



NAT Traversal for VoIP and Internet Communications using STUN, TURN and ICE

Eyeball AnyFirewall™ Technology White Paper

Table of Contents

1	Introduction	1
2	The NAT and Firewall Traversal Problem for VoIP	1
2.1	The NAT/Firewall Traversal Challenge	2
2.2	NAT Traversal Solution Requirements	4
3	Solving the NAT Traversal Problem	5
4	How ICE Methodology Works	6
5	Eyeball AnyFirewall™ Technology	9
5.1	Eyeball AnyFirewall Engine	10
5.2	Eyeball AnyFirewall Server	11
5.3	VoIP Call Example Using the AnyFirewall Engine	12
6	Conclusions	13
7	About Eyeball Networks	14
8	References	14

Last Modified: 2012

Copyright © 2001-2012 Eyeball Networks Inc. Patented and patents pending. All rights reserved.

Eyeball, Eyeball.com, its logos, AnyBandwidth™ Technology and AnyFirewall™ Technology are trademarks of Eyeball Networks Inc. All other referenced companies and product names may or may not be trademarks of their respective owners.

1 Introduction

Increased penetration of broadband Internet is driving the adoption of voice over Internet Protocol (VoIP) both for consumer and business markets, and VoIP is now on its way to becoming the main mode of multimedia communications in the coming years. High quality multimedia communications along with rich presence, universal mobility and availability, and low cost are some of the benefits VoIP brings to end-users. For operators, it promises new revenues from new and converged services, service bundling, increased customer loyalty, and lower capital and operation expenses (capex/opex) by building and running a single IP-based network for all communications services.

Ironically, the main driving force behind VoIP adoption also poses one of the biggest challenges – VoIP calls do not work well in many broadband situations. In short, the problem is as follows:

- More than 90% of PCs or end-devices access the broadband service using private IP addresses. These private IP addresses get mapped into real Internet addresses using a mechanism called Network Address Translation (NAT), which is implemented in all broadband access devices (also called broadband routers) and sometimes again in the service provider network.
- Most users have one or more packet-filtering firewalls to protect them from hacker attacks or malicious users on the Internet. Firewall features are implemented in most broadband routers these days and also in operating systems such as Windows XP and Vista.
- Most VoIP solutions in the market do not work well through NATs and firewalls – callers may sometimes fail to connect to each other or the quality and performance may be unacceptable.

In this white-paper we investigate the root-causes and challenges for the NAT/firewall traversal problem for VoIP, summarize recent work and progress made by standards-bodies and the industry on NAT traversal, and finally, present a comprehensive solution to this problem comprising a client-side SDK and a scalable carrier-grade server.

2 The NAT and Firewall Traversal Problem for VoIP

Figure (1) illustrates the NAT and firewall traversal problem for VoIP. Home users connect to the Internet using broadband routers from various vendors, such as Belkin, D-Link, LinkSys and NetGear. Users connect using wireless hotspots at Internet cafes or hotels, which also use routers from vendors, such as NetGear and SonicWall. Also, business users connect to the Internet using firewall products such as Cisco PIX, Juniper NetScreen, and CheckPoint FW1. Businesses sometimes also set up web-proxies such as Squid or Microsoft ISA for accessing the Web by their employees. Now suppose all the users behind all of these NAT and firewall devices/solutions want to make VoIP calls between each other. Will all of these calls work using traditional VoIP technologies? The answer is no. Most VoIP calls will not work through these NATs and firewalls. This is referred to as the NAT and firewall traversal problem – or simply the NAT traversal problem.

Recently, there have been a lot of mobile phone products with Wi-Fi VoIP features (single-mode Wi-Fi or dual mode cellular plus Wi-Fi). According to Infonetics Research, the number of Wi-Fi phones will double or triple each year until 2009 reaching a worldwide market of \$3.7 billion [12]. Now suppose a lot of users have these Wi-Fi phones, and they move around with these phones to use them from wherever they are - at homes, Internet cafes and offices. They will not only face the same NAT traversal problem, but worse, their connections and thus their network configurations, may change frequently. This further emphasizes the NAT traversal problem as it will severely limit user capability to communicate.

