KAMAILIO
ELASTICITY OF VOIP PLATFORMS

Scale On Demand

load balancing
replication
smart routing

KAMAILIO

VoIP - VoLTE

telephony
voice - video
instant messaging
presence
webrtc

Deploy Everywhere

bare metal
cloud platforms
mixed environments

Last Years In Review

refactoring, more flexibility, new modules

voip - ims - volte - presence - instant messaging - webrtc

continuous development since 2001

Highlights 2017 - 2018

new modules and scripting languages

- new modules
  - acc_diameter,
  - app_jsdt
  - ims_ocs
  - nsq
  - pua_rpc
  - rabbitmq
  - ss7ops
  - app_sqlang
  - call_obj
  - evrexec
  - ims_diameter_server
- new modules
  - acc_json
  - acc_python3
  - db_redis
  - app_ruby
  - phonenum
  - sipdump
  - pua_json
  - topos_redis
  - keepalive
- config scripting languages (kemi)
  - native - startup optimizations
  - lua - many libraries, routing logic hot reload
  - javascript - embedded code, routing logic hot reload
  - python2/3 - lots of libraries, hot reload
  - ruby - lots of libraries, hot reload
  - squirrel language - embedded code, routing logic hot reload

https://www.kamailio.org/wiki/features/new-in-5.1.x

https://www.kamailio.org/wiki/features/new-in-5.0.x

https://www.kamailio.org/wiki/features/new-in-devel

Highlights 2017 - 2018

kemi framework

- introduced in Kamailio v5.0.0
  - use other scripting languages for writing SIP routing logic (route blocks)
  - exports a function one time and becomes available in embedded interpreters
  - ***allows reloading routing script without restart***

KEMI Framework
kamailio embedded (interpreter) interface

- Kamailio Configuration File - Two Main Roles
  - Kamailio application initialization
    - Done once at startup (passive scope)
    - Global parameters, loading modules and modules'parameters
    - Many values can be changed at runtime via RPC (no restart)
  - Rules for handling SIP traffic
    - Done during runtime to decide the routing of SIP messages
    - No reload without restart for native kamailio.cfg scripting language
    - KEMI routing scripts can be reloaded without restart (v5.0+)
- Scripting languages
  - Native scripting language
    - built from scratch, routing blocks with set of actions
  - Kamailio Embedded Interface (KEMI) languages
    - reuse existing (popular) scripting languages
    - replace the routing blocks from native scripting language
    - allow reloading of scripts without restart and more features
  - Inline execution of scripting languages
    - can be executed inside native scripting language
    - support for Lua, JavaScript, Python, Perl, .Net (C#, …), Squirrel, Java, Ruby

```
298  # Routing to foreign domains
299  route[SIPOUT] {
300          if (uri==myself) return;
301
302          append_hf("P-hint: outbound\r\n");
303          route(RELAY);
304          exit;
305  }
```

KEMI Framework
kamailio embedded (interpreter) interface

Native (Custom) Routing Language Since 2001

Kamailio Embedded Interpreter Interface - K E M I

- introduced in Kamailio v5.0.0
  - split parts: **passive** (global and module params) and **active** (routing blocks)
  - live reload of active part
- working for next scripting languages (embedded interpreters)
  - Lua
  - JavaScript
  - Python - Python3
  - Squirrel
  - Ruby
- not yet updated for next existing embedded interpreters
  - can be used with inline execution
  - Perl
  - Mono (.NET, C#, …)
  - Java (?)
- *native language still available*
  - *the old kamailio.cfg scripting language is still there, maintained and developed*

```
298  # Routing to foreign domains
299  route[SIPOUT] {
300          if (uri==myself) return;
301
302          append_hf("P-hint: outbound\r\n");
303          route(RELAY);
304          exit;
305  }
```

KEMI Framework
sip routing logic in lua

- app_lua
  - existing since 2010 offering inline execution of Lua scripts
  - https://www.kamailio.org/docs/modules/stable/modules/app_lua.html
- highlights
  - very small interpreter, fast
  - decent number of extensions (native lua libraries)
  - popular in gaming space, also in RTC (Asterisk, Freeswitch)

https://github.com/kamailio/kamailio/blob/master/misc/examples/kemi/kamailio-basic-kemi-lua.lua

```
324  -- Routing to foreign domains
325  function ksr_route_sipout()
326          if KSR.is_myself(KSR.pv.get("$ru")) then return 1; end
327
328          KSR.hdr.append_hf("P-Hint: outbound\r\n");
329          ksr_route_relay();
330          KSR.x.exit();
331  end
```

