VoIP –High availability

Failover, load sharing, redundancy and a scalable service can be implemented with the Mizu VoIP server using any of the standard methods to achieve highly available VoIP service with 99.999% uptime or more.
Please note that here we discuss failover and load sharing for the mizu server itself (for incoming calls) and not for outgoing calls. Failover and load sharing for outgoing calls can be easily configured using the Mizu server routing (including server failover based on bad statistics, in-call failover, round robin and percent based routing, LCR and BRS).
Failover for incoming calls means that your system is capable to serve the users even if one (or more) of your servers fails (hardware or software failure, termination or malfunction). We will focus on failover here, but the same methods can be used also for load sharing.
Access to VoIP is done using the SIP, WebRTC or H.323 protocol, thus you have to make sure that your failover method will work on the protocol(s) used by your endusers and other peers.
For all these methods to work, you will need at least 2 separate servers, preferably geo distributed.
You can use one or more of the following methods to implement a highly available service.

HA architectures

Client based failover

With this method the failover role is handled entirely by the client endpoints and nothing have to be done on the server side.
If the VoIP endpoint cannot access your primary server, will retry to secondary server.
Pros:

- This is the most efficient and powerful method if supported by the VoIP endpoints and peers

Cons:

- Needs client side support

Note: the Mizutech VoIP clients has failover/backup server support on both SIP and WebRTC.

Load-balancer

You can use a load balancer to distribute the traffic across the VoIP servers.
Load balancers are using more robust software with a high throughput and are to be run on a reliable hardware. Hardware performance is usually not a problem as they don't need to implement complex call control logic.

Pros:

- Transparent for the endpoints (no need for any support by the endpoints)

Cons:

- The load-balancer might be a single point of failure (but that can be also failover or distributed or with a hot backup)

SBC

You can use a session border controller to distribute the traffic across your VoIP servers.

Pros:

- Transparent for the endpoints (no need for any support by the endpoints)
- SBC's can handle many other tasks such as media processing, protocol conversion (WebRTC to SIP) or transcoding

Cons:

- Costly: SBC might need similar hardware performance then the VoIP server
- The SBC will be a single point of failure (but that can be also failover or distributed or with a hot backup)

DNS A record

Simply setup a proper DNS name for your VoIP server. If the primary server fails with IP A, restore it from the last backup to a secondary server with IP B and reassign your DNS record to this second server (to IP B).
http://en.wikipedia.org/wiki/Domain_name

Pros:

- This is the simplest method
- Easy to implement and cheap

Cons:

- DNS refresh propagation can take some time (2 hours in average; sometime up to 24 hours) to propagate to all users. Meantime users with the old DNS will not be able to use the service
- The reconfiguration usually have to be done manually

DNS SRV record

This works in a similar way then the A record, but it has some advantages.
http://en.wikipedia.org/wiki/SRV_record

Pros:

- Cheap and simple method
- The failover can take effect automatically
- Can be also used for load sharing (records with different weights)

Cons:

- Your domain name provider might not provide access to SRV settings
- Some SIP and most WebRTC clients doesn't support SRV lookup
- Refresh propagation can take some time (however this is usually faster and less relevant then in case of using only A record)

Failover DNS

Some providers have similar service to reassign the dns automatically if your primary server fails

Pros:

- Cheap

Cons:

- Not reliable (usually it is ping based)
- DNS propagation will take time

IP takeover

If the primary server fails with IP A, assign IP A to your secondary server.
This can be implemented multiple ways but usually requires manual interaction:

- If the two servers are near each other, then simply change the Ethernet cable (Change also OS settings ipconf)
- Some hosting providers offers "Virtual IP" so you can easily migrate your IP to any of your servers
- If you are an ISP, then you can use BGP to implement this

Pros:

- No need for any client side support. Works with no delay (once the IP change were made –usually manually)

Cons:

- Can't be geographically distributed

Shared virtual IP

Is a method to duplicate all network traffic (so everything can be seen on both the primary and secondary servers)

Pros:

- Transparent for the endpoints

Cons:

- Can't be geographically distributed (will not protect for power outage)

Dual Wan

This might be an alternative only if you are running your server from your home/office by using two different providers.

