompassing the clients and an “outside” where all servers are located. In line with the client-server paradigm, the NAT assumes that the clients inside the NAT initiate all traffic. When a client initiates a session to a server outside the NAT, the NAT replaces the client’s private IP address with a global IP address of the NAT. The reply from the server is therefore sent to this address. When the reply is received by the NAT, it replaces the global IP address of the NAT with the client’s private IP address and forwards the packet to the client. All subsequent communication between the client and the server is subject to this kind of address translation.

The NAT approach assumes that all peers inside the NAT operate as clients and communicate with servers outside the NAT. A peer that is located outside a NAT cannot contact a peer located inside the NAT (although the opposite is possible). If two peers are located behind different NATs, peer-to-peer communication is not possible. Another problem is that NATs operate transparently to the peers. If the two peers cannot communicate because either one or both peers are located behind a NAT, the failure will not be detected by the peer-to-peer application.

The wide and rapidly growing deployment of NATs had direct influence on the development of peer-to-peer communication. After April 1996, for example, the development on the Speak Freely peer-to-peer application (mentioned above) was discontinued. The reason was the increasing usage of NATs, which destroyed the peer-to-peer architecture that the Internet was built upon.

Nowadays, NATs are so widely deployed that communication on the Internet that is peer-to-peer (in its strict architectural sense) cannot be guaranteed to work.

Overlay networks and Peer-to-Peer communication

The term peer-to-peer communication was re-introduced by an application called *Napster*. This was not an application for peer-to-peer communication using text or voice, but to let users share mp3-files over the Internet. After that, the term Peer-to-Peer (P2P) was

associated with file sharing. Napster provided a server where users could upload an index of the mp3-files available at the computer from which they connected to the network. The Napster clients could ask the server “Where is this file”, and the server would answer with the IP-address and port of the client that had provided the information. The requesting client could now contact the other Napster client directly and download the mp3-file, giving the clients the impression of communicating peer-to-peer. The Napster server was an easy target for legal authorities, and was soon forced to close its service.

Other file sharing applications were soon to follow where the centralized server of Napster was removed, enabling peers to form a distributed Peer-to-Peer overlay network (“P2P network”). The fundamental principle behind the P2P networks is that each and every node has equal importance in the network that is formed. Rather than a large number of client machines contacting one or more central servers, nodes interact directly with each other. Each node that participates in the P2P network provides server-like functionality and may act as both server and client within the system. The Gnutella Protocol [3] is an example of a technology used to realize a P2P network (Figure 1). In a Gnutella network, every client is a server, and vice versa. These so-called Gnutella servents perform tasks normally associated with both clients and servers. They provide client-side interfaces through which users can issue queries and view search results, while at the same time they also accept queries from other servents, check for matches against their local data set, and respond with applicable results. Due to its distributed nature, a network of servents that implements the Gnutella protocol is highly fault-tolerant, as operation of the network will not be interrupted if a subset of servents goes offline.

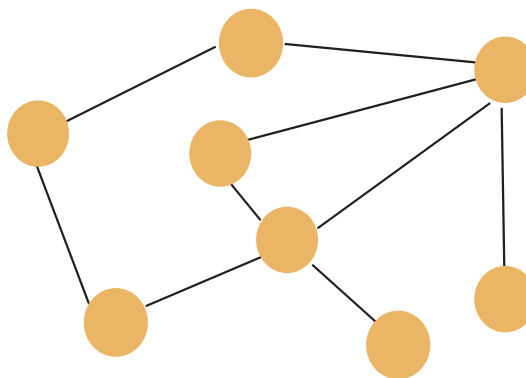


Figure 1 The architecture the Gnutella P2P overlay network

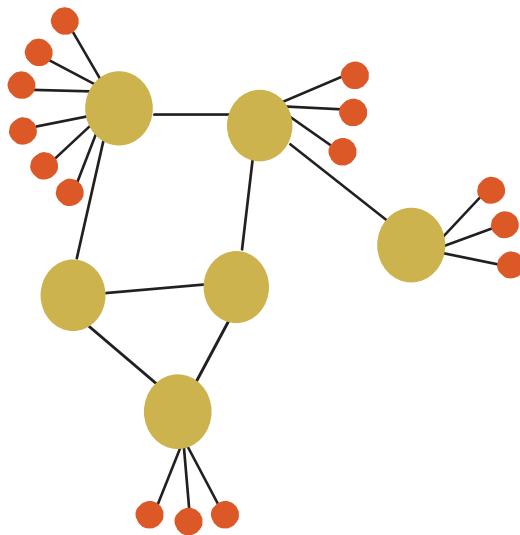


Figure 2 The architecture of the FastTrack P2P overlay network

With an overlay network, it is possible to implement distributed techniques to communicate freely, despite the possible presence of NATs in the network. This has made overlay networks an inevitable part of modern peer-to-peer communication in today's Internet. It should be noted that P2P networks are not peer-to-peer communication in the strict architectural sense. That is, underneath the P2P network peers that are located inside a NAT might always operate as clients, while peers not located inside a NAT might always operate as servers. However, for the P2P network, on the contrary, this underlying networking is concealed. In the overlay network, all nodes are peers that operate on equal terms. In this paper, we use the terms "P2P" or "Peer-to-Peer" (with capital "P"s) to distinguish overlay network architectures from network architectures that are "peer-to-peer" (with lower-case "p"s) in its architectural sense.