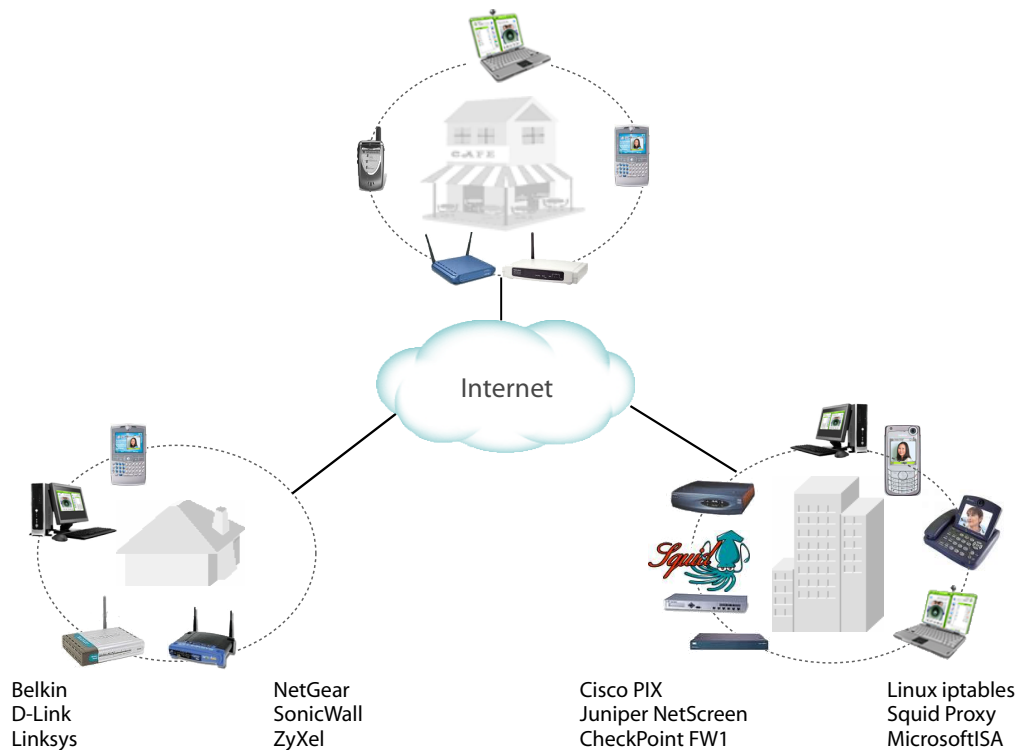


Figure 1: Broadband users connect to Internet using a plethora of NATs and firewalls. For widespread adoption, VoIP calls need to work through them seamlessly.

2.1 The NAT/Firewall Traversal Challenge

Homes and businesses are increasingly installing intermediary devices between their computers and their Internet connections. These devices – usually routers – provide a number of capabilities, with the most common being that of a NAT and/or a firewall.

NAT traversal is complicated by many contributing factors:

1. NATs break VoIP protocols

The idea of a NAT is to allow several devices to share a single public IP address. Figure (2a) shows how a router connects several hosts using private IP addresses to the Internet using a single public IP address. The router allows these hosts to access the public Internet by modifying each IP packet to and from these computers by using a two-way mapping between private IP addresses and transport ports to the router's public IP address and transport ports. The rewriting of addresses by the NAT is usually performed using a lookup table, where mappings between internal address/port pairs and external address/port pairs are stored.

This technique facilitates sharing a single public IP address among many hosts that use private IP addresses. However, this technique imposes a few problems for VoIP calls. Figure (2b) shows the problem when Carol makes a VoIP call using SIP from behind her NAT device. To establish the call, Carol needs to share the IP address and a UDP transport port where she will receive voice data. However when Carol uses the private IP address and local UDP port to receive voice for the SIP call, voice packets from the remote party connected to public Internet will never reach Carol because private IP addresses are not routable in the public Internet.

Another property of NATs is that the port mapping is kept only if there is traffic in both directions. For example, if Carol is in a call with Ellen, and for a while only Ellen talks (i.e. Carol does not send any packets to Ellen), then Carol's NAT may close, which effectively terminates the call.

2. Firewalls do not allow uninvited packets and close inactive connections

The main purpose of a firewall is to protect an internal network from unauthorized access by entities on external networks. Firewalls normally allow incoming traffic from external hosts only if the session was initiated from the internal network. Therefore, incoming calls, coming from applications on un-trusted external sources, are filtered out by the firewall, and the application fails to establish connection between the end users. Firewalls are not only present in most routers, but are also available in most modern operating systems (e.g. Windows firewall in Windows XP).

Figure (2) shows the problem as described above. The firewall allows media from Ellen to reach Carol, because Carol initiated the call. However, the incoming call from Dave could not pass through the firewall, as no data packets were sent to Dave from Carol. Therefore, the call between Carol and Dave fails to establish. A firewall can, however, be configured in any number of ways, such as only allowing TCP traffic out to the public Internet and preventing the use of UDP.

3. Cascaded NATs

NAT configuration may be cascaded, which adds one more level of complexity to the problem. In this scenario, one router is connected to the Internet using public IP addresses, and provides a private IP address to a second set of routers. Each of the second set of routers may itself provide separate private IP addresses to one or more hosts. For VoIP, the challenge is for any host connecting to any of these routers may call each other, or it may also call any other host in the public Internet (or behind yet another router in another location).

UPnP gateways expect application control

4. Sometimes residential routers expect application control using the UPnP protocol to access the Internet. If UPnP is enabled on a router, which is the default case for many Asian countries such as Japan and Korea, the VoIP application needs to speak with the UPnP protocol to enable sending/receiving of data to/from the Internet.

Enterprise firewalls block UDP and sometimes enforce web-proxies

5. Most businesses or enterprises use strong firewall rules where UDP is usually blocked. Thus all communications need to use TCP transport. In some the cases only Internet communications that these businesses allow is browsing the Internet through some web-proxies (such as Squid or Microsoft ISA). In such environments, VoIP calls cannot use UDP, and therefore need to use TCP transport or HTTP-tunneling.

