Build Mobile Phone Connectivity Over IP Networks

Harri Kukkonen
Helsinki University of Technology
Harri.Kukkonen@tkk.fi

Abstract

Connecting two mobile phones through IP networks is challenging since mobile phones usually don't have a specific IPv4 address and connected through NATs and firewalls. Various methods can be used to build up connections for mobile devices over IP networks, such as Teredo, IPsec tunneling, OpenVPN, ICE, STUN and TURN. All those methods have their advantages and disadvantages, this paper evaluates those methods and tries to identify the best approaches to build data connections for mobile phone devices over IP networks.

KEYWORDS: mobile, connectivity, nat traversal, teredo, ipsec, openvpn, ice, stun, turn

1 Introduction

Mobile phones are usually connected to the Internet using IPv4 protocol through various systems like Network Address Translators (NAT) [18] and firewalls. Usually NAT prevents new TCP or UDP connections from being opened towards the client behind NAT, which in turn makes it impossible for two mobile nodes to connect each other. Although firewalls are not as restricting, the end result is most of the time the same as with NAT.

To solve this problem several issues need to be addressed, most import of which is connecting to the client from Internet through NAT. Methods that are used to accomplish this are usually referred to as NAT Traversal. Multiple different techiques exist, many of which require a public server between the clients to assist in creating a connection. One of such methods, Traversal Using Relay NAT (TURN) [14], employs a public server to relay all traffic between the two clients. Another technique is Session Traversal Utilities for NAT (STUN) [15], which utilizes a public server to find the clients external addresses that are visible to the Internet. The public server is also used to open ports in NAT for UDP communication. Clients can then use their external addresses and the open ports for communication with other clients.

Teredo [6] is a complete solution that encapsulates IPv6 traffic inside the IPv4 UDP protocol so it can be routed through NATs and the IPv4 Internet. In addition to the normal IPv4 address, globally routable IPv6 addresses are also assigned for the Teredo hosts. The IPv6 addresses are then used for communication with other Teredo hosts or normal native IPv6 hosts. For NAT Traversal, Teredo uses a similar method to STUN i.e. by employing a public server to determine the NAT type and open UDP ports for communication between clients. Maintaining these servers does not require much bandwidth since they only forward ICMPv6 packets between the clients. Teredo relays, however, require lot of network bandwidth. Teredo hosts use the nearest relay to forward all their IPv6 traffic to other Teredo or IPv6 hosts over UDP IPv4. Relays are also utilized to receive packets from other Teredo or IPv6 hosts over IPv4.

Another technique is Interactive Connectivity Establishment (ICE) [13], which focuses mainly on getting UDP-based media streams such as VoIP, video or instant messaging to traverse through NATs. ICE utilizes both STUN and TURN to find out if the clients are behind NATs and with which IP addresses and ports connections between the clients could be established.

IPsec Tunneling [9] is used to secure Virtual Private Network (VPN) [17] connections to a safe environment where nodes can communicate to each other more easily. When creating secure VPN connections IPsec is used in a tunnel mode in which entire IP packets, data and header, are encrypted and encapsulated into new IP packets with new headers. NAT Traversal is implemented using Nat Traversal in the IKE (NAT-T) [10] that encapsulates the IPsec packet with another UDP and IP headers to protect the original IPsec header from being modified by the NAT.

OpenVPN is an alternative to IPsec tunneling and provides a complete package for creating secure VPN connections [1]. It uses OpenSSL library for all encryption and authentication functionality. OpenVPN prefers running connections over UDP, which is also the default choice, but also TCP is supported. It works by multiplexing all connections to a single TCP or UDP port and is capable of passing through most HTTP Proxies and doing NAT traversal.