Pros:

- You will not depend on a single ISP

Cons:

- Costly
- Your router must have support for this

Clusters

Clustering can be implemented using the OS clustering capabilities.

Pros:

- It might be preferred if you already have a cluster

Cons:

- Complicated and not reliable if the administrator doesn't have experience
- Costly

Containers

High availability and scalability can be achieved by using containers such as Docker.

In a container you might deploy the app service, the database service or both.

Platforms such as Kubernetes can be used for management, automation and auto scaling.

Pros:

- Can be used to create large systems with full automation

Cons:

- Complicated configuration

Virtual servers

Most vendors (VMware, Microsoft, etc) have already implemented live migration for virtual instances.

Note that you will need to use at least two hosts to protect yourself against hardware failures.

Pros:

- Robust and easy management

Cons:

- Costly

Database high availability

This is not a failover method in itself but we are listing it here because it plays an important role.
The Mizu VoIP server has two main components: the app service and the database service.
From high availability point of view, you might treat them as a single unit or use separate servers/containers for them.
The service has built-in database redundancy failover methods implemented. Your primary database server can automatically synchronize with secondary database server using log shipping, backup failover or other methods described here.
The VoIP app server(s) can automatically contact the secondary server if fails to connect to primary server.
Note that you don't necessarily have to use dedicated servers for database only. Database and VoIP service can be run on the same box.

Recommended method

Recommended configuration

You will have to use two servers (you can add more for load sharing if your traffic is high). The servers can be geographically distributed. Setup both the database and the VoIP service on both of the servers. At one time only one database can be the active; however there is no restriction for the number of VoIP app servers, thus you can run the primary database on one server and the VoIP service on both, especially if your traffic is high.
For a cheap and efficient failover or load balancing setup with Mizu servers we recommend combining the following methods:

- dns a record: assign a (sub)domain name to your primary VoIP app server
- dns srv record: if the VoIP service is running on both servers, then add both of them to your SRV records
- client based failover: can be implemented for all Mizu customized softphones/webphone (third party clients without this support can still use dns based failover)
- setup database auto failover as described here

This method will also add high scalability for your VoIP platform: add as many app servers as you wish, based on your traffic and load. A simple diagram can be seen here.

Emergency plan

If secondary (not active) database fails:

- Effect: No visible effect for customers (this server doesn't have an active role)
- Action: No urgent action is required. Repair/rebuild as soon as possible to avoid service loss if also the main database goes down.

If primary (active) database fails:

- Effect: No visible effect for customers (the VoIP app service will auto connect to the secondary database)
- Action: No urgent action is required. Repair/rebuild as soon as possible to avoid service loss if also the secondary database goes down.

If one VoIP app server fails:

- Effect: clients with built-in failover will connect to the other server. Users might have to redial. Third party VoIP endpoints without failover support will have some outage (until dns changes are propagated for then)

- Action: A simple urgent action is required: Change your dns records to point only to the other server

More capacity and redundancy

- Use any number of VoIP service. These are easy to setup and start and you can use as many as you wish
- Use any scaling (up and/or out) method for database servers

Hardware recommendation:

- For VoIP app service: multiple cheap servers (one mind-range CPU, 8 GB RAM, slow 60 GB disc or more if recording or excessive logging is needed)
- For DB service: two high-end servers. A powerful server can serve many app servers with huge amount of traffic (8-16 CPU core, 32+ GB RAM, 2 separate SSD disk)

Note that these are recommended only for high amount of traffic.
Low amount of traffic can be easily served with one single server running both the VoIP and the database service. For example the recommended hardware for 1000 simultaneous calls is one server with one modern CPU, 8+ GB RAM, 2x256 GB disk). For database we currently recommend SSD disks due to their high value for price.

Related

Softswitch webpage
Large scale VoIP webpage
HA database
Database failover
Admin guide