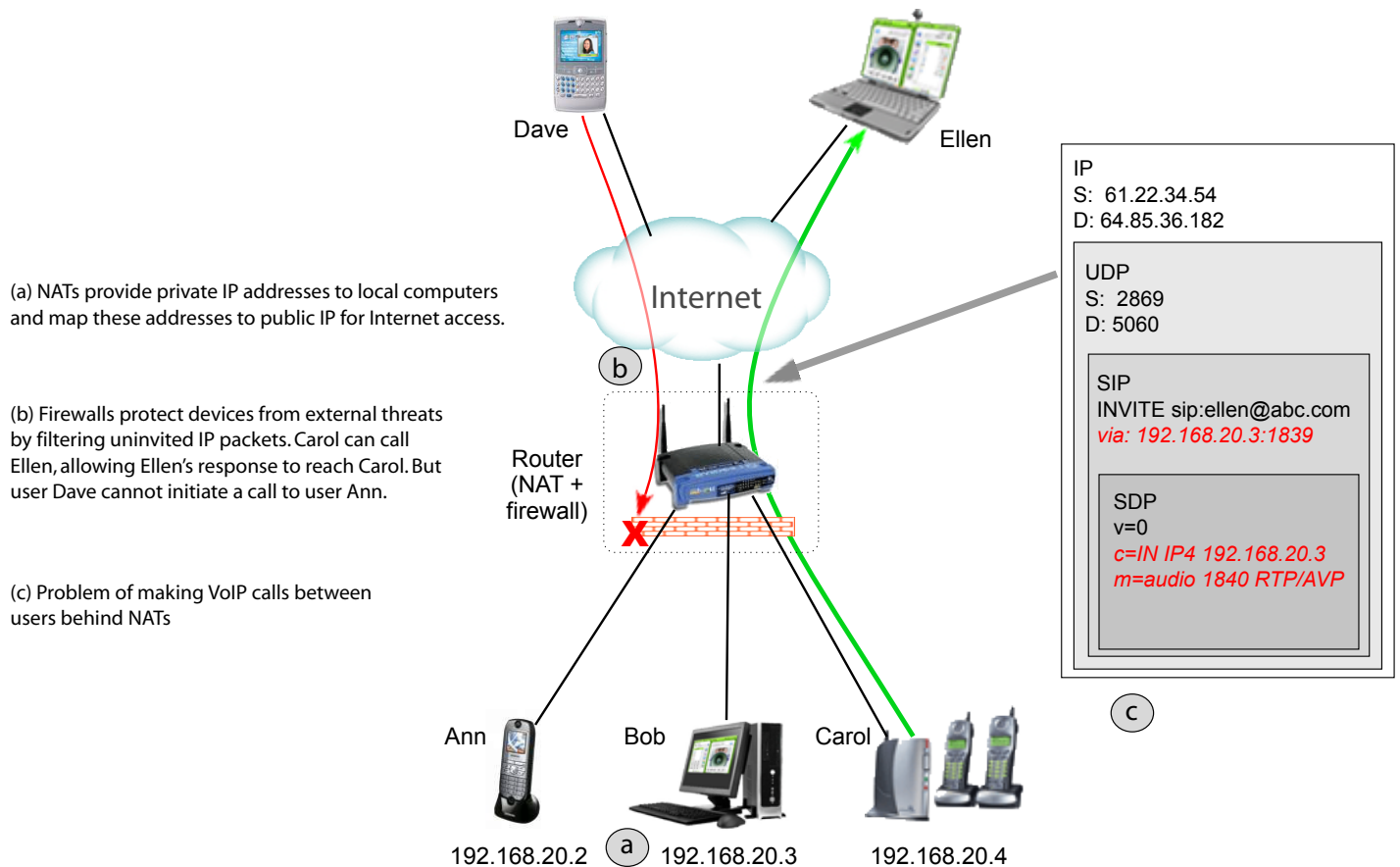


Figure 2: The NAT traversal challenge

While NATs/firewalls play a very important role in securing and enhancing the usability of an internal network, they impose a significant problem in setting up VoIP calls between end users. Application developers cannot make assumptions about how traffic would pass into or out of these private networks.

2.2 NAT Traversal Solution Requirements

We have seen how NAT/firewalls present a challenge to VoIP call completion. As shown in Figure (1), there are different kinds of NATs/firewalls in use, each of which may be set up differently, making VoIP calls difficult to complete.

A typical solution to the problem described above is that a VoIP application will require a range of specific port numbers to be left open in the firewall. This approach introduces a severe security risk because an intruder, with knowledge of these open ports, can create malicious software to take advantage of the fact that the firewall is letting traffic in through the open ports. Leaving ports open defeats the purpose of installing a firewall in the first place.

Another problem with opening ports is that manual configuration is required by end-users or network administrators. Home users often lack the necessary technical knowledge to correctly make this adjustment, or may even be unable to do so if their ISP controls their firewall product, as is the case with certain cable and DSL service providers. For internal users, their network administrator may also be unable, or more likely unwilling, to open the required ports that the VoIP applications need to function properly. Either way, users are required to take extra steps to enable end user communications and, more often than not, will give up in frustration.

Some key features that are expected from a NAT traversal solution include:

- **Guaranteed call completion with maximized peer-to-peer calls:** The solution must ensure 100% call completion rate between users, regardless of the NAT/firewall types used. Moreover, it needs to maximize peer-to-peer calls in order to reduce load on relay servers.
- **Security:** The NAT traversal solution must not compromise the security settings of the NAT/firewall.
- **Ease of integration with existing products or services:** It is vital for the NAT traversal solution to be easily integrated with existing VoIP products or services, with minimal amount of work and time.
- **Standard compliance and interoperability:** The solution must interoperate with equipment from different vendors. Therefore, the solution must be based on some standards to ensure successful communication between devices with different settings.
- **Service scalability:** The solution needs to be scalable so that it can be used for a large number of participants.
- **Optimized call completion time:** The solution needs to make sure that the calls are established in a reasonable amount of time.

3 Solving the NAT Traversal Problem

To solve the NAT traversal problem, the industry has attempted a few solutions:

1. **Application Level Gateway (ALG):** An ALG acts as a protocol-aware firewall, monitoring traffic and permitting traffic flows for specific applications. This solution, however, does not ensure security or authenticity, and is difficult to deploy.
2. **Session Border Controller (SBC):** An SBC addresses some of the problems that ALGs fail to resolve. However, this solution is not scalable for large numbers of concurrent calls. Moreover, it introduces additional delay and packet loss with the ultimate consequence of inferior end-user experience. Since SBCs use proprietary methods for NAT traversal, they do not work with SBCs from other vendors and/or third party solutions.

- IETF STUN, TURN and ICE: The IETF (Internet Engineering Task Force) has devised a suite of protocols, namely STUN (Session Traversal Using NAT) [1], TURN (Traversal Using Relay NAT) [2], and ICE (Interactive Connectivity Establishment) [3], to address the limitations of the currently available NAT traversal solutions. STUN allows the applications discover the public IP address and port mappings that the applications can use to communicate with its peer. TURN, on the other hand, allocates a public IP/port on a globally reachable server and uses it to relay media between communicating parties. ICE is a framework that defines how to use the STUN and TURN protocols to solve the NAT traversal problem, by choosing the best possible interconnection method between two users. Since ICE incorporates STUN and TURN methods, sometimes ICE is also used to refer to the complete STUN, TURN, and ICE solution.
- 3.