2 Teredo

Teredo consists of several different components, namely clients, servers, relays and host-specific relays. Clients are IPv4 or IPv6 hosts, the endpoints of communication, that support the Teredo interface and communicate to other Teredo clients or non-Teredo nodes in the IPv6 network. Servers assist clients in communication by providing them an address prefix that is used to configure a Teredo based IPv6 address; they also facilitate connection creation of Teredo clients to other Teredo clients or IPv6 hosts. Servers are nodes that have addresses both in IPv4 Internet and IPv6 Internet and listen for Teredo traffic on UDP port 3544. Teredo relays route the traffic between Teredo clients in IPv4 network and IPv6 hosts and listen for Teredo traffic on UDP port 3544 in the same way as Teredo servers. Teredo host-

specific relays run on the same host as the Teredo client and are used when the client in the IPv4 network communicates with an IPv6 host that also has IPv4 connectivity. In this case all traffic can stay in the IPv4 network without the need to use external Teredo relays. IPv6 addresses assigned to clients in Teredo have much information encoded within. Total length of the address is 128-bits and it is divided in the following way.

- 32-bits, Teredo prefix, 2001:0000::/32
- 32-bits, IPv4 address of the clients Teredo server.
- 16-bits, Flags, may describe the type of NAT server used.
- 16-bits, UDP port used by the NAT server.
- 32-bits, NAT server's public address.

UDP port and the IPv4 address of the NAT server are used in connecting to the client located behind the NAT server from the Internet.

For connection initialization and upkeep, Teredo nodes use 'bubble packets' which contain IPv4 header (20 bytes), UDP header (8 bytes) and IPv6 header (40 bytes) without any payload. The actual traffic payload is wrapped inside the same headers, which creates in total 68-bytes of overhead for each packet when using Teredo.

2.1 Connection initialization

All Teredo clients first utilize the Teredo server to determine what kind of NAT server is between them and the Internet by using a qualification prodecure that is a simplified version of STUN. The ability for the server to contact clients at any time is maintained by sending regularly, approximately every 30 seconds, bubble packets to the server, which then sends a bubble packet as a reply thus ensuring that UDP packets can be sent using the same port back to the client through NAT.

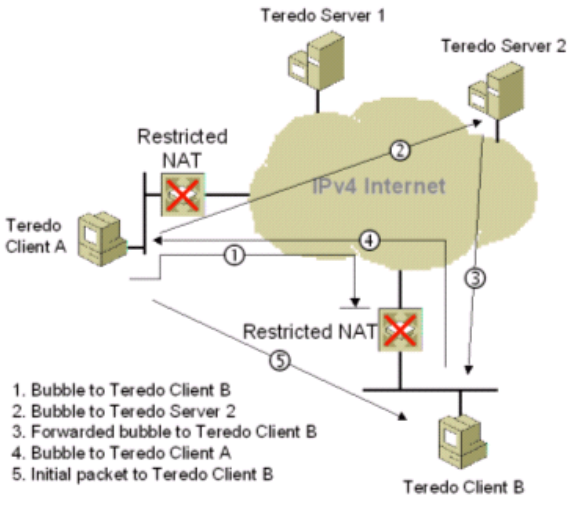

Figure 1: Teredo connection initialization, source Microsoft TechNet[12]

Complexity and details of connection initiation between clients depends on the type of NATs used and whether the clients are located in IPv4 or IPv6 Internet. A similar principle does, however, apply in most cases so a case between two clients behind restricted NAT systems initiating communication is described here as an example (See Figure 1).

When client A wants to connect to client B, it first tries to send a bubble packet directly to B but the NAT in front of B blocks the packet. This does however open the NAT in front of client A so client B can subsequently send a reply. In the next step client A uses client B's Teredo server, whose address is determined from client B's Teredo IPv6 address, to forward a bubble packet to client B, which goes through B's NAT. Client B replies directly to client A, which creates an opening in client B's NAT, and finally the clients are able to communicate directly to each other.

