HIGH QUALITY TELEPHONY USING A FAIL-SAFE MEDIA RELAY SETUP

Pawel Kuzak
VoIP Backend Development

Agenda

- 1&1 - Member of United Internet AG
- VoIP Network Overview
  - Quality of Service
- Media relay - RTPengine
  - Extended VoIP Architecture
  - Hardware and Network configuration
  - Challenges
- Redundancy
  - RTPengine setup
  - Redis Keyspace Notifications
- Roundup

1&1 – Member of United Internet AG

Access	Consumer	1&1
	Business	versatel
Applications	Consumer	GMX, WEB.DE, mail.com, 1&1, united internet media, affilinet
	Business	fasthosts, home.pl, united domains, 1&1, arsys, InterNetX, sedo
	Partners	VIRTUALMINDS 25.10%, PROFITBRICKS The IaaS-Company. 28.57%, Stay Open. OX 29.02%, epages 25.10%, uberall 30.36%
	Listed investments	DRILLISCH 20.11%, himedia 10.46%, hipay 8.37%*, ROCKET INTERNET 8.31%

* Spin-off of Hi-Media S.A.

1&1: Internet services of United Internet AG

united internet

Access | **Applications**

Networks

User equipment

Content

Standard software

Motivated team
- Around 8,200 employees, thereof approx. 2,700 in product management, development and data centers

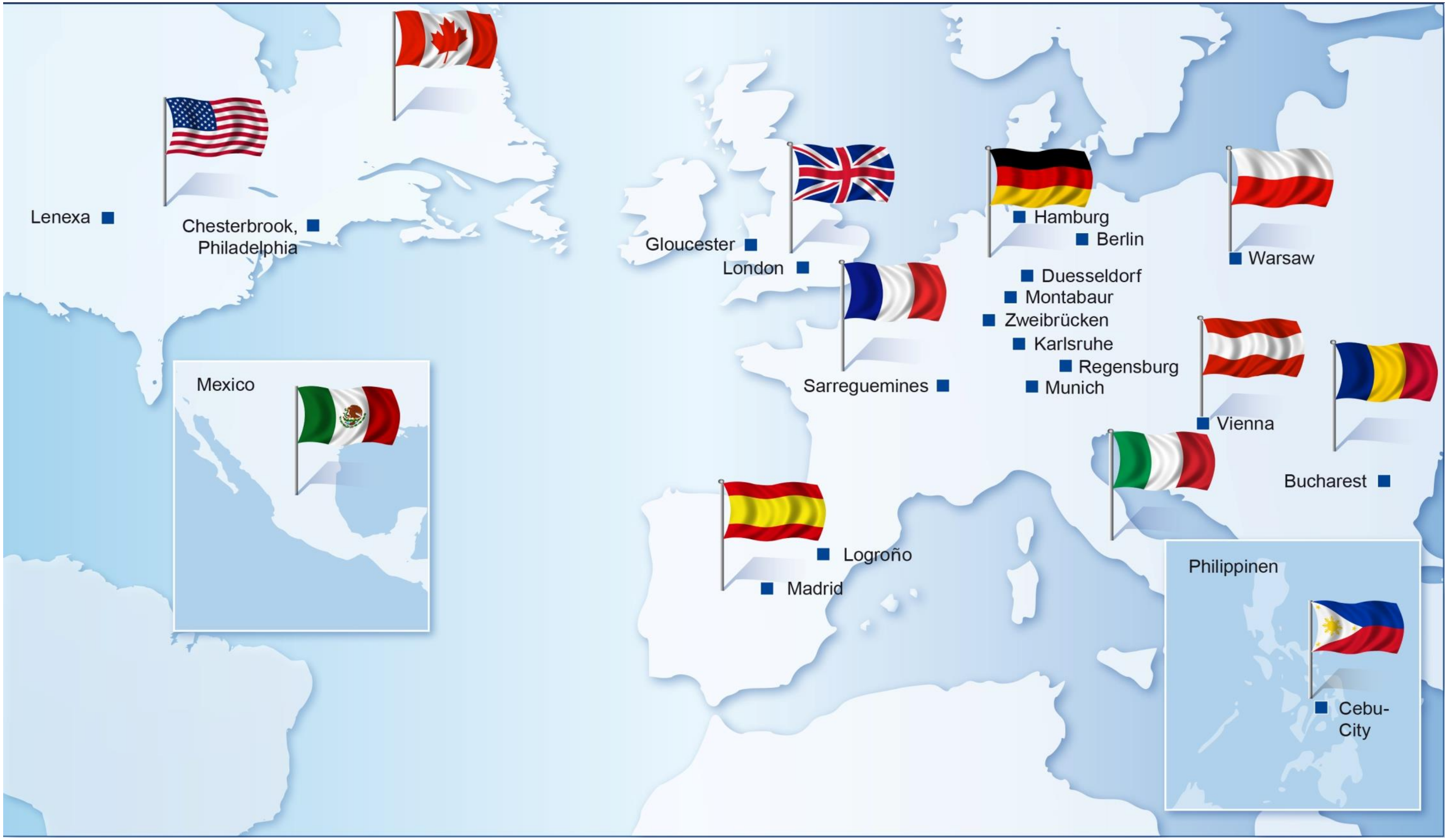
Sales strength
- Approx. 3.2 million new customer contracts p.a.
- 50,000 registrations for free services on a daily basis