Leading vendors including Microsoft, Cisco, Nortel, Lucent Alcatel, Huawei, Avaya, Juniper, Tanberg, Tekelec, Nokia, and Sony Ericsson have adopted ICE for NAT traversal. CableLabs, the technology consortium of

cable system operators who are also the largest VoIP operators in USA, has also incorporated ICE support into the CableLabs IMS specification for next-generation communications architecture.

The next section presents an overview on how NAT-traversal using ICE methodology works.

4 How ICE Methodology Works

In this section we assume that calls are established and controlled with SIP [13]; however, call completion can work with any appropriate rendezvous protocol. An overview of the ICE methodology is shown in Figure (3). Making a call starts by sending a SIP INVITE message with an SDP describing on which IP address(es) and port(s) the application can receive audio and/or video packets. These addresses and ports are known as candidates. Candidates are obtained from the AnyFirewall Engine and are inserted into the SDP of a SIP INVITE message, which is sent to the callee.

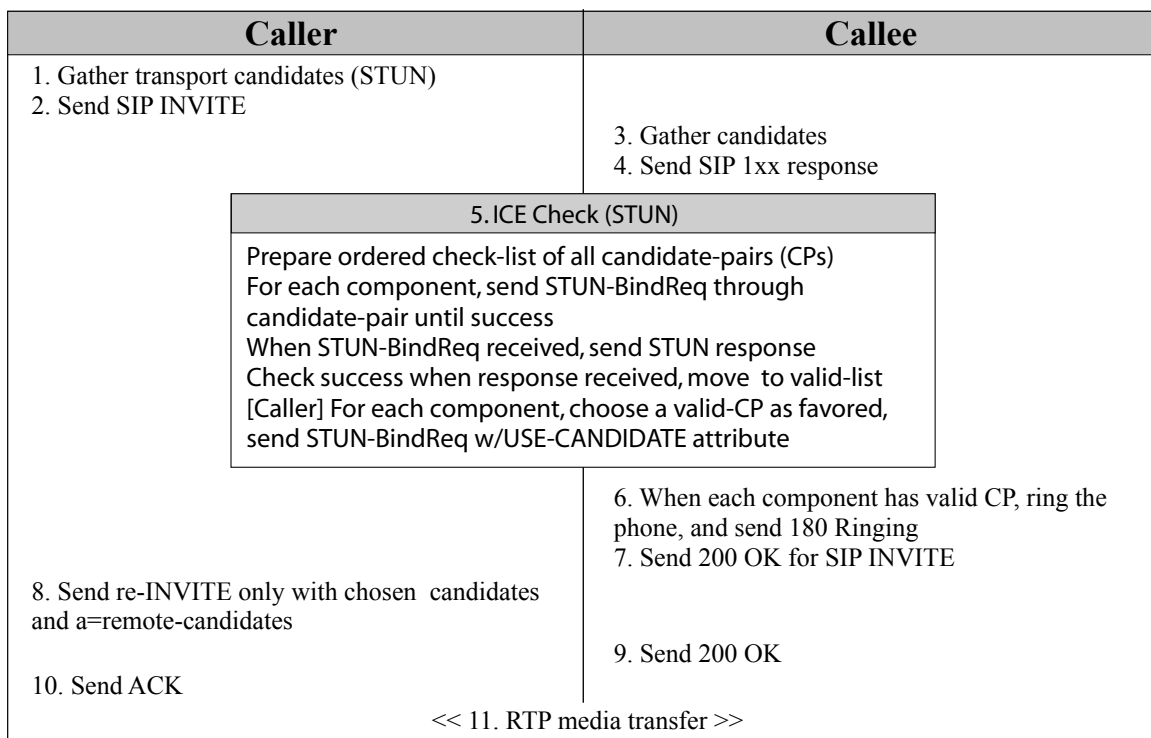


Figure 3: SIP call flow using ICE

Specifically, a candidate is an IP address and port at which one peer can receive data from another peer.

There are 3 types of candidates:

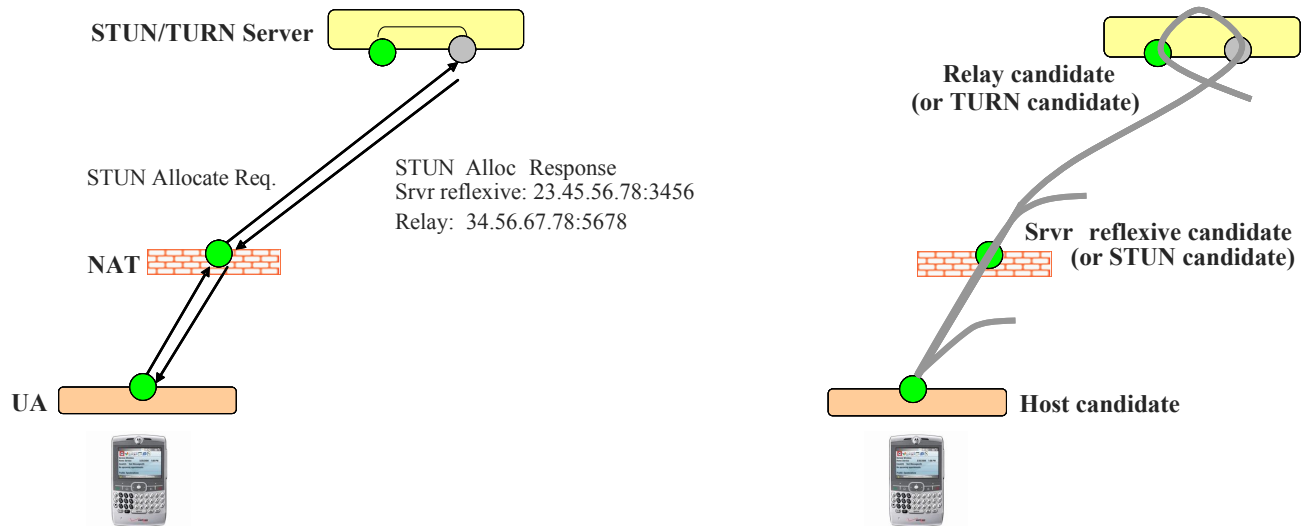
1. Local candidate: a local IP address of the client.
2. Reflexive or STUN candidates: an IP address of the client's NAT (assuming they are only behind a single NAT). These are determined from another entity, and then communicated back to the client.
3. Relay or TURN candidate: an address on a relay server that has been allocated for use by the client.