KEMI Framework
sip routing logic in python

- app_python - app_python3
  - introduced in 2010 for inline execution of Python scripts
  - https://www.kamailio.org/docs/modules/stable/modules/app_python.html
  - https://www.kamailio.org/docs/modules/devel/modules/app_python3.html
- highlights
  - popular scripting language with extensive number of extensions
  - object oriented, perceived as slower than other scripting languages
  - reloading of the routing script not implemented yet

https://github.com/kamailio/kamailio/blob/master/misc/examples/kemi/kamailio-basic-kemi-python.py

```
332     # Routing to foreign domains
333     def ksr_route_sipout(self, msg):
334         if KSR.is_myself(KSR.pv.get("$ru")) :
335             return 1;
336
337         KSR.hdr.append("P-Hint: outbound\r\n");
338         self.ksr_route_relay(msg);
339         return -255;
```

KEMI Framework
Kamailio API Exports

- index of exported functions
  - KSR static or dynamic object
  - function names and parameters map (pretty well) over the ones for native kamailio.cfg scripting

http://kamailio.org/docs/tutorials/devel/kamailio-kemi-framework

https://github.com/kamailio/kamailio/blob/master/misc/examples/kemi/

Highlights 2017 - 2018

Non-SIP Processing Or Forwarding Proxy

- **Receiving And Then Forwarding HEP3 Packets To Different Destination**
- Actually just a wrapper for **send(...)** and the variants
  - **sip_capture_forward(uri)** - uri can be a static string or can contain script variables
- ***sipcapture module***
  - https://kamailio.org/docs/modules/devel/modules/sipcapture.html
- **Non-SIP Processing - Example: Compressing/Decompressing Data**

```
event_route[sipcapture:request] {
    ...
    if(src_ip==1.2.3.4) {
        sip_capture_forward("sip:2.3.4.5:5090");
        return 0;
    }
    ...
}
```

```
...
loadmodule "app_perl.so"
loadmodule "corex.so"
...
# ----- app_perl params -----
modparam("app_perl", "filename", "/usr/local/etc/kamailio/custom_compress.pl")
modparam("app_perl", "modpath", "/usr/local/lib64/kamailio/perl")

# ----- corex params -----
modparam("corex", "network_io_intercept", 32)
modparam("corex", "min_msg_len", 32)
modparam("corex", "msg_avp", "$avp(msg)")
...
event_route[network:msg] {
    if (is_incoming()) {
        if (perl_exec_simple("do_uncompress", "" + $mb + "")) {
            xlog("L_INFO", "Received message '$avp(msg)' \n");
        } else {
            xlog("L_INFO", "Received message '$mb' \n");
            $avp(msg) = $mb;
        }
    } else {
        xlog("L_INFO", "Sending message '$mb' \n");
        if (!perl_exec_simple("do_compress", "" + $mb + "")) {
            $avp(msg) = $mb;
        }
    }
}
```

```
use strict;
use warnings;
use IO::Compress::Gzip qw(gzip $GzipError) ;
use IO::Uncompress::Gunzip qw(gunzip $GunzipError) ;

sub do_compress() {
    my $input = shift;
    my $output;

    gzip \$input => \$output
        or eval {
            Kamailio::log(L_WARN, "GZIP failed: $GzipError\n");
            $output = $input;
        };

    Kamailio::AVP::add("msg", $output);
}

sub do_uncompress() {
    my $input = shift;
    my $output;

    gunzip \$input => \$output
        or eval {
            Kamailio::log(L_WARN, "GUNZIP failed: $GzipError\n");
            $output = $input;
        };

    Kamailio::AVP::add("msg", $output);
}
```

Highlights 2017 - 2018

Enhancements

- Lots of improvements to IMS/VoLTE extensions
- Latency statistics and routing for dispatcher (load balancer)
- Topology stripping (topos module)
- Many Redis-related additions, including cluster support, pipeline and backend for topos
- Stronger algorithms for authentication (SHA256)
- More elasticity for Kamailio nodes authentication - auth with keys (auth_xkeys)
- Event API sockets - json based interaction with external apps - evapi module
- Asynchronous processing - suspend/resume routing SIP messages
- Optimizations for mobile networks - push notifications
- Distributed platforms - propagation among nodes for user location, dialogs, presence
- Message queues - DMQ, RabbitMQ, NSQ
- Refactored dispatcher (load balancer) internals: linked list replaced with AVL tree indexing, XAVPs instead of AVPs
- RTP recording and transcoding capabilities to **rtpengine**