Operational excellence
- 49 million accounts in 11 countries

7 data centers
- 70,000 servers in Europe and USA

Powerful network infrastructure
- 41,000 km of fiber network

Locations

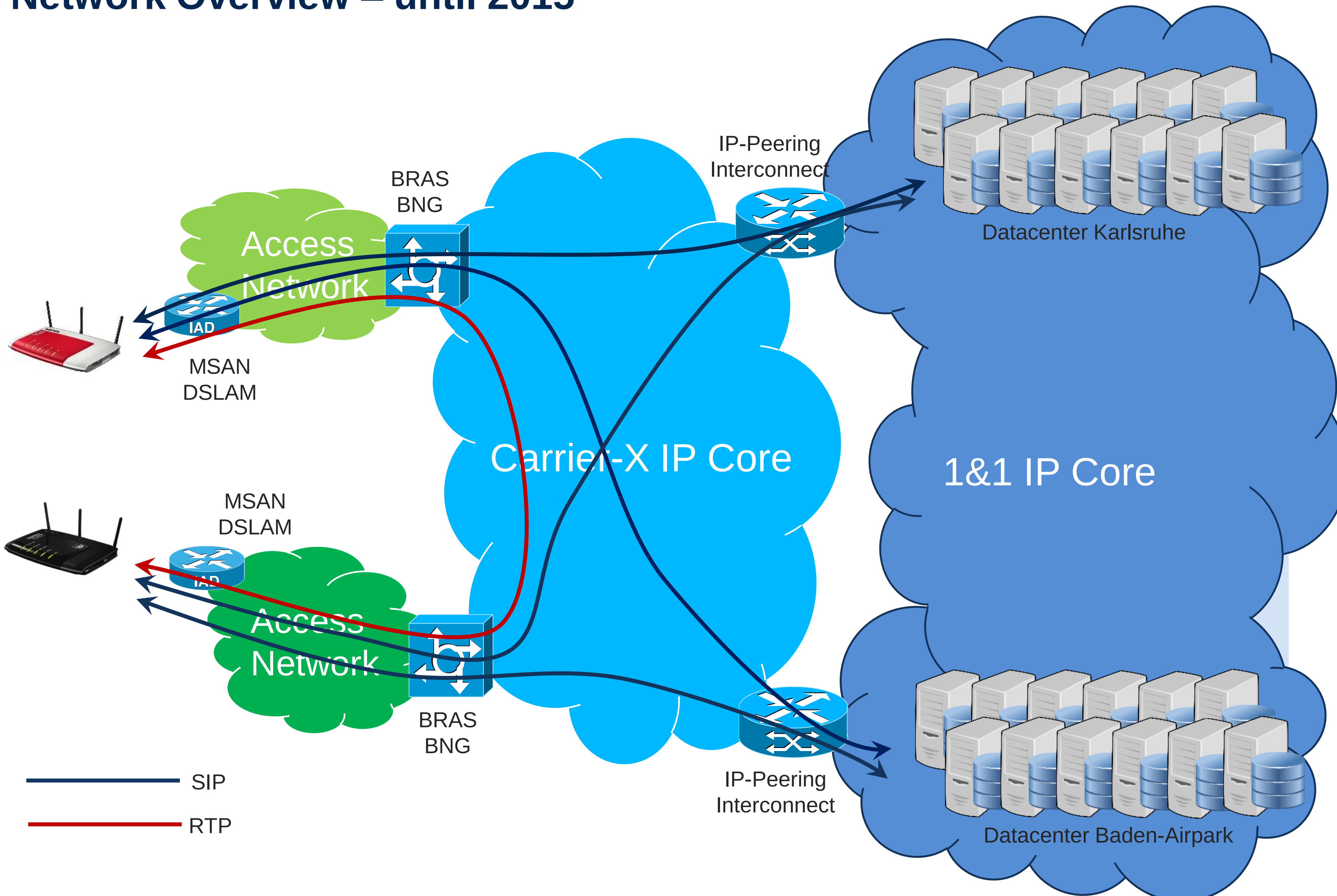


Lenexa
Chesterbrook, Philadelphia
Mexico
Gloucester
London
Sarreguemines
Hamburg
Berlin
Duesseldorf
Montabaur
Zweibrücken
Karlsruhe
Regensburg
Munich
Warsaw
Vienna
Bucharest
Logroño
Madrid
Philippinen
Cebu-City