Traffic can always be sent successfully using relay candidates, unless a firewall blocks all traffic towards the client, in which case no legitimate firewall traversal technique can ever work. The problem with using relay candidates, however, is that they require server resources, and relayed traffic introduces additional delay, loss and jitter in the traffic stream.

We now describe how ICE methodology works for SIP-based calls, as related to the steps in Figure (4):

1. Caller gathers transport candidates

As Figure (4a) illustrates, the caller determines its server reflexive and relay candidates for a connection. The client sends an ALLOCATE request to the server, which instructs the server to allocate an IP address and port on the server (the relay candidate). Upon successfully allocating the relaying address for the client, the server returns the caller's IP address and port as seen by the server (the client's server reflexive candidate). The reflexive candidate contains the public address that the client is using, which is usually the address of a NAT that the client is behind.



(a) Using STUN to find candidates.

(b) 3 candidates for media reception:

- Host candidate
- STUN candidate
- TURN candidate

```
a=candidate:1 1 UDP 2130706431 192.168.1.102 1816 typ host
a=candidate:2 1 UDP 1694498815 23.45.56.78 3456 typ srflx
a=candidate:3 1 UDP 16777215 34.56.67.78 5678 typ relay
```

(c) Using candidates in SIP offer/response.

Figure 4: Gathering of candidates by user agent

2. Caller sends a SIP INVITE

After gathering the candidates, the caller encodes them in the call setup message (e.g. a SIP INVITE), as shown in Figure (4c), and sends the message to the called party.

3. Callee gathers transport candidates

Upon receiving the SIP INVITE, the called party also gathers its candidates in the same manner as the caller does in step 1.

4. Callee sends SIP 1xx response

In response to the SIP INVITE, the callee sends its ICE candidates within the SDP of a SIP provisional response, such as a SIP 183 (Session Progress), to the caller. The message should be sent reliably (i.e. with retransmission), and should not be considered successfully sent until either a 200 OK is received in acknowledgement, or a connectivity check from the caller is received by the callee, as is described in the next section.

Both conducts ICE connectivity checks

5. Once the callee has sent its ICE candidates, and once the caller receives them, they each start performing ICE connectivity checks. At this point, both parties know about their peer's potential transport candidates. Each possible pair of local and remote candidates is formed, creating a number of candidate pairs. A connectivity check is done by sending STUN messages from the local candidate to the remote candidate for each pair, starting with the highest priority (i.e. most preferred) candidate pair first. Both parties exchange STUN messages in this way to determine the best possible candidate pair that they can use to communicate. Once a valid (i.e. successful) message has been sent both ways on a single candidate pair, the connectivity check may stop and media can be sent/received using that candidate pair.

Figure (5) shows how the ICE connectivity checks are carried out between user agents by sending STUN messages between candidate pairs. For simplicity, the Figure shows only a subset of the candidate pairs that could be checked.

