Distributed Presence Kamailio + JSON

VoIPxSWITCH

About me

- Emmanuel Schmidbauer
- VoIP Engineer at TextNow
- ~10 years experience in SIP (still learning…)
- Kamailio Developer

What I'm going to talk about

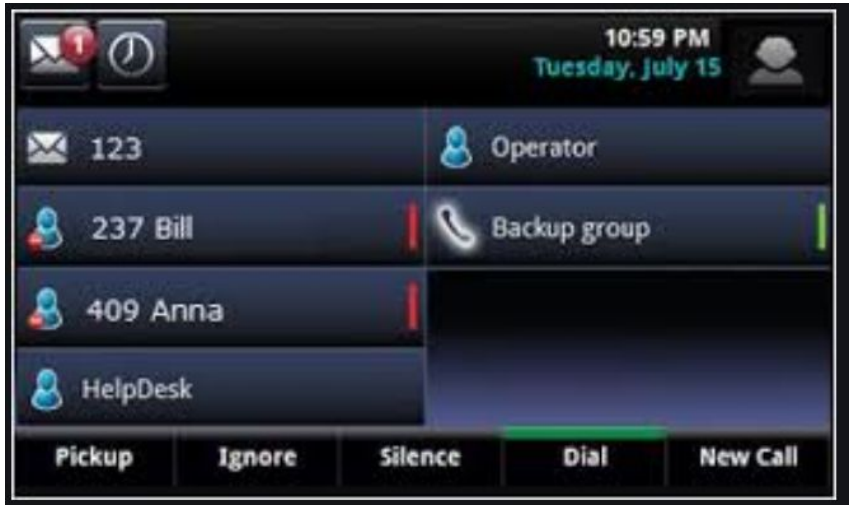
- presence modules
- distributed presence design

What I'm *NOT* going to talk about

- Not going into detail of SIP request methods
- Not going to talk about anything RFC specific

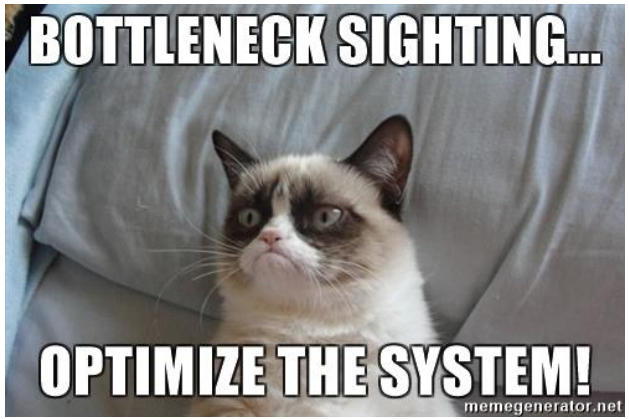
What is presence?

- **Busy Lamp Field (BLF)**
- **Message Waiting Indicator (MWI)**
- **Shared Call Appearance (SCA)**
- **Probably more…**



question: Why Kamailio Presence + JSON?
answer: scaling!

Back in 2016…

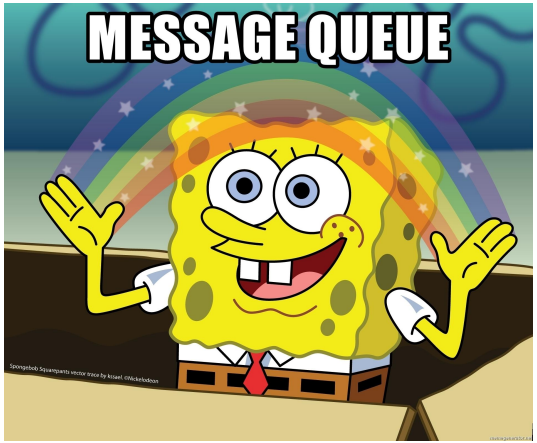


- **Problem:** How do you scale presence?
  FreeSWITCH presence choked after ~800 SIP Users w/ 4 line keys

- **Solution:** *Kamailio!!! But how? IDK - let's just build it!*
  Several weeks later, the **nsq** module was born....
  - https://kamailio.org/docs/modules/devel/modules/nsq
  - https://nsq.io

What is nsq?

lmgtfy...



NSQ is a realtime distributed messaging platform designed to operate at scale, handling billions of messages per day. It promotes distributed and decentralized topologies without single points of failure, enabling fault tolerance and high availability coupled with a reliable message delivery guarantee.