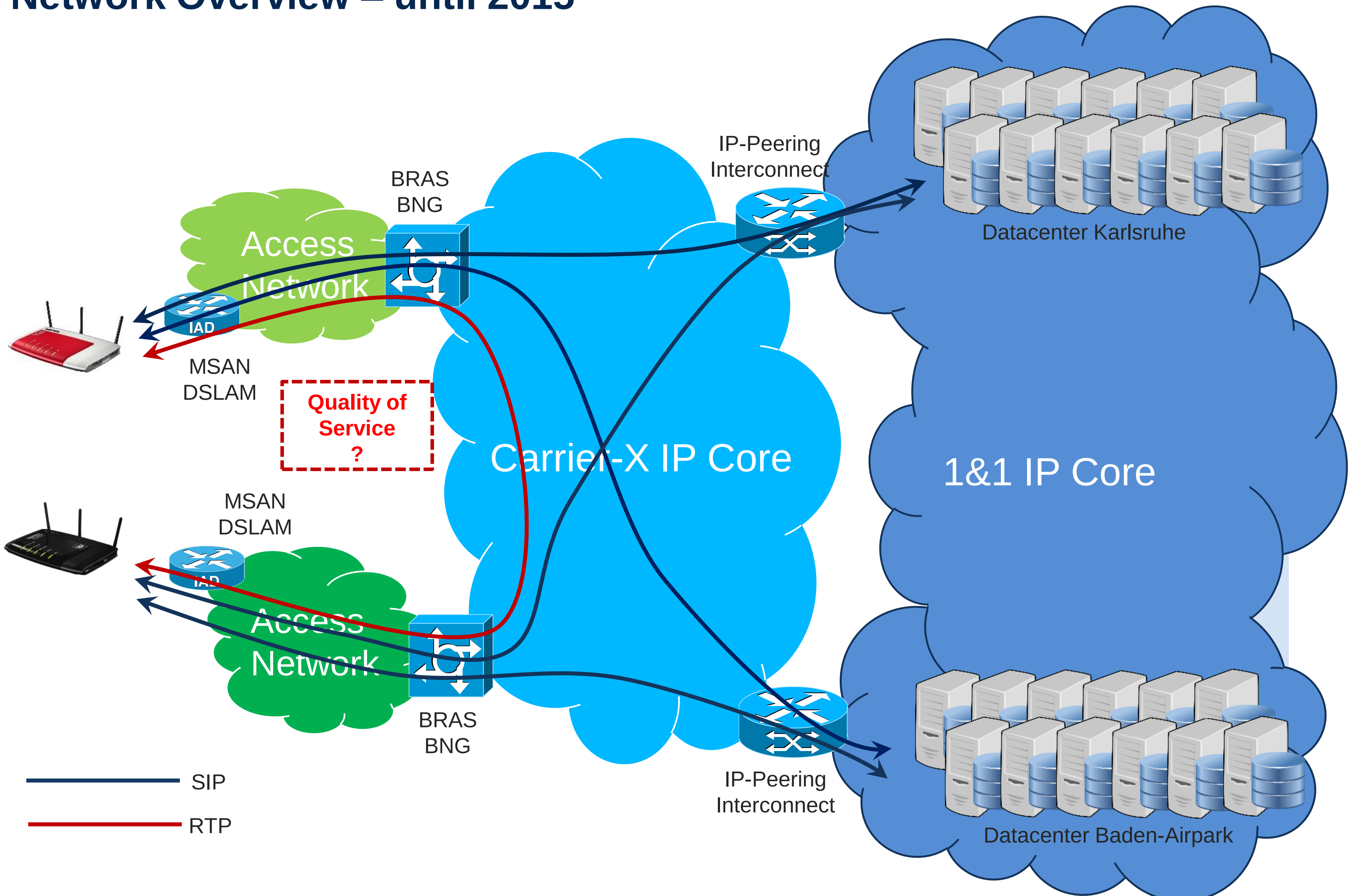
VoIP Backend – Some numbers

- > 100 servers forming a geo-redundant active-active setup
- ~ 4 Mio. VoIP customers
- ~ 12 Mio. registered numbers
- > 500 Calls/Second (Peak)
- > 120.000 concurrent callls (Peak)
- > 15.000 REGISTER requests per Second (Peak)
- 99,999% availability
- SIP interconnections with
  - Versatel
  - QSC
  - Telefonica
  - Vodafone

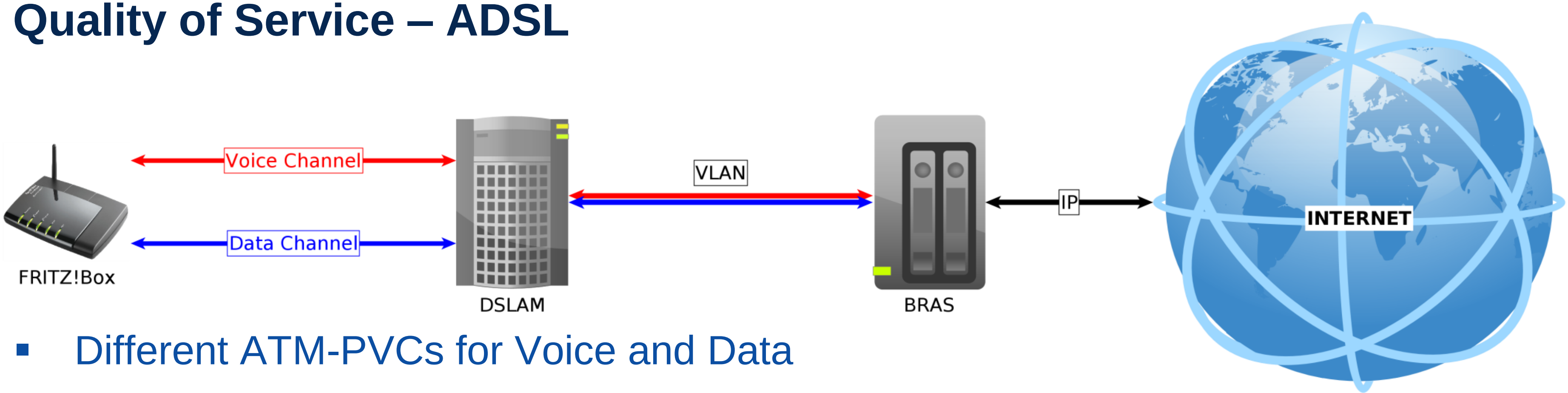
Network Overview – until 2015



Network Overview – until 2015



Quality of Service – ADSL



- Different ATM-PVCs for Voice and Data
- One PPPoE session per ATM-PVC
- Different VLANs for Voice and Data in Backbone