VoIP - VoLTE

telephony for fixed and mobile networks

IPv4-IPv6
UDP/TCP/TLS/SCTP
SCTP
multi-homing
multi-streaming, statistics
UDP Raw Sockets
DNS NAPTR & SRV

SIP proxy, redirect
and registrar server
user registration
with
digest authorization

Load Balancing
Least Cost Routing
DID - Prefix Routing
Aliases
Ring Groups
Speed Dial
ENUM Routing

NoSQL
Memcached
Redis
Cassandra

Instant Messaging
End-to-End
Offline Storage
Embedded MSRP Relay
IM Conferencing

NAT traversal
Security
permissions
anti-DOS attacks
User call preferences
Call Processing
Language

IMS
Extensions

Presence Services

End-to-End Presence
SIMPLE Presence Server
Embedded XCAP Server
Dialog Info (blinking lamps)
Resource Lists Service (RLS)
Profile and Conference Events
Presence User Agents

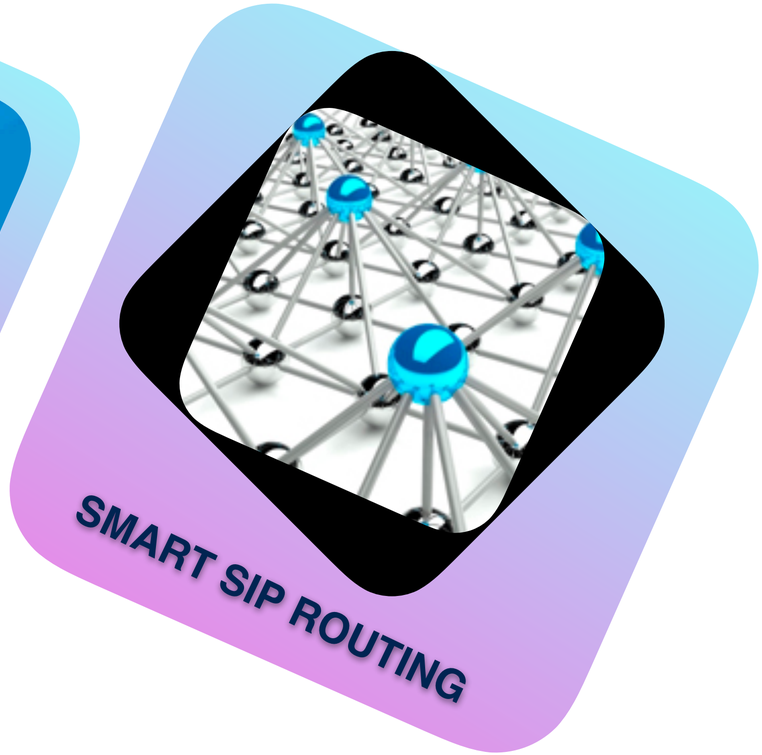
DNS Caching System

srv load balancing
fast failover
blacklisting
avoid long blocking

Embedded
HTTP
client
server
WebSocket

RabbitMQ
NSQ

https://www.kamailio.org/w/features/
https://www.kamailio.org/events/
https://www.youtube.com/channel/UCElq4JNTPd7bs2vbfAAYVJA/videos



