OpenSER Admin Course

Voice System SRL

http://www.voice-system.ro

http://www.openser.org

High Availability

- Murphy's Law holds: Everything Can Go Wrong
- When we talk about system unavailability, the reasons are not only due failures, but also human errors in maintenance procedures!

- PSTN offers 99.999% availability
- What about VoIP/SIP ?

- In SIP things are more complicated as there is a huge puzzle:
  - puzzle of infrastructure – you as provider do not own/control the whole infrastructure
  - puzzle of specifications – SIP is RFC specified and interoperability may be a huge issue.
  - puzzle of softwares – PSTN is mainly about boxes, SIP is about software

High Availability - Levels

There are different levels where you have to look for High Availability:

- network connectivity
  - is the service still reachable via IP network
- hardware availability
  - hardware is a well know subject to crashes
- software availability
  - is the software still running at process level?
- software functionality
  - if still running, is the software doing what it is suppose to do?

Recipe

- High Availability is not a standalone component
  - it depends and interacts with a lot of other things.
- What you need to archive HA:
  - to have a replacement (hot or cold)
  - to monitor the platform
  - to detect the faults in ~ realtime
  - to automatically trigger the failover with a minimum of downtime

- most of HW failures are related to harddrives or network boards
- This is a fortunate case of failures as you do not have to specifically deal with. Why?
  - any HW failure translates into OS blocking or some software blocking
  - OS blocking leads to a network failure type as the IP connectivity is lost
  - software blocking leads to software failure

Conclusion: HW failures perfectly maps over other types of failure, so you do not need to handle them.

Refers to the must of having your platform reachable via network.

It does not mean the IP connection is related to a specific HW. The platform (as functionality spread over a set of machines) must be reachable via network.

We can talk about Network Availability at two levels:

- IP level – use IP related tools/mechanism to ensure the connectivity
- Application level – use smart detection and re-routing logics in high level applications