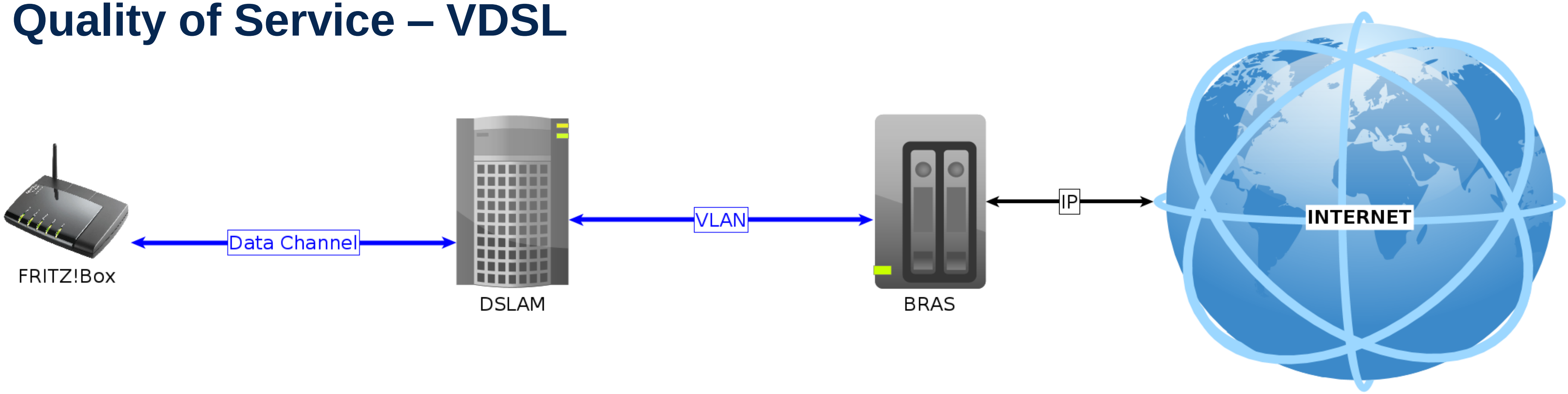
Upstream (F!B → Internet):

- F!B just seperates Data traffic from Voice traffic
- Based on the ATM-PVC, the DSLAM decides whether to use the prioritized VLAN to BRAS
- BRAS uses DSCP (sets TOS Bit)

Downstream (Internet→ F!B):

- Based on the target IP, the BRAS knows which VLAN to use
- Traffic may be routed best-effort to BRAS

Quality of Service – VDSL



- No distinction between Voice and Data
- One VLAN for Voice and Data in Backbone

Quality of Service – VDSL

Carrier won‘t prioritize IP traffic unless it doesn‘t come from a pre-defined IP range

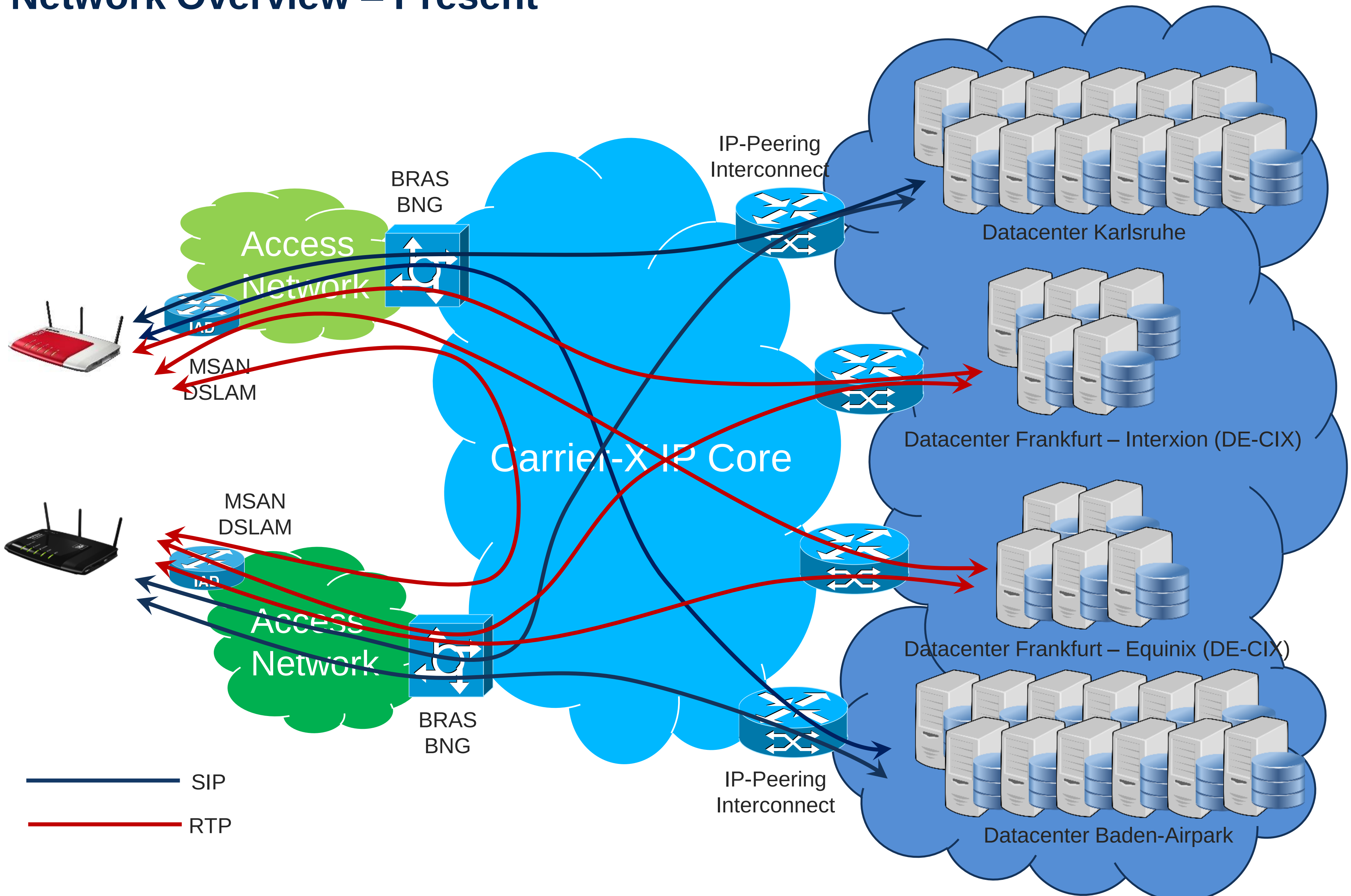
→ Prioritization of end-to-end RTP traffic not possible
  → IPs spread
→ Need to bundle RTP traffic
→ Need media relay