The most popular P2P networks today are overlay networks that use the second-generation P2P protocol FastTrack [4]. The FastTrack network was based on the Gnutella protocol [3], but was extended with the addition of *Supernodes* to improve scalability (Figure 2). Since most P2P-users have a higher download rate than upload rate, their links often became congested with search queries. To avoid this, P2P-users with such connections can upload its index of offered files to a supernode. The supernode can then answer search requests on behalf of the P2P-user with the slow connection. The supernodes in a P2P network can also act as proxy for P2P-users behind a NAT, since two P2P-users, each located behind their respective NAT, have no means of communicating peer-to-peer. The supernode functionality is built into the client software, and a P2P-user with a fast network connection and a computer with a global routable IP-address can become a supernode.

IP Telephony as a peer-to-peer application

Nowadays, end-to-end IP telephony comes in two forms. One form is where the end-user subscribes to a VoIP service from an IP telephony operator. The user uses some IP telephony equipment that is connected to the user's Internet connection, and the operator implements some IP telephony infrastructure on the Internet that the user connects to. This resembles traditional telephony in the way that the subscriber connects to a predefined network infrastructure. It also resembles the client-server paradigm with the users acting as clients and the IP telephony infrastructure acting as the server.

Another form of IP telephony is Peer-to-Peer. The IP telephony infrastructure mainly consists of the overlay network formed by all the other IP telephony peers. This is the main focus of this article. In the following we will present a number of VoIP Peer-to-Peer applications.

Skype

The Peer-to-Peer application Skype [5] is an example of a VoIP application that combines the Peer-to-Peer ability of modern P2P-networks with telephony. Skype operates over the FastTrack P2P-network, and takes advantage of the supernode functionality that effectively provides relay servers for Skype users behind a NAT. Skype users can also call traditional telephone numbers for a fee (SkypeOut) or receive calls from traditional phones (SkypeIn). SkypeOut and SkypeIn are realized by implementing gateways between the Skype Peer-to-Peer network and traditional PSTN telephony networks.

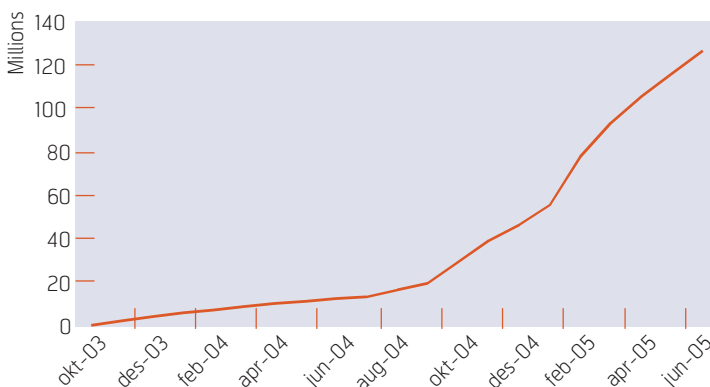


Figure 3 Downloading rate of Skype

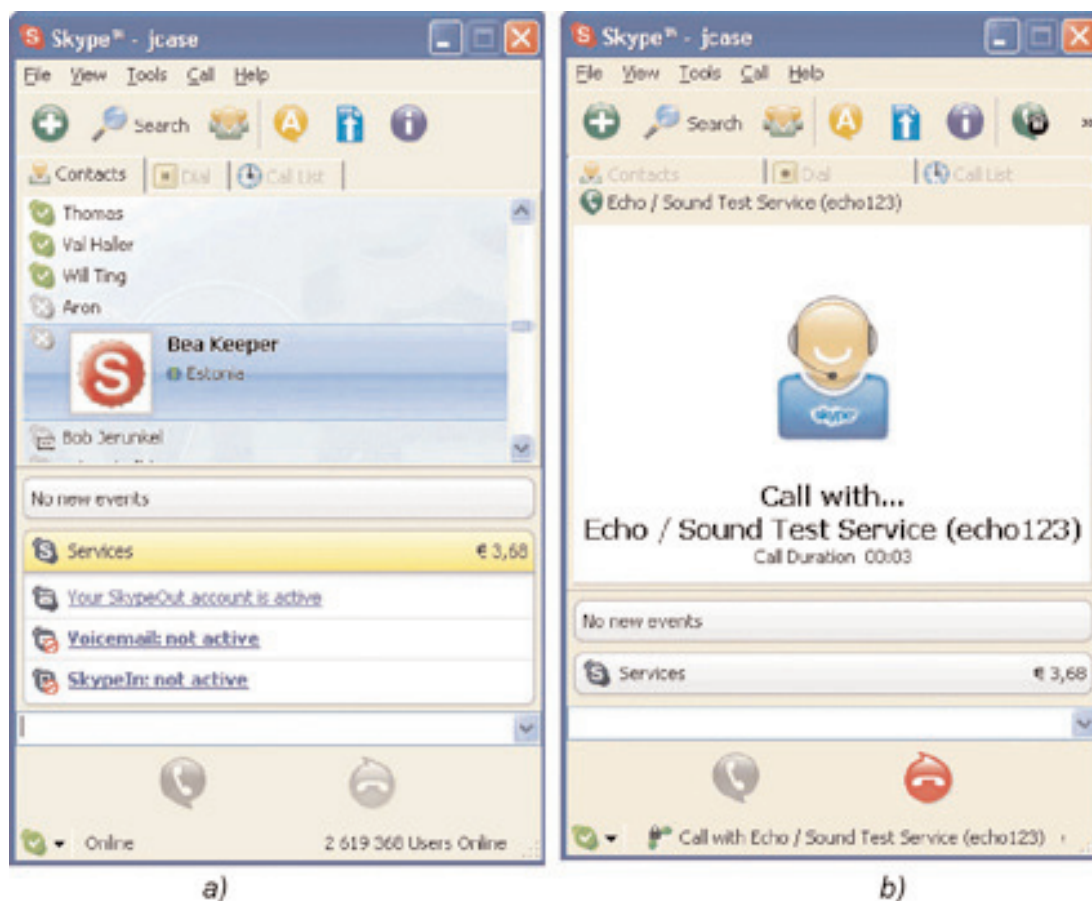


Figure 4 The Skype user interface. a) Contacts and Event, b) Talking

Skype is the fastest growing application in the history of the Internet. The downloading rate is shown in Figure 3. As other instant messaging (IM) applications, Skype has capabilities for voice-calls, instant messaging, audio conferencing, and buddy lists.

