Mizutech VoIP Tunneling and Encryption

Technical details

About

The Mizu VoIP tunneling solution is a set of client and server side software capable to bypass voip blockades, firewalls, NAT's, STUN and HTTP proxies.
For a detailed presentation please visit this link: http://www.mizu-voip.com/Products/VoIPTunnel.aspx

All configuration and management can be done by Mizutech support. However you can use the tunneling guide and the softswitch admin guide to manage the VoIP tunneling server yourself.
Once the server is installed, you will usually need to perform only a few management tasks which are the followings:

- edit your registrar, inbound and outbound server(s) settings (all these settings can point to a single server)
- monitoring (running calls, CDR records, statistics)
- backup and cloning

Install, configuration and support services are included with each license plan. Contact serversupport@mizu-voip.com

Included software

Server:

On the server side a special VoIP software is used whose task is to encrypt all communication from/to the clients and to forward the calls to your server(s) using the common SIP protocol.

- Mizu softswitch with built-in tunneling (This is a full featured softswitch including user management and billing features)

or

- Mizu Tunneling Server (This can be used with any existing softswitch or voip servers. This software is based on the Mizu softswitch but unlike a softswitch it doesn't require provisioning and billing configuration)

Client:

The followings are available on the client side:

- Cross platform browser webphone (platform independent java applet. Runs from any J2SE capable browser)
- Windows softphone
- iPhone softphone
- Android softphone
- mvoiptunnel (client software that can be integrated with any third party softphone on the Windows platforms)
- mvoiptunnelclient (client service to enable tunneling services for any third party SIP clients: ATA's, IP phones, softphones, etc)

Technical details

The tunneling server is capable to use various tunneling and encryption transport methods to bypass all VoIP filtering and firewalls running over a full featured class 4/5 SIP stack with all common functions included (registrar and proxy/SBC functionality, voice calls, video calls, presence, IM, all DTMF methods, PBX functions like call hold, transfer, conference, etc and with support for all common codec's: g729, g723, g711, GSM, speex, opus wideband and ultra-wideband, iLBC, g726, g722, L16)

The tunnel service uses both standard (TLS/DTLS/SSL/RSA/Blowfish) and proprietary encryption. There is a few different built-in encryption methods based on symmetric keys to speed-up the connect times. The most used encryption method is Blowfish due to its low CPU usage. Additionally it can use compression to further reduce bandwidth utilization. Optionally also double encryption can be enabled (encryption with 2 different methods). For key exchanges usually the RSA protocol is used.

The server software is optimized to the Microsoft Windows operating systems with an MS-SQL backend (although database operations are not so critical compared with a traditional softswitch. The database is used only to store the configuration,

temporary user records and CDR records for local statistics). The server application is running as a standard NT service and is capable for both client side and server side load balancing and failowering.

The tunneling server can be used in the following ways:

- one tunneling server - one softswitch (this is usual. The tunneling server can be installed in the same box where your softswitch is running or on a separate box)
- one tunneling server - many softswitch (if you have many softswitch, each with low traffic)
- many tunneling server - one softswitch (if your softswitch can run more than 10000 simultaneous calls, than the performance of one tunneling server is not enough and you need to use more tunneling servers)

Performance while using encrypted transport with tunneling and encryption on a Xeon E5620 with 6 GB RAM (64 bit Windows Server 2008):

- simultaneous calls: 3500
- registered users: 45000
- BHCA: 432000 (120 calls/sec)
- signaling message processing/sec: 1600

The client will try to use the "best" available transport protocol. There are 11 different built-in transport protocols that can be used. The following table will present the most important categories.