Requirements for media relay:

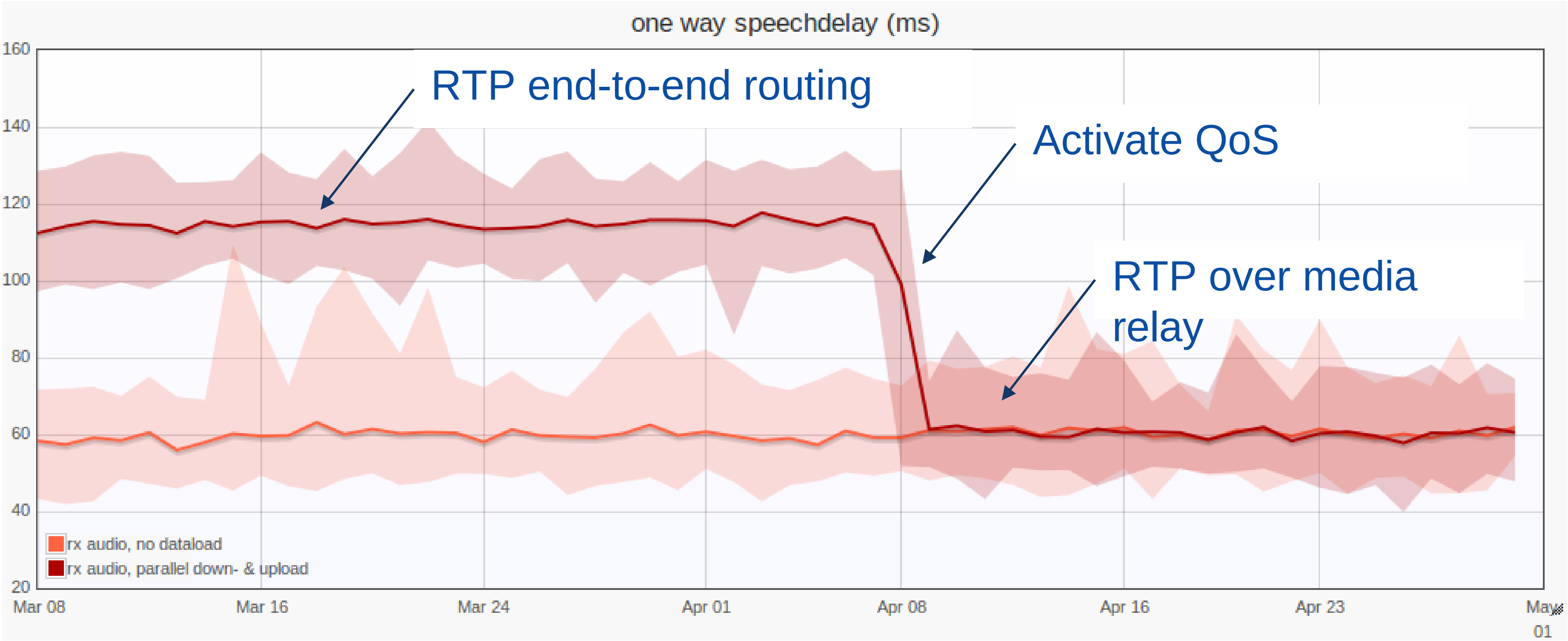
- Open source
- Kamailio integration
- High performance
- High availability
- Scalability
- TOS – Bit setting

→ RTPengine (https://github.com/sipwise/rtpengine)

Network Overview – Present



QoS impact – VDSL: Speech delay VDSL-ISDN



Media relay – Some numbers

- Currently 6 server in 2 data centers
  - Dell R430, Intel X520 Network 10 Gbit/s, 2x 10 Core Xeon E5 w/ HT
  - Running RTPengine controlled via Kamailio‘s rtpengine module
- Full redundant active-active setup
- Each server handling 3.000 concurrent sessions (Peak)
  - ~ 300.000 packets/s incoming and outgoing
- Each server able to handle 18.000 concurrent sessions (Peak)
  - Worst case backup scenario
  - ~ 18.000.000 packets/s incoming and outgoing
- Successfully tested 25.000 concurrent sessions