The Skype protocol is proprietary and encrypted and is thus not public knowledge. The Skype application has been studied in [6] where an attempt to decrypt its protocol was made. It was discovered that there are three types of nodes in the Skype overlay net-

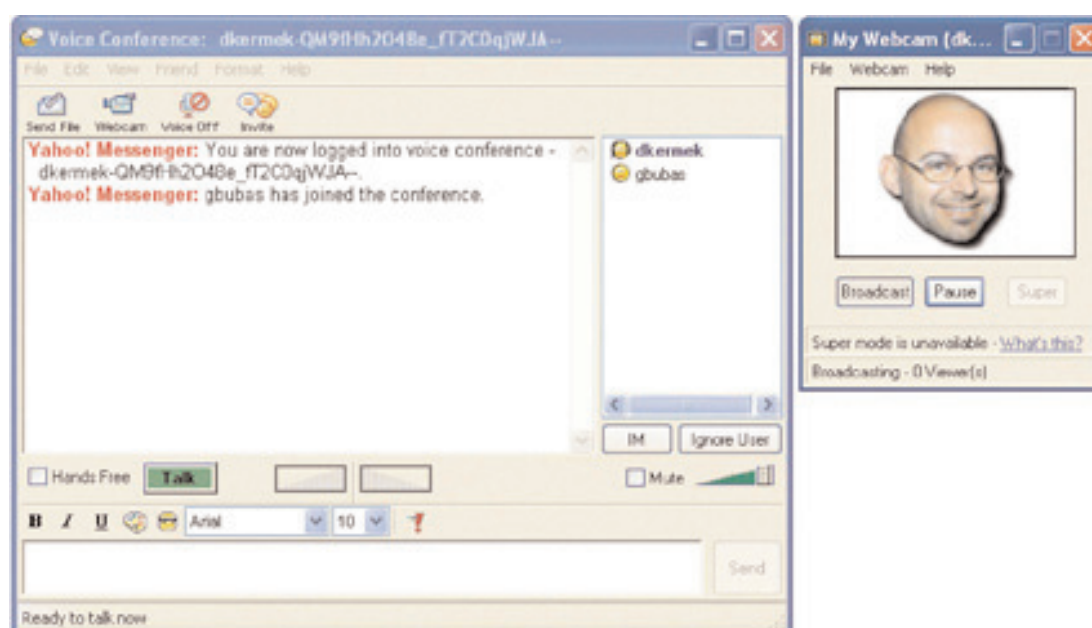


Figure 5 Yahoo Messenger user interface

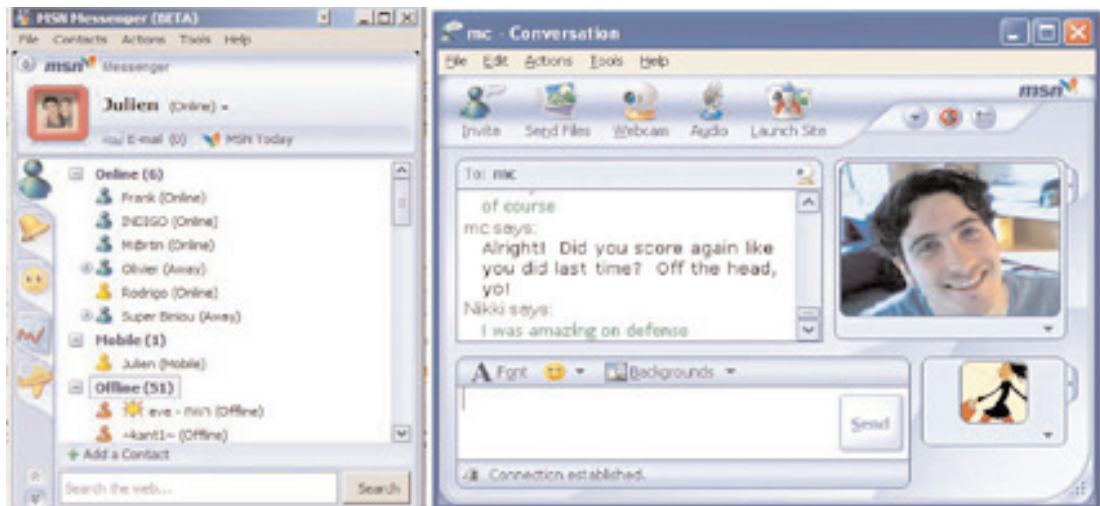


Figure 6 MSN Messenger user interface

work: ordinary hosts, supernodes and a login server where user names and passwords are authenticated and stored. An ordinary host must connect to a supernode to register with the Skype login server. All communications between any pair of Skype users consisting of any combination of voice, video, text chat or file transfer are carried over an encrypted Skype “session layer” that is established between the communicating users before messaging begins.

The Skype user interface is illustrated in Figure 4.