SCALABILITY AND ELASTICITY

growing fast and dealing with it

locations, devices, calls, attacks

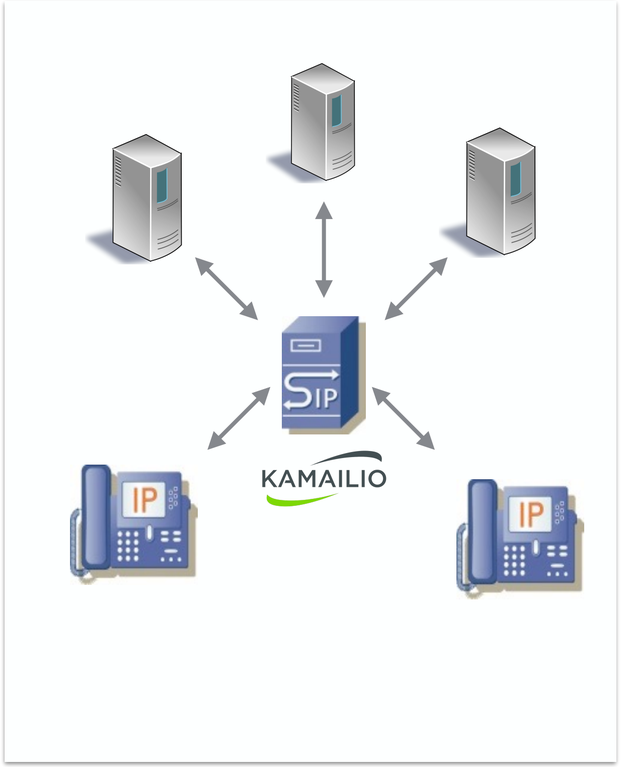
Scalability - Elasticity

Load Balancer - Dispatcher - Edge Proxy

dispatcher module

- list of balancing nodes from file or database
- monitoring of nodes (activate/inactivate automatically)
- re-route in case of failure
- various algorithms: hashing, weight distribution, round robin, call load distribution, priority routing
- reload list of nodes without restart

```
# Dispatch requests
route[DISPATCH] {
        # round robin dispatching on gateways group '1'
        if(!ds_select_dst("1","4")) {
                send_reply("404", "No destination");
                exit;
        }
        xdbg("--- SCRIPT: going to <$ru> via <$du>\n");
        t_on_failure("RTF_DISPATCH");
        route(RELAY);
        exit;
}
# Re-route in case of failure
failure_route[RTF_DISPATCH] {
        if (t_is_canceled()) {
                exit;
        }
        # next node - only for 500 or local timeout
        if (t_check_status("500") || (t_branch_timeout() && !t_branch_replied())) {
                if(ds_next_dst()) {
                        t_on_failure("RTF_DISPATCH");
                        route(RELAY);
                        exit;
                }
        }
}
```



Scalability - Elasticity

Transparent proxy - Anycast

- **Expectations**
  - Forward without adding/removing headers required in SIP specifications
  - Mainly about to not alter Via stack
    - Do not add for SIP request
    - Do not remove for SIP response
  - Set source IP ($si) and local socket ($Ri)
- It was there for very long time - **send(...)** and the variants
  - Easy with deployments running on a single socket
- ***Corex module***
  - Added some helper functions to help on deployments with multiple network interfaces
  - https://kamailio.org/docs/modules/devel/modules/corex.html


```
# before forwarding request
via_add_srvid(“1”);
$xavp(via=>node) = LOCALIP;
via_add_xavp_params("1");
```

```
# response does not belong to a transaction, relay to pair srv
if(!t_check_trans()) {
 append_hf(“X-Src-Addr: $su\r\n”);
 $var(d) = $(hdr(Via){s.select, ,-1}{param.value,node});
 sendx(“sip:$var(d)”, “LOCALSOCKET”, “$mbu”);
 drop();
}
```

```
# once receiving a transparent proxied SIP response
set_recv_socket(“ANYCASTSOCKET”);
set_source_address(“$hdr(X-Src-Addr)”);
```