3 IPsec tunneling

IPsec is to encrypt and/or authenticate transferred data. If only authentication is required IPsec encapsulates the data payload inside an AH (Authentication Header). When it is necessary to also encrypt the payload data, ESP (Encapsulating Security Payload) is utilized. Encapsulating data inside new headers causes overhead, 24 bytes for AH and 24 to 40 bytes when using ESP. Additionally tunnel mode adds 20 bytes of headers on top of everything else. Using encryption requires lot of calculation power, which is an important factor when considering suitability for mobile use.

IPsec requires the use of IKEv2 (Internet Key Exchange Protocol) [8] protocol when creating new connections between clients. IKEv2 establishes shared authentication keys and security parameters called SA (Security Association) so a secure encrypted IPsec connection can be formed between the clients. Cryptographic calculations required by this introduce some need for additional computational power, which results in bigger power consumption.

3.1 Connection initialization

When negotiating a new connection, NAT-T first determines if both clients can perform NAT-T and if there are NAT servers between them. If both are true, NAT-T is used before normal negotiation is performed.

The NAT Traversal algorithm used in IPsec, NAT-T [11], encapsulates the ESP header with another UDP header using the same ports as IKE. This enables the original IPsec headers to pass unchanged through NAT.

In normal connection negotiation IKEv2 establishes the connection by sending a `IKE_SA_INIT` message and waiting for a reply. The first message negotiates used cryptographic algorithms, exchanges nonces and does Diffie-Hellman key exchange so further communication can be encrypted. The second message pair `IKE_AUTH` authenticates the first message pair, exhcanges identities and certificates and a child SA is established.

4 OpenVPN

Like IPsec, OpenVPN provides a secure tunnel for network traffic. There are however some fundamental differences.

Security layer in OpenVPN is located in the application layer, implemented using OpenSSL [19], rather than ip layer as with IPsec. This gives the possibility to use all of the different ciphers implemented in OpenSSL as the cryptographic algorithm, also authentication is handled by OpenSSL. Authentication can be done using pre-shared secret key, certificates or username and password although non-encrypted tunnels are also an option.

Traffic inside OpenVPN is usually encapsulated inside UDP packets, TCP may also be used but it is not recommended. There are several reasons why UDP should be preferred, most important being the lower network overhead when using UDP. Encapsulating TCP packets inside TCP packets may also cause problems because the congestion control protocol is applied for both streams. In a case of large amount of packet loss their efforts to correct the problem can cascade and break the connection altogether.

Traffic overhead caused by OpenVPN depends on configuration, but UDP encapsulation without encryption causes on average 51 bytes of overhead. Encryption adds more to this but the exact amount is dependant on the used cipher [3].

4.1 NAT Traversal

NAT Traversal in OpenVPN is implemented by hole punching [2], either with UDP or TCP depending on the encapsulation used, which does not support symmetric NAT's. UDP hole punching, which works in a similar fashion as Teredo's NAT traversal, is presented here as an example.

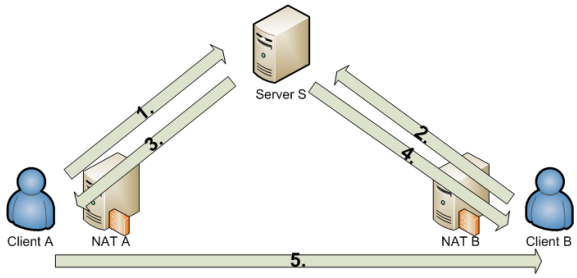

Figure 2: UDP Hole Punching

1. Client A sends a packet to server S which creates an opening in NAT A for communication back to client A.
2. Client B sends a packet to server S which creates an opening in NAT B for communication back to client B.
3. Server S sends client A client B's port information.
4. Server S sends client B client A's port information.
5. Client A can now communicate directly with client B and vice versa.

