SANOG 25

Building Robust IPTSP Based on Open Source Technology

Anowar Hasan Sabir, BDCOM Online Ltd. Bangladesh

Session Goal

To provide you a understanding of Building IPTSP, Based on Open source technology includes Asterisk, Kamailio and some other things...

Agenda

Is Asterisk not enough?

- Why Kamailio?
- Installing Kamailio.
- Configuring them to work together.
- Expanding them for bigger game.

Is Asterisk Not Enough?

- Asterisk covers everything but does it fit for all purpose?
    - Asterisk - A complete PBX in software
    - Voicemail, conferencing, IVR, queuing, as well as standard calling function
    - Highly extensible - can handle virtually any task imaginable
    - It has some sister apps that has similar feature such as FreePBX, Callweaver, Yate, etc.

Is Asterisk Not Enough? ....

Without a doubt Asterisk is the BEST ip pbx you can have right now.
Its best for small and medium scale office.
Contact Center.
And mostly its nice for developing any telephony application you can think of.

But Think of a IPTSP. number of users may reach millions. even more.
Number of concurrent call. 1000. 5000 even more..
Is asterisk itself the answer?

Is Asterisk Not Enough? ....

In a single server handling millions registration, more than thousand concurrent call is impossible.

Have to deploy an array of server.
Managing the servers and maintain the number plan and routing is not impossible but a nightmare.

So we can think of something else that can be a supporting hand for Asterisk.

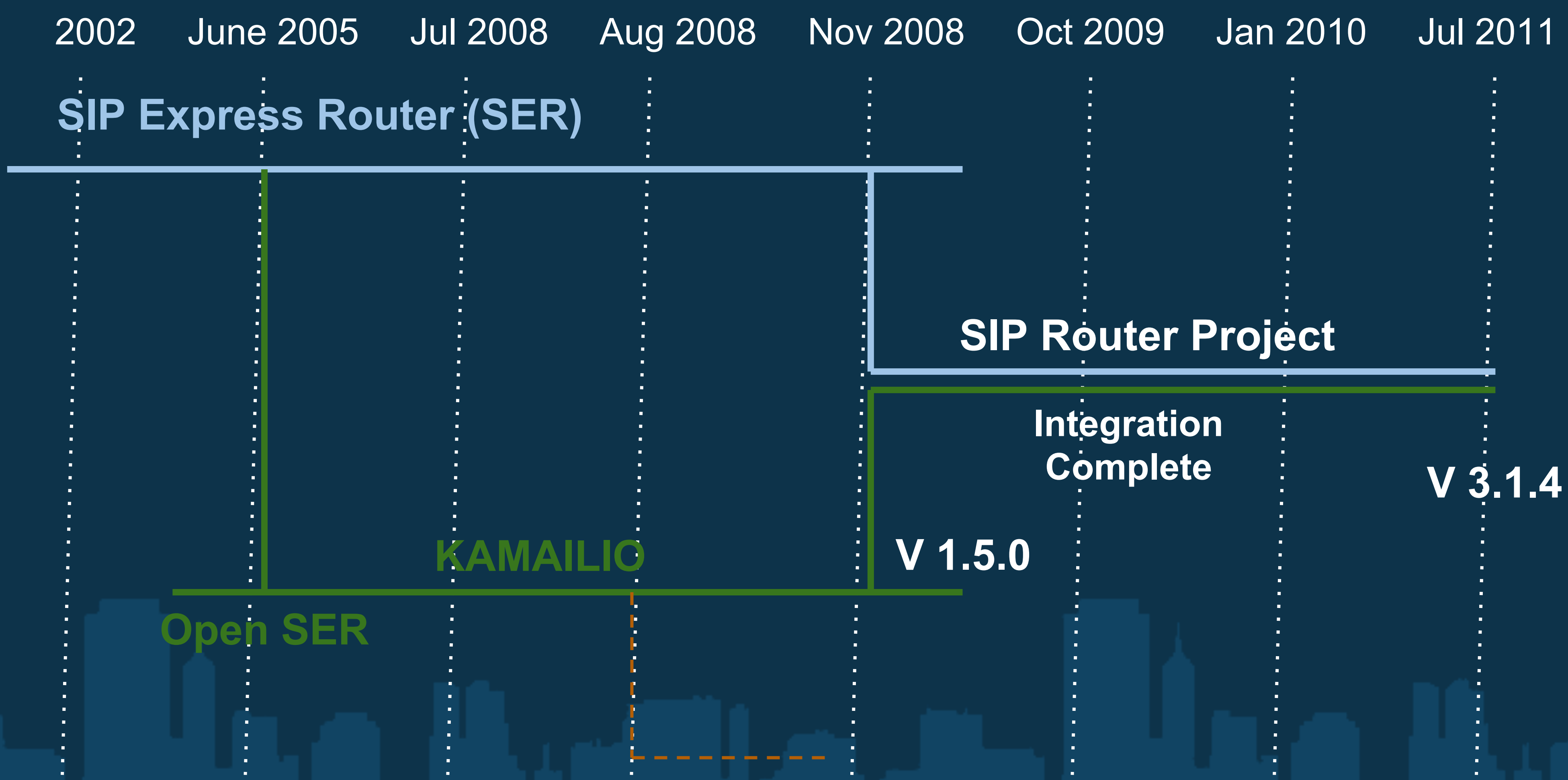
Why Kamailio?