Scalability - Elasticity

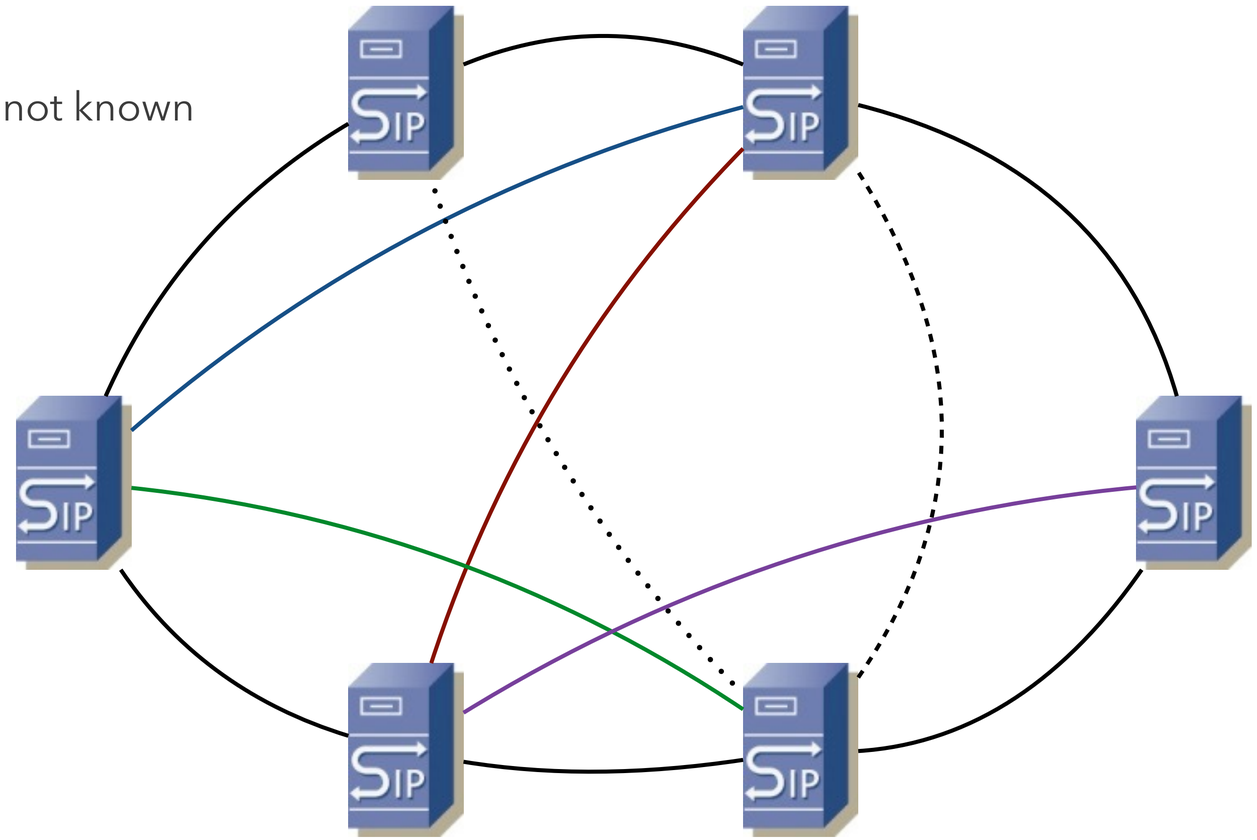
Transparent proxy - Anycast

https://github.com/kamailio/kamailio/tree/master/misc/examples/mixed/kamailio-minimal-anycast.cfg

Scalability - Elasticity

how to trust sip traffic between sip server nodes

- **permissions module**
  - IP based authorization
  - *but elasticity is not very elastic*
  - e.g., new instances on cloud with IP not known beforehand



Scalability - Elasticity

sip server node authentication

- **auth_xkeys module**
  - sign headers with a shared secret
  - API-key-like header

```
modparam("auth", "algorithm", "SHA-256")
...
modparam("auth_xkeys", "xkey", "id=abc;name=xyz;value=secret;expires=72000")
...
auth_xkeys_add("X-My-Key", "abc", "sha256", "$Ri:$fu:$ru:$hdr(CSeq)");
...
if(!auth_xkeys_check(“X-My-Key", "abc", "sha256", "$si:$fu:$ru:$hdr(CSeq)")) {
    send_reply("403", "Forbidden");
    exit;
}
remove_hf("X-My-Key");
...
```

Scalability - Elasticity

authentication - security - privacy

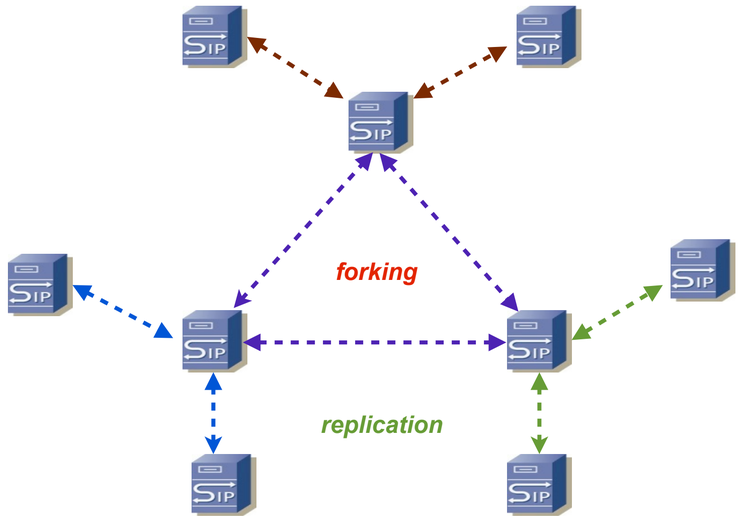
- **auth* modules**
  - *sha256 hasing algorithm*
  - *one time nonce, etc.*
- **tls module**
  - scaling to couple of hundred thousands connections
  - optimize by trusting the accepted connections
  - avoid second round trip for authentication

```
# check connection id for authentication

modparam("htable", "htable", "conids=>size=8;autoexpire=900;")
request_route {
…
  if(proto==TLS) {
    if($sht(conids=>$fU@$fd) == $conid) {
      # skip authentication - traffic on known connection
      …
    } else {
      # usual authentication
      route(AUTH);
      # associate connection id with From username@domain
      $sht(conids=>$fU@$fd) == $conid;
    }
  }
…
}
```