	Direct UDP (no tunneling or encryption)	Encrypted UDP	Direct TCP or via http proxy connect	HTTP 1.1. or chunked transfer	HTTP 1.0 with or without proxy
Quality in good network conditions	10	10	11	9	7
Quality in bad network conditions (8% packet loss, 150 msec jitter)	8	8	6	5	3
Server Processing Load	1x	1,5x	2,5x	3,5 X	4x
Delay	50 msec	50 msec	75 msec	250 msec	400 msec
Protocol Overhead and Ethernet bandwidth usage with G.729	0% (32 Kbits)	1% (32 Kbits)	5% (35 Kbits)	10% (38 Kbits)	50% (50 Kbits)
Circumstances	Clear SIP/RTP protocol	When SIP/RTP is blocked (voip filtering)	When all UDP is blocked or from behind SOCKS proxies	From behind HTTP proxies or corporate firewalls	In very rare circumstances if everything other is blocked behind an outdated corporate firewall

With the bandwidth optimizer enabled, you can even cut the network usage! (with around 5-60% for 10 simultaneous calls)

The following steps are performed by the clients:

1. On startup the clients will load its configuration (this can be hardcoded, stored in a local configuration file or downloaded from predefined location)
2. The client first will try to connect directly to your server(s) (to the "most available server" or based on autoprovisoning in case if you will have multiple servers). This is an optional step.
3. Select a random and usable UDP port (based on a sophisticated algorithm)

4. Try encrypted UDP communication (to the “most available tunneling server” or based on preconfigurations in case if you have multiple tunneling servers)
5. Will do a quick test to detect if UDP quality is lowered (with simulated signaling and RTP traffic).
6. If this fails then will try direct TCP connection
7. If this fails then will try tunneled TCP connection via socks or proxy
8. If this fails then will try SSL connection
9. If this fails then will try proxy traversal with connect, if there is a local http proxy
10. If this fails then will try http 1.1 with or without proxy (usually to port 80)
11. If this fails then will try http 1.0 streaming with chunked length encoding with or without proxy
12. If this fails then will try http 1.0 with or without proxy
13. If this fails then it might try alternative proxies

This is a simplified overview how the connect procedure and all this is done during startup (at login, first connect or call) and will take 1-3 seconds. If can’t connect then it will try to use alternative routes (distributed network service).
While using UDP transport the packets are sent with variable packet length to harden the detection by voip blockades.
The communication with the server can be tunneled in one single stream or distributed on multiple streams (based on timing or different streams for uploads and downloads)
The following negotiations can be done after startup:

1. During the subsequent registrations, if the protocol is UDP and the communication is blocked, then will switch to TCP automatically
2. During a call if call quality is lowered (high packet loss, jitter, etc) then will switch to the next available transport protocol automatically (without call interruption)
3. While using TCP, it can switch to HTTP automatically if TCP is blocked or network conditions have changed during the session. HTTP routing can bypass 90% of the HTTP proxies (except proxies where streaming is not supported)

The clients will remember its previous activity and in subsequent startups might act in a different way. For example if there was more than 3 subsequent UDP to TCP switch during a call (due to high packet loss), then in the next startup it will not try UDP at all.

We improve our software continuously to be able to detect and handle future blockades. This means that the above mentioned negotiations might be changes with time if necessary. The tunneling clients are also capable to work with multiple servers. In this case even if one IP is blocked, the clients will be able to work with the remaining IP’s or domains.
The software uses also some other technologies to prevent voip blockades which are not revealed here.

Both client and server side load balancing and failowering are supported.
When using client side load balancing, the client will use the “most available” server which means a combination of the followings:

- the server used before by the same client
- the most closest server (where the network round-trip time is the smallest)
- the number of users and simultaneous calls on the servers
- the CPU load and RAM usage of the servers

Server requirements

- OS: Windows Server 2003/2008/2012/2016 (Windows XP/Vista/7/8/10 for testing)
- CPU: depending on the usage (the server can take advantage from up to 32 cpu core)
- RAM: depending on the usage (minimum 1 GB for 200 simultaneous calls)
- Disk: 40 GB with logs turned off and or 192 GB for maximum trace level
- Network: ~10 Mbit/s for 300 simultaneous calls (depending on the transport protocol and the codec used)

Typical recommended configuration for VoIP service providers between 500 and 5000 simultaneous calls: 2 servers with 4 or 8 CPU core, 4 GB RAM, and 256 GB disk space. By using 2 servers you can separate the application server from the database server and also you will have a hot backup.