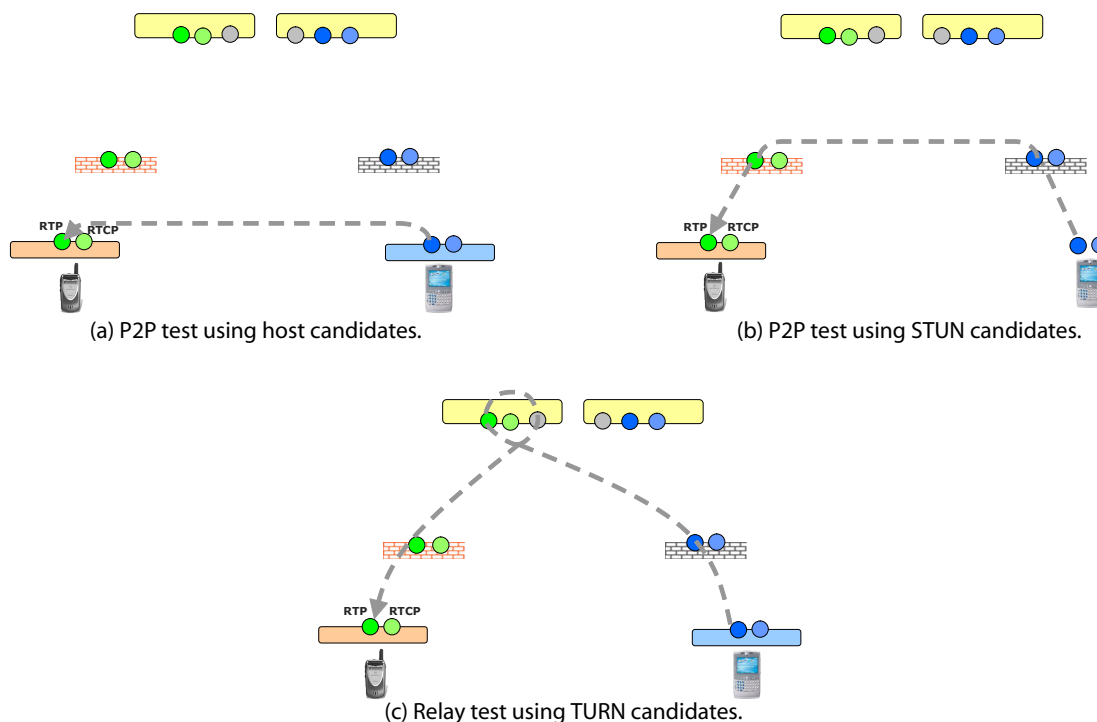


Figure 5: Performing ICE connectivity check by user agents

6. Callee sends SIP 180 ringing

A 180 RINGING message is sent to the caller after the connectivity checks are completed, signaling that the callee's phone has begun ringing. Ringing the callee's phone happens now, after signaling is complete, so that there is no delay in receiving audio once the phone is picked up.

7. Callee sends SIP 200 OK to invite

If the user accepts the call, the callee sends a final response to the caller (200 OK).

8. Caller sends SIP re-invite

If the candidates chosen differ from the original connection candidate put into the media and connection lines of the SDP of the SIP message (i.e. the candidate chosen differs from the one thought to have worked), then a new SIP INVITE message is sent to the callee with the agreed upon candidates in the m/c line of the enclosed SDP message.

Callee sends SIP 200 OK to re-invite

9. If a re-invite is sent, then it must be acknowledged, which is done with a SIP 200 OK message.

Caller sends SIP ACK

10. A final acknowledgement is sent from the caller to the callee indicating the completion of the call setup.

Voice/video media transport starts

11. Now that the call has been established, both caller and callee can send media to/from their successful candidate addresses. (usually using RTP protocol).

5 Eyeball AnyFirewall Technology

Eyeball has developed AnyFirewall Technology to ensure seamless traversal of media across different NATs, firewalls, UPnP gateways, & web proxies. This comprises of two products:

- I. AnyFirewall Engine (AFE) – the industry's leading firewall and NAT traversal SDK offering the most comprehensive implementation of STUN, TURN and ICE.
- II. AnyFirewall Server (AFS) - a carrier-grade STUN and TURN server ready for licensing and mass deployment.

Here are a few highlights about Eyeball NAT traversal solutions:

- Developed using industry standard protocols: Based on IETF standards and drafts; STUN-rfc 5389 [1], TURN-rfc 5766, ICE-rfc 5245, ICE-TCP [4], nat-behaviour-discovery-01[5], and UPNP[10].
- 100% call completion: In addition to implementing ICE for NAT/firewall traversal, UPnP and HTTP proxy tunneling are provided to ensure 100% call completion.
- High peer-to-peer call completion rate: More than 95% of calls are completed peer-to-peer in UDP-enabled networks.
- Small SDK footprint: The standard footprint is less than 300kB, but smaller footprints are available for embedded devices and other environments where available memory is limited.
- Multiple platforms: AFE is available on Windows, Linux and MacOS, with other platform support available upon request.

- Easy to integrate: AFE API is based on the standard Berkeley socket API, which is used in most operating systems. This allows AFE to be integrated quickly into existing products.
- Complete solution: AnyFirewall Server (a standards-based STUN/TURN relay server) and AnyFirewall Engine (a standards-based ICE client) provide a complete solution for NAT traversal.
- Service scalability: AnyFirewall Server supports more than 10,000 concurrent calls on a single server computer, with more calls supported by simply adding more computers.
- Product maturity: Eyeball has been a leader in NAT traversal solutions for over 5 years. Our products are field tested by millions of end-users all over the world.

Sections (5.1) and (5.2) present AnyFirewall Engine and Server solutions respectively, and section (5.3) provides a typical call completion scenario using these solution.

5.1 Eyeball AnyFirewall Engine

AnyFirewall Engine provides a feature-rich NAT traversal SDK for application developers and device makers. Here are a few technical highlights:

- Most comprehensive implementation of STUN, TURN, and ICE, plus optional features such as UPnP gateways and HTTP tunneling through web-proxies.
- Automatic selection of transport modes (UDP or TCP), and transparent translation of UDP to TCP when using TCP relaying.
- Supports symmetric RTP and smart keep-alives for signaling and media connections.
- Multiparty calls with hybrid UDP, TCP and HTTP streams.
- Traversal for voice, video, instant-messages and file-transfer.
- Minimized call completion time by pre-fetching and caching candidates.
- Simple C/C++ API familiar to TCP/IP socket programmers.
- Works with 3rd party SIP/XMPP stacks & voice/video engines.
- PC and embedded system support including Microsoft Windows, Linux, and Windows Mobile.

The rich set of APIs offered by AFE enable developers to write VoIP or other peer to peer applications without the concern of firewall traversal problems. Figure (6) shows the diverse kinds of VoIP applications that can be built using AFE. AFE also integrates with third party application protocol stacks and media engines.

