FreeSWITCH SIP and WebRTC

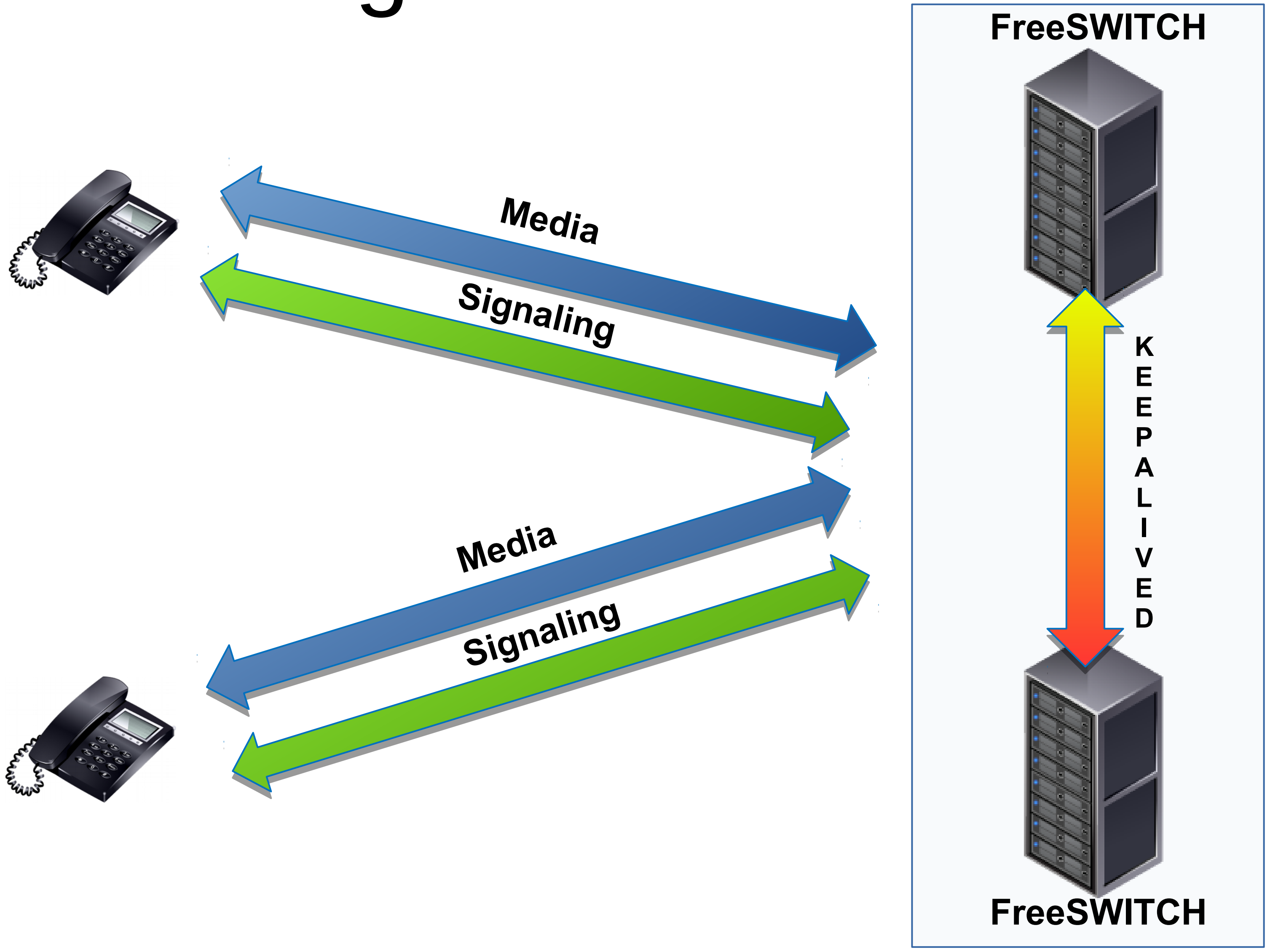
Load Balancing and High Availability in Real World

Giovanni Maruzzelli
gmaruzz@OpenTelecom.IT

High Availability: **Double It ALL**

- LAN Switch and Cabling
- FreeSWITCH Server
  - Virtual (Floating) IP address
  - HeartBeat, Keepalived, Corosync
- File System
  - DRBD
  - Rsync
  - BTSync
  - GlusterFS
- Database
  - Master-Master (Active-Passive)

Two Single FreeSWITCHes



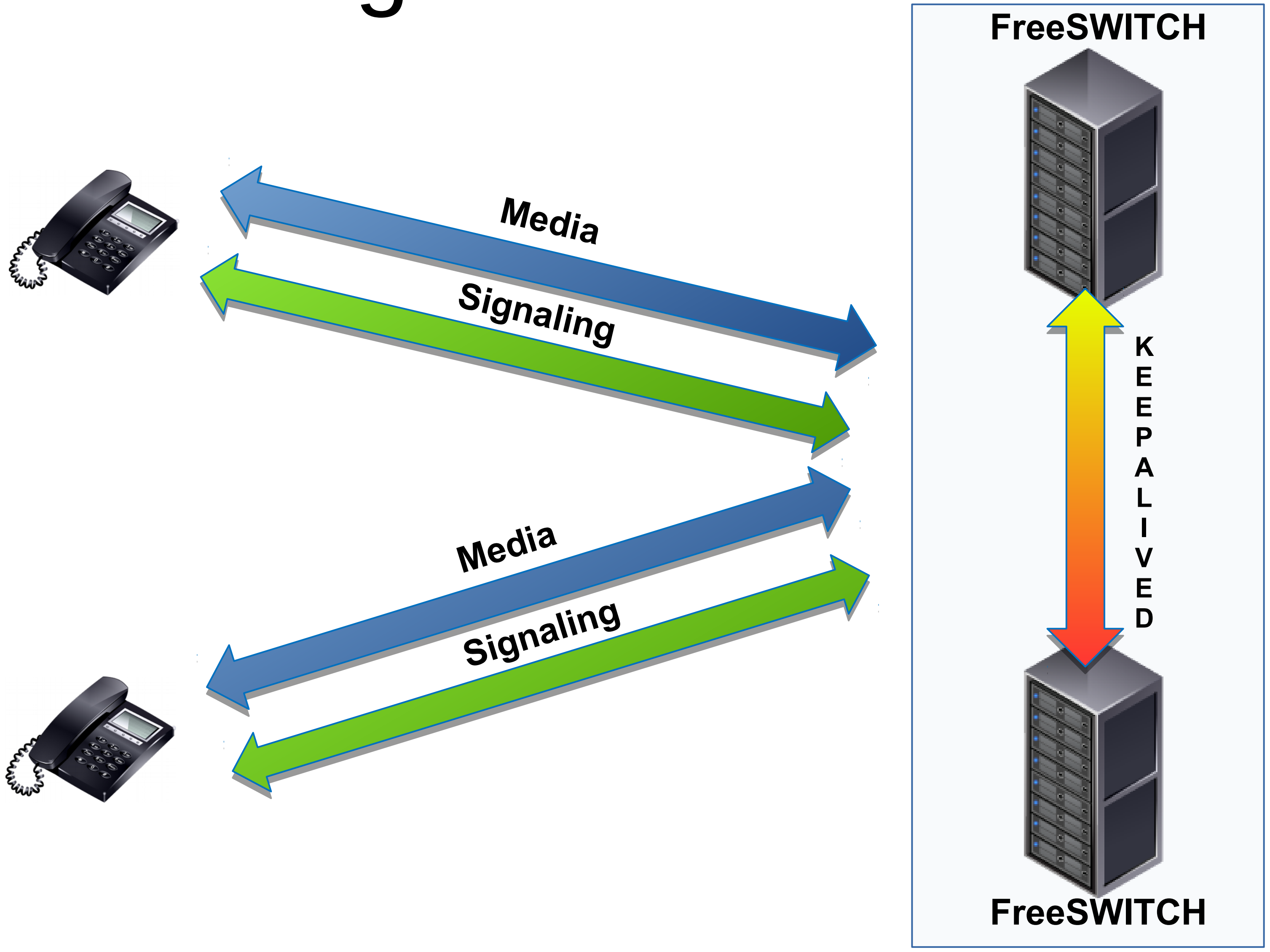
Two Single FreeSWITCHes

- Two Single FreeSWITCHes: **ACTIVE - PASSIVE**
  - Rsync or DRBD or BTSync or GlusterFS:
    - Has its own Configuration
    - Keeps its own State
    - Writes and Reads Voice Mail
  - Manages NAT Handling (Media and Signaling)
  - Mixes Conference Participants' Media
  - Parks and Unparks Calls
  - Manages Queues and ACDs

Two Single FreeSWITCHes

- **One BIG FS Machine is Constantly IDLE**
- BIG FS IDLE = $$$
- After a while you don't know if it will work at all
- You will probably start using the IDLE machine for some small things, then some other, and then…
- Scales Only Vertically = $$$$$

Two Single FreeSWITCHes

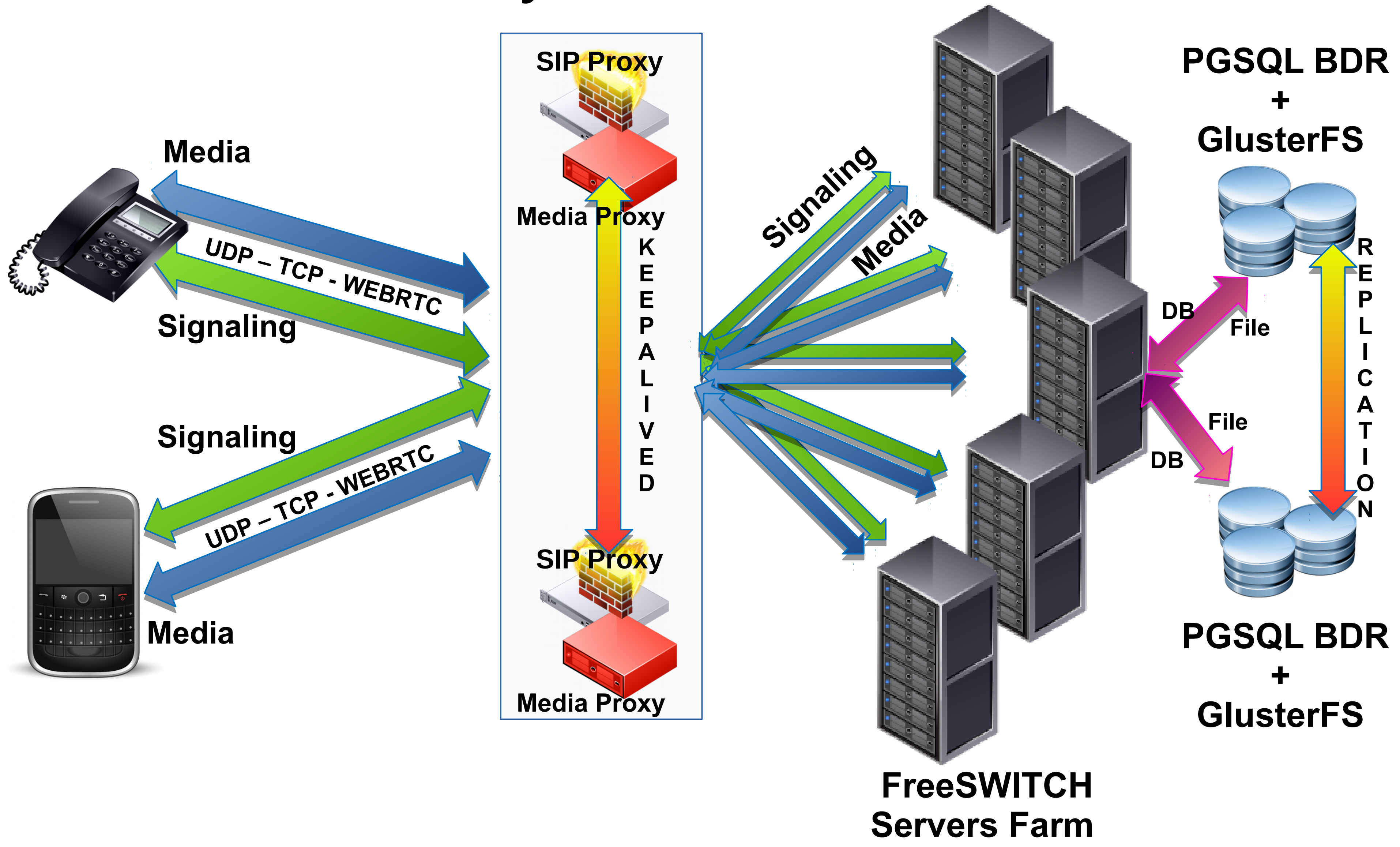


Many FreeSWITCHes

- **ALL FS Boxes are Constantly ACTIVE**
  (and most other boxes are too)
  - HA Database:
    - Keeps its own State
  - Distributed FileSystem:
    - Has its own Configuration
    - Writes and Reads Voice Mails
  - HA Load Balancers and Proxies:
    - Manages NAT Handling (RTP Media and SIP Signaling)
  - Partitioning (with Failover):
    - Mixes Conference Participants' Media
    - Parks and Unparks Calls
    - Manages Queues and ACDs