Scalability - Elasticity

replication

- **tm - corex - dmq module**
  - distributed message queue using SIP messages
  - auto-discovery of the nodes
  - propagation for user location, presence, dialogs (active calls), cached items (htable module)




```
KDMQ sip:usrloc@10.6.0.175:5060 SIP/2.0
Via: SIP/2.0/UDP 10.6.0.173;branch=z9hG4bK9cbc.6c89b87100000000000000000000000.0
To: <sip:usrloc@10.6.0.175:5060>
From: <sip:usrloc@10.6.0.173:5060>;tag=390c95c339281829d3cea6f43c8512cb-0295
CSeq: 10 KDMQ
Call-ID: 7f7531821a1befa0-18771@10.6.0.173
Content-Length: 499
User-Agent: kamailio
Max-Forwards: 1
Content-Type: application/json

{“action”:1,”aor”:”subscriber_name@subscriber_domain",
"ruid":"uloc-559bdf82-4953-b",
“c":"sip:subscriber_name@192.168.1.238:54270",
...}
```

Scalability - Elasticity

smart routing

- **smart routing - dispatching - load balancing**
  - *dispatcher module*
    - *round robin, call load distribution, weight based distribution, hashing algorithms, serial/parallel forking…*
  - *rtjson module*
    - *json-based routing, serial/parallel forking…*

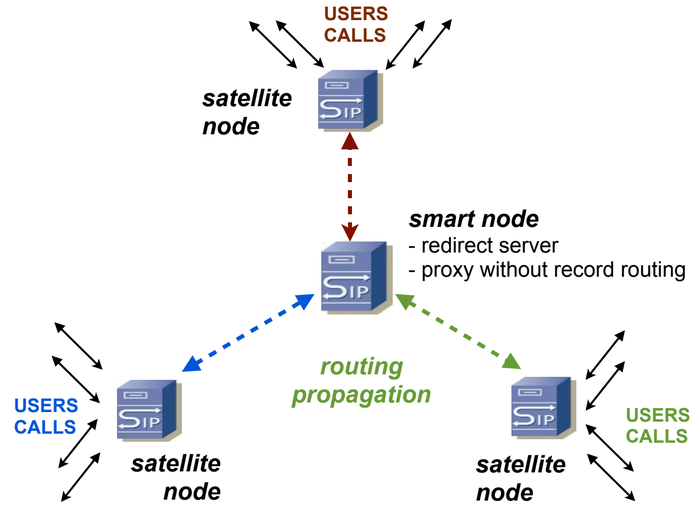
```
{
    "version": "1.0",
    "routing": "serial",
    "routes": [
        {
            "uri": "sip:127.0.0.1:5080",
            "dst_uri": "sip:127.0.0.1:5082",
            "path": "<sip:127.0.0.1:5084>, <sip:127.0.0.1:5086>",
            "socket": "udp:127.0.0.1:5060",
            "headers": {
                "from": {
                    "display": "Alice",
                    "uri": "sip:alice@127.0.0.1"
                },
                "to": {
                    "display": "Alice",
                    "uri": "sip:alice@127.0.0.1"
                },
                "extra": "X-Hdr-A: abc\r\nX-Hdr-B: bcd\r\n"
            },
            "branch_flags": 8,
            "fr_timer": 5000,
            "fr_inv_timer": 30000
        },
...
```