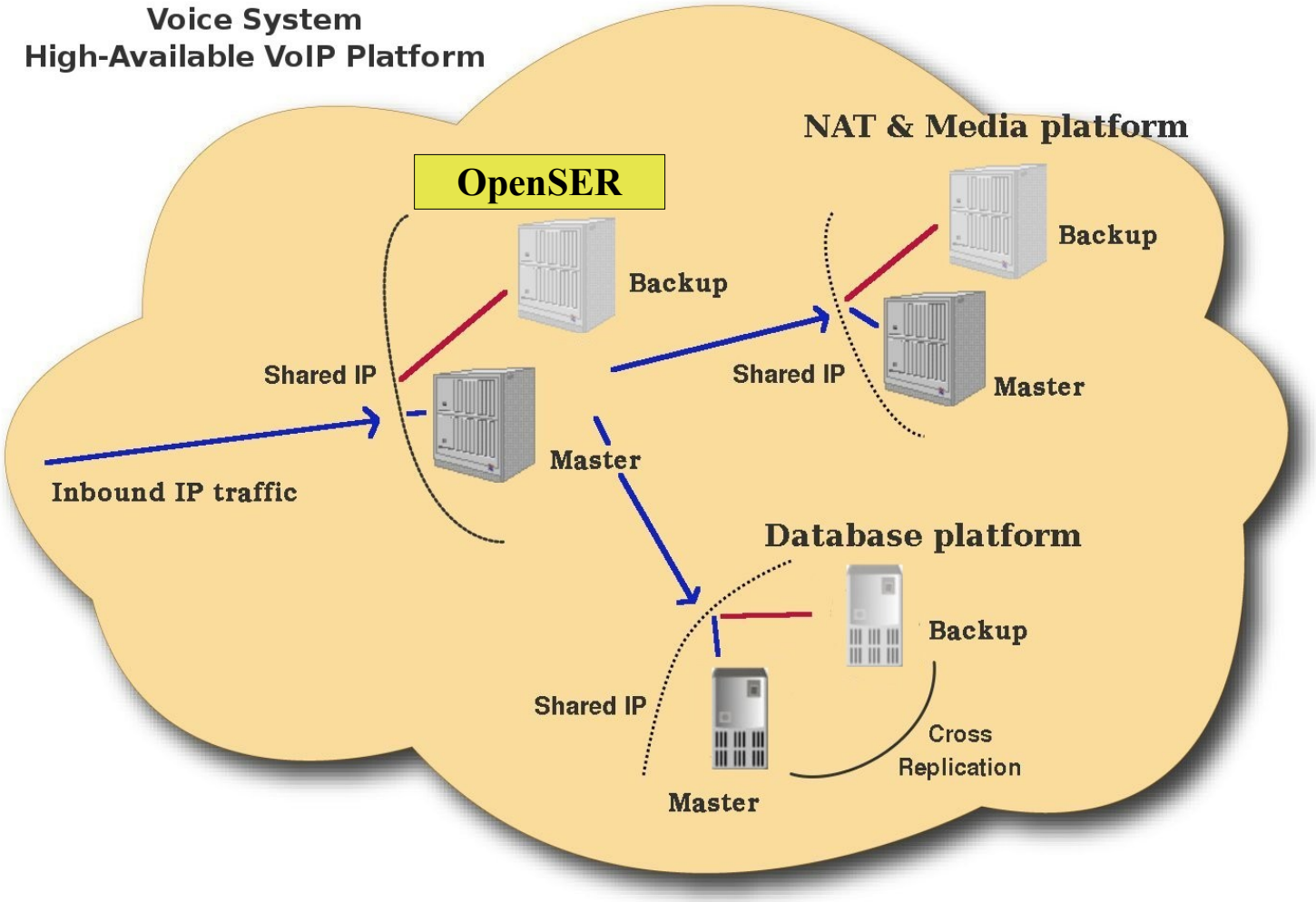
Network Availability at IP level

- IP managing – your platform is reachable all the time via the same IP address. Actually it's more about IP availability:
  - shared IP among several servers – VRRP
  - IP migration between servers – HeartBeat
- DNS managing – your platform is reachable all the time via the same DNS name.
  - DNS round robin
  - DNS SRV

Network Availability I – IP managing

- VRRP
  - more difficult to install and configure as OpenSER does not support interface changes at runtime.
  - stock VRRP cannot be used with OpenSER
  - allows hot backup configuration – a synchronized OpenSER ready to take over
  - the switching time is minimum (3 secs) as no software start is required
- HEARTBEAT
  - no special configuration/changes required
  - implements only cold back configuration – start the OpenSER instance when the server become active
  - the switching time is higher as time to warm (start applications) the newly only server is required.
  - simpler as now runtime sync. is required.

Network Availability I – VRRP showcase



If IP manging is done on the servers (platform) side, the DNS managing has impact on client (phones) side.

When using DNS there is a great concern regarding:

- delays in solving DNS queries
- delays in propagating DNS changes
- how the caller device uses the DNS information (RFC compliant) – does the device do DNS query for each requests or it does it only once as startup?
- routing in sequential request due Record Routing – as IP may change, DNS names must be used in RR also – this introduces an unwanted burn on the platform (revers DNS is required for each sequential request)

Network Availability at application level

Use a high-level application to transparently handle the connectivity among equivalent point:

- OpenSER for SIP routing – use blacklist, LCR, Dynamic Routing, Dispatcher modules to detect and alternatively route the SIP traffic.
- MySQL Cluster – ensure the data persistence across the Data Nodes in case of failure

This kind of Network Availability is suitable for HA inside the platform, as you have there a controlled environment and well defined architecture.

OpenSER can be deployed along dedicated tools for monitoring at process level.

The monitoring tool will run on the same machine with the monitored software.

Main task:

- see if the application runs (check at process level)
- if not running, try to restart the dead application.

Main utility:

- deals in a simple and efficient ways with the software failures (crashes)
- also it can be extended to monitor some critical parameters in order to see the healths of the applications. Ill apps can be rapidly cured by restart.

A well fitted tool is MONIT – a monitoring tool that have powerful capabilities when comes to monitoring, handling applications, triggering actions and sending notifications.