Kamailio is a distribution of SER.
Sip Express Router. commonly known as SER can be the aid for Asterisk for broader perspective. because ..

- Kamailio can handle over 5000 call setups per second.
- On a systems with 4GB memory, Kamailio can serve a population over 300 000 online subscribers
- System can easily scale by adding more Kamailio servers

kamailio is simple first because it doesn't care about everything that done by asterisk. is a simple sip router.

Kamailio development.



Kamailio features

Robust and Performant SIP (RFC3261) Server flavours

- Registrar server
- Location server
- Proxy server
- SIP Application server
- Redirect server

Kamailio features

Flexibility

- small footprint - suitable for embedded devices - the binary file is small size, functionality can be stripped/added via modules
- plug&play module interface - ability to add new extensions, without touching the core, therefore assuring a great stability of core components
- modular architecture - core, internal libraries and module interface to extend the server's functionality
- impressive extension repository - overall 150 modules are included in the Kamailio source tree

Kamailio features.

SIP Routing Capabilities

- stateless and transitional stateful SIP Proxy processing
- serial and parallel forking
- NAT traversal support for SIP and RTP traffic
- load balancing with many distribution algorithms and failover support
- flexible least cost routing
- routing failover
- replication for High Availability (HA)

Kamailio features

Transport Layers

- support for communication via UDP, TCP, TLS and SCTP
- IPv4 and IPv6
- transport layer gatewaying (IPv4 to IPv6, UDP to TLS, a.s.o.)
- SCTP multi-homing and multi-streaming

Kamailio features

Secure Communication

- Digest SIP User authentication
- Authorization via ACL or group membership
- IP and Network authentication
- TLS support for SIP signaling
- transparent handling of SRTP for secure audio
- TLS domain name extension support
- authentication and authorization against database (MySQL, PostgreSQL, UnixODBC, BerkeleyDB, Oracle, text files), RADIUS and DIAMETER

Kamailio features

Accounting

- event based accounting
- configurable accounting data details
- multi-leg call accounting
- storage to database, Radius or Diameter

Kamailio features

Extensibility APIs

- Perl Programming Interface - embed your extensions written in Perl
- Java SIP Servlet Application Interface - write Java SIP Servlets to extent your VoIP services and integrate with web services
- Lua Programming Interface
- Python Programming Interface