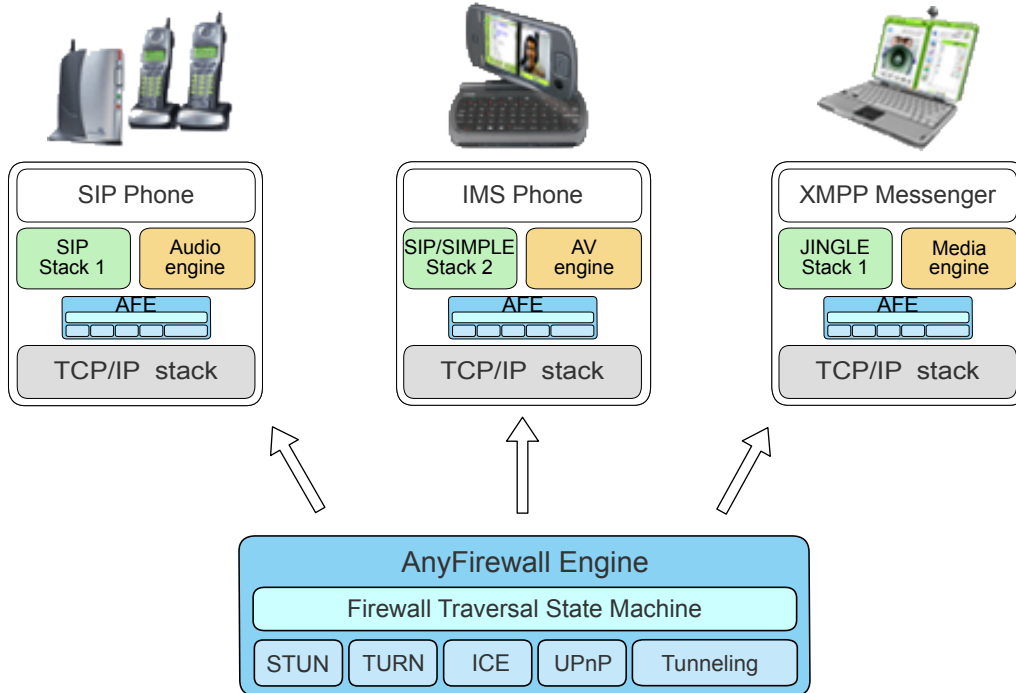


Figure 6: VoIP applications built using the AFE API

5.2 Eyeball AnyFirewall Server

AnyFirewall Server is a carrier-grade server for NAT/firewall discovery, signaling and media relay based on STUN and TURN IETF RFCs. Here are some of the features of the AnyFirewall Server.

- The first standards-based NAT and firewall traversal server for VoIP. It incorporates STUN, TURN, supports HTTP tunneling as a fallback and supports traversal of media and signaling including voice, video, IM and file-transfer.
- Provides scalable firewall traversal for large deployments by completing most calls use peer-to-peer media transport, using load balancing based on DNS SRV lookup and supporting more than 10,000 concurrent calls per CPU.
- Interoperable with third party clients and end-points, and SIP servers from Cisco, Huawei, Nortel, Tekelec and Ubiquity.
- Supports wiretapping of calls by forcing relay usage for certain using (for CALEA requirements).
- Ready for deployment in IMS infrastructure (stand-alone server or integrated into CSCF).
- Runs on standard Linux systems (standard PC or carrier-grade servers).
- Easy to set up using either text-based configuration; interactive command line interface; or web-based provisioning, monitoring, and usage statistics.

Figure (7) shows how this solution is deployed in an operator's network to achieve 100% call completion. AFE and the AnyFirewall Server together provide a comprehensive solution meeting all the requirements for solving the NAT/firewall traversal problem, as discussed in Section 2.2.

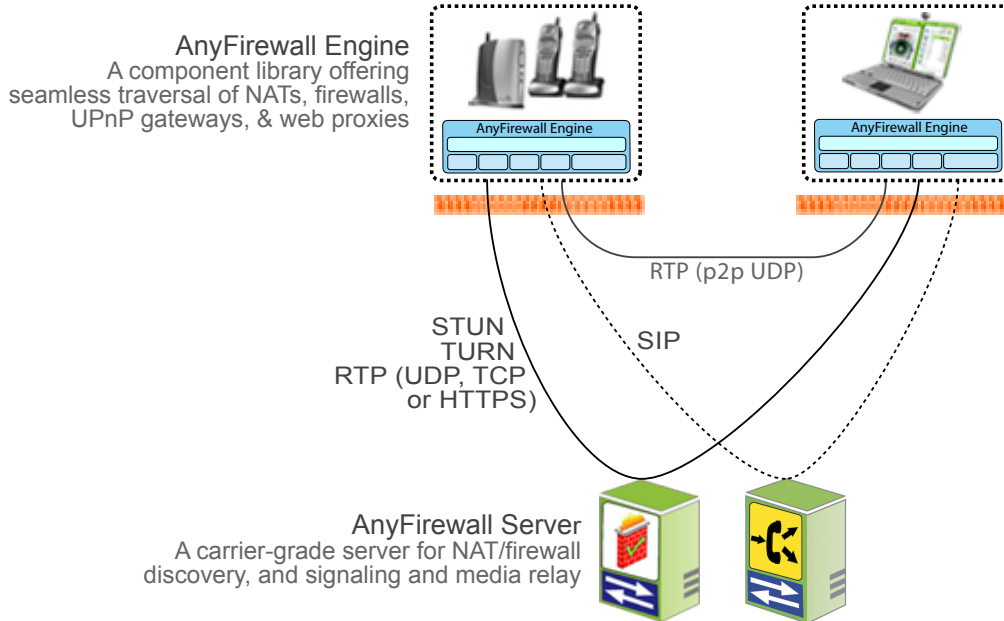


Figure 7: Eyeball AnyFirewall solution deployment

5.3 VoIP Call Example Using the AnyFirewall Engine

Eyeball AnyFirewall Engine uses the concept of channels to simplify application programming. Figure (8) shows how the AFE fits in a typical VoIP application. Each channel is accessed via a set of functions similar to the socket API. Like sockets, each channel represents an endpoint for sending and receiving data. However, channels hide the underlying complexity required for the firewall traversal process, such as the STUN, TURN, and ICE functionality. To make adding the functionality of AFE to an existing application easy, calls to the socket API are replaced with similar AFE API calls. For example, to send and receive data using AFE, an application calls the `Send()` or `Recv()` on a channel, instead of using the `send()` and `recv()` functions of the socket. Furthermore, AFE provides the `Select()` function for channels, which models the behavior of the socket API function, `select()`.