MONIT is able to provide critical data about the application (as process), like CPU and memory usage;

Also it can monitor non-application resources (like file systems);

It provides web interface for simpler provisioning.

Software functionality comes on top of software availability – once we know the application is running, we want to be sure it still does what it is supposed to.

The application's functionality is to be checked via periodic tests. A larger set of tests will ensure a better testing.

NOTE: too many and too often testing may decrees the platform output. Try to merge more functionalities in one test and make them simple.

Ex: periodical registration – it checks the OpenSER's overall capability to process SIP traffic; checks the DB interaction – for authentication and location); checks memory availability.

Tools:

- monit
  - it can be used for simple tests; it has build in support for testing the simple functionality of well know server – mysql, http, ftp, etc
  - also, custom but simple tests can be build on top of it
- nagios
  - more powerful than monit as it allows building of highly complex test scenarios.
  - also it can be integrated with other tools – like sipsak.

Replication

Hot backups

- When it comes to hot backups, we consider:
  - the application must already be running at failover time
  - the application must be up-to-date (from input data pov)
- The difficult task is to guarantee the synchronization (at runtime) between the active and stand-by server. This is essential in order to “present” to the end-user the same service/behavior after a failover.
- Unfortunately, for Media based services (Media Server, RTPproxy), synchronization of media sessions between active and stand-by server is not yet possible.

SIP synchronization

- In order to have two synchronized entities, we need to perform data replication from the active to the stand-by entity.
- What runtime data must be replicated for OpenSER?
  - everything stored in memory cache
    - configuration data
    - user location data
  - processing data (call information)

As OpenSER is only transaction stateful and not dialog stateful, there is no call data to be replicated. A call can start on active server and terminate on the stand-by as the proxy does not maintain any dialog related data.

At transaction level, the processing of a transaction may be interrupted and lost during a failover event. This is not an issue, as due the SIP protocol characteristics (like retransmissions), the processing of the lost transaction will be started from scratch on the new active machine.

SIP synchronization – configuration data

- all configuration data that it is loaded from database at startup and cached in memory during OpenSER's lifetime is usually provision via DB update and DB reload into cache.
- the reload is triggered via MI – Management Interface.
- it is important to apply the same set of reloads on both the active and stand-by server

A usual approach is to use the same database for the read-only data for both active and stand-by OpenSER. Issuing the same MI reload command on both will guarantee synchronization of the instances.

SIP synchronization – user location

- if the user location support is configured to use cache (due performance issues) it is required to implement data replication.
- this case is more complex as the data is learned by proxy at runtime; bulk replication is not possible as user location is highly time sensitive information.
- replication must be done in realtime, record by record, directly between the two instances of OpenSER (active and stand-by).
- for communication between the instances, a separate set of IP addresses may be required (the active IP may be shared)
- during user location replication, there is more data than contact URI that needs to be replicated:
  - received URI and local socket
  - NAT state
  - expire time

SIP synchronization – user location - showcase

```
# first see if it's a replicated REGISTER
if ( !(src_ip==LOCAL_PAIR_IP && src_port==LOCAL_PAIR_PORT) )
{
    # standard processing of a REGISTER request
    ......
    # replicate the REGISTER
    # add additional data to request
    .....
    add_sock_hdr("Local-Sock");
    force_send_socket( LOCAL_IP:LOCAL_PORT );
    t_replicate("sip:LOCAL_PAIR_IP:LOCAL_PAIR_PORT");
} else {
    # it's a replicated REGISTER
    # set the flag for retrieving sock_info from header
    setflag(XX);
    # save contact, but only in cache
    save("location","0x01");
}
```

As previously described, MySQL replication and HA can be achived by using MySQL cluster, but how to set and use a MySQL Clusters is not the goal of this presentation.