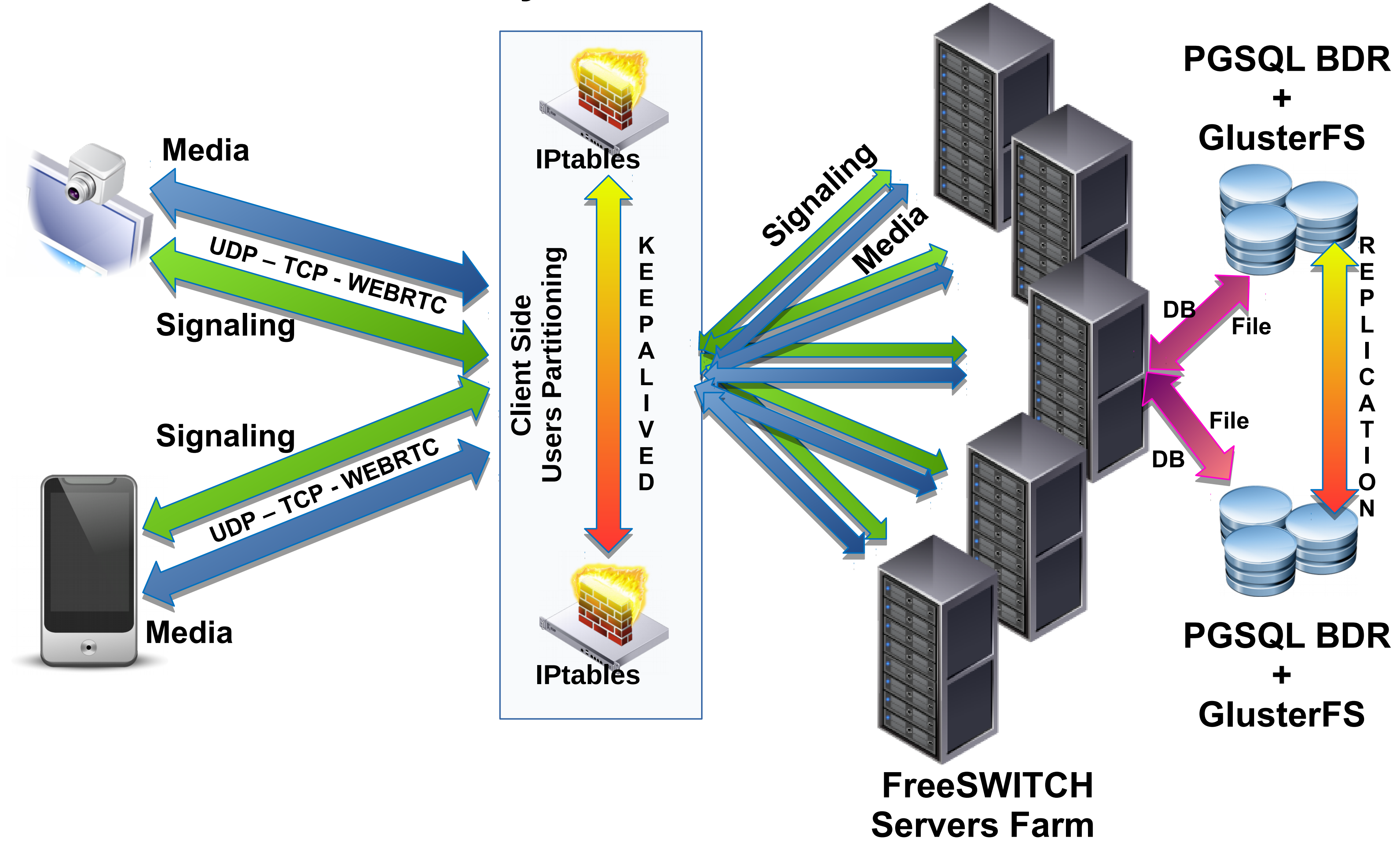
SIP

Many FreeSWITCHes



VERTO

Many FreeSWITCHes



Many FreeSWITCHes

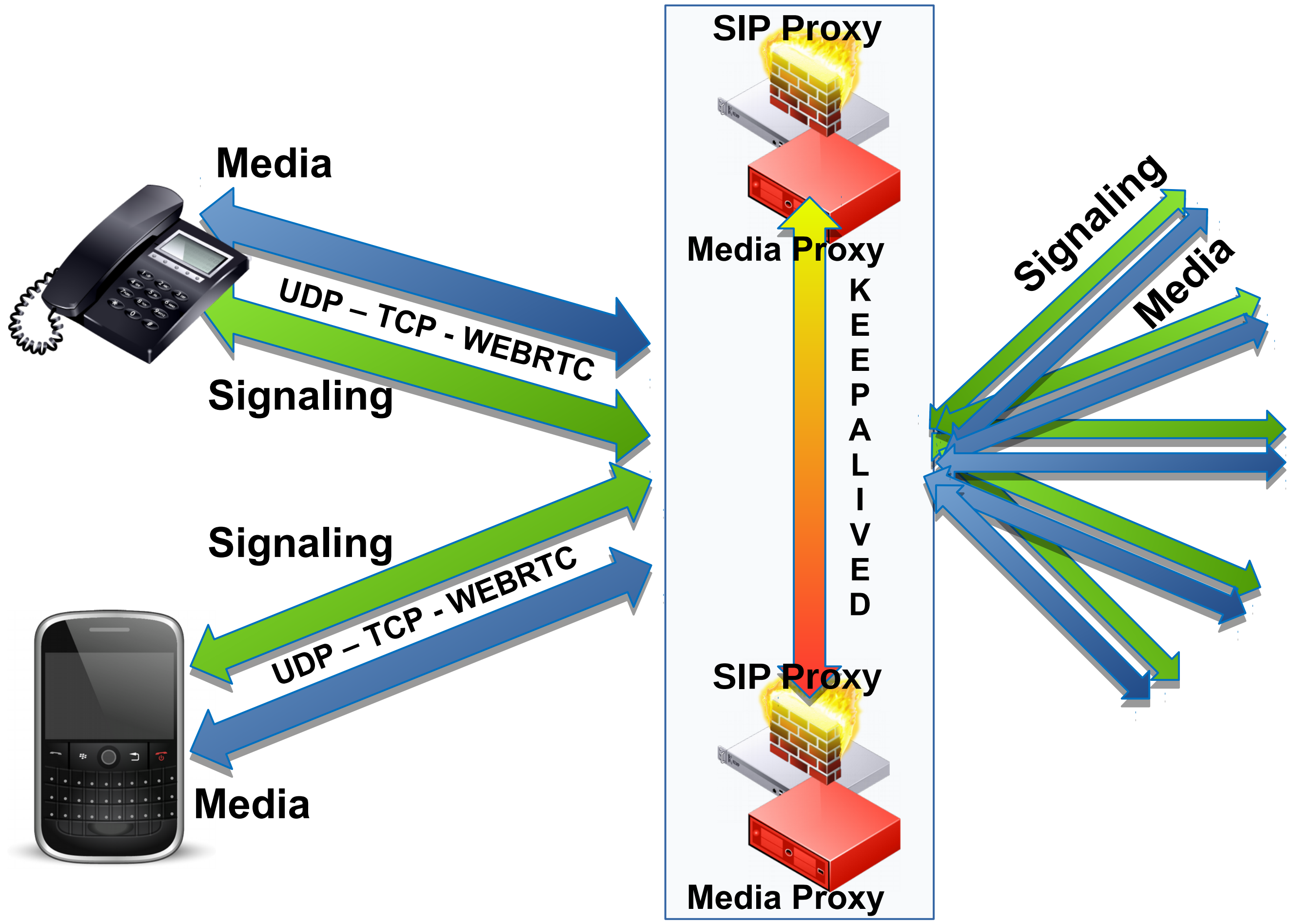
- One Load Balancer is **Constantly IDLE**
- LITTLE LB IDLE = ₵₵₵
- Entire platform is constantly exercised
- Scales Horizontally = ₵₵₵

SIP and NAT

- Client is behind NAT
- Client sends from its own IP:port a REGISTER request to Location Server IP:port, and in doing so it opens a pinhole in the NAT, waiting for server's answer
- NAT pinhole is only able to receive packets from same IP:port couple (Client/Server) it was open by, and for a limited period of time (30 seconds?)
- Location Server sends periodically from same IP:port an OPTIONS message to Client IP:port, Client answers, and in doing so it maintains the pinhole open (FS sends each 23 secs)
- When there is an incoming call for Client, Server sends the INVITE from same IP:port to Client IP:port

SIP

Load Balancing and Proxies



Where to put the **SIP** Registrar

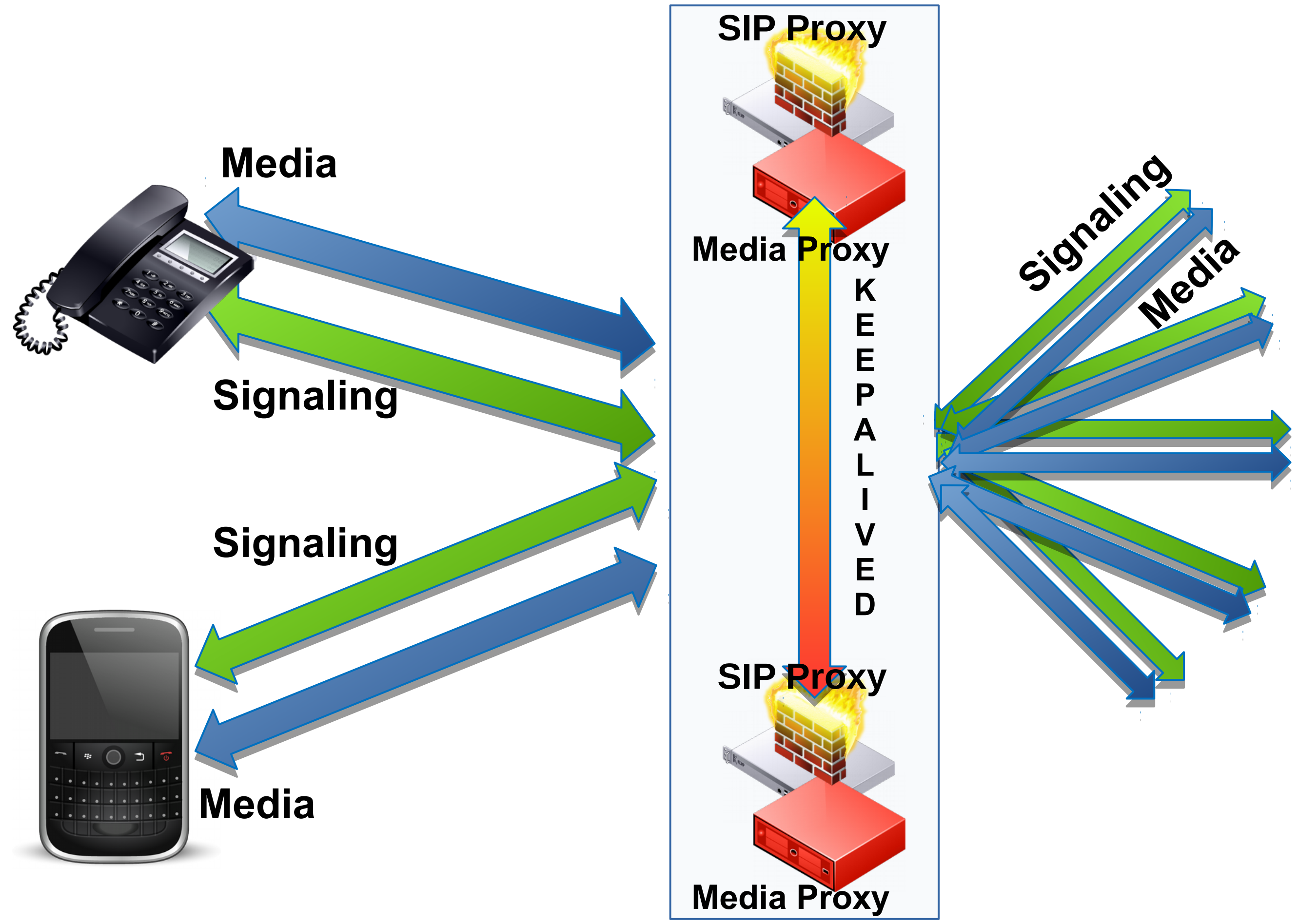
- **ON LB (SIP Proxy) MACHINE**, directly interacting with Clients
  - REGISTER and NAT Keepalive (OPTIONS, NOTIFY) are high volume, low load transactions
  - One robust box (in active-passive HA) will be able to serve tens of thousands clients
  - This is the most straightforward topology
- **ON FreeSWITCH MACHINES**, load balanced by LB
  - FreeSWITCHes act as registrars, load balanced, all using the same database
  - Need to record on which individual FreeSWITCH a client is registered, and send him calls from it
  - This topology is similar to a single FreeSWITCH