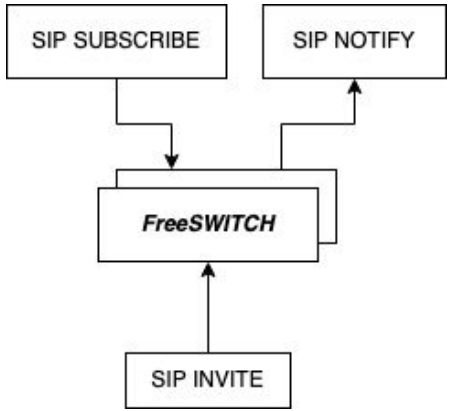
github.com › nsqio › nsq

nsqio/nsq: A realtime distributed messaging platform - GitHub

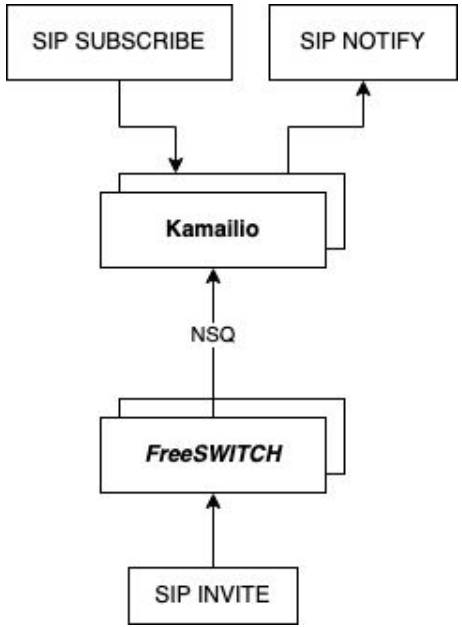
Why an nsq module?

- already had an ***nsq*** cluster
- already had event callbacks hooked to *FreeSWITCH* ESL publishing to our ***nsq*** cluster
- influenced by ***2600hz's*** kazoo module

Before and After NSQ



Using only ***FreeSWITCH*** for presence



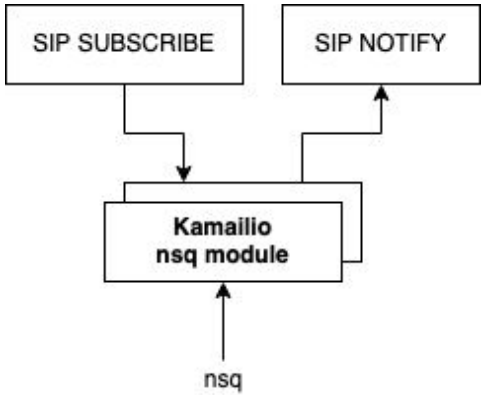
Using ***FreeSWITCH & Kamailio*** with ***NSQ*** for presence

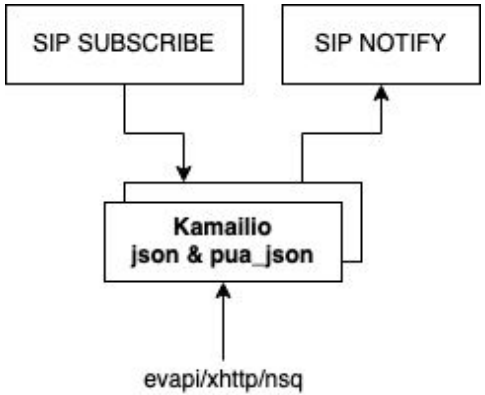
Not just NSQ anymore

- after sometime, we wanted more extensibility and less of a “blackbox” module
- moved `json` transformations out of **nsq** module and into the json module
  - extended **json** module API
- created new module pua_json
  - extended the **presence** module API
- shifted **nsq** module to just a message consumer
  - leveraged **nsq**, **json**, **pua_json**, **presence** modules to handle presence
- now it’s possible to use several other module to publish presence
  - evapi
  - xhttp
  - *many more...*

