High Availability For Kamailio And RTPEngine

19 Years Kamailio Development Celebration - Sept. 2-3, 2020

Dr. Yufei Tao

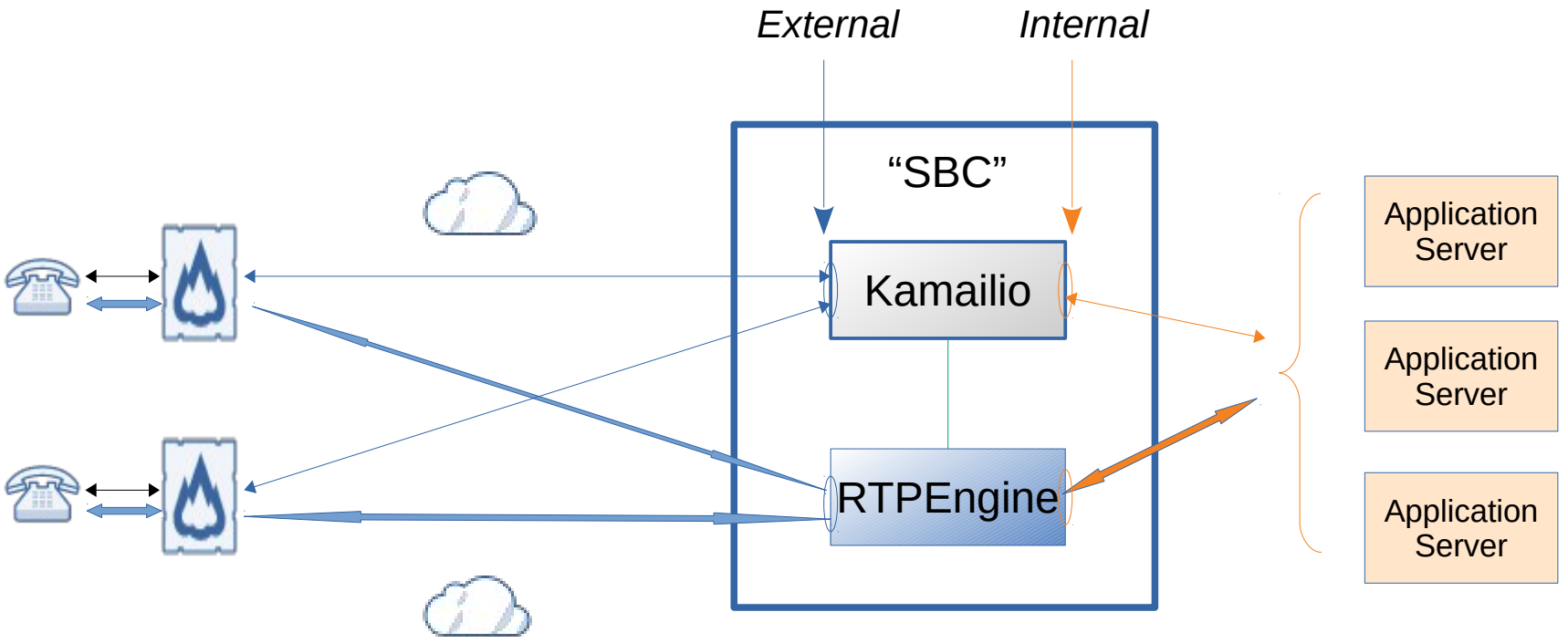
Syntec
Integrated contact centre systems

Tao Communications Ltd.

Kamailio + RTPEngine

- Kamailio for SIP signalling
- RTPEngine for media
- Many use cases
  - Proxies - relay across different network interfaces
  - NAT traversal
  - GW for conversions:
    - SIP transport protocols
    - codec transcoding
    - encryption/decryption of SIP/media
    - ...