SIP Call Distribution: DISPATCHER & LOAD BALANCER

- SIP Proxy can be used for relaying requests to multiple boxes using “static” algorithms (eg: round robin or weighted) or “dynamic” algorithms (that take care of actual number of active calls on each machine)
- All proxy's algorithms are able to “ping” destinations, retry on failed destination, disable the failed box from list, and re-enable it when is back in order

SIP

Load Balancing and Proxies



SIP NAT & Media Relaying

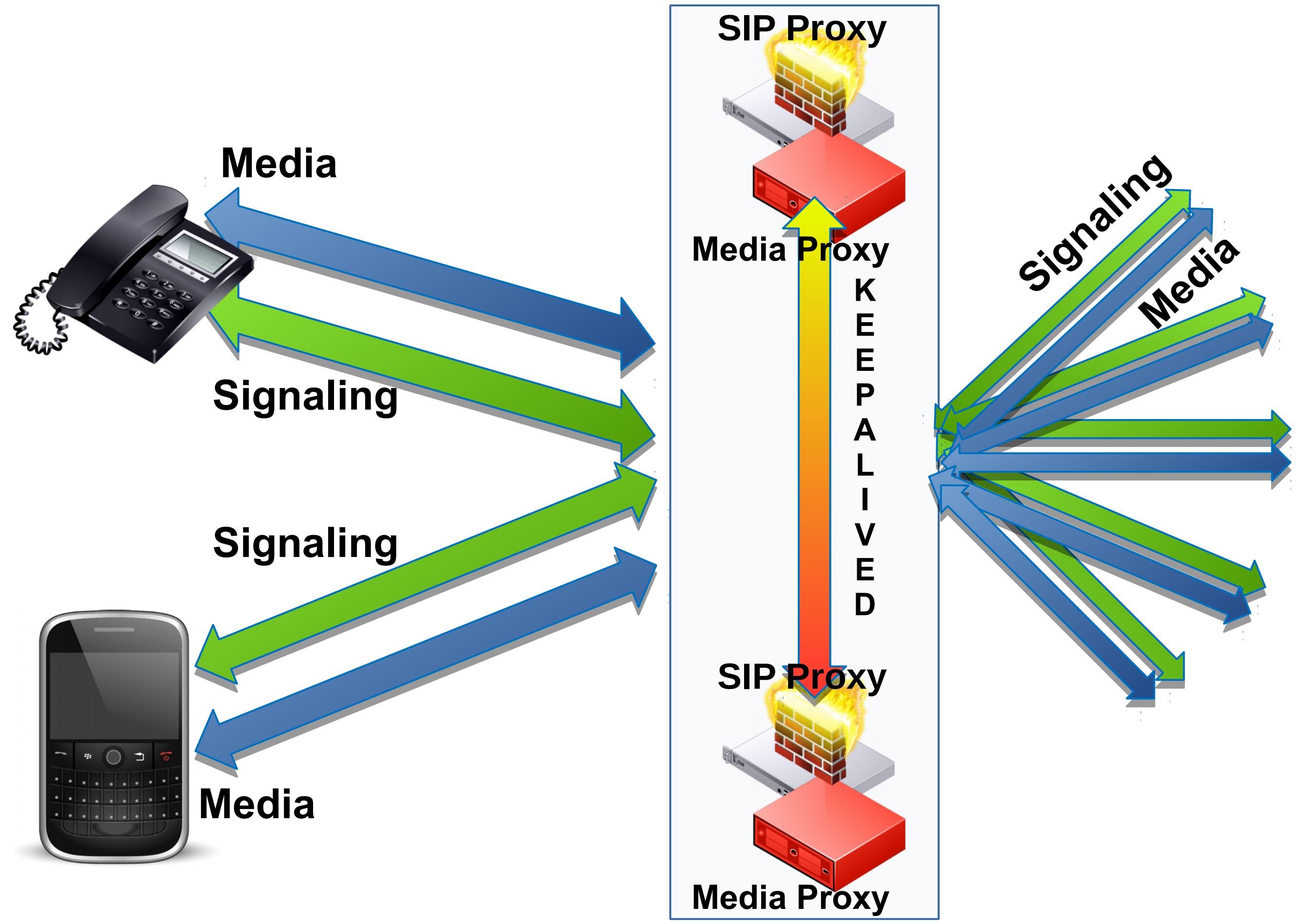
- There are special cases of clients behind NATs that cannot directly sends packets to each other. In those cases ONLY way for them to communicate is via the mediation of a server
- Also, you need to relay media in any case, if you're load balancing servers that are not directly reachable from clients

SIP Media Relaying

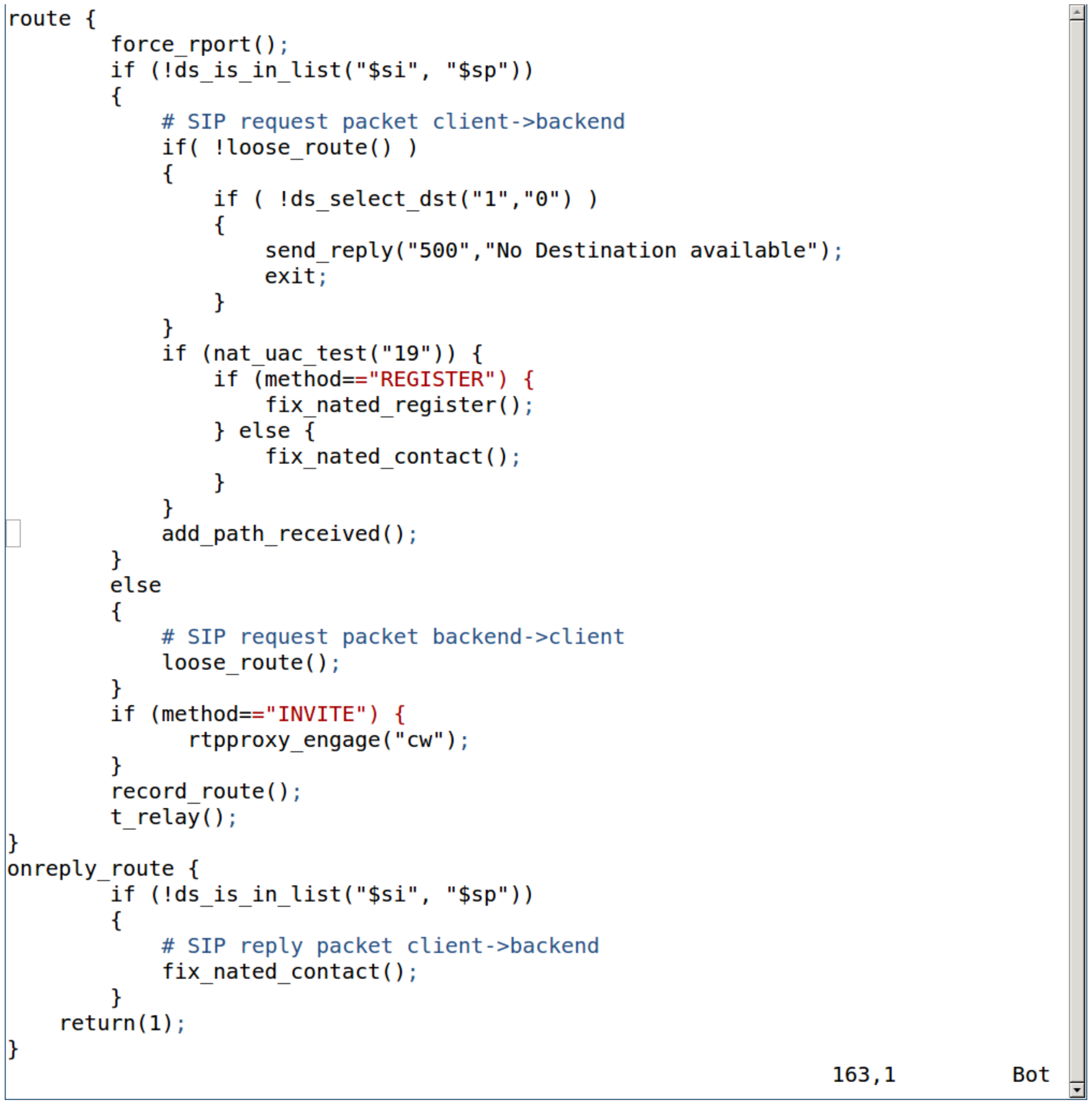
- SIP (signaling) proxy has nothing to do with media flow, it does not touch RTP
- It can modify SIP headers, and SDP bodies, so clients behind restrictive NATs will use a third party as a relay, and it can pass commands to that relay (eg: so the relay knows which client must be relayed to which)
- Original relay software is “Rtpproxy”
- More recent and advanced (eg: kernel space, etc):
  - MediaProxy
  - RtpEngine
- All of them can scale indefinitely

SIP

Load Balancing and Proxies



Adapted from: https://txlab.wordpress.com/tag/xlab1/ pass-through proxy by Stanislav Sinyagin



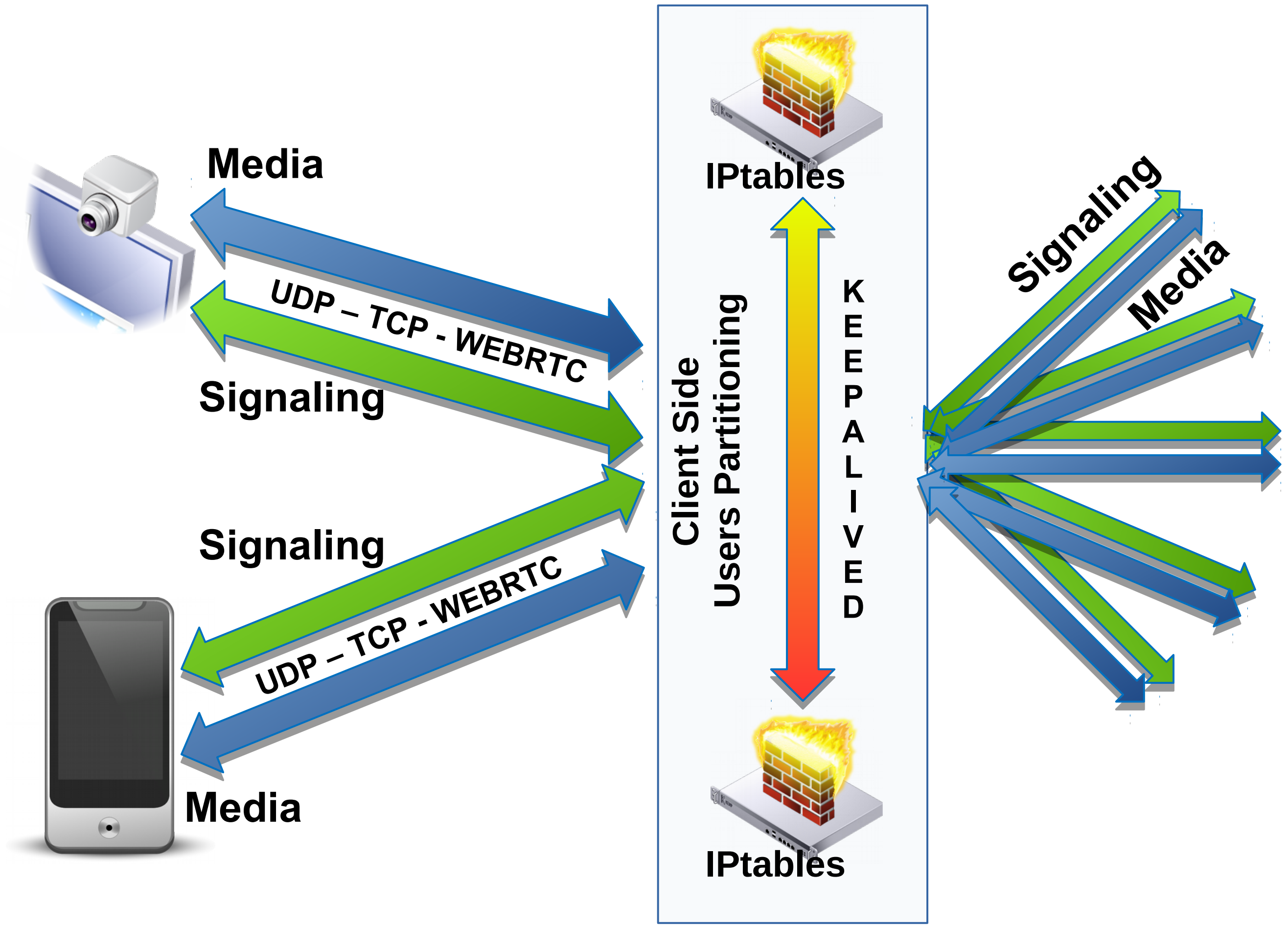
```
route {
        force_rport();
        if (!ds_is_in_list("$si", "$sp"))
        {
            # SIP request packet client->backend
            if( !loose_route() )
            {
                if ( !ds_select_dst("1","0") )
                {
                    send_reply("500","No Destination available");
                    exit;
                }
            }
            if (nat_uac_test("19")) {
                if (method=="REGISTER") {
                    fix_nated_register();
                } else {
                    fix_nated_contact();
                }
            }
            add_path_received();
        }
        else
        {
            # SIP request packet backend->client
            loose_route();
        }
        if (method=="INVITE") {
              rtpproxy_engage("cw");
        }
        record_route();
        t_relay();
}
onreply_route {
        if (!ds_is_in_list("$si", "$sp"))
        {
            # SIP reply packet client->backend
            fix_nated_contact();
        }
    return(1);
}
                                                              163,1          Bot
```