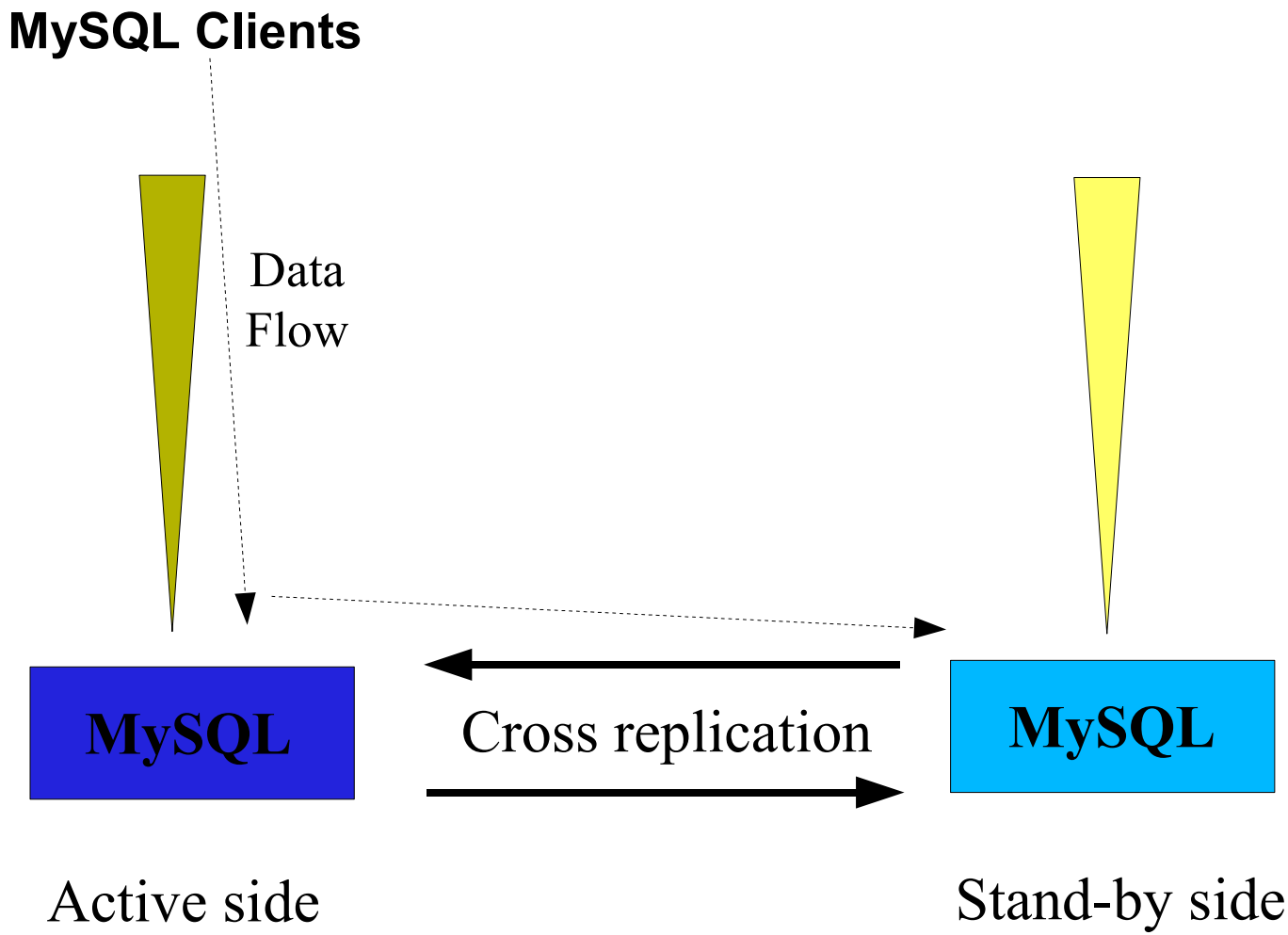
To achieve MySQL HA in a non-cluster configuration, you need to use mysql replication.

As it is not a backup scenario, but a HA one, any of the mysql server can be master or slave at a time, so all the mysql servers must be equivalent.

=> it must be a cross replication relation between the mysql servers.