for more visit http://www.kamailio.org/w/features/

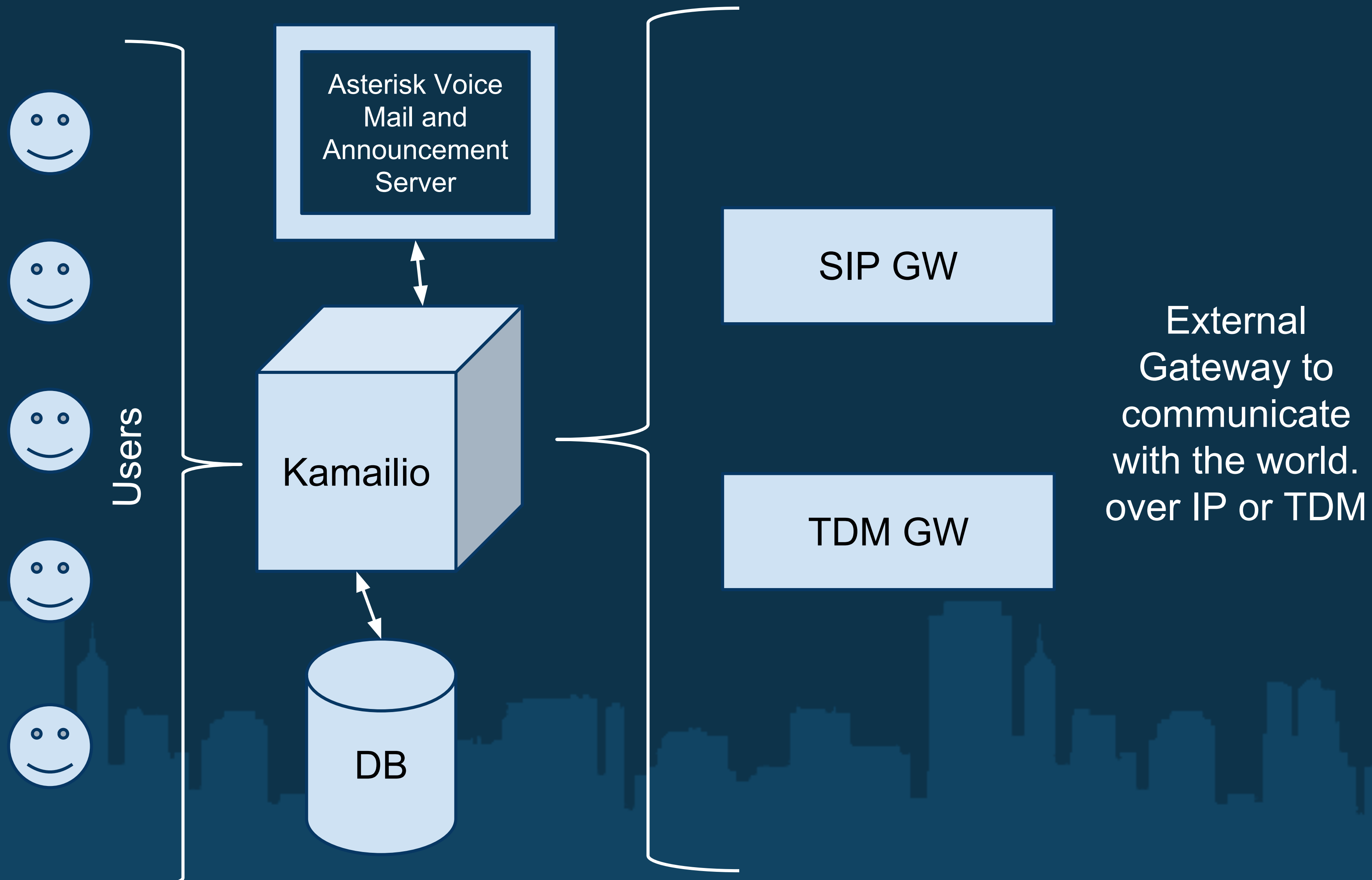
Why Kamailio+Asterisk?

If kamailio can do this all why do we need Asterisk with it?
because we can't do all in our telephony need in kamailio:
Like:

- We can't create IVR, Announcement in Kamailio.
- We can't do Voicemail in Kamailio.
- We can't translate protocol in Kamailio, like SIP->H323 or SIP->IAX2 etc.
- mainly we can't do TDM. like PRI,R2,SS7

So for complete telephony operation we go Kamailio+Asterisk

Kamailio+Asterisk



Configuring Asterisk

To work as a voicemail server for kamailio we have to create

- A SIP Trunk with Kamailio in Asterisk sip.conf.
- Add Voicemail Users in voicemail.conf. and
- Add specific dialplan in extension.conf

Configuring Asterisk

Add SIP Trunk in sip.conf

```
[voicemail-trunk]
type=friend
host=your.ser.domain.com
context=voice-mail-trunk
nat=no
qualify=yes
canreinvite=no
disallow=all
allow=alaw
allow=ulaw
insecure=port,invite
```

Configuring Asterisk

In voicemail.conf add your voice mail users.

```
[default]
extension_number => voicemail_password,user_name,
user_email_address,user_pager_email_address,user_option(s)
```

in extension.conf

```
[voice-mail-trunk]
exten => _XXXX,1,Voicemail(${EXTEN},u)
```

Configuring Kamailio

Before Configuring Kamailio for voicemail we have to create users for Kamailio. to create sip user we have to type ..

```
# kamctl add 1000 mysecret0
# kamctl add 1001 mysecret1
# kamctl add 1002 mysecret2
# kamctl add 1003 mysecret3
```

With this we just added 4 sip user in our kamailio server.