Yahoo Messenger

Yahoo Messenger is an instant messenger application provided by Yahoo and has been widely available for many years. It is heavily integrated with Yahoo’s other services, such as online shopping, web search, multi-user online games and email. In its latest release it has included a VoIP service. In earlier versions Yahoo Messenger also offered voice communication, but only between computers where only one of the users was located behind a NAT. Yahoo Messenger now offers VoIP also when both users are located behind NATs.

In June 2005 Yahoo acquired a VoIP company called DialPad. This company has technology for connecting VoIP calls on the Internet with the PSTN network. It is assumed that this technology will be integrated with Yahoo Messenger. The Yahoo Messenger user interface is illustrated in Figure 5.

MSN Messenger

MSN Messenger is Microsoft’s instant messaging client for Windows computers. The major use of the software is for instant messaging, although other features which now come as standard include support for

voice conversations, full screen audio video conversations, transferring files, and built-in multi-user online games.

The protocol used by MSN Messenger is closed, but it is common knowledge that the basic elements are as follows [7]:

Notification Server (NS): The connection to a notification server is the basis of an MSN Messenger session, as it handles the users’ presence information. If you are disconnected from the notification server, you are no longer online to your buddies. The main purpose of the notification server is to handle presence information about the user and the principals whose presence he has subscribed to.

Switchboard (SB): The switchboard handles instant messaging sessions between principals. In other words, each person in an MSN chat corresponds to a connection to a shared switchboard session. Invitations to other services such as file transfer and Net-Meeting are also sent and received through the SB.

In August 2005 Microsoft bought a small VoIP company called Teleo [8]. This company has a VoIP application that is able to connect to the public telephony network. Most likely this technology will be integrated with MSN Messenger, providing the same VoIP capabilities as Skype. Even though Skype has had a huge success, it is nowhere near MSN Messenger when it comes to the number of installations.

The MSN Messenger user interface is illustrated in Figure 6.

Google Talk

The newest contender on the IM/VoIP arena is Google. In August 2005 Google announced Google Talk, which is a small IM application that is integrated with Google Mail and offers VoIP communication between other Google Talk users. The Google Talk user interface is illustrated in Figure 7.

SIP in a peer-to-peer telephony setting

In order to realize end-to-end IP telephony, in terms of initiating and terminating calls, the IETF developed the *Session Initiation Protocol* (SIP) [9]. It allows for direct signalling between two end-users, where one user can call another by sending an “invite” message using a URI on the form `SIP:bill@online.no` (Figure 8). The message contains the IP address, protocol (typically RTP/UDP), port number and possible codecs preferred by the caller. The callee sends a reply, containing its own address, port number and the codec. When this negotiation is complete, the reply is finally acknowledged (Figure 8), and an RTP session is set up in each direction to carry the actual telephone conversation.

Although SIP is designed to work in a strict peer-to-peer setting (as shown in Figure 3), it is normally used in combination with a SIP server. This is shown in Figure 9. (Here, only the initial invite message and the resulting media session are shown.)

SIP fits well in a peer-to-peer setting, since it is designed for direct communication between the clients. First, all servers in SIP are optional. Second, even when a server such as a proxy server is utilized, the packets are routed on a peer-to-peer basis after the initial exchange. Although SIP is based on an inherent peer-to-peer architecture, most applications using SIP today register their IP addresses with the SIP server so that the other users can reach them. In fact, the use of SIP servers may also be used to realize certain forms of mobility.

Although SIP was originally designed with IP telephony in mind, it might in principle be used to set up all kinds of communication sessions, such as chat sessions, video conferencing, and so forth. Furthermore, instance messaging and presence service (e.g. in terms of buddy-lists) are features that are now available with SIP. Instant messaging and presence service are commonly implemented by a large number of different peer-to-peer applications. Hence, SIP is a very suitable candidate technology both for IP telephony and peer-to-peer networking.

Peer-to-Peer systems inherently have high robustness and fault tolerance, because the network is self-organizing without the use of centralized servers. A sensi-

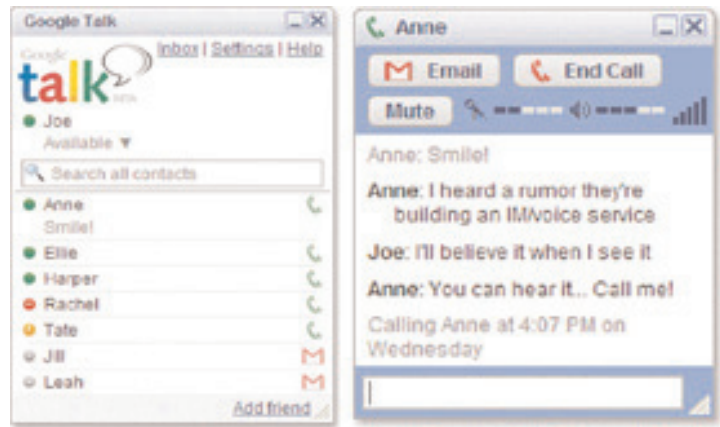


Figure 7 Google Talk user interface

ble approach might be a hybrid solution that will benefit from the scalability offered by centralized systems such as SIP and the reliability offered by P2P. A solution using P2P-technology in combination with SIP is suggested in [10][11]. Unlike a conventional SIP architecture, the P2P SIP system requires no central servers, and the peers connect directly to a few other peers, forming an overlay network. P2P enabled SIP nodes can then communicate with other P2P enabled SIP nodes to establish sessions.