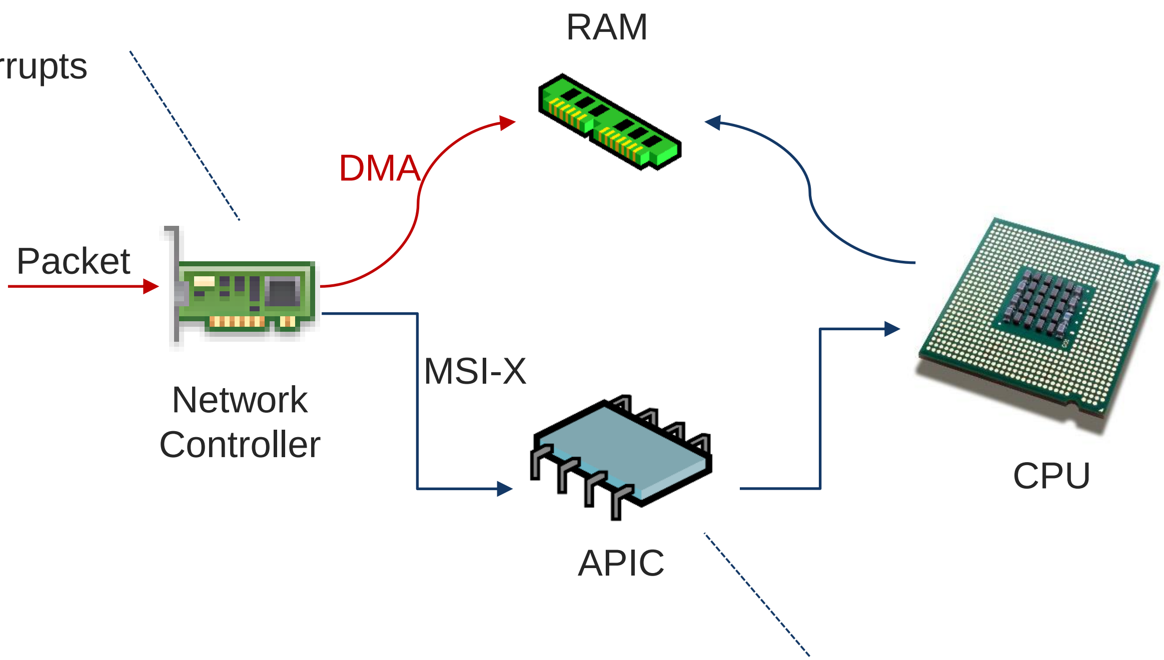
Limiting factors:

- Bandwidth ➔ Extend hardware
- Available sockets ➔ Use more IPs
- Interrupts ➔ Tune configuration

Media relay – Interrupt handling configuration

Intel® Flow director / N-tuple filter / Hash filter:
→ Map RTP/RTCP to same queue