VERTO and NAT

ICE

VERTO

Load Balancing and Proxies

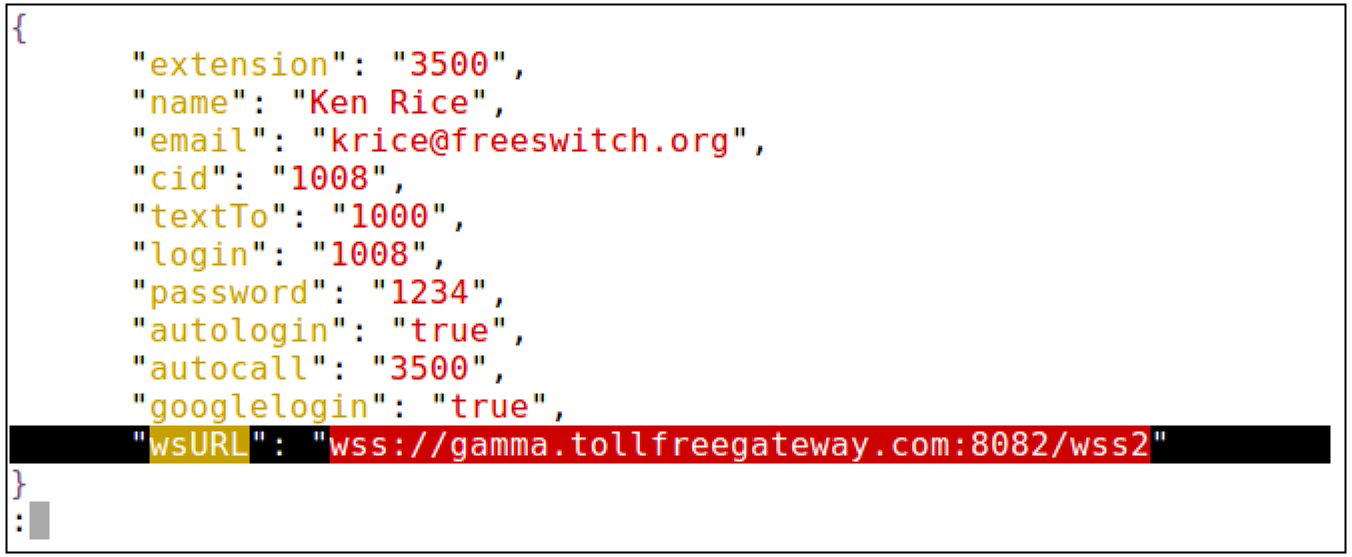


VERTO User Partitioning

- **VERTO**, at this moment, has **NO TRUNKING**
  - Each FreeSWITCH Server is a VERTO Island!
  - As of today, you use **SIP to Trunk** from one FS VERTO server to another VERTO server

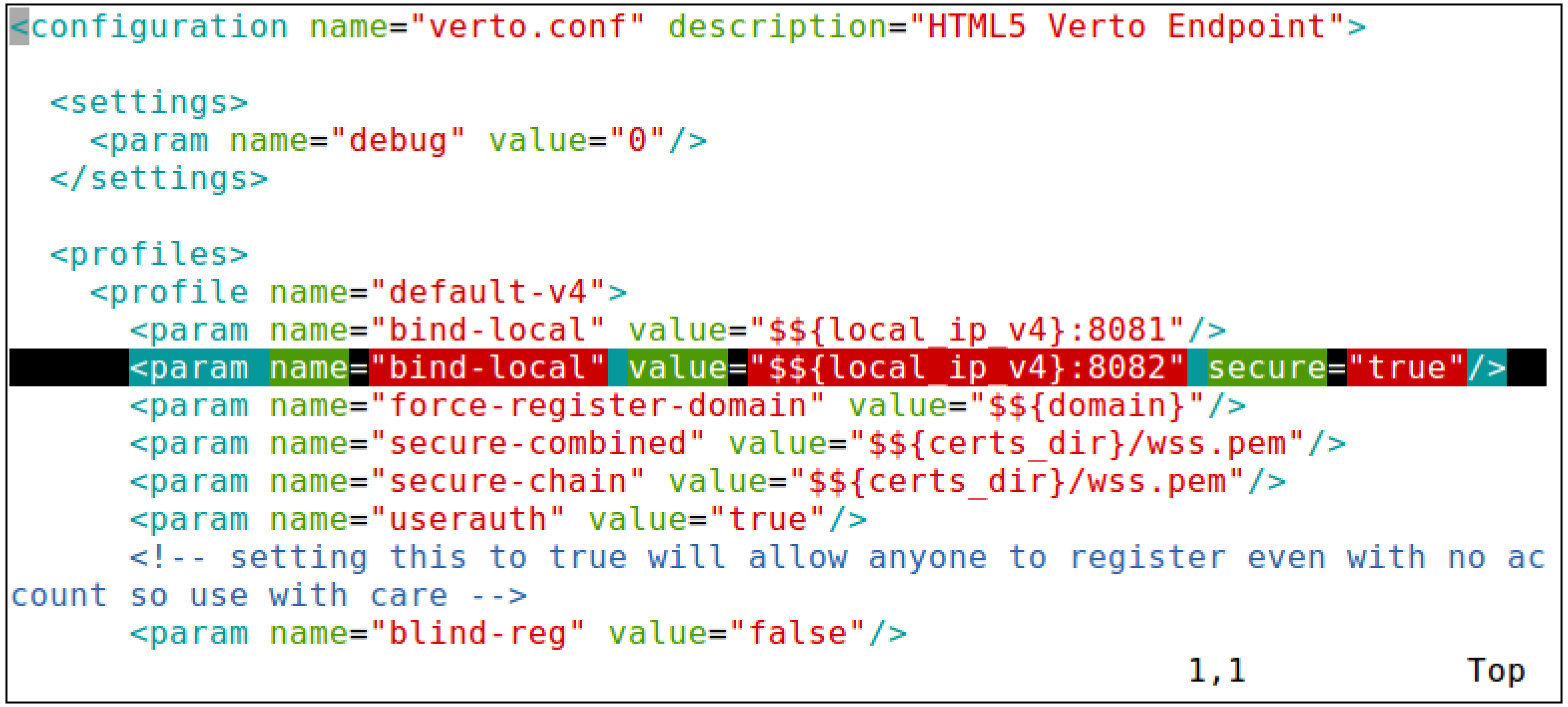
- **VERTO**, at this moment, has **no external** “VERTO **proxies**” and “VERTO **registrars**”
  - VERTO users (extensions) atm must be partitioned at client side
  - Client is under our control! (is a web page!)
  - Each users partition (by domain and/or by extension) is sent to a specific FS server via port forwarding

VERTO Client IP PORT



```
{
        "extension": "3500",
        "name": "Ken Rice",
        "email": "krice@freeswitch.org",
        "cid": "1008",
        "textTo": "1000",
        "login": "1008",
        "password": "1234",
        "autologin": "true",
        "autocall": "3500",
        "googlelogin": "true",
        "wsURL": "wss://gamma.tollfreegateway.com:8082/wss2"
}
:
```

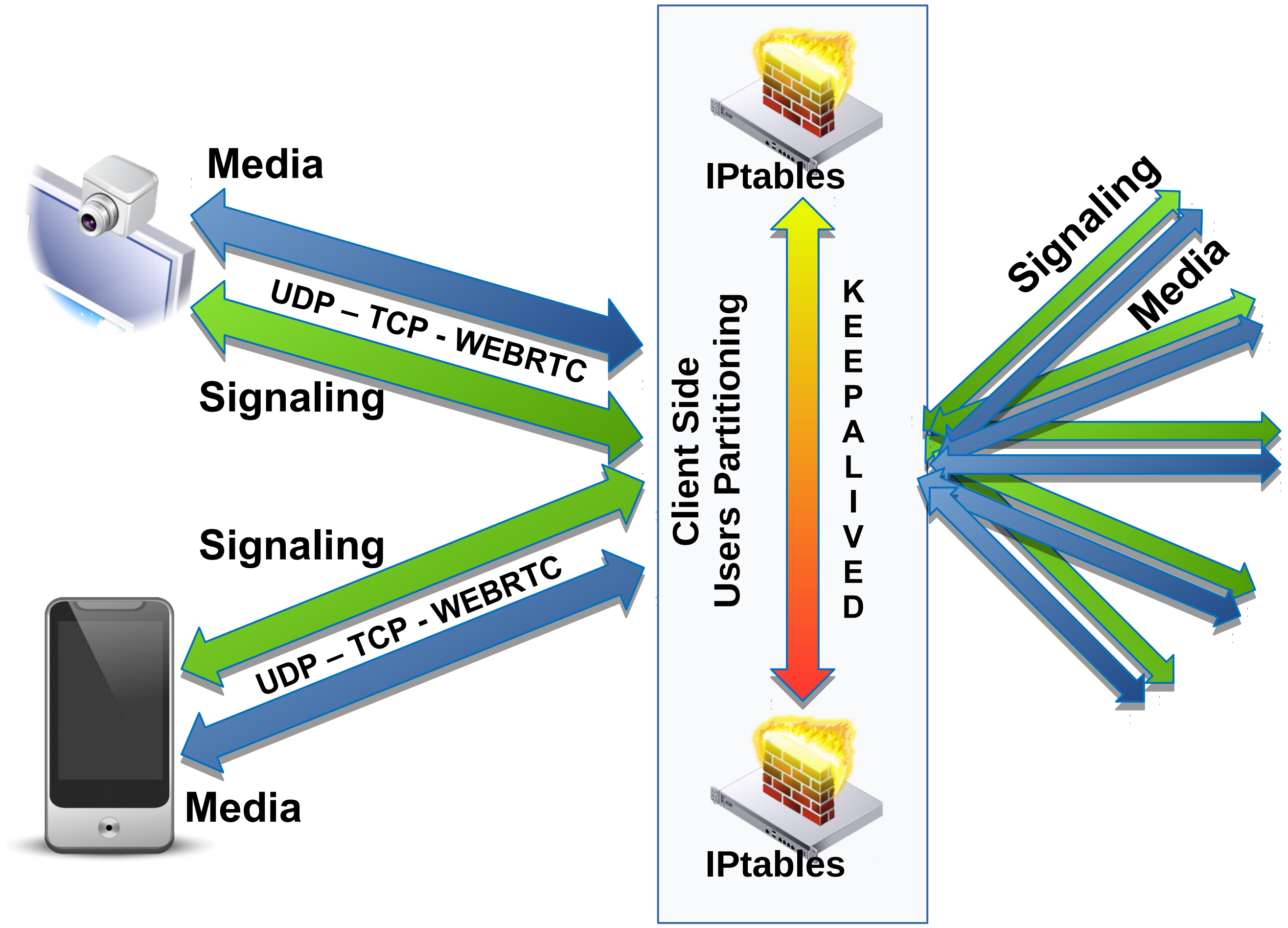
VERTO Server IP PORT



```
<configuration name="verto.conf" description="HTML5 Verto Endpoint">

  <settings>
    <param name="debug" value="0"/>
  </settings>

  <profiles>
    <profile name="default-v4">
      <param name="bind-local" value="$${local_ip_v4}:8081"/>
      <param name="bind-local" value="$${local_ip_v4}:8082" secure="true"/>
      <param name="force-register-domain" value="$${domain}"/>
      <param name="secure-combined" value="$${certs_dir}/wss.pem"/>
      <param name="secure-chain" value="$${certs_dir}/wss.pem"/>
      <param name="userauth" value="true"/>
      <!-- setting this to true will allow anyone to register even with no ac
count so use with care -->
      <param name="blind-reg" value="false"/>
                                                                  1,1           Top
```