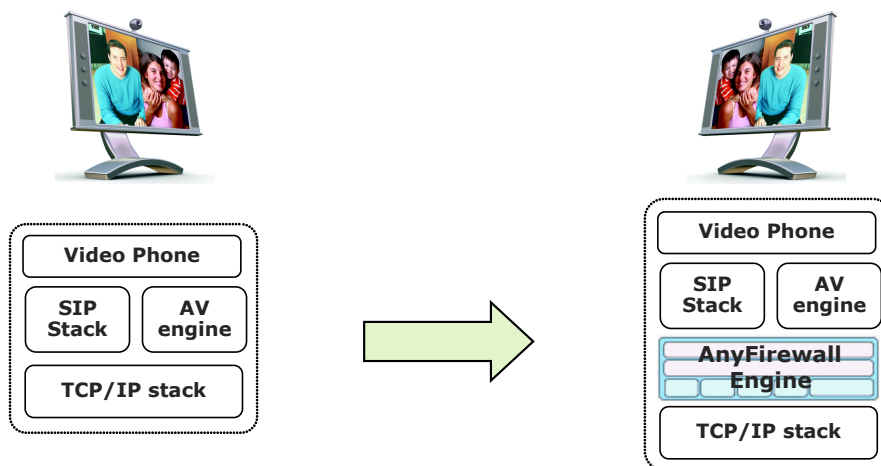


Figure 8: Integration of the AnyFirewall Engine into a VoIP application

Figure (9) shows the sequence of AFE API calls that take place on the caller and callee user-agents in order to establish a call. Please note that the sequence of calls here is similar to the sequence of ICE methodology steps in figure (3). Once a call is completed, the usual Send(), Recv(), and Select() functions are used, as in standard socket API.

Caller	Callee
<p>1. iRTPChannel = Create ("AF_CHANNEL_RTP") iSession = CreateSession ("audio " + iRTPChannel) sAudioSDPInfo = MakeOffer(iSession)</p> <p>2. sInvite = BuildINVITE(sAudioSDPInfo) Send(iSIPChannel, sInvite)</p> <p>5. Receive PRACK with SDP ProcessAnswer(iSession, s183PRACK) // Caller starts ICE check here</p> <p>7. On receive 200 OK, start RTP stack (send/receive media)</p>	<p>3. Receive INVITE on SIP channel iRTPChannel = Create ("AF_CHANNEL_RTP") iSession = CreateSession ("audio " + iRTPChannel) sAudioSDPInfo = MakeAnswer(iSession, sIn) // Called party starts ICE check here</p> <p>4. s183PRACK = Build183PRACK(sAudioSDPInfo) Send(iSIPChannel, s183PRACK)</p> <p>6. On ICE_CHECK_COMPLETED event, ring the phone Once the call is accepted, s200OK = Build200OK() Send(iSIPChannel, s200OK)</p> <p>8. start RTP stack (send/receive media)</p>

Figure 9: SIP call setup using AFE API

6 Conclusions

NAT devices and firewalls are major barriers to widespread adoption of VoIP and IETF work STUN, TURN, and ICE provide an excellent methodology to address this issue. Unfortunately, ICE is a very complex methodology. Internet engineering experts at IETF have taken more than 6 years to develop the protocol. Real implementation requires considerable insight and experience to get it right. Eyeball has invested many years to develop its technology, and its NAT traversal technology has been field-tested for more than 7 years. If you want to integrate ICE in your products and services, and are working with time-to-market and development cost constraints, getting the NAT traversal solution from technology experts at Eyeball may be your best option. Otherwise you may risk going into a prolonged cycle of development which may cause your project to go out of budget and delay your product launch.

Eyeball provides comprehensive and award-winning NAT-traversal solutions comprising two products: (a) AnyFirewall Engine – a feature-rich NAT traversal SDK incorporating ICE, and (b) AnyFirewall Server – a carrier-grade STUN and TURN server. VoIP application developers, device makers, and service providers can now integrate and deploy these solutions into their products in order to guarantee VoIP and video call completion across NATs, firewalls, and web proxies.

7 About Eyeball Networks

Eyeball Networks Inc. is a pioneer and world leader in VoIP, video telephony, and instant messaging software. We enable service providers, application developers, and device makers to deliver carrier-grade communications services with guaranteed call completion and quality. Today, more than 20 million subscribers and 200 service providers and device makers benefit from Eyeball's innovations, including its AnyBandwidth™, AnyFirewall™, and AntiSPIT™ technologies to guarantee the best possible IP communications experience over any Internet connection, across any NAT or firewall, and on any device.

For more information, please visit <http://www.eyeball.com>.

Eyeball Networks Inc. Tel. 604.921.5993
102 - 100 Park Royal Fax 604.921.5909
West Vancouver, BC
Canada V7T 1A2
Sales: sales@eyeball.com

8 References

1. IETF STUN RFC 5389: Session Traversal Utilities for NAT (STUN)
2. IETF TURN RFC 5766: Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)
3. IETF ICE RFC 5245: Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols
4. IETF STUN RFC 5780: NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)
5. IETF SIP RFC 5626: Managing Client Initiated Connections in the Session Initiation Protocol
6. IETF Symmetric RTP RFC 4961: (RTCP) Common Local Transmit and Receive Ports (Symmetric RTP) Protocols
7. IETF RFC 3581: An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing
8. IETF RFC 3489: STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)
9. UPNP Forum specification: Internet Gateway Device (IGD) Standardized Device Control Protocol V 1.0
10. CableLabs specification: PacketCable 2.0 NAT and Firewall Traversal Technical Report (PKT-TR-NFT-V02-061013)
11. Wi-Fi Phones Biannual Worldwide Market Share and Forecast, Infonetics Research, January 2006
12. IETF RFC 3261: SIP –Session Initiation Protocol