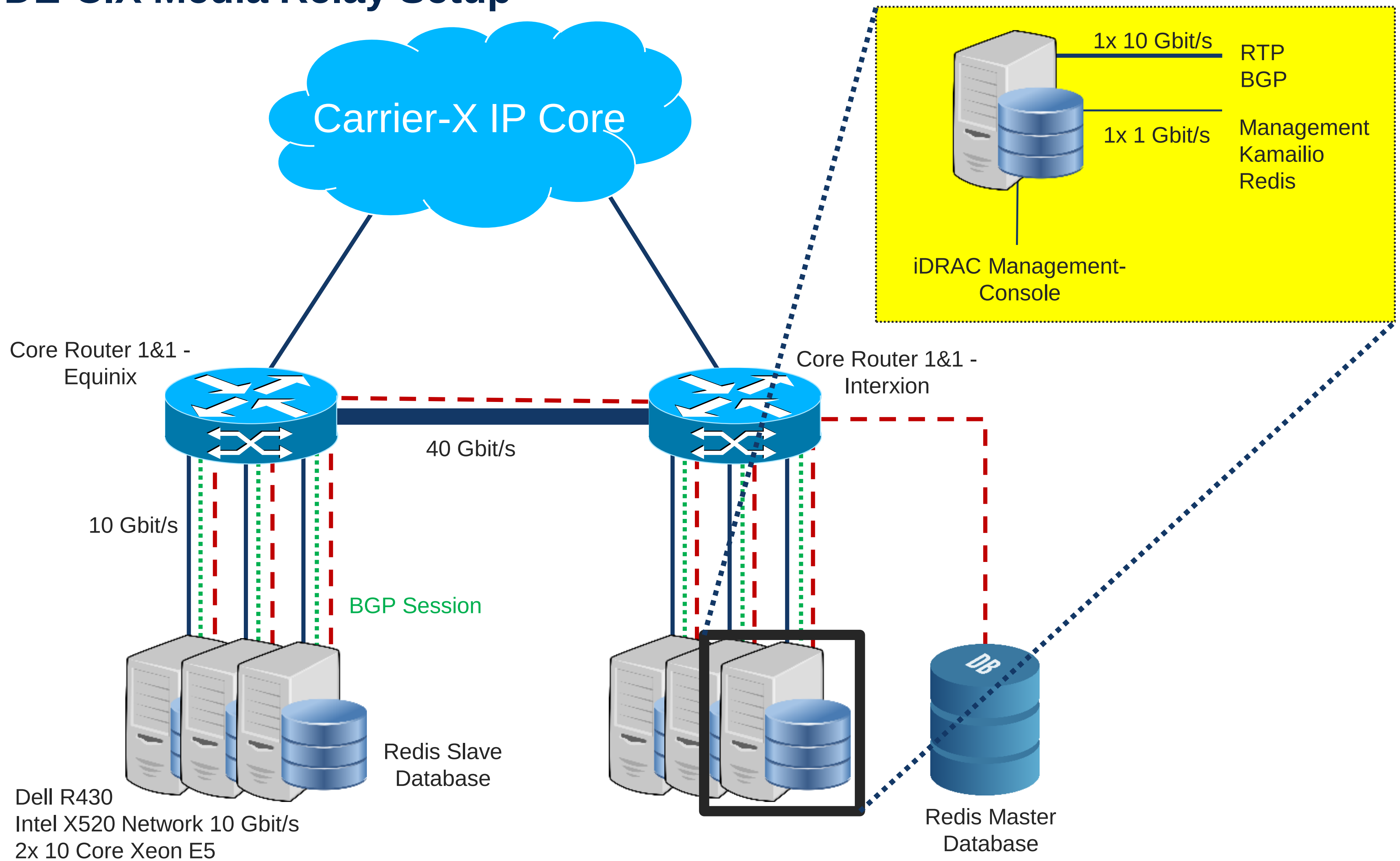
IRQ coalescing:
→ „Bundle“ interrupts



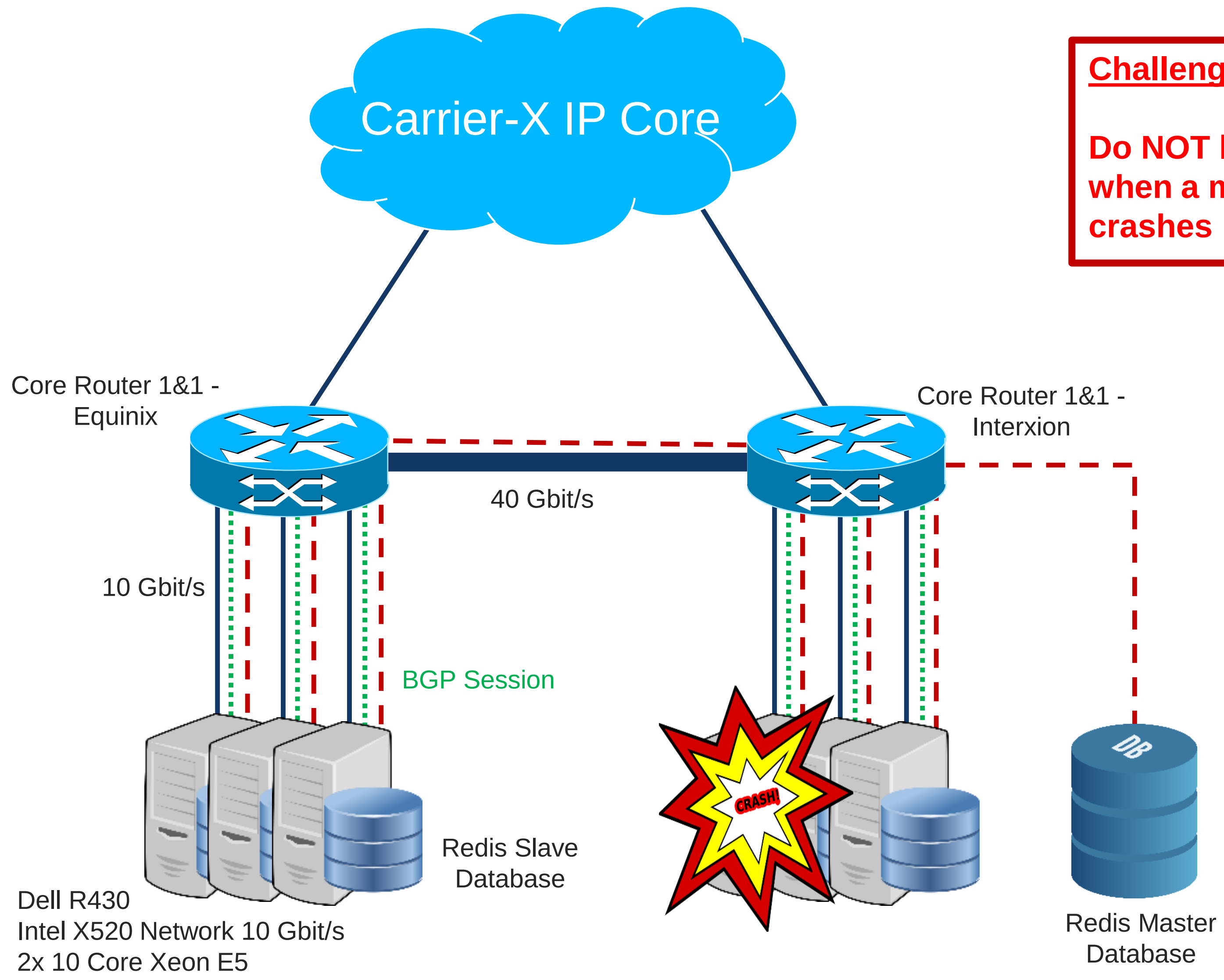
Configured IRQ-CPU core affinity:
→ Map IRQ/queues to same CPU cores

→ Optimize utilization of CPU caches
→ Improve performance
→ Session mapped 1:1 to CPU core