Scalability - Elasticity

hierarchical routing propagation

```
# Handle SIP registrations
route[REGISTRAR] {
    if (is_method("REGISTER")) {
        if(isflagset(FLT_NATS)) {
            setbflag(FLB_NATB);
            # uncomment next line to do SIP NAT pinging
            ## setbflag(FLB_NATSIPPING);
        }
        if (!save("location"))
            sl_reply_error();

    $uac_req(method)=“KUSRLOC"
    $uac_req(ruri)="sip:store@smartnode.kamailio.org";
    $uac_req(furi)="sip:server@sat1.kamailio.org";
    $uac_req(hdrs)="Content-Type: text/kusrloc\r\n";
    pv_printf(“$uac_req(body)”, "$fu@fd");
    uac_send_req();

    exit;
    }
}
```

```
route[TOMAIN] {
    $du = "sip:SMARTNODEIP";
    route(RELAY);
    exit;
}

# USER location service
route[LOCATION] {
#!ifdef WITH_SPEEDDIAL
    # search for short dialing - 2-digit extension
    if($rU=~"^[0-9][0-9]$")
        if(sd_lookup("speed_dial"))
            route(SIPOUT);
#!endif
#!ifdef WITH_ALIASDB
    # search in DB-based aliases
    if(alias_db_lookup("dbaliases"))
        route(SIPOUT);
#!endif

    $avp(oexten) = $rU;
    if (!lookup("location")) {
        if(src_ip!=SMARTNODEIP)
            route(TOMAIN);
...
}
```




```
modparam(“htable”, “htable”, “kusrloc=>size=10;autoexpire=7200;”)
...
request_route {
    # add here ip authorization, etc…

    # handle location update notification
    if(method=="KUSRLOC" && $rU=="store") {
        $sht(kusrloc=>$rb) = “sip:” + $si + “:” + $sp + “;transport=” + $pr;
        send_reply("200", "Stored");
        exit;
    }
    # handle standard SIP requests
    if($sht(kusrloc=>$rU@$rd)!=$null) {
        $du = $sht(kusrloc=>$rU@$rd);
        t_relay();
        exit;
    }
    send_reply(“404”, “Not found”);
    exit;
}
```