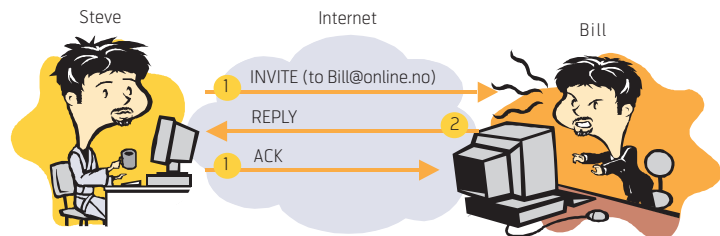


Figure 8 The SIP signaling (peer-to-peer)

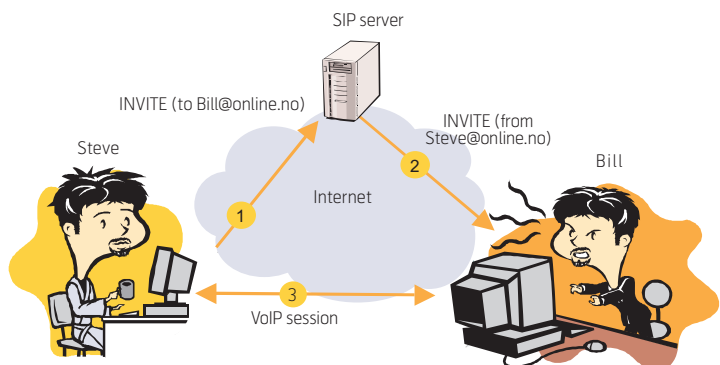


Figure 9 SIP signaling using a SIP server

A shortcoming of SIP is when both users are located behind a NAT, SIP cannot manage to establish the VoIP session without taking special measures to traverse the NAT. The problem of NAT traversal will be discussed in more detail later in the next chapter.

Enablers and showstoppers for peer-to-peer communication

Network Address Translation (NAT)

The P2P applications, SIP and others, all experience some problems when there is a NAT in the communication path. The operation of a NAT is to be an active unit placed in the data path, usually as a functional component of a border router or site gateway. A NAT intercepts all IP packets, and may forward the packets onward with or without alteration of the contents of the packets, or it may choose to discard the packets. The essential difference from a conventional router or a firewall is the ability of the NAT to alter an IP packet before forwarding it. NATs are similar to stateful firewalls, and different from routers, in that they are topologically sensitive. They have an “inside” and an “outside,” and undertake different operations on intercepted packets depending on whether the packet is going from inside to outside, or in the opposite direction.

The header of an IP packet contains the source and destination IP addresses. If the packet is being passed in the direction *from* the inside *to* the outside, a NAT rewrites the source address in the packet header to a different value, and alters the IP and TCP header checksums in the packet at the same time to reflect the change of the address field. When a packet is received *from* the outside destined *to* the inside, the

destination address is rewritten to a different value, and again the IP and TCP header checksums are recalculated. The “inside” does not use globally unique addresses to number every device within the network served by the NAT. The inside (or “local”) network may use addresses from private address blocks, implying that the uniqueness of the address holds only for the site. The operation of a NAT is illustrated in Figure 10.

A variant of the NAT is the *Port-Translating NAT*, or NATP. This form of NAT is used in the context of TCP and *User Datagram Protocol* (UDP) sessions, where the NAT maps the local source address and source port number to a public source address and a public-side port number for outgoing packets. Incoming packets addressed to this public address and port are translated to the corresponding local address and port.

There are many types of NATs, and different types of NATs require different solutions for NAT traversal. Generally, one can divide NATs into:

Symmetric NAT: Here the NAT mapping refers specifically to the connection between the local host address and port number and the destination address and port number and a binding of the local address and port to a public side address and port. This is the most restrictive form of NAT behavior under UDP, and it has been observed that this form of NAT behavior is becoming quite rare, because it prevents the operation of all forms of applications that undertake referral and handover.

Full-cone NAT: A full-cone NAT is the least restrictive form of NAT behavior. It implies that any remote