DE-CIX Media Relay Setup



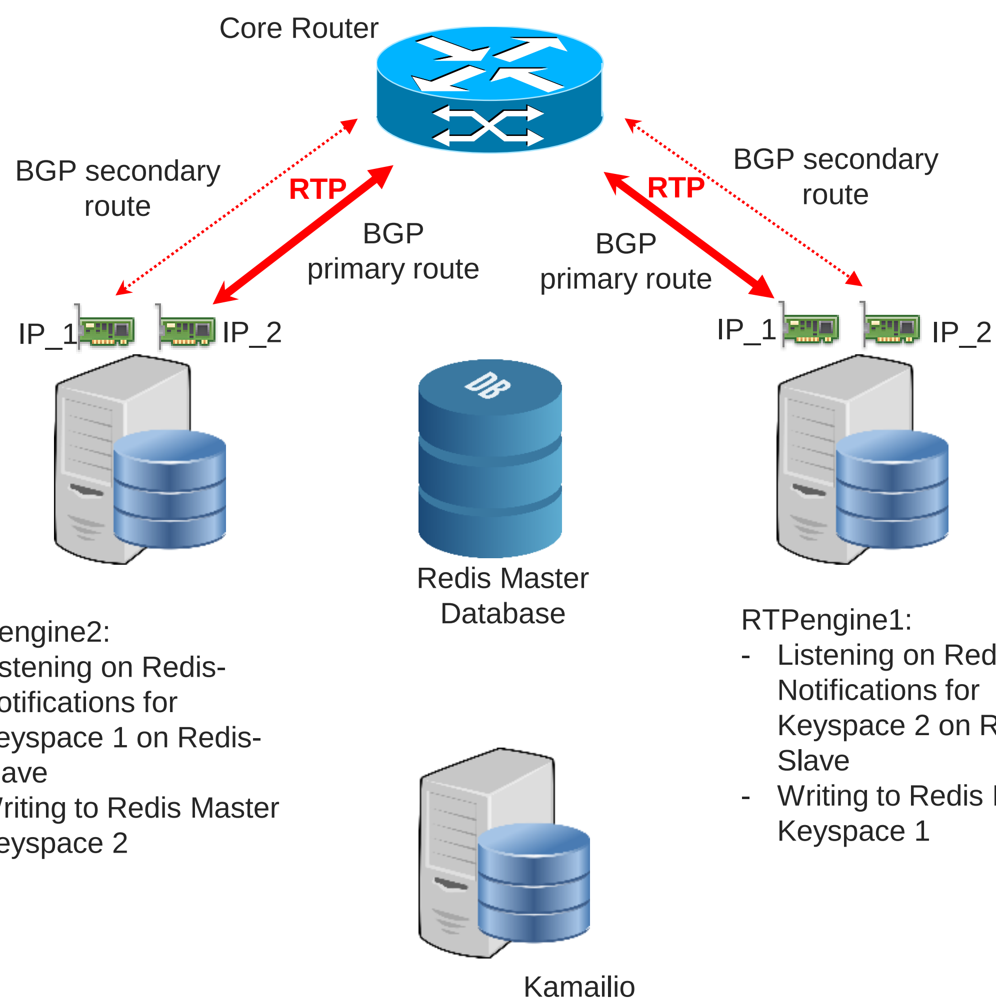
RTPengine Redundancy



Challenge:

Do NOT lose sessions when a media relay crashes

RTPengine Redundancy – Simplified illustration

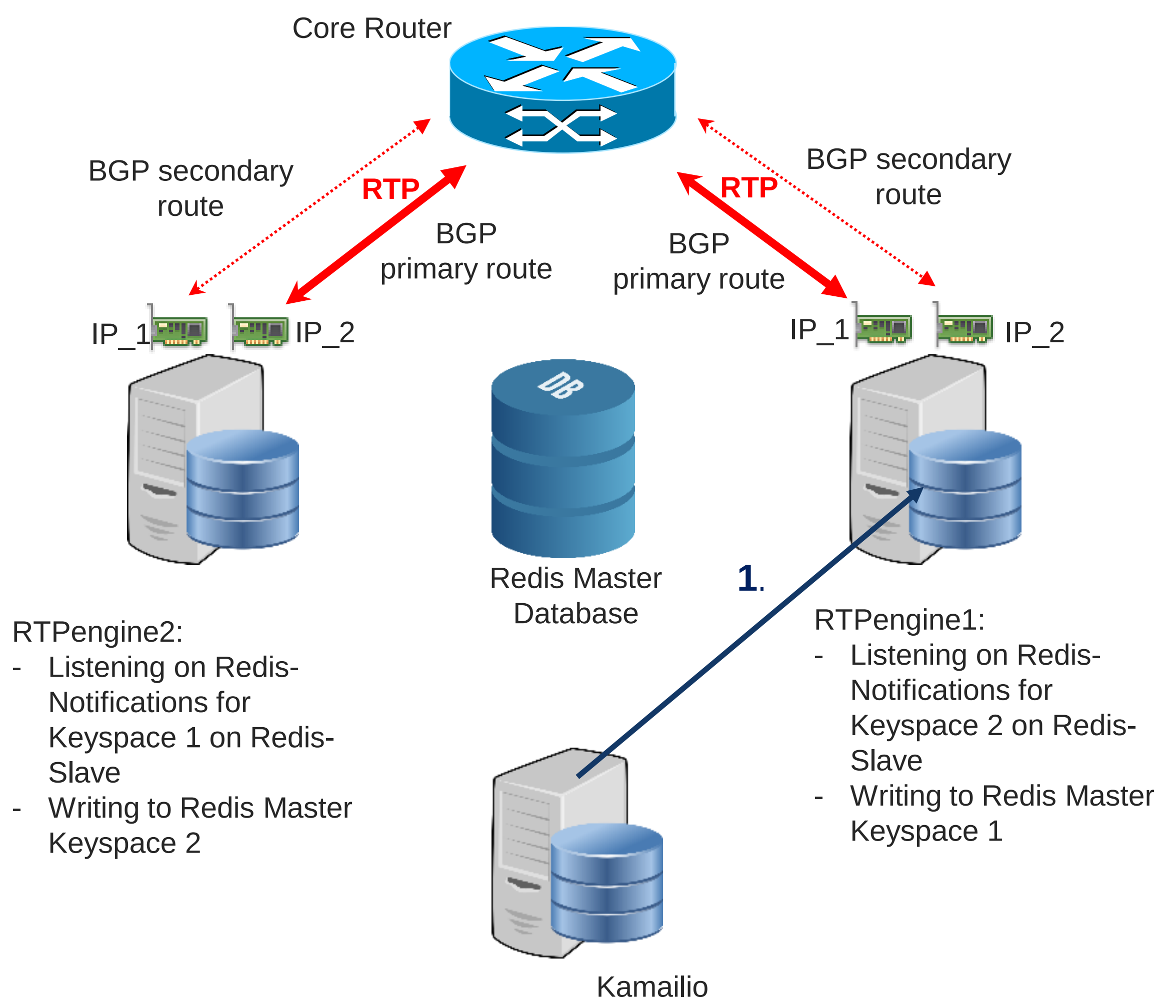


RTPengine2:
- Listening on Redis-Notifications for Keyspace 1 on Redis-Slave
- Writing to Redis Master Keyspace 2