Example “SBC” Setup



NAT traversal for remote endpoints

Relaying across network interfaces

Routing/LB for other servers

High Availability

- 1. Reliable server
  - Happy path is the easiest
  - Edge cases take more effort to handle – e.g. consider
    - Unsuccessful calls; missing call terminations ...
    - memory leak in scripts – clear htable ...
- 2. Failover setup to allow services to continue in case of...
  - Failures - application/server/host/network: software or hardware
  - Maintenance – upgrade etc.

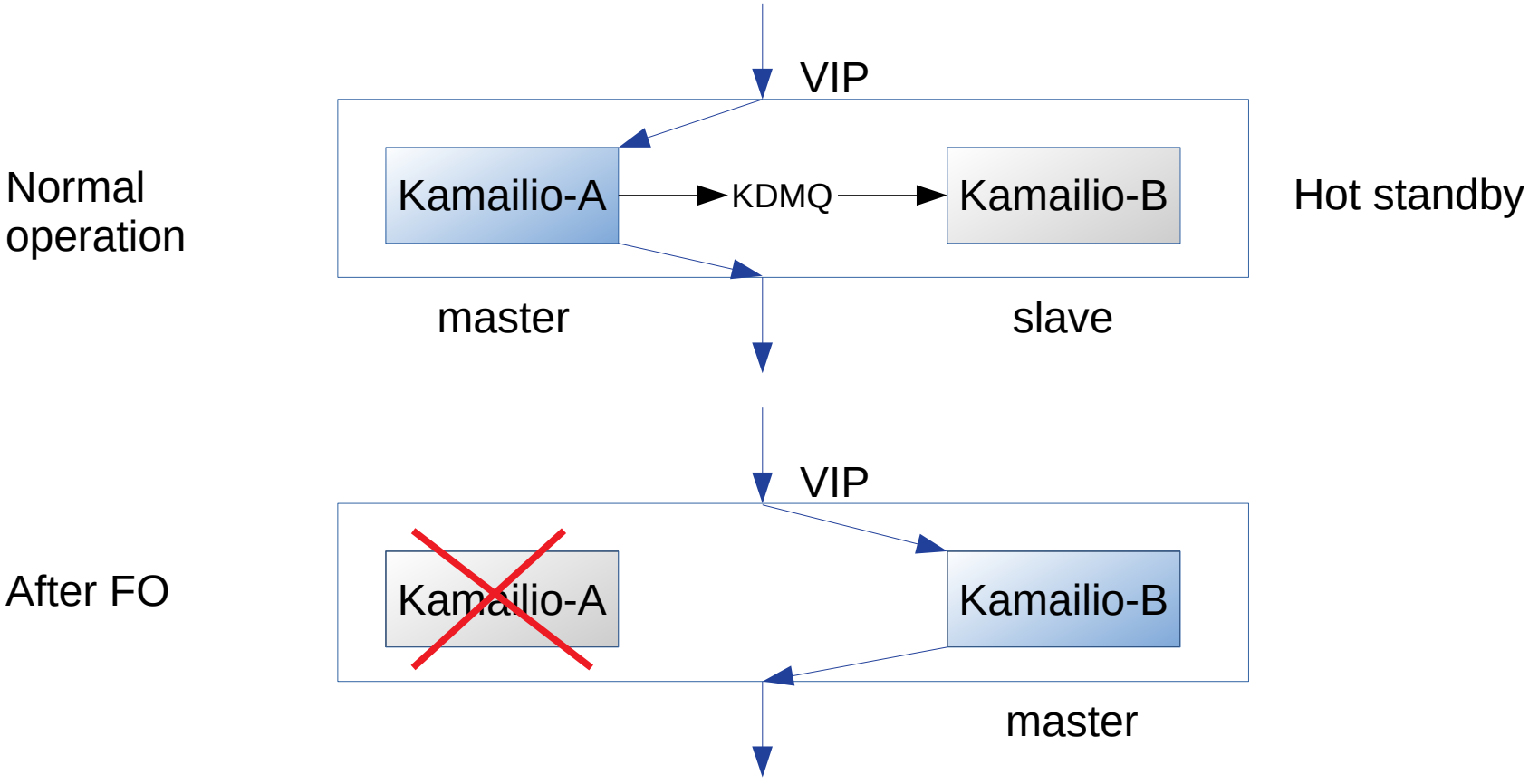
1. Reliable Servers

KISS Principle:

- To make the server more robust and easier to implement/debug/maintain/upgrade
- Many Kamailio modules to help you create services
- However you don’t have to
  - use existing modules
  - use ready-made DB tables

    … if they’re an overkill for your needs
- Simple proxy can be kept simple
  - Get the SIP headers right and send to the right place on the right socket
  - May craft your own htables, e.g. per call info: $sht(MyDlg=>$ci::<key>)
    - To store e.g. tags, call direction etc. to play media to a call leg ...

2. FO Setup Example



Failover (FO)

- Keepalived is a popular choice
  - Ability to monitor network interface and application statuses
  - Assigns VIPs to the default master node
  - When master node down, move VIPs to slave
- Applications: listen on the VIPs

  Enable nonlocal bind in sysctl's conf file:

  net.ipv4.ip_nonlocal_bind = 1

keepalived.conf – master node:

```
vrrp_script check_apps {
  script "/etc/keepalived/mychecks.sh"
  interval 1 # check every 1 second
}

vrrp_instance VI_SIP {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    nopreempt
    virtual_ipaddress {
        <Your_VIP> dev eth0
    }
    track_script {
      check_apps
    }
    # Run script on state changes
    notify /etc/keepalived/keepalive-notify.sh
}
```

Kamailio Routing Configuration

- Stateful mode - Using TM Module
  - Automatically handles transaction layer tasks e.g. retransmissions, timeouts, sending local ACKs …
  - Depended on by other useful modules you may need
- No support for replication of transactions
- But still possible to handle most FO scenarios

FO Outside Transaction

After call establishment

- Happens most often
  - Call duration is long
  - FO not inside transaction

Node A:
Handles INVITE-OK-ACK
Sets up routing for initial INVITE:
- Find correct destination
- [Set correct send socket]
- Record route

Node B:
Handles in-dlg transactions e.g. BYE-OK
- Use loose_route()

FO Inside Transaction - Response

- OK to INVITE hits B
  - Gap between 1xx and OK can be many seconds
- Error response hits B

Node A:
Handles INVITE-1xx
Sets up routing for initial INVITE:
- Find correct destination
- [Set correct send socket]
- Record route

Node B:
Handles Response-ACK
- Response: handled using Via headers (may need to force socket)
- ACK for OK: relayed using loose_route()
- ACK for error response: different

FO Inside Transaction - CANCEL

- INVITE-1xx hits node A
- CANCEL-OK, 487-ACK hits node B
  - May find destination for it same way as for INVITE

> *RFC 3261 Section 9.1:*
> *"The following procedures are used to construct a CANCEL request. The Request-URI, Call-ID, To, the numeric part of CSeq, and From header fields in the CANCEL request MUST be identical to those in the request being cancelled, including tags".*

  - Or may use saved destination from processing INVITE

```
# Without FO: CANCEL relayed only when
#    relevant INVITE transaction exists
# With FO: find destination and relay

request_route {
        …

        if (is_method("CANCEL")) {
                if (t_check_trans() < 0) {
                     route(MY_ROUTING);
                }
                route(RELAY);
        }
        …
}
```

FO – ACK to Error Response

Usually

```
route[WITHINDLG] {
        ...
        if ( is_method("ACK") ) {
                if ( t_check_trans() ) {
                        # no loose-route, but stateful ACK;
                        # must be an ACK after a 487
                        # or e.g. 404 from upstream server
                        route(RELAY);
                        exit;
                } else {
                        # ACK without matching transaction
                        #  ignore and discard
                        exit;
                }
        }
        …
}
```

With FO

```
route[WITHINDLG] {
        …
        if ( is_method("ACK") ) {
                if ( !t_check_trans() ) {
                        # Find or use saved destination
                        route(MY_ROUTING);
                }

                # Relay anyway
                route(RELAY);
        }
        …
}
```

Difference in this ACK by node B compared to that sent by node A (with no FO) ?