VERTO

Load Balancing and Proxies

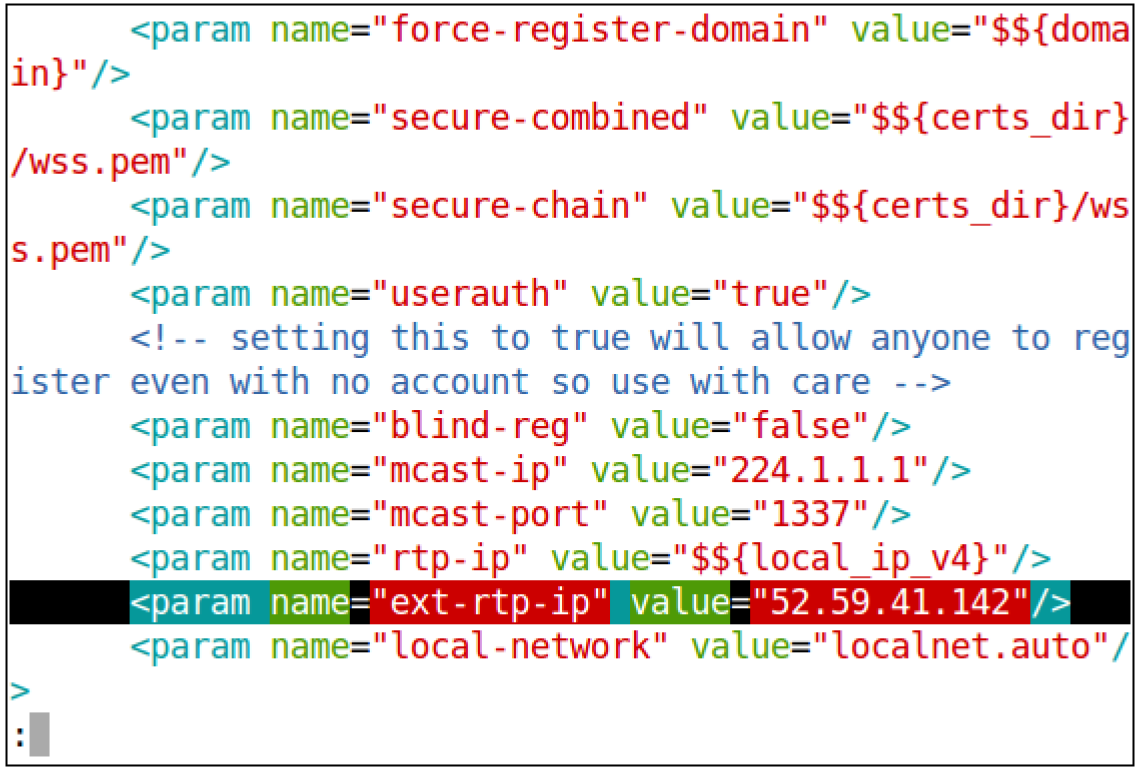


VERTO Call Distribution:

RTP IP, IPTables & IP Ranges

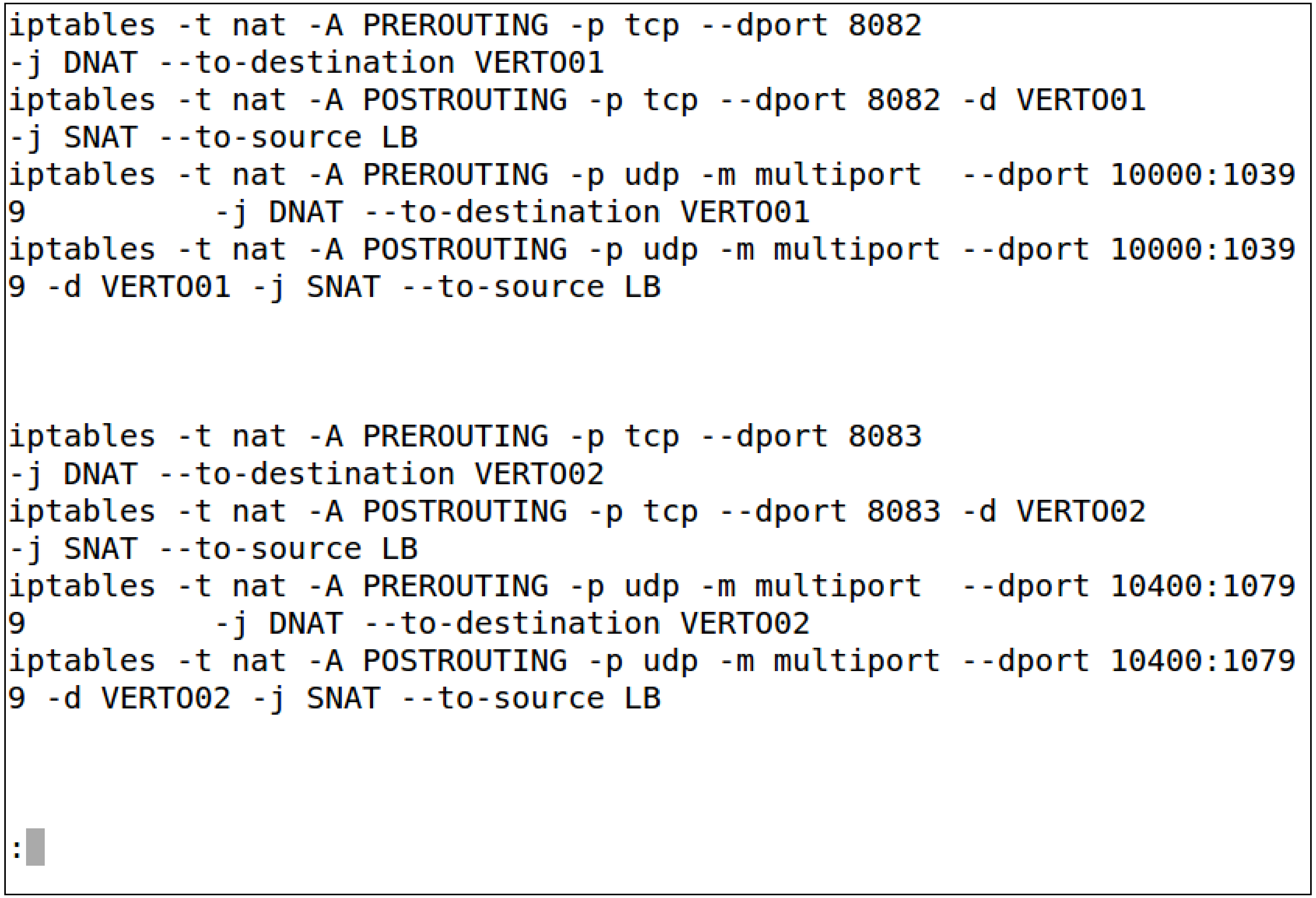
- All FreeSWITCH servers have ext-rtp-ip set to LB address in verto.conf.xml
- Each FreeSWITCH server has its own range of RTP ports set in switch.conf.xml
- IPTables will forward RTP back and forth from LB to the correct FreeSWITCH
- If a FreeSWITCH server dies, clients will automatically reconnect to the new instance of that server (that's the beauty of TCP wss)

VERTO RTP IP



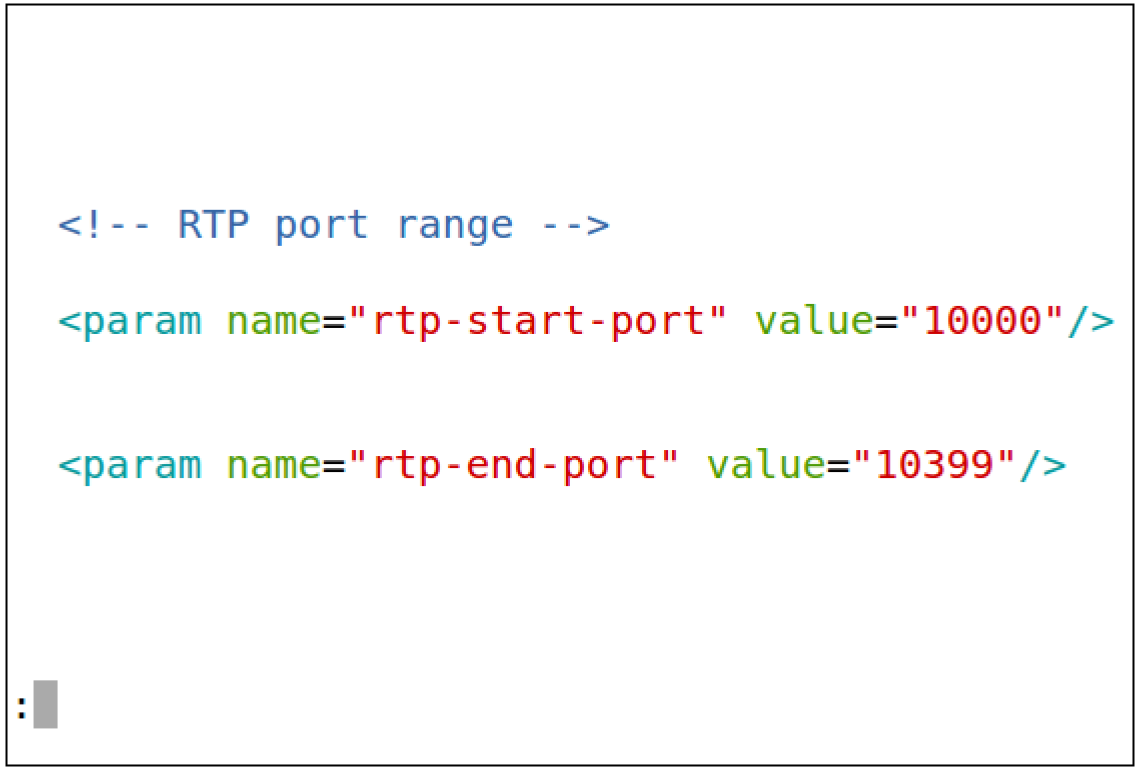
```
      <param name="force-register-domain" value="$${doma
in}"/>
      <param name="secure-combined" value="$${certs_dir}
/wss.pem"/>
      <param name="secure-chain" value="$${certs_dir}/ws
s.pem"/>
      <param name="userauth" value="true"/>
      <!-- setting this to true will allow anyone to reg
ister even with no account so use with care -->
      <param name="blind-reg" value="false"/>
      <param name="mcast-ip" value="224.1.1.1"/>
      <param name="mcast-port" value="1337"/>
      <param name="rtp-ip" value="$${local_ip_v4}"/>
      <param name="ext-rtp-ip" value="52.59.41.142"/>
      <param name="local-network" value="localnet.auto"/
>
:
```

VERTO IPTables



```
iptables -t nat -A PREROUTING -p tcp --dport 8082
-j DNAT --to-destination VERTO01
iptables -t nat -A POSTROUTING -p tcp --dport 8082 -d VERTO01
-j SNAT --to-source LB
iptables -t nat -A PREROUTING -p udp -m multiport  --dport 10000:1039
9          -j DNAT --to-destination VERTO01
iptables -t nat -A POSTROUTING -p udp -m multiport --dport 10000:1039
9 -d VERTO01 -j SNAT --to-source LB



iptables -t nat -A PREROUTING -p tcp --dport 8083
-j DNAT --to-destination VERTO02
iptables -t nat -A POSTROUTING -p tcp --dport 8083 -d VERTO02
-j SNAT --to-source LB
iptables -t nat -A PREROUTING -p udp -m multiport  --dport 10400:1079
9          -j DNAT --to-destination VERTO02
iptables -t nat -A POSTROUTING -p udp -m multiport --dport 10400:1079
9 -d VERTO02 -j SNAT --to-source LB



:
```