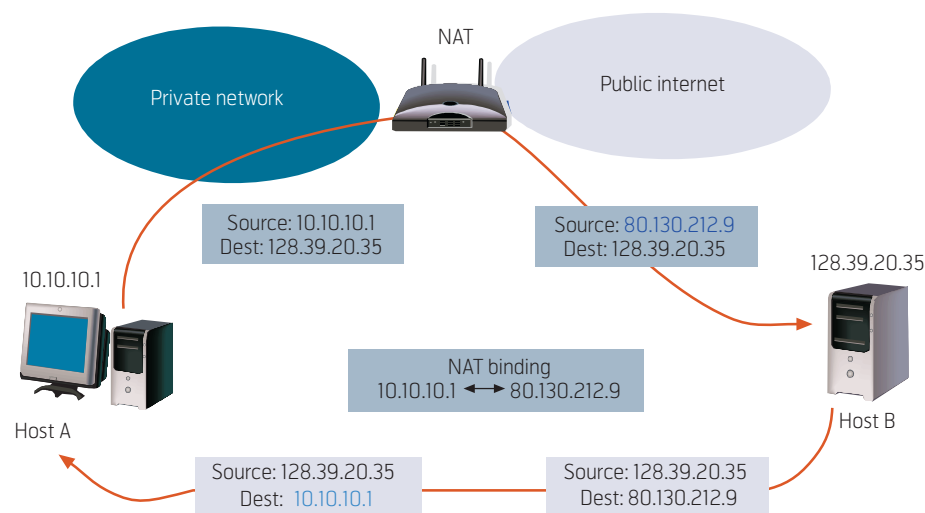


Figure 10 Operation of a NAT

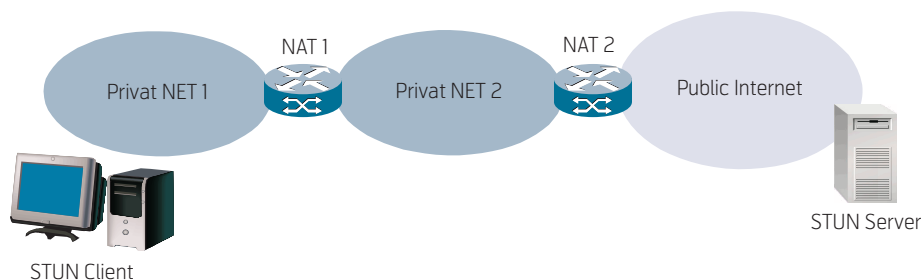


Figure 11 STUN Scenario

host on any remote port address can use the binding of a local address and port to a public-side address and port, when the binding has been established.

Restricted-cone NAT: A restricted-cone NAT is one where the NAT binding is accessible only by the destination host, although in this case the destination host can send packets from any port address after the binding is created.

Port-restricted-cone NAT: A port-restricted-cone NAT is one where the NAT binding is accessible by any remote host, although in this case the remote host must use the same source port address as the original port address that triggered the NAT binding.

Since the common use of IP was dominated by client/server communication, NATs were unfortunately designed under the assumption that no communication is peer-to-peer. These middleboxes assume that the server is located on the publicly routed Internet. However, as applications developed, it soon became clear that solutions for NAT traversal were necessary.

Simple Traversal of UDP through NATs (STUN)

For applications to be able to discover if they are behind a NAT, the *Simple Traversal of UDP through NATs* (STUN) protocol [12] was developed. STUN is a probe system that examines the interchange between a STUN client that may lie behind a NAT and a STUN server that is positioned on the public side of the NAT. The protocol is a simple UDP request-response protocol that uses embedded addresses in the data payload, and compares these addresses with header values in order to determine the type of NAT that may lie in the path between the client and server. The basic operation of STUN is to use a common request of the form: "Please tell me what public address and port values that were used to send this query to you." A typical STUN scenario is shown in Figure 11.

STUN attempts to determine the type of NAT by a structured sequence of requests and responses. This sequence is illustrated in Figure 12. The client sends an initial request to the STUN server. If the public address and port in the returned response are the same as the local address, then the client can conclude that there is no NAT in the path between the client and the server. If the values differ, the client can conclude that there is a NAT on the path. STUN then uses subsequent requests to determine the type of NAT.

Traversal Using Relay NAT (TURN)

STUN allows a client to obtain a transport address (and IP address and port) that may be useful for receiving packets from a peer. However, addresses obtained by STUN may not be usable by all peers. Those addresses work depending on the topological conditions of the network. Therefore, STUN by itself cannot provide a complete solution for NAT traversal.