Extra via header from previous hop

Kamailio FO - Summary

- Replication
  - htable: may be used to store call info
  - DMQ: for replication so slave has all info it needs to take over when master is down
- Summary
  - For a simple Kamailio proxy it is possible to handle FO for most cases
    - Most important case: established calls, call attempts
  - A couple of seconds for VIPs to migrate

RTPEngine FO

- Hot standby RTPEngine – avoid loss of media
- FO more noticeable than for Kamailio
- Easier to handle from scripting point of view
- FO based on Redis keyspace notifications
  - 1&1 Presentation at KW16 by Pawel Kuzak:

    https://www.kamailio.org/events/2016-KamailioWorld/Day2/20-Pawel.Kuzak-High-Quality-Telephony-Using-A-Fail-Safe-Media-Relay-Setup.pdf
  - Sipwise doc:

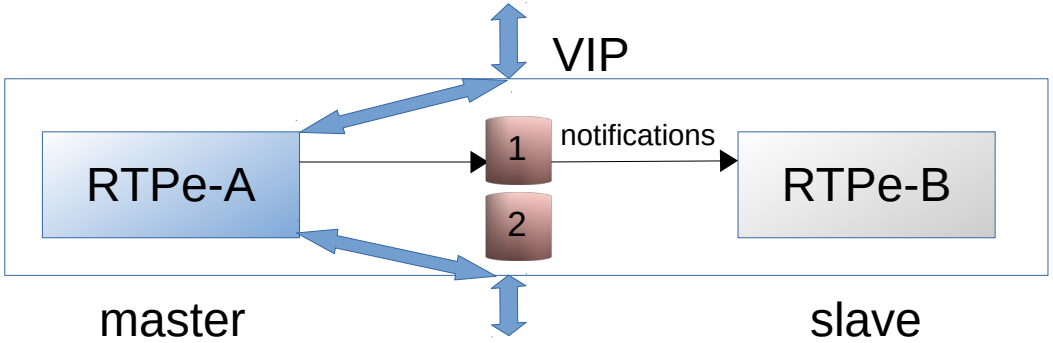
    https://github.com/sipwise/rtpengine/wiki/Redis-keyspace-notifications

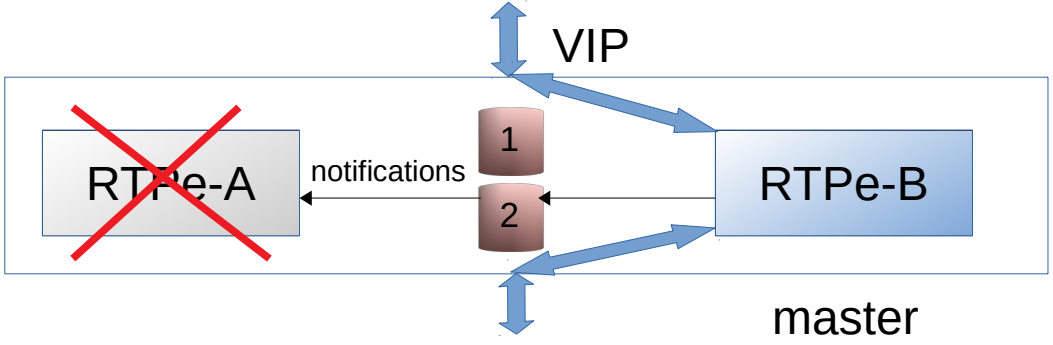
RTPEngine FO – Hot Standby

<u>Redis keyspace notification</u>

Node A:
- write to db1
- subs to db2

Node B:
- write to db2
- subs to db1





Summary

- It is possible to achieve reasonable HA using Kamailio and RTPEngine
- There are different ways to combine them
  - Experiment and find what’s best for your requirements and environment
- There are more ways – share your ideas!

Dr. Yufei Tao
yufei.tao@gmail.com