Not just NSQ anymore

Before and After

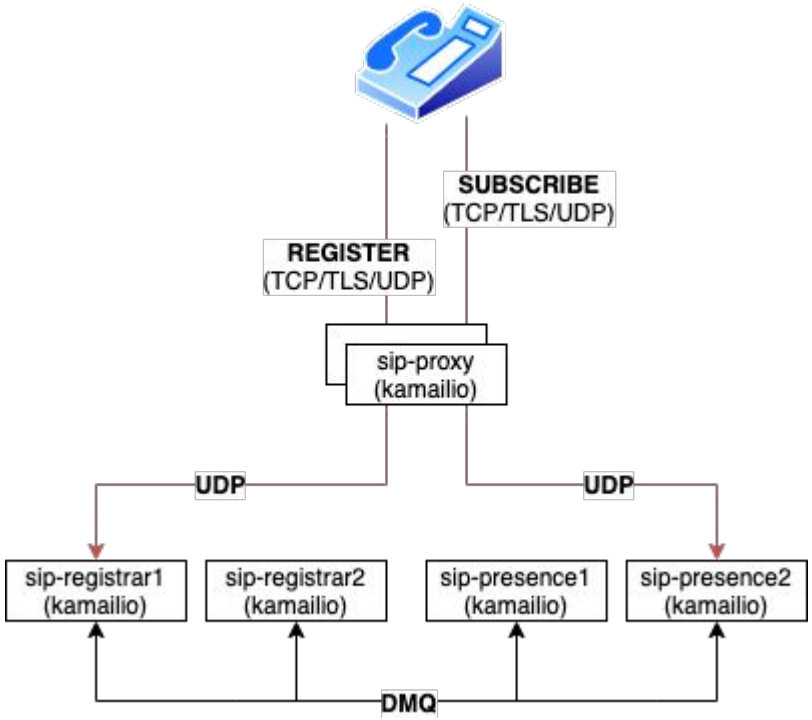




Before: the “blackbox” that was nsq

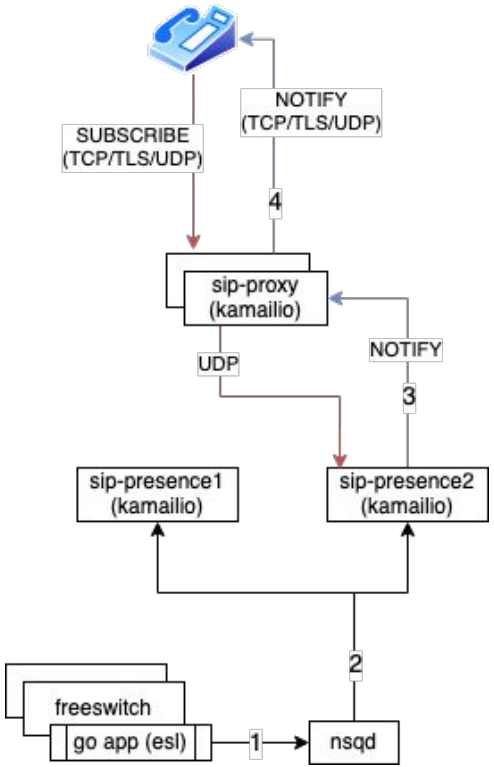
After: updates from evapi, xhttp, nsq, and many more...

Scaling SUBSCRIBEs: skip the auth



- **“edge” proxy** to remove overhead of tls/tcp
- ***dispatcher*** module to distribute traffic
- **fault-tolerance:** any node can fail/be taken out and cluster will be operational
- **dmq_usrloc:** share user location data
- check registrations data on ***SUBSCRIBE*** instead of performing authentication

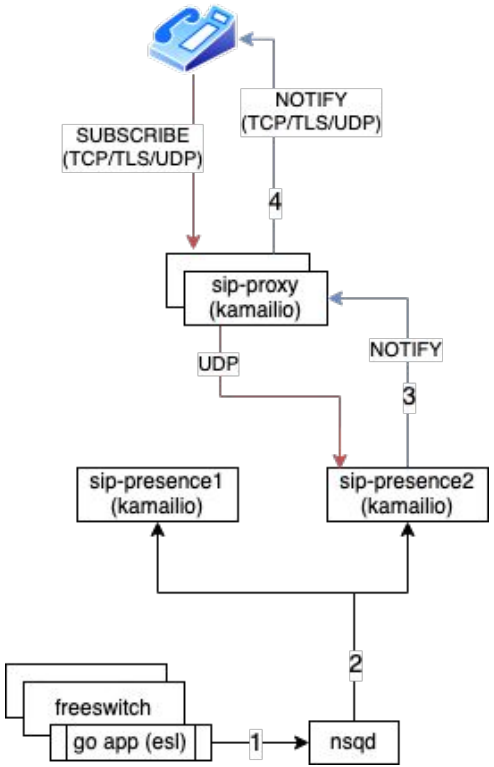
```
route[SUBSCRIBE] {
    if (is_method("SUBSCRIBE")) {
        if (!(registered("location", "$fu", 4) == 1)) {
            $var(retry) = 10 + ($RANDOM mod 300);
            append_to_reply("Retry-After: $var(retry)\r\n");
            send_reply("500", "Retry Later");
            exit;
        }
```