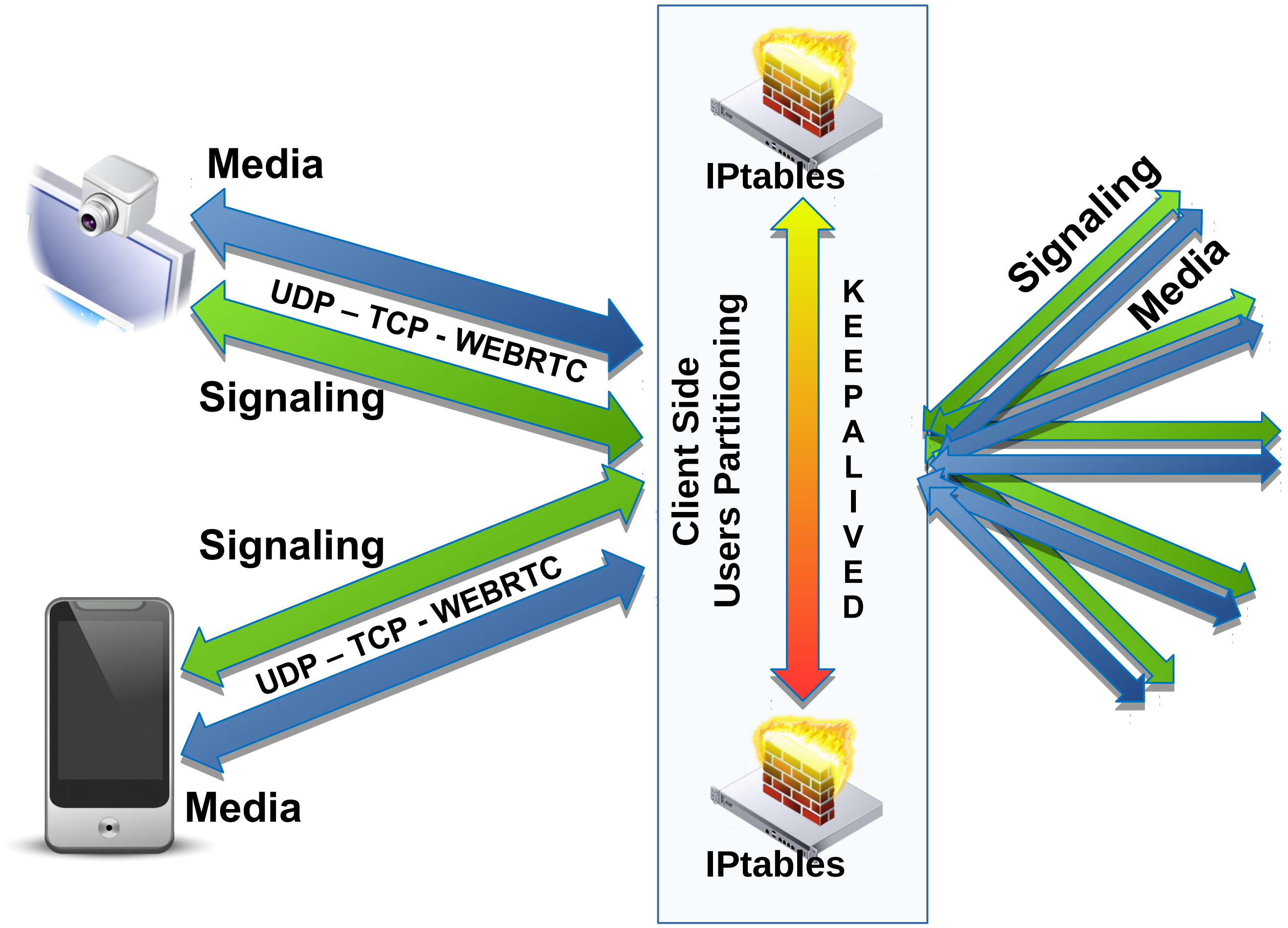
VERTO RTP Range



```
<!-- RTP port range -->
<param name="rtp-start-port" value="10000"/>

<param name="rtp-end-port" value="10399"/>


:
```

VERTO

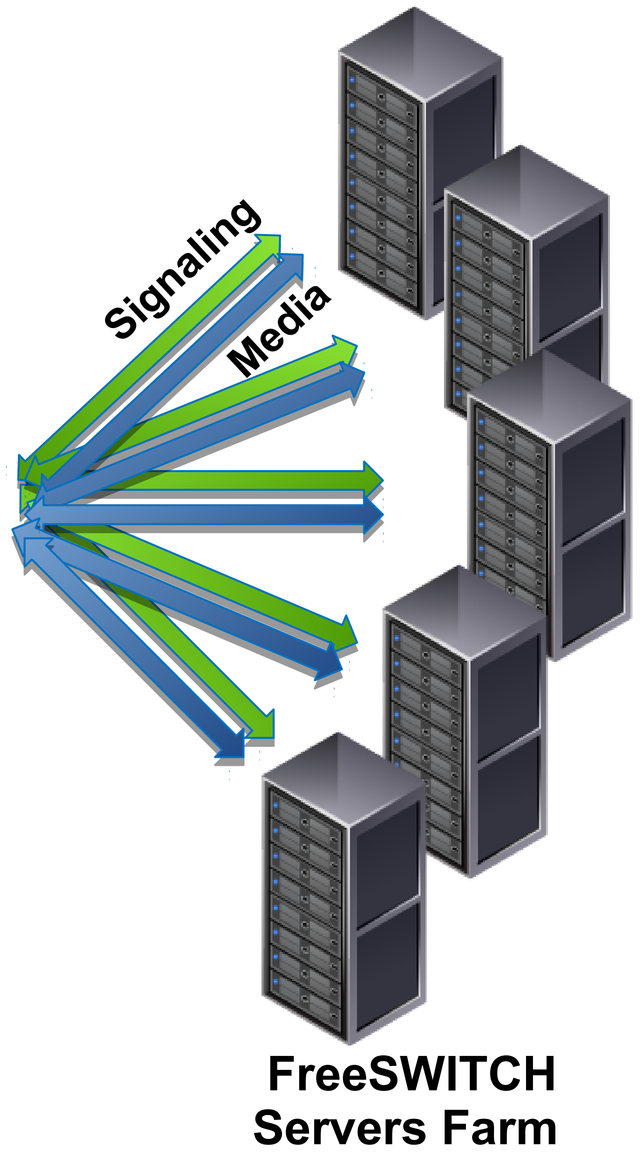
Load Balancing and Proxies



Keepalived

- Keepalived is a simple way to move a “Virtual” IP address from one Load Balancer server to another
- Virtual IP address will be the only published and accessed address
- Keepalived will check Proxy is alive and working (eg, with sipsak) on the “primary” load balancer. If primary has failed, Virtual IP address will be moved to “secondary” (or “standby”) load balancer
- All other machines (clients, FS servers, etc) will not perceive any change

FreeSWITCHes' Farm



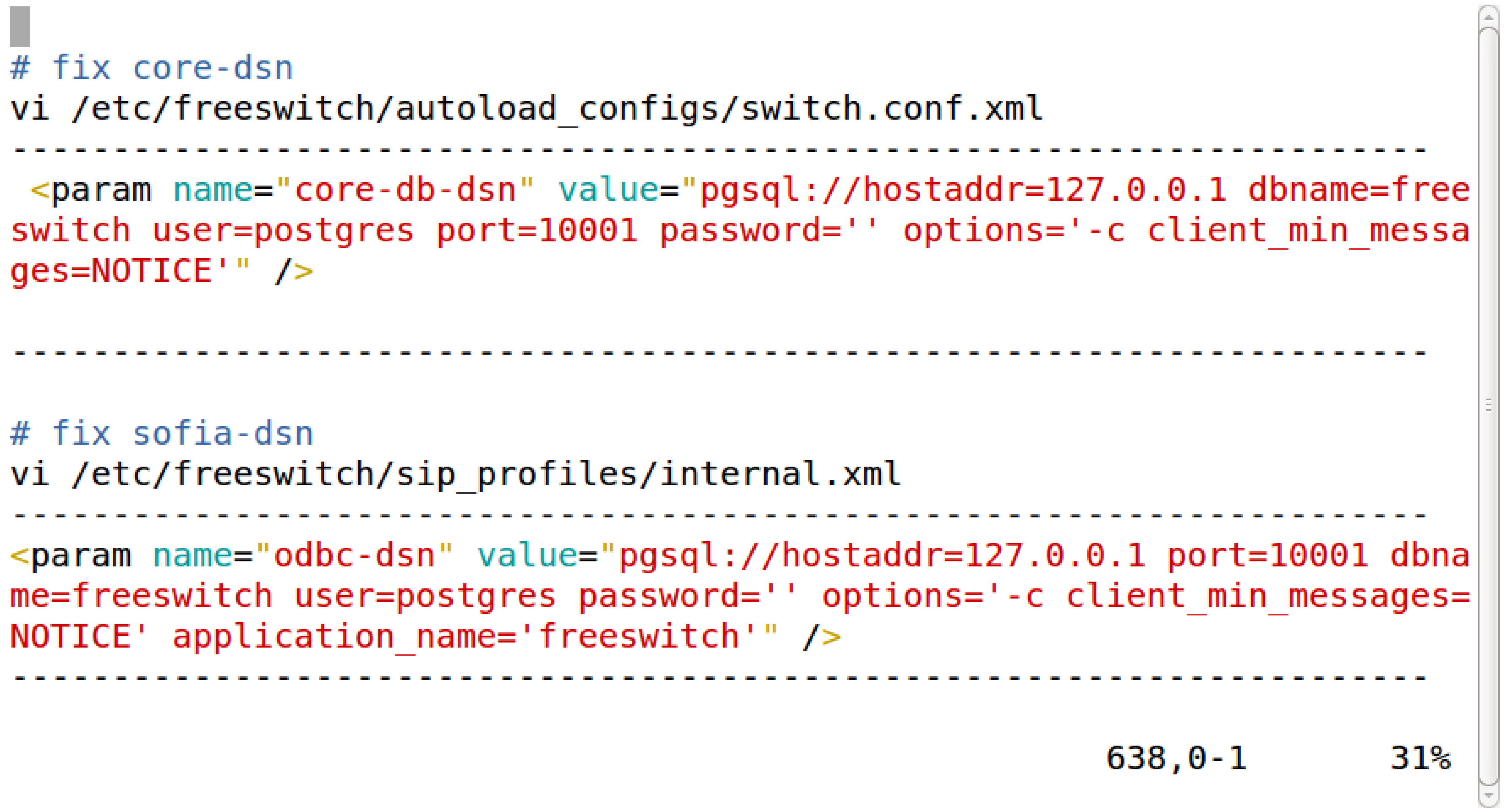
FreeSWITCHes' Farm

- FreeSWITCH gets its own configuration from XML
- By default, that XML is kept in files in a local directory
- GlusterFS client permits to access that directory from many Fses (another way is to use mod_xml_curl to access XML via HTTP)
- VoiceMail metadata resides in DB, while actual audio messages are shared by GlusterFS

FreeSWITCHes' Farm

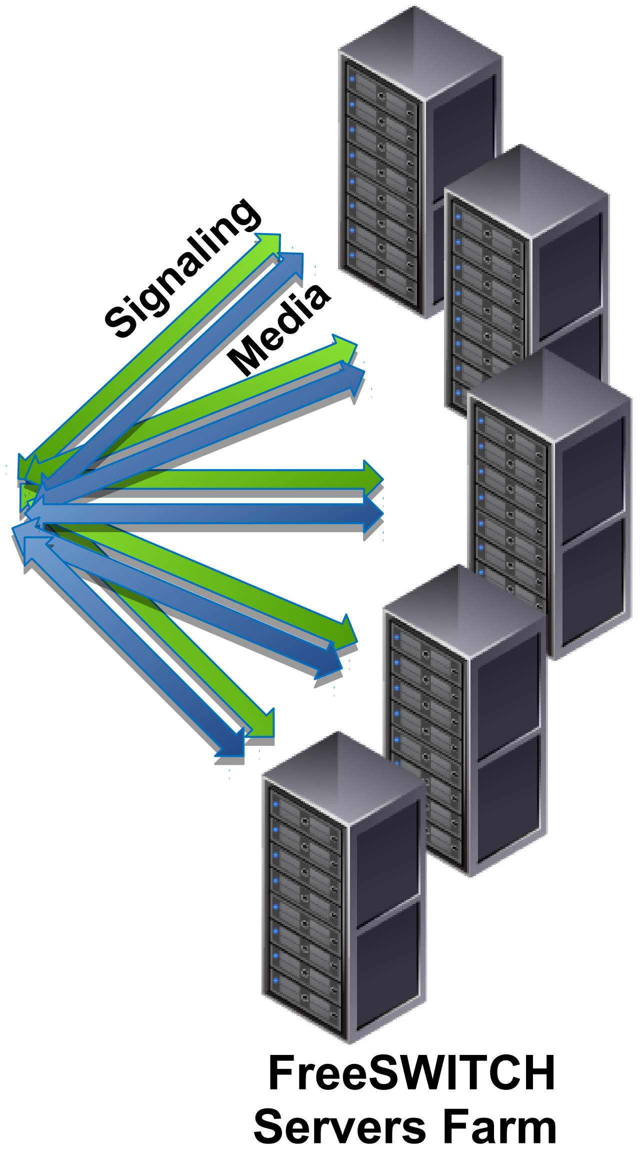
- FreeSWITCH uses an internal database to keep state and persistance about SIP registrations, call states, etc
- By default, that database is kept on SQLite files in a local directory
- With PGSQL in CORE, and by setting mod_sofia, all the FS guts will reside in a remote PostgreSQL, shared by many FSes