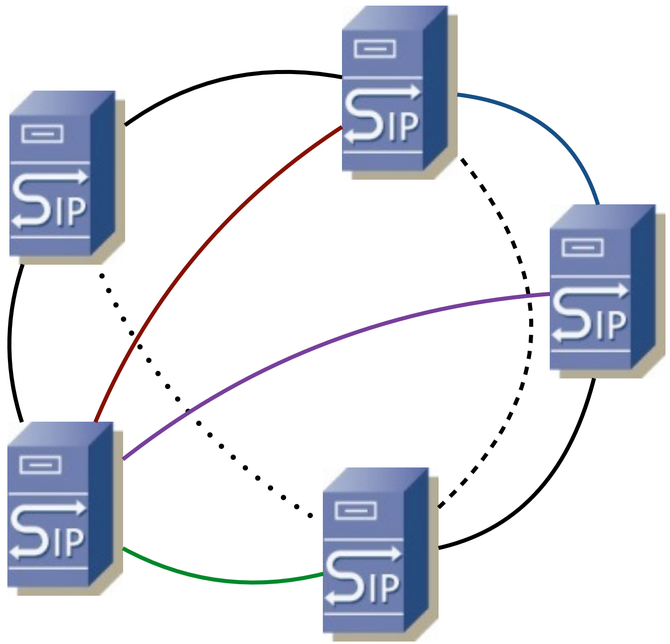
satellite node

smart node

Scalability - Elasticity

horizontal routing propagation

```
request_route {
…
    if(is_method("KDMQ")) {
      if($rU == "store") {
          $sht(kusrloc=>$rb) = “sip:” + $si + “:” + $sp + “;transport=” + $pr;
          send_reply("200", "Stored");
          exit;
      } else {
          dmq_handle_message();
      }
   }
…
}

# Handle SIP registrations
route[REGISTRAR] {
    if (is_method("REGISTER")) {
        if(isflagset(FLT_NATS)) {
            setbflag(FLB_NATB);
            # uncomment next line to do SIP NAT pinging
            ## setbflag(FLB_NATSIPPING);
        }
        if (!save("location"))
            sl_reply_error();

    if(registered(“location”, “$fu”)) {
        dmq_bcast_message(“store”, "$fu@fd", “text/plain”);
    }
    exit;
    }
}
```




```
route[TOHOMENODE] {
    if($sht(kusrloc=>$rU@$rd)!=$null) {
        $du = $sht(kusrloc=>$rU@$rd);
        t_relay();
        exit;
    }
}

# USER location service
route[LOCATION] {
#!ifdef WITH_SPEEDDIAL
    # search for short dialing - 2-digit extension
    if($rU=~"^[0-9][0-9]$")
        if(sd_lookup("speed_dial"))
            route(SIPOUT);
#!endif
#!ifdef WITH_ALIASDB
    # search in DB-based aliases
    if(alias_db_lookup("dbaliases"))
        route(SIPOUT);
#!endif

    $avp(oexten) = $rU;
    if (!lookup("location")) {
        route(TOHOMENODE);
…
}
```

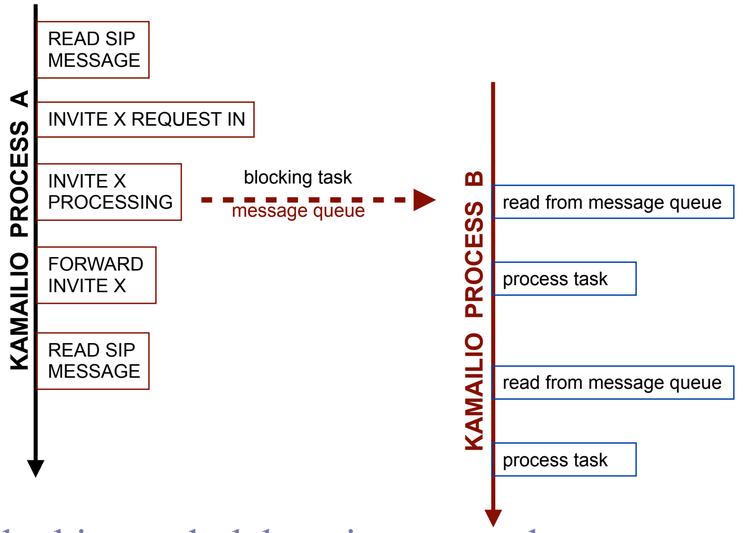
satellite node

smart node

Scalability - Elasticity

vertical scaling - asynchronous processing

- **message queues and event sockets**
  - rabbitmq module
  - nsq module
  - evapi module
- **asynchronous processing**
  - suspend transaction - resume processing
  - async module
  - mqueue + rtimer module
  - async DB queries (insert, update, delete, …)
  - async HTTP queries



blocking task delegation to another process

```
async_workers=8
...
request_route {
    ...
    async_task_route("RESUME");
    ...
}
route[RESUME] {
   t_relay();
   exit;
}
...
```

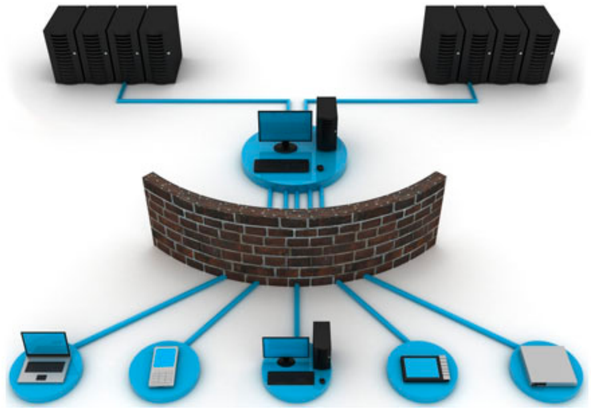
Scalability - Elasticity

topology hiding - topology stripping

- **topoh module**
  - encrypts and encodes addresses in headers
  - no backend needed
  - shared secret across server nodes

- **topos module**
  - strip headers when sending
  - re-add headers when receiving
  - make it look like originated by kamailio (b2bua)
  - SQL database or Redis backends

Note: inside kamailio.cfg, SIP messages have all headers, like being received without topoh or topos



```
loadmodule “topos.so”

event_route[topos:msg-outgoing] {
  if($sndto(ip)=="10.1.1.10") {
    drop;
  }
}
```

THANK YOU!

SEE YOU AT THE NEXT KAMAILIO WORLD CONFERENCE!

SPRING 2019

Daniel-Constantin Mierla
Co-Founder Kamailio Project
@miconda
asipto.com