Scaling with NSQ

- **sip-proxy**
  - multi-homed SIP load-balancer with **dispatcher**
  - converts SIP TCP/TLS to UDP
  - server-side NAT handling
  - distributes *active watchers* to cluster of presence servers
- **sip-presence**
  - handle_subscribe()
  - active watchers & presence data
  - nsq consumer
- **freeswitch**
  - handles all calling
  - custom go app binds to “call events” via *FreeSWITCH* ESL
  - sends presence *NSQ* messages to *nsqd*

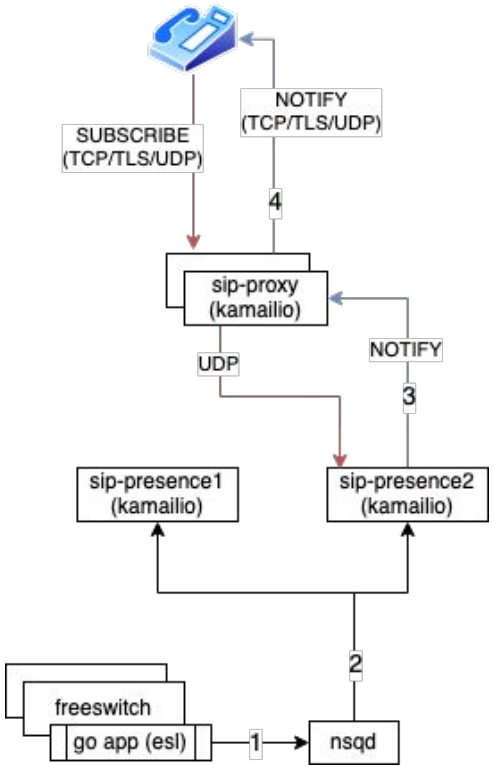
SIP PROXY ROLE




```
route[SUBSCRIBE] {
    record_route();
    if (!ds_select_dst("1", "1")) {
        send_reply("404", "No destination");
        exit;
    }
    route(RELAY);
}

route[RELAY] {
    if (is_method("NOTIFY")) {
        $var(retcode) = t_relay();
        if ($var(retcode) < 0) {
            # send reply to indicate active watcher has expired
            send_reply("408", "user cannot be reached");
        }
        exit;
    }
    t_relay();
    exit;
}
```

SIP PRESENCE ROLE




```
route[SUBSCRIBE] {
    if (is_method("SUBSCRIBE")) {
        if (!(registered("location", "$fu", 4) == 1)) {
            $var(retry) = 10 + ($RANDOM mod 300);
            append_to_reply("Retry-After: $var(retry)\r\n");
            send_reply("500", "Retry Later");
            exit;
        }
        if (!t_newtran()) {
            sl_reply_error();
            exit;
        }
        if (!handle_subscribe()) {
            xlog("L_WARN", "unsupported message [$rm] from [$fu] to [$tu]");
        }
        t_release();
        exit;
    }
}

event_route[nsq:consumer-event-presence-update] {
    pua_json_publish($nsqE);
}
```

FreeSWITCH ROLE

```
<extension name="ext100">
  <condition field="destination_number" expression="^100$">
    <action application="set" data="presence_id=${sip_from_uri}"/>
    <action application="bridge" data="[^^:presence_id=100@test1.voipxswitch.com]sofia/internal/sip:100@test1.voipxswitch.com;fs_path=sip:registar1.voipxswitch.com"/>
  </condition>
</extension>
```