We can now configure our sip client and try dialing each other. you don't have to configure anything for calling your local user.

Basics of Kamailio configuration.

if you see the asterisk configuration folder folder you will find almost 20/30 files in a basic setup. All of them may not important for you.

In Kamailio there is only 2.
kamailio.cfg and
tls.cfg

Only things is important is kamailio.cfg

Basics of Kamailio configuration.

kamailio.cfg has 3 major part.

1. Where we load module

```
loadmodule "dispatcher.so"
```

2. Where we define module parameter

```
modparam("dispatcher","list_file","/path/dispatcher.
list")
```

3. Where we route call. the main part.

```
# - processing of any incoming SIP request starts with this
route
    route {
        # per request initial checks
        route(REQINIT);
        # NAT detection
        route(NAT);
```

Basics of Kamailio configuration.

Every Call Starts with

```
route {
      }
```

And end in this. there is some sub other function like

```
failure_route[FAIL_ONE] {
                        }
```

And

```
route[PSTN] {
            }
```

this two is very important for us.

Basics of Kamailio configuration.

To use Asterisk voicemail we have to route failed call to asterisk. to do so just add this in you failure_route.

```
    failure_route[FAIL_ONE] {
     ... ...... ..
            if (t_check_status("486|408|480")) {
                 sethostport("asterisk server host:5060");
                 append_branch();
                 t_relay();
            }
    }
```

if Asterisk and kamailio both installed in the same machine asterisk must be bind in a different port than 5060 and the port should be defined hare.

Basics of Kamailio configuration.

```
if (t_check_status("486|408|480")) {
```

In this clause 486 or 408 and 480 are SIP Response code.
this are

- 486 Busy Here
- 480 Temporarily Unavailable
- 408 Request Timeout

That's explain everything.

Basics of Kamailio configuration.

For Routing Calls to PSTN Gateways you just have to route calls that does't match in your local extension to another server which can handle PSTN or External SIP GW.

All number which doesn't match your local extension will be automatically routed to

```
route[PSTN] {

    }
```

Basics of Kamailio configuration.

```
route[PSTN] {
# route to PSTN dialed numbers starting with '+' or '00' or '0'
    if(!($rU=~"^(\+|00|0)[1-9][0-9]{3,20}$"))
        return;

    $ru = "sip:" + $rU + "@your pstn gw ip;

    route(RELAY);
    exit;
}
```

This way your calls will be routed to PSTN Gateway. which is also a asterisk server with TDM card or an external sip provider.

Expanding for Bigger Game....

Now you have a better sip server that can locally handle lot more call than your only asterisk server can handle.

But the problem is if you have everything in a same machine or have a single Asterisk server as PSTN Gateway how many call it can handle.

And how can you ensure redundancy.

Kamailio has several solution for that. but what I frequently used is dispatcher.

Expanding for Bigger Game....

You already seen how to add dispatcher module and set dispatcher file parameter. here is a simple dispatcher file:

dispatcher.list

```
# gateways
1 sip:yourgatewayip1:5060
1 sip:yourgatewayip2:5060
```

the first column is gateway set id, next is server address and port.
you can add as many set as you wish.

Expanding for Bigger Game....

In-place of `$ru = "sip:" + $rU + "@your pstn gw ip;`
add

```
ds_select_dst("1", "4");
```

in ds_select_dst() method first parameter is the gateway set id. and the second parameter is the algorithm used to select the destination address. here 4 means round-robin (next destination).

Expanding for Bigger Game....

All the algorithm are like this.

- "0" - hash over callid
- "1" - hash over from uri.
- "2" - hash over to uri.
- "3" - hash over request-uri.
- "4" - round-robin (next destination).
- "5" - hash over authorization-username (Proxy-Authorization or "normal" authorization). If no username is found, round robin is used.
- "6" - random (using rand()).
- "7" - hash over the content of PVs string. Note: This works only when the parameter hash_pvar is set.
- "X" - if the algorithm is not implemented, the first entry in set is chosen.

The outcome....