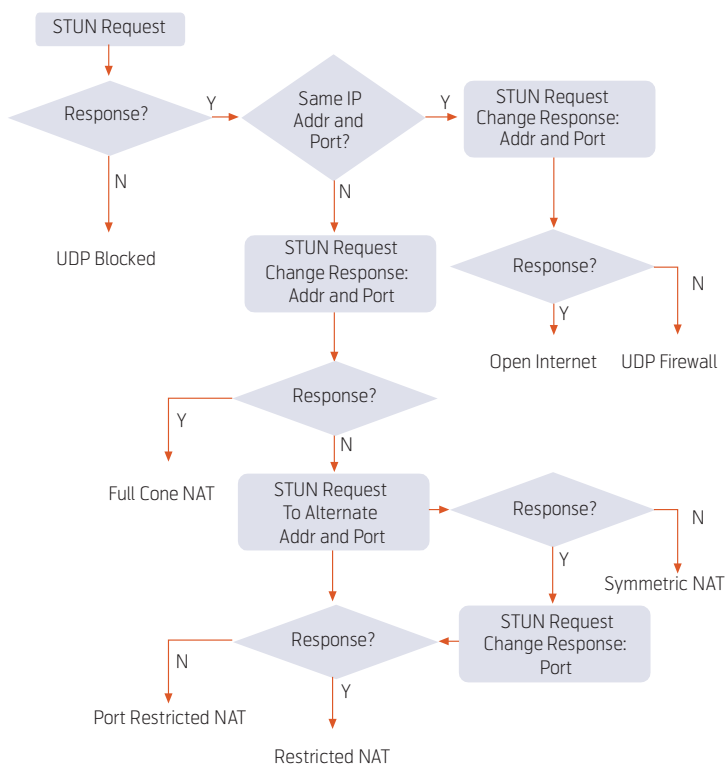


Figure 12 NAT discovery process using STUN

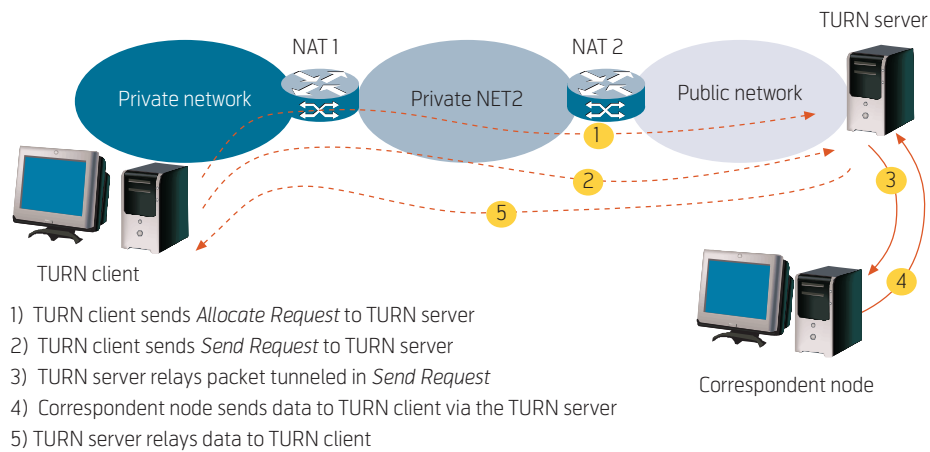


Figure 13 Operation of TURN

A complete solution requires that a client obtain a transport address that is commonly available from the public Internet. This can be accomplished by relaying incoming data to the client through a server that resides on the public Internet.

Traversal Using Relay NAT (TURN) [13] is a protocol that allows for an element behind a NAT or firewall to receive incoming data over TCP or UDP connections. It is most useful for elements behind symmetric NATs or firewalls that wish to be on the receiving end of a connection to a single peer. TURN does not allow for users to run servers on well-known ports if they are behind a NAT. It supports the connection of a user behind a NAT to only one single peer. In that regard, its role is to provide the same security functions as provided by symmetric NATs and firewalls, but to *turn* them into port-restricted NATs.

Unlike a STUN server, a TURN server provides resources to clients that connect to it. Therefore, only authorized clients can gain access to a TURN server. This requires that TURN requests are authenticated. TURN assumes the existence of a long-lived shared secret between the client and the TURN server in order to achieve this authentication. The client uses this long-lived shared secret to authenticate itself in a *Shared Secret Request*, sent over *Transport Layer Security (TLS)*. The *Shared Secret Response* provides the client with a one-time username and password. The TURN protocol is illustrated in Figure 13.

NAT Traversal of peer-to-peer IP telephony

Skype provides a solution for NAT and firewall traversal, and it is assumed that the Skype client uses a variation of the STUN and TURN protocols to determine the type of NATs and firewalls it is behind.

In [6] it was found that the Skype client is initialized with a set of bootstrap supernodes. These are stored in a host cache and contain IP-addresses / port pairs of active supernodes. A Skype client will learn the IP address of a login server from one of the supernodes, and will attempt to login there. It is not known whether it is the supernodes or the login server that act as a STUN/TURN server. It might even be some of the ordinary nodes themselves. It is shown in [6] that when the caller and the callee are both behind port restricted NATs, the traffic is relayed by another online Skype node.

It is well known that Yahoo and MSN Messenger are also using a STUN/TURN alike technology.

IPv6 as the future of peer-to-peer communication

NAT boxes are rapidly being deployed in combination with IEEE 802.11 wireless technology. As more and more peer nodes are located behind a NAT, the number of available peers with a public IP address will decrease. If the rate of new Skype users behind a NAT becomes higher than the rate of new Skype users with a public IP address, peer-to-peer telephony services such as Skype will be forced to deploy relay servers to maintain the same level of availability. This will transform the peer-to-peer architecture of Skype into a traditional client/server architecture. This new infrastructure will come with a price tag in the form of server boxes, bandwidth, maintenance and management. This is also the case for Yahoo and MSN Messenger.

Since the IPv4-based Internet is no longer peer-to-peer, there will be a need for middleboxes that bypass problems caused by NATs and firewalls. Some of the problems peer-to-peer services will face caused by this are:

- STUN and TURN servers are single point of failures in a network.
- STUN/TURN servers need maintenance and management
- STUN/TURN servers are vulnerable to DoS attacks
- STUN and TURN workarounds are per-application solutions. Since applications are not modular, every application will need to have its own solution for NAT and firewall traversal.

One commonly held belief is that the deployment of IPv6 [12] will eliminate the problem of NATs within the Internet. There will be plentiful public IPv6 address space available and no particular reason to deploy NATs in an IPv6 realm. That does not say that IPv6 NATs will not be implemented, nor used. Indeed IPv6 NATs are already available, and they are being used, albeit to a small extent. NATs are, rightly or wrongly, considered to be part of a security solution for a site because of their filtering properties that prevent incoming packets from entering the site. So it is reasonable to predict that some use of NAT will be seen in IPv6, although the level of IPv6 NAT will probably not be anywhere near that of NAT in IPv4.

IPv6 provides the necessary address space so that NATs are unnecessary. Hence, by converting to IPv6, the client/server communication paradigm that has been forced upon us by the deployment of NATs will evaporate. One of the most compelling consequences of a transition to IPv6 is that the simple and powerful Internet model is restored. Not only will IPv6 allow for the end-to-end principle in Internet communication. First and foremost, it will restore the peer-to-peer communication paradigm on the Internet. This will pave the way for a series of peer-to-peer services, which will for certain include voice communication. When IPv6 is deployed, Skype, MSN and Yahoo can easily be upgraded to support IPv6. IPv6 will be able to reduce the complexity of these VoIP applications, since there will no longer be any need for STUN/TURN solutions. The need for relaying of VoIP sessions will also be reduced to a minimum, since all users will be able to communicate peer-to-peer.

IPv6 will also reduce the complexity of developing peer-to-peer applications and services. Hopefully this will stimulate the development of other peer-to-peer voice services. Microsoft has stated that they see IPv6 as an enabler for peer-to-peer services [15] and are already supporting IPv6 in all their latest operating systems. They also provide APIs and development kits for P2P applications/services [15].