MySQL synchronization II

- MySQL replication is not something difficult to configure, but it may be tricky to design.
- it requires a good understanding of the databases and data flow in order to eliminate any conflicts due replication.
- mysql replication needs special care when it comes to dealing with errors – by default, any error will stop the replication process. A large volume of data pending for replication may prevent the replication process to restart.


MySQL Clients
Data
Flow
MySQL
Cross replication
MySQL
Active side
Stand-by side

BREAK

- balancing algorithms
  - hash over callid
    - ensures that all requests within a dialog goes to same box
  - hash over from uri
    - ensures that all requests from same user goes to same box
  - hash over to uri
    - ensures that registrations of an AoR goes to same box
  - hash over request-uri
    - ensures that requests to same destination are processes by same box
  - round-robin (next destination)
    - well-known algorithm with pretty fair distribution

- method type
  - REGISTER
  - MESSAGE
  - INVITE, CANCEL, BYE, ACK
- service type
  - registrar
  - proxy
  - presence server
  - media gateway
- QoS
  - bandwidth
  - features
  - resource availability

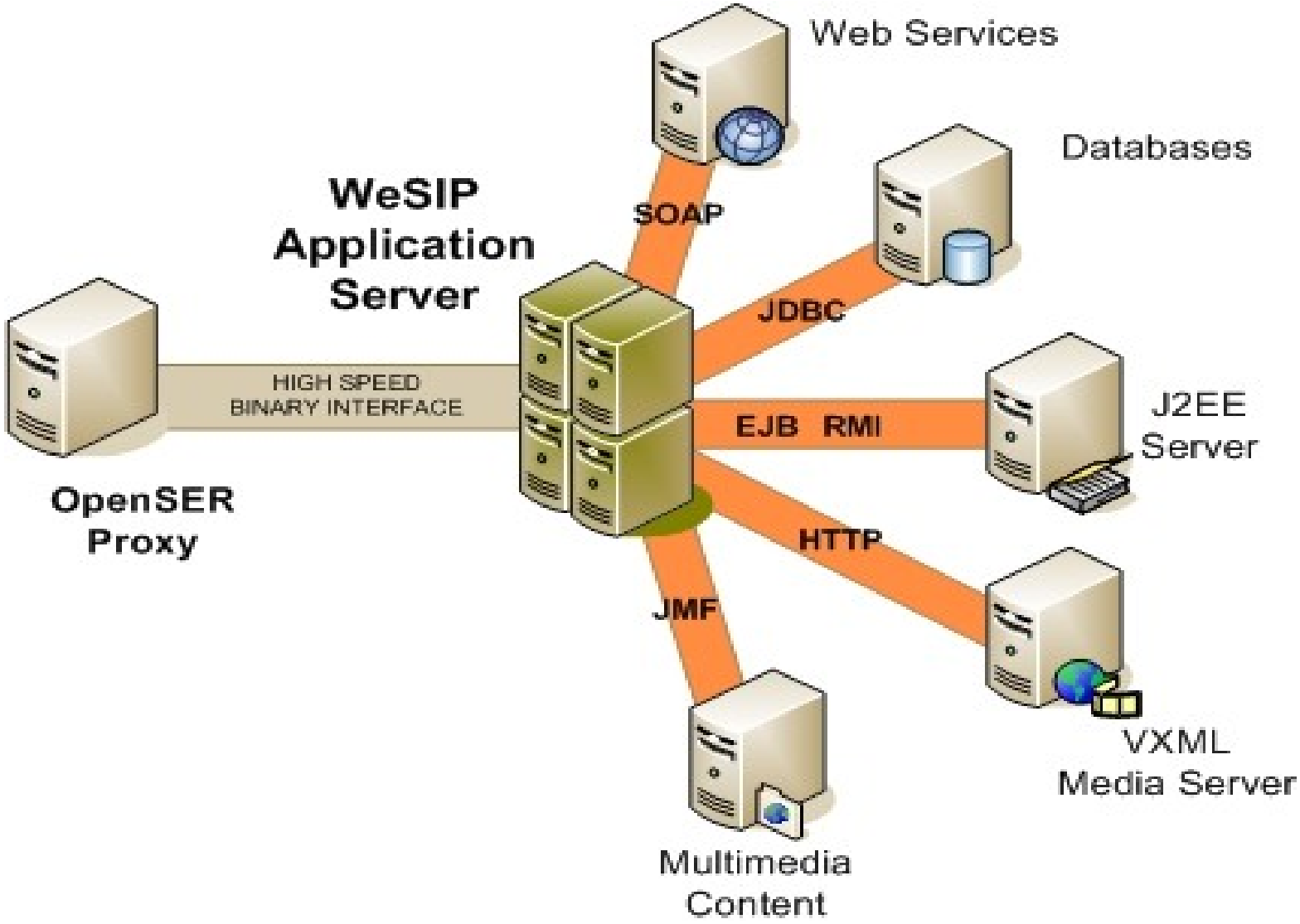
- distribution of logical entities
  - better scaling
  - resource allocation
  - security
  - performances
- localization
  - qos
  - languages
  - accommodation
- robusness
  - failover
  - disaster recovery