- **presence_id**=[user@domain]

```
root@sip-comm1:~# telnet localhost 8021
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Content-Type: auth/request


auth ClueCon

Content-Type: command/reply
Reply-Text: +OK accepted

events CHANNEL_CALLSTATE

Content-Type: command/reply
Reply-Text: +OK event listener enabled plain
```

```
Event-Name: CHANNEL_CALLSTATE
...
Channel-Call-State: RINGING
Presence-Call-Direction: inbound
Channel-Presence-ID: 102%40test1.voipxswitch.com
Channel-Call-UUID: 7766c1cd-d51c-4f03-a281-1479acb3c58e
Caller-Destination-Number: 100
```

examples... nsq module + go

```
# ----- nsq params -----
modparam("nsq", "consumer_workers", 16)
modparam("nsq", "topic_channel", "KamailioWorld:Demo#ephemeral")
modparam("nsq", "lookupd_address", "nsqlookup-01")
modparam("nsq", "consumer_event_key", "Event-Category")
modparam("nsq", "consumer_event_subkey", "Event-Name")
```

```
event_route[nsq:consumer-event-presence-update] {
    pua_json_publish($nsqE);
}
```

```
type presenceUpdate struct {
    CallID    string `json:"Call-ID,omitempty"`
    Category  string `json:"Event-Category,omitempty"`
    Name      string `json:"Event-Name,omitempty"`
    Package   string `json:"Event-Package,omitempty"`
    Expires   string `json:"Expires,omitempty"`
    Direction string `json:"Direction,omitempty"`
    From      string `json:"From,omitempty"`
    FromUser  string `json:"From-User,omitempty"`
    FromRealm string `json:"From-Realm,omitempty"`
    To        string `json:"To,omitempty"`
    ToUser    string `json:"To-User,omitempty"`
    ToRealm   string `json:"To-Realm,omitempty"`
    State     string `json:"State,omitempty"`
}
```

```
u := presenceUpdate{
    CallID: "94829AEB-4A1B-4750-AEB6-C363D8FCD267",
    Category: "presence",
    Name: "update",
    Package: "dialog",
    Expires: "3600",
    From: "sip:park+01@example01.voipxswitch.com",
    FromUser: "park+01",
    FromRealm: "example01.voipxswitch.com",
    To: "sip:user01@example01.voipxswitch.com",
    ToUser: "user01",
    ToRealm: "example01.voipxswitch.com",
    Direction: "initiator",
    State: "confirmed",
}
b, err := json.Marshal(u)
if err != nil {
    return err
}
err = producer.Publish(ctx, "KamailioWorld", b)
if err != nil {
    return err
}
```

more examples...xhttp module + curl

```
listen=tcp:127.0.0.1:8080
tcp_accept_no_cl=yes

loadmodule "xhttp.so
loadmodule "json.so"
loadmodule "pua_json.so"
```

```
event_route[xhttp:request] {
    if ($(rb{json.parse,Event-Package}) == "dialog") {
        pua_json_publish($rb);
    }
    xhttp_reply("200", "OK", "application/json", "");
}
```


```
❯ curl -d '{ \
        "Call-ID": "7C102584-3F2B-4678-8151-BECD02E4E6FD", \
        "Event-Category": "presence", \
        "Event-Name": "update", \
        "Event-Package": "dialog", \
        "Expires": "3600", \
        "To": "sip:user01@example01.voipxswitch.com", \
        "To-User": "user01", \
        "To-Realm": "example01.voipxswitch.com", \
        "From": "sip:park+01@example01.voipxswitch.com", \
        "From-User": "park+01", \
        "From-Realm": "example01.voipxswitch.com", \
        "State": "confirmed" \
}' http://localhost:8080/
```

It works, but does it scale?

YES! It scaled to around 300k active watchers but then…. DISK IO got in our way....

DISK IO?!?! WHY?

There ***was*** a limitation in the presence module where presence records could not run **in-memory** mode. This meant reading/writing to a DB each time a presence update was sent to kamailio.

IN-MEMORY MODE (5.4 release)

- support for **in-memory** mode was on my TODO list for a long time. Unfortunately, many other things got prioritized ahead of it...
- Daniel to the rescue!!
  - recent release of 5.4 include support for full **in-memory** mode
  - http://kamailio.org/docs/modules/devel/modules/presence#presence.p.publ_cache

```
...
modparam("presence", "publ_cache", 2)
modparam("presence", "subs_db_mode", 0)
...
```

Thank you!

Contact Info

Emmanuel Schmidbauer
eschmidbauer@gmail.com

https://blog.voipxswitch.com/

https://www.linkedin.com/in/eschmidbauer/