5 ICE, STUN and TURN

5.1 STUN and TURN

In STUN an external STUN server located in the open Internet is requested by the client A to determine what type of NAT server is between it and the Internet. The STUN server does this by sending series of UDP packets to the client and observing the changes NAT servers make to the source address of the reply packet. Based on the nature of these changes STUN can determine NAT server's type and the address and port which can be used to send UDP packets to client A from Internet. Information is then forwarded back to the client. STUN does have some limitations, it does not work with symmetric NATs and the connection address and port determined for the client might not work for all other clients trying to connect client A, depending on the network topology.

TURN also requires the use of external servers in the open Internet. Bandwidth requirements are however much steeper than with STUN, as the TURN server functions as a relay between two clients transferring all traffic through the server.

5.2 ICE

ICE provides NAT Traversal for media-stream protocols that are based on offer-answer methodology, such as SIP [16] which is used as the example case here. ICE is based on using multiple methods to achieve connectivity between two clients, most important being STUN and TURN.

Step 1. Address gathering

All possible addresses and ports that can be used to create a connection to a client are gathered. All locally defined addresses clients have, like the ip addresses of network interfaces or the address acquired from an existing VPN connection, are stored. Then STUN is used to gather addresses and ports that can be used to get through NAT servers, same is done using TURN, which uses external relay servers to achieve connectivity through NAT.

Step 2. Priorization

Client prioritizes the gathered addresses for optimal latency and bandwidth usage in candidates. More direct routes such as local or VPN interfaces are usually preferred over using STUN. Relayed addresses like those collected using TURN are set as last.

Step 3. Encoding

Extra information is added and encoded into the SIP request called SDP, such as the preferred candidate addresses and ports for connection for each different media stream.

Step 4. Offer and answer

The SIP request is sent to a client to start a call, if the target client also supports ICE it performs the same three steps and sends its own candidate addresses and ports back to the caller.

Step 5. Checking

When the caller and target client have exchanged their SDP's they construct pairs of the other client's and their own candidate addresses for each media stream creating all possible combinations. Priority for each pair is then calculated by combining the priority of both addresses. After priorization each pair is tested for connectivity using a STUN transaction. Because the number of address combinations grows exponentially testing is done in priority order and sequentially to avoid flooding the network. When a working combination is found testing of connections is stopped.

Step 6. Completing

Found working address pair is usually the one with highest priority because of the sequential testing. Final check is then made, usually initiated by the caller, to confirm that the found address pair is the same for both. After this the called client can ring the phone to start the call.

Doing this many checks introduces a delay before a call is started, but eliminates for example ghost calls that could happen with other methods when connection was tried only after ringing. If connecting then failed no sound was transferred. Additionally the checking is complex only when connecting using direct means does not succeed, as those are checked first in the best case the delay before starting the call caused by ICE is minimal.

6 Comparison of methods

All of the methods analyzed in this paper are primarily meant for communication between two clients, so each point-to-point connection requires a new tunnel. OpenVPN is the exception, it can be used in bridge mode to connect two separate LANs together. But when considering mobile use and especially VoIP communication the bridge mode does not give OpenVPN an advantage over the others.

All of the methods support IPv6 communication, although Teredo's implementation seems to be most advanced in this group. Teredo's IPv6 support does have a drawback. A Teredo relay is needed if one of the clients is in IPv4 Internet and other in IPv6 Internet. Additionally because the relay has to transmit all traffic, not just facilicate connection establishment, the bandwidth requirements are very high.

Teredo, IPsec tunneling and OpenVPN have similar feature sets when it comes to NAT Traversal. Only symmetric NAT's are not supported. ICE offers also symmetric NAT support through the use of TURN. Implementation and complexity does differ between the methods, OpenVPN and IPsec tunneling implement their NAT Traversal without the need for external servers. Teredo needs an external Teredo server and ICE requires STUN servers and TURN servers. There is a method to enable symmetric NAT support also in Teredo, called SymTeredo [5]. However, the method remains on theoretical level, currently there are no practical implementations of SymTeredo.