BREAK

- Perl API
  - OpenSER Perl Library
    - OpenSER::Message
    - OpenSER::URI
    - OpenSER::AVP
    - OpenSER::Utils
    - OpenSER::LDAPUtils
    - OpenSER::Constants
  - exported most of openser-available functions and structures to Perl
  - write Perl applications and embed execution in openser config file

```
my $ruri = $m->getRURI();
OpenSER::log(L_INFO, "from a Perl script");
```

- Java SIP Servlets – WeSIP
  - adds a J2EE layer to OpenSER
  - you can use the java programming language
  - huge set of available libraries, and the family of J2EE facilities
  - like SOAP, EJB or JDBC, to develop your SIP applications in the form of SIP Servlets
  - integration with media servers, content sources or even IMS entities like the HSS
  - tight integration with HTTP/HTML
    - click-to-dial
    - call status
  - ability to create IP PBX/Centrex features


Web Services
Databases
WeSIP
Application
Server
SOAP
JDBC
HIGH SPEED
BINARY INTERFACE
EJB RMI
J2EE
Server
OpenSER
Proxy
HTTP
JMF
VXML
Media Server
Multimedia
Content

- end-to-end
  - default, just basic SIP routing
- client-to-server
  - rich presence server
  - XCAP
  - SLA/BLA
  - user location presence
  - programmable interface to presence server
  - simple-xmpp presence
  - design for distribution
  - easy to add new XML-based presence info aggregation
  - message waiting indication
  - RLS


Ready
Your username is:admin
X-Lite
XFER
HOLD
RECORD
AA
AC
DND
CONF
FLASH
REDIAL
MUTE
SPEAKER PHONE
CLEAR
Powered by
COUNTERPATH
Calls & Contacts
Detach
Available
Contacts
Name
Friends
Home
Work
(No Group)
system - U:7/C:0.8/M:...
system - U:7/C:0.8/M:22052k/L:0.19,0.08,0.08
SoftPhone: system@192.168.2.5
Calls


Ready
Your username is:admin
X-Lite
1
2
XFER
HOLD
RECORD
AA
AC
DND
CONF
FLASH
REDIAL
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0
#
CLEAR
MUTE
SPEAKER PHONE
Powered by COUNTERPATH
Calls & Contacts
Detach
Available
Contacts
Name
Friends
Home
Work
(No Group)
system - U:1/T:0/R:21
system - U:1/T:0/R:21
SoftPhone: system@192.168.2.5
Contacts
Calls

- im chatting
  - default sip routing
  - no matter of content/type
- offline storage
  - msilo module
  - store message if user offline
  - deliver when user becomes online
- im conferencing
  - IRC style
  - admin privileges
  - public/private chat rooms

- gateway to sms
  - sms module
  - simple gateway via AT modem
- gateway to xmpp
  - instant messaging
  - presence
  - voice?!?!
- gateway to ...
  - email – via exec module
  - smpp – via perl and kannel
  - icq, aol, yahoo, msn, ... - via jabber module

BREAK