Concluding remarks

There is no doubt that 'Voice over IP' telephony is growing rapidly, and it is equally certain that it will grow even faster as the solutions continue to develop and as ever-expanding populations of potential users come into contact with the concept. Skype has pretty much dominated the VoIP scene the last year or so, as IP telephony finally hit the Internet. However, Skype has the following properties that may cause problems in the future:

- The protocol is proprietary, unlike open standards such as SIP.
- It provides a single service that makes calls or instant messages and does not provide an architecture for new services.
- It depends on the availability of public IPv4 addresses

Skype's VoIP technology is based on software from the company Global IP Sound (GIPS) [16]. This software is also available to others. Other VoIP applications supporting this technology are Google Talk [17] and MSN Messenger [18]. Gizmo [19] is another free VoIP application based on GIPS, and while Skype is proprietary, Gizmo has been built using an open source philosophy around the emerging SIP standards. In addition to being based on the SIP open standard, Gizmo is also committed to interconnecting its IP telephony system with those operated by other organizations. As more VoIP applications become available, Skype will face serious challenges, especially as SIP matures.

All peer-to-peer VoIP services will, however, have to deal with NATs. Despite their shortcomings and despite the problems NATs create for numerous applications, many NATs are deployed in the IPv4 realm. However, it is highly unlikely that a NAT-based architecture will scale to support peer-to-peer services, and IPv6 is therefore a prerequisite for a continued growth of peer-to-peer communication.

References

- 1 Egevang, K, Francis, P. *The IP Network Address Translator (NAT)*. May 1994. (RFC 1631)
- 2 Srisuresh, P, Egevang, K. *Traditional IP Network Address Translator (Traditional NAT)*. January 2001. (RFC 3022)
- 3 *The Gnutella Protocol*. March 10, 2006 [online] – URL: http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

- 4 *FastTrack Protocol*. March 10, 2006 [online] – URL: <http://en.wikipedia.org/wiki/FastTrack>
- 5 *The Skype VoIP application*. March 10, 2006 [online] – URL: www.skype.com
- 6 Baset, S A, Schulzrinne, H. *An Analysis of the Skype Peer-to-peer Internet Telephony Protocol*. March 10, 2006 [online] – URL: <http://arxiv.org/ftp/cs/papers/0412/0412017.pdf>
- 7 *MSN Messenger Protocol*. April 27, 2006 [online] – URL: <http://www.hypothetic.org/docs/msn/index.php>
- 8 *MSN Teleo*. March 10, 2006 [online] – URL: <http://teleo.msn.com/>
- 9 Rosenberg, J, Schulzrinne, H et al. *SIP: Session Initiation Protocol*. June 2002. (RFC3261)
- 10 Bryan, D, Jennings, C. *A P2P Approach to SIP Registration*. [online] – URL: <http://www.ietf.org/internet-drafts/draft-bryan-sipping-p2p-02.txt> (work in progress, published 5 March 2006)
- 11 Johnston, A. *SIP, P2P, and Internet Communications*. [online] – URL: <http://www.ietf.org/internet-drafts/draft-johnston-sipping-p2p-ipcom-02.txt> (work in progress, published March 5, 2006)
- 12 Rosenberg, J, Weinberger, J, Huitema, C, Mahy, R. *STUN – Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*. March 2003. (RFC 3489)
- 13 Rosenberg, J, Mahy, R, Huitema, C. Obtaining Relay Addresses from Simple Traversal of UDP Through NAT (STUN). [online] – URL: <http://www.ietf.org/internet-drafts/draft-ietf-behave-turn-00.txt> (published February 27, 2006)
- 14 Deering, S, Hinden, R. *Internet Protocol, Version 6 (IPv6) Specification*. December 1998. (RFC2460)
- 15 *Windows Peer-to-Peer Networking*. March 10, 2006 [online] – URL: <http://www.microsoft.com/windowsxp/p2p/default.aspx>
- 16 *Global IP Sound*. March 10, 2006 [online] – URL: <http://www.globalipsound.com/>
- 17 *Voice codecs in Google Talk*. March 10, 2006 [online] – URL: <http://www.google.com/talk/developer.html#codecs>
- 18 *Microsoft and Global IP Sound Partner*. March 10, 2006 [online] – URL: <http://blog.tmcnet.com/blog/tom-keating/voip/microsoft-and-global-ip-sound-partner.asp>
- 19 *The Gizmo Project*. March 10, 2006 [online] – URL: <http://www.gizmoproject.com/>

Paal E. Engelstad completed his PhD on resource discovery in Mobile Ad hoc and Personal Area Networks in 2005. He has also a Bachelor and Masters degree (Honours with Distinction) in Applied Physics from NTNU, Norway, and a Bachelor degree in Computer Science from University of Oslo, Norway. After working five years in industry, he joined Telenor R&D where he focuses on IETF and IP technology (e.g. IP mobility, IPv6, QoS, MANET and AAA-issues) and IEEE wireless technologies (e.g. 802.11, 802.15 and 802.16). Paal Engelstad has published 28 refereed papers and holds three patents (two pending).

email: Paal.Engelstad@telenor.com

Geir Egeland holds a B.Eng (Hons) from the University of Bristol and has for the last ten years worked as a research scientist in the field of mobile networks. He is currently with Telenor R&D where his work is mainly focused on mobility for IP networks, with a particular emphasis on MANET and IPv6. Geir Egeland was formerly employed by the Norwegian Defence Research Establishment (NDRE) as a research scientist working on design and analysis of MAC and routing protocols for mobile ad hoc network.

email: geir.egeland@telenor.com