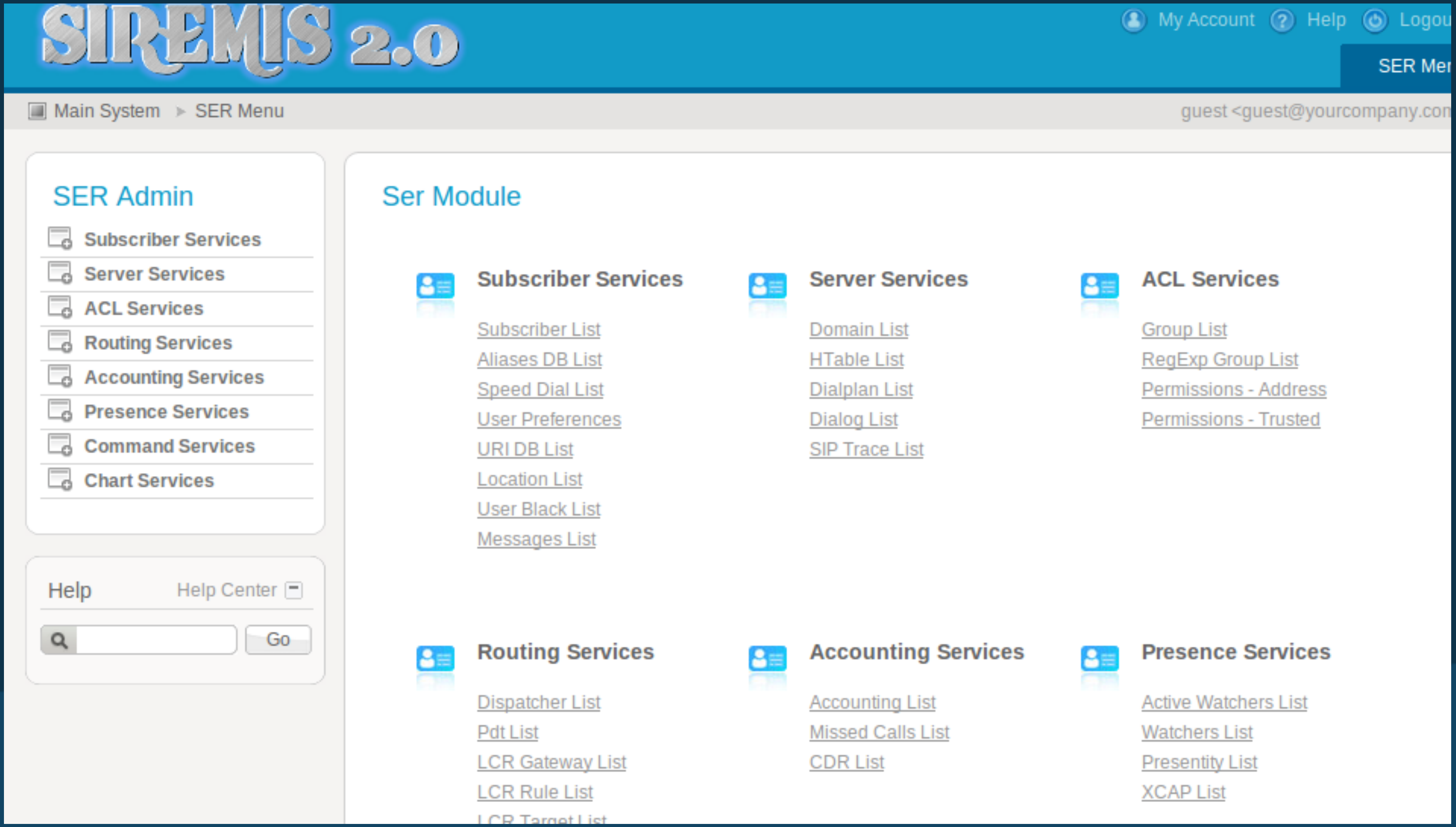
Now we know how to add multiple gateway.
And how we can distribute calls between them with balanced load.

By this way the gateways are not only expanded it also ensure redundancy and high availability.

kamailio server itself can be configured as HA in multiples server and extend its capacity.

Finally

Finally we can use graphical interface to manage kamailio by installing SIREMIS. from http://siremis.asipto.com/



Any Questions?

For more reading ....

Asterisk.
https://wiki.asterisk.org/wiki/display/AST/Home

Kamailio
http://www.kamailio.org/w/documentation/

SIREMIS
http://siremis.asipto.com/

Thank You.

keep in touch in http://gplus.to/sujon