Teredo server and STUN servers have low bandwidth requirements and are able to serve a large number of clients at the same time, so this is not an overly large drawback. Both TURN servers and Teredo relays require lot of bandwidth as they transmit all traffic between clients, but it is the only way to enable connections for clients behind symmetric NAT's.

All of the methods introduce some level of network overhead. Teredo and IPsec are the heaviest protocols, OpenVPN is slightly better and ICE has the lowest network overhead.

All except Teredo support some form of encryption for the encapsulated data. OpenVPN and IPsec support the best encryption ciphers and key management with ICE taking a lighter approach. All forms of encryption do increase network overhead and especially computational overhead, particularly the encryption methods used by OpenVPN and IPsec. The lack of encryption is therefore not a big problem because we are targeting mobile use. In addition, encryption brings with it the need for key exchange, which complicates and lengthens the problematic initial connection establishment even more.

7 Conclusion

When the theoretical and practical benefits and disadvantages of these various methods are compared ICE seems to be the best candidate for use with mobile VoIP traffic as it is specially designed for UDP media streams. It is the only one that can traverse through every kind of NAT server between clients and is also the lightest protocol of the analyzed protocols. It also has a relatively mature implementation stack for various platforms thus supporting it does not require extraordinary amount of effort. Teredo is also a good choice, however it does not bring any improvement over ICE and has higher network overhead. IPsec is still complicated to set up and isn't the best for this sort of ad-hoc connectivity. The same applies for OpenVPN: the protocols are designed more for static configurations and complete security than lightness and flexibility.

References

[1] M. Feilner. *OpenVPN: Building and Integrating Virtual Private Networks*. Packt Publishing, 2006.

[2] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators.

[3] M. Hall. Performance Analysis of OpenVPN on a Consumer Grade Router.

[4] H. Hansen. IPsec and Mobile-IP in Mobile Ad hoc Networking. *Department of Computer Science, Helsinki University of Technology, http://www. hut. fi/hansen/papers/adhoc*.

[5] S.-M. Huang, Q. Wu, and Y.-B. Lin. Enhancing teredo ipv6 tunneling to traverse the symmetric nat. *Communications Letters, IEEE*, 10(5):408–410, may 2006.

[6] C. Huitema. Teredo: Tunneling IPv6 over UDP through network address translations (NATs). Technical report, RFC 4380, February 2006, 2006.

[7] N. Kang, L. Lo Iacono, C. Rulan, and Y. Kim. Efficient application of IPsec VPNs in wireless networks. In *Wireless Pervasive Computing, 2006 1st International Symposium on*, Jan. 2006.

[8] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), Dec. 2005. Updated by RFC 5282.

[9] S. Kent and K. Seo. RFC 4301: Security architecture for the Internet protocol. *The Internet Society, Standards Track*, 2005.

[10] T. Kivinen, B. Swander, A. Huttunen, and V. Volpe. Negotiation of NAT-Traversal in the IKE. Technical report, RFC 3947, January 2005, 2005.

[11] T. Kivinen, B. Swander, A. Huttunen, and V. Volpe. Negotiation of NAT-Traversal in the IKE. RFC 3947 (Proposed Standard), Jan. 2005.

[12] Microsoft. Teredo Overview. *World Wide Web, http://technet.microsoft.com/en-us/library/bb457011.aspx, Last visited 17.3.2010*.

[13] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. Technical report, 2007.

[14] J. Rosenberg, R. Mahy, and P. Matthews. Traversal Using Relays around NAT (TURN). Technical report, 2009.

[15] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard), Oct. 2008.

[16] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630.

[17] C. Scott, P. Wolfe, and M. Erwin. *Virtual private networks*. O'Reilly Media, Inc., 1999.

[18] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), Jan. 2001.

[19] E. Young, T. Hudson, and R. Engelschall. OpenSSL. *World Wide Web, http://www.openssl.org/, Last visited 17.3.2010*.

[20] J. Zao and M. Condell. Use of IPSec in mobile IP. *November 1997*, 10:381–390.