FreeSWITCHes' Farm DSN



```
# fix core-dsn
vi /etc/freeswitch/autoload_configs/switch.conf.xml
---------------------------------------------------------------------
 <param name="core-db-dsn" value="pgsql://hostaddr=127.0.0.1 dbname=free
switch user=postgres port=10001 password='' options='-c client_min_messa
ges=NOTICE'" />


---------------------------------------------------------------------


# fix sofia-dsn
vi /etc/freeswitch/sip_profiles/internal.xml
---------------------------------------------------------------------
<param name="odbc-dsn" value="pgsql://hostaddr=127.0.0.1 port=10001 dbna
me=freeswitch user=postgres password='' options='-c client_min_messages=
NOTICE' application_name='freeswitch'" />
---------------------------------------------------------------------

                                                      638,0-1       31%
```

FreeSWITCHes' Farm

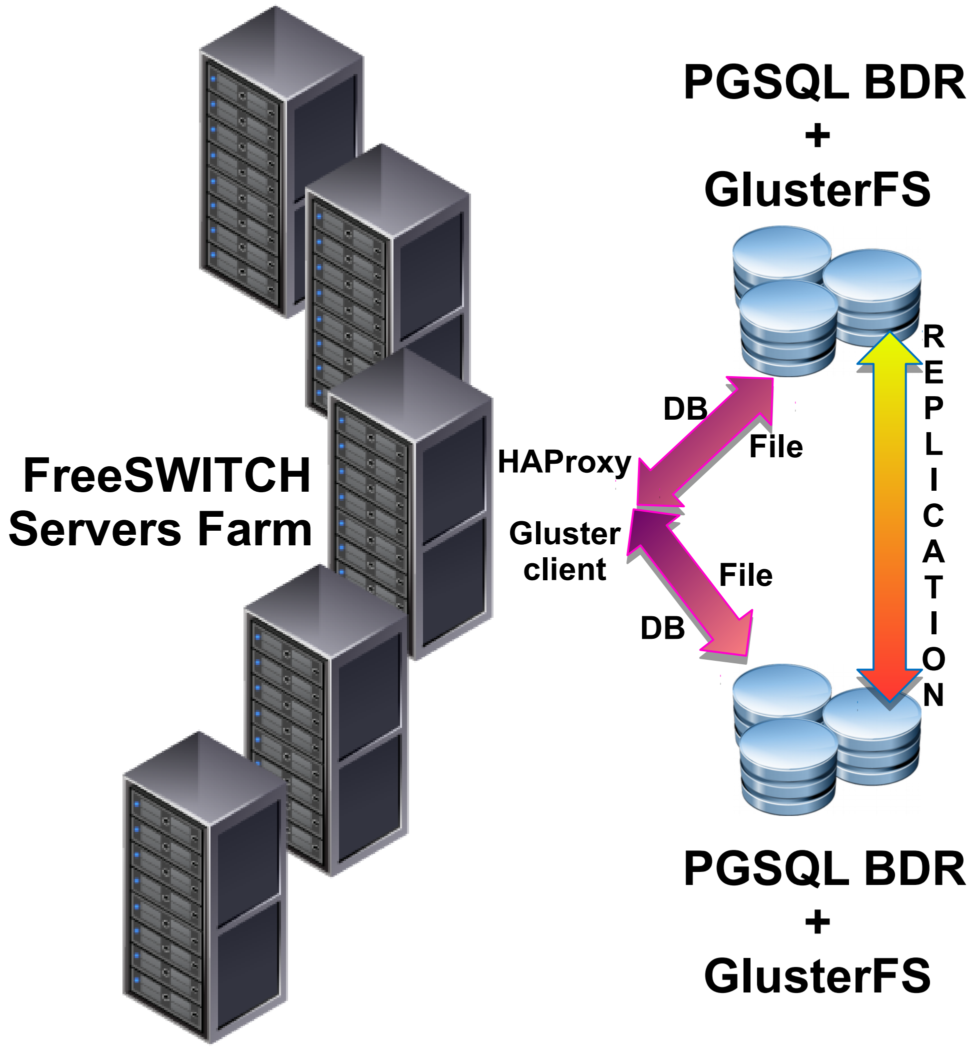


FreeSWITCHes' Farm

- On each FreeSWITCH machine we put an HAProxy
- PostgreSQL will be accessed by HAProxy
- HAProxy wil automatically balance between PGSQL servers, and failover when needed

PERSISTENCE:

GlusterFS & PostgreSQL BDR

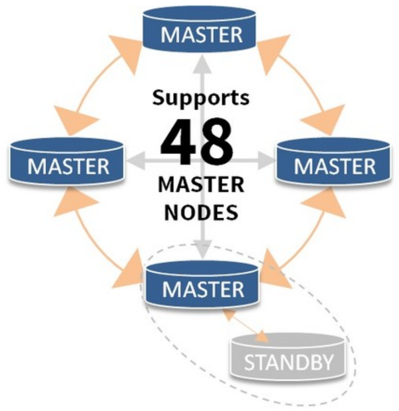


GlusterFS

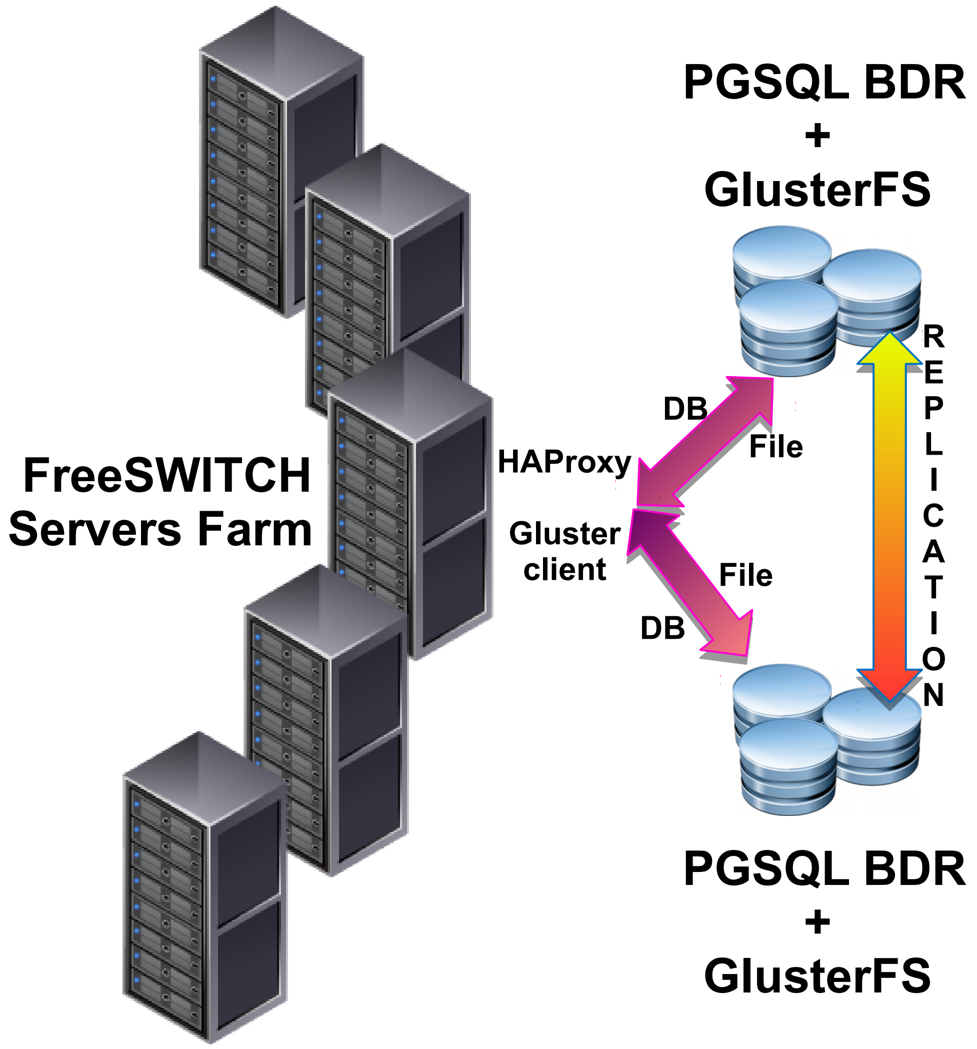
- GlusterFS is a distributed filesystem
- Gluster SERVERs “export” local BRICKs
- Gluster CLIENTs “mount” remote BRICKs
- Any modifications made by clients is automatically synched in realtime on all servers and all clients
- If a server fails, clients automatically failover in realtime to another server

PostgreSQL BDR

- Bi Directional Replication (BDR) is a new addition by 2ndQuadrant to PostgreSQL. Is being integrated into mainline and will be in a future official release
- BDR allows for master-master low latency clustering
- BDR automatically replicate new tables and table modifications
- To use BDR you must have uniq Pks inserted (UUIDs)
- Two ways for doing that from FS



PERSISTENCE:
GlusterFS & PostgreSQL BDR



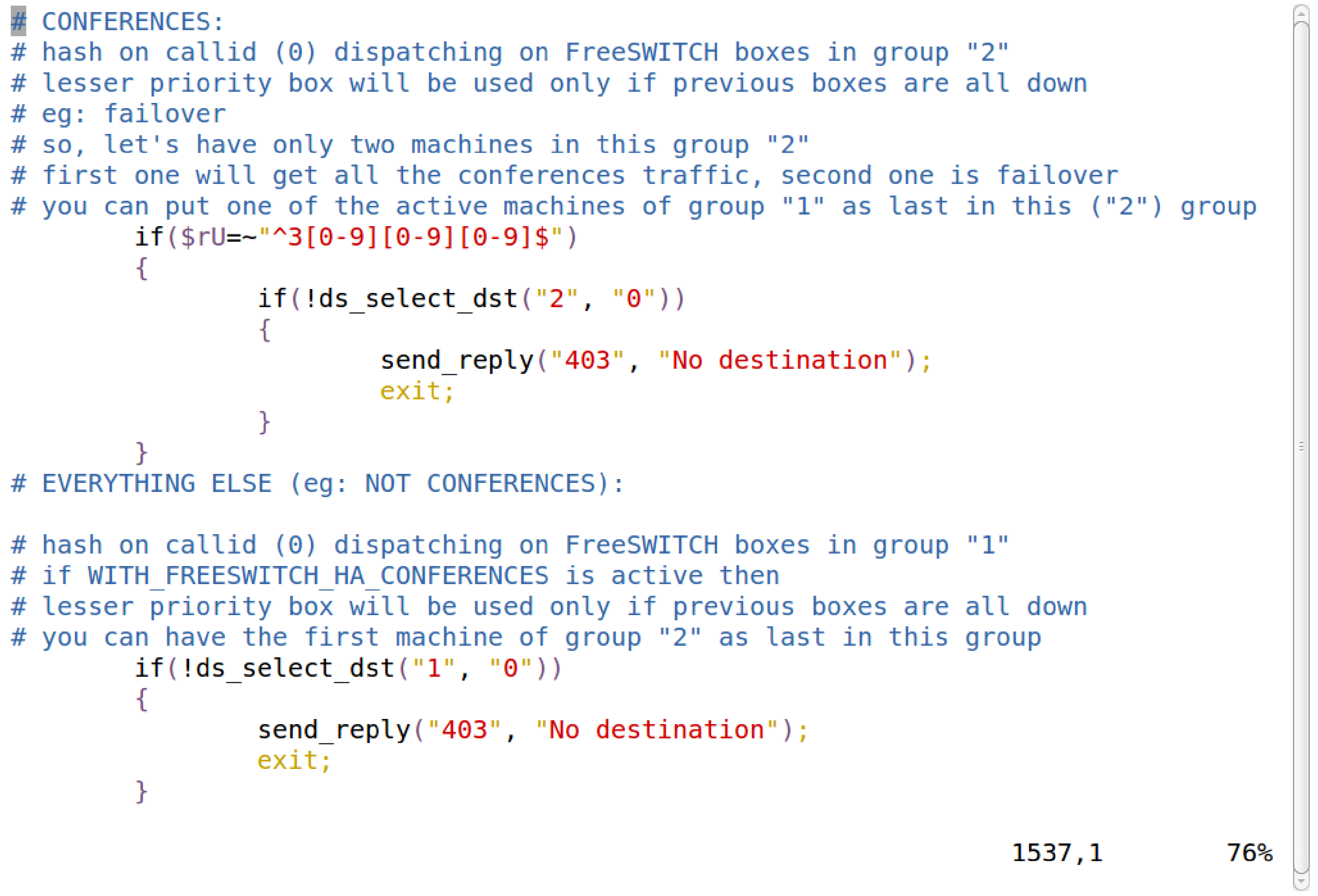
Special Cases

- Load Balancing is predicated on a server farm of equivalent and equipollent (eg: interchangeable) servers

- There are cases for which this is not true:
  - Conferences
  - Call queues
  - Call centers

- **ANSWER IS: Partitioning!**

Special Cases



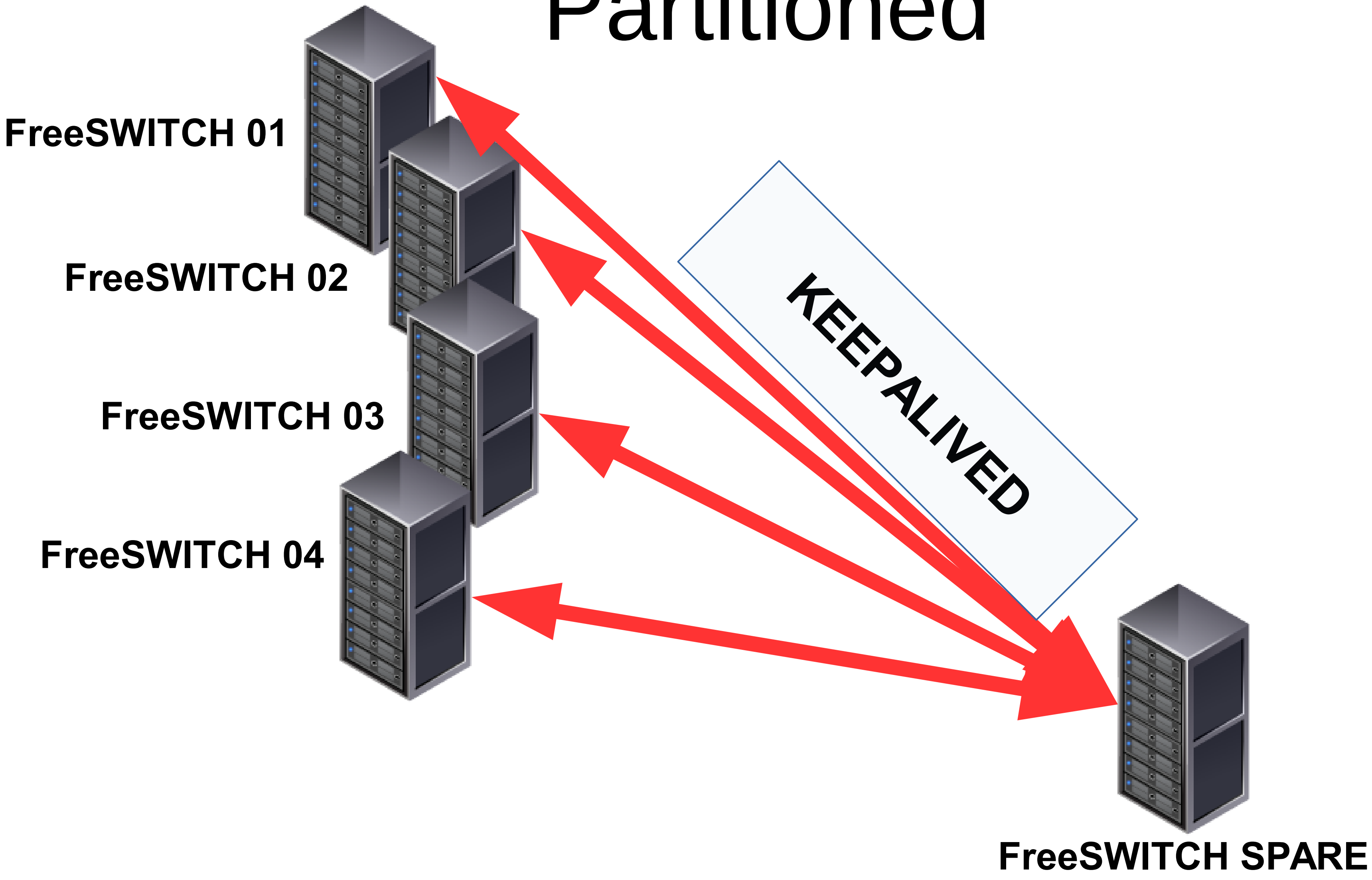
```
# CONFERENCES:
# hash on callid (0) dispatching on FreeSWITCH boxes in group "2"
# lesser priority box will be used only if previous boxes are all down
# eg: failover
# so, let's have only two machines in this group "2"
# first one will get all the conferences traffic, second one is failover
# you can put one of the active machines of group "1" as last in this ("2") group
        if($rU=~"^3[0-9][0-9][0-9]$")
        {
                if(!ds_select_dst("2", "0"))
                {
                        send_reply("403", "No destination");
                        exit;
                }
        }
# EVERYTHING ELSE (eg: NOT CONFERENCES):

# hash on callid (0) dispatching on FreeSWITCH boxes in group "1"
# if WITH_FREESWITCH_HA_CONFERENCES is active then
# lesser priority box will be used only if previous boxes are all down
# you can have the first machine of group "2" as last in this group
        if(!ds_select_dst("1", "0"))
        {
                send_reply("403", "No destination");
                exit;
        }

                                                                 1537,1        76%
```

Special Cases (Multi Tenancy)

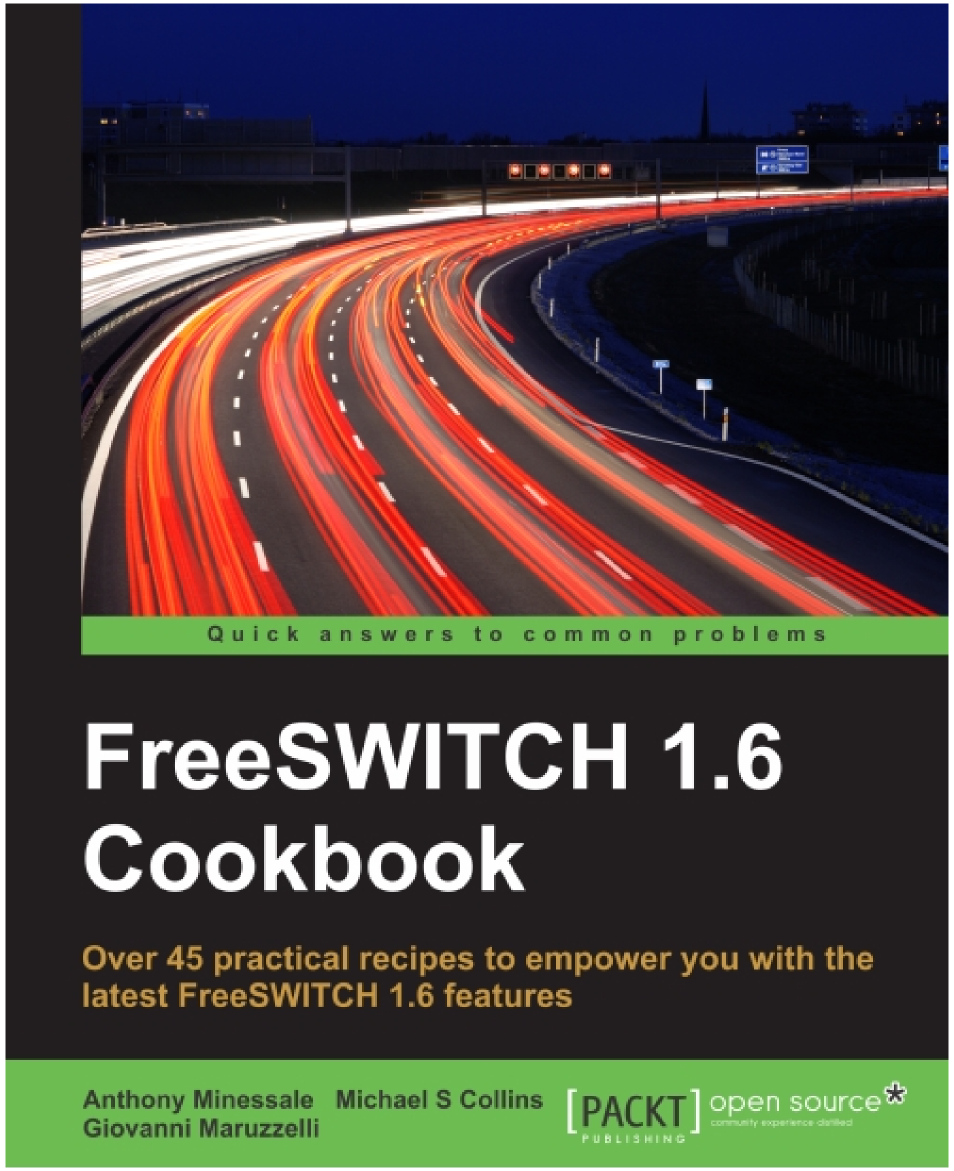
- Multi Tenant = Multiple SIP/WebRTC domains, managed independently
- Farm is partitioned on Domains by the Proxy, each domain goes to a particular machine
- This solves the conferencing-queues-transfer issues (eg locality of calls/users)

- High Availability by one or more SPARE machines, ready to take the role if the failed machine

FreeSWITCHes' Farm Partitioned



www.packtpub.com



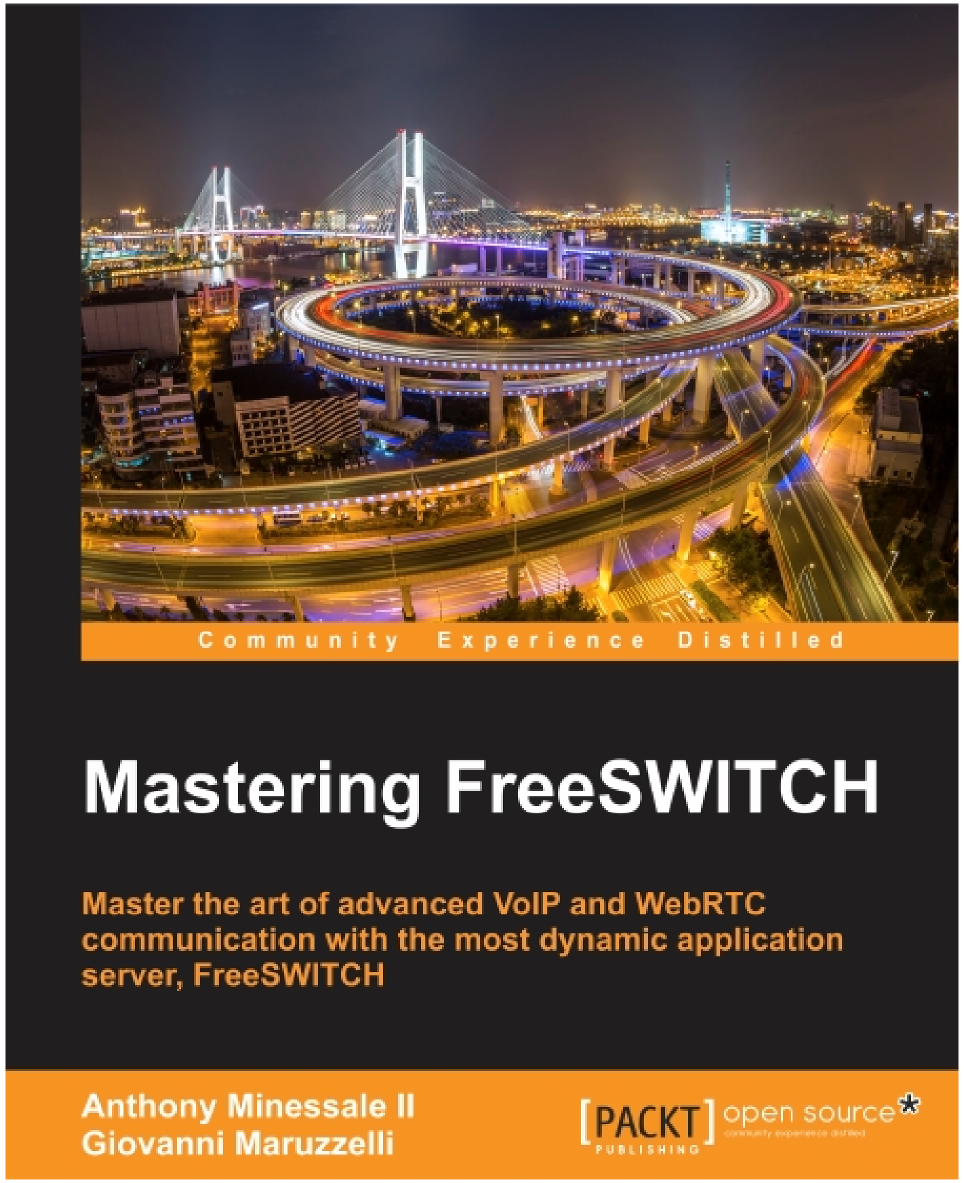
Quick answers to common problems

FreeSWITCH 1.6 Cookbook

Over 45 practical recipes to empower you with the latest FreeSWITCH 1.6 features

Anthony Minessale Michael S Collins Giovanni Maruzzelli

www.packtpub.com



Community Experience Distilled

Mastering FreeSWITCH

Master the art of advanced VoIP and WebRTC communication with the most dynamic application server, FreeSWITCH

Anthony Minessale II
Giovanni Maruzzelli

[PACKT] open source*
PUBLISHING

Thank You

QUESTIONS ?

Giovanni Maruzzelli
gmaruzz@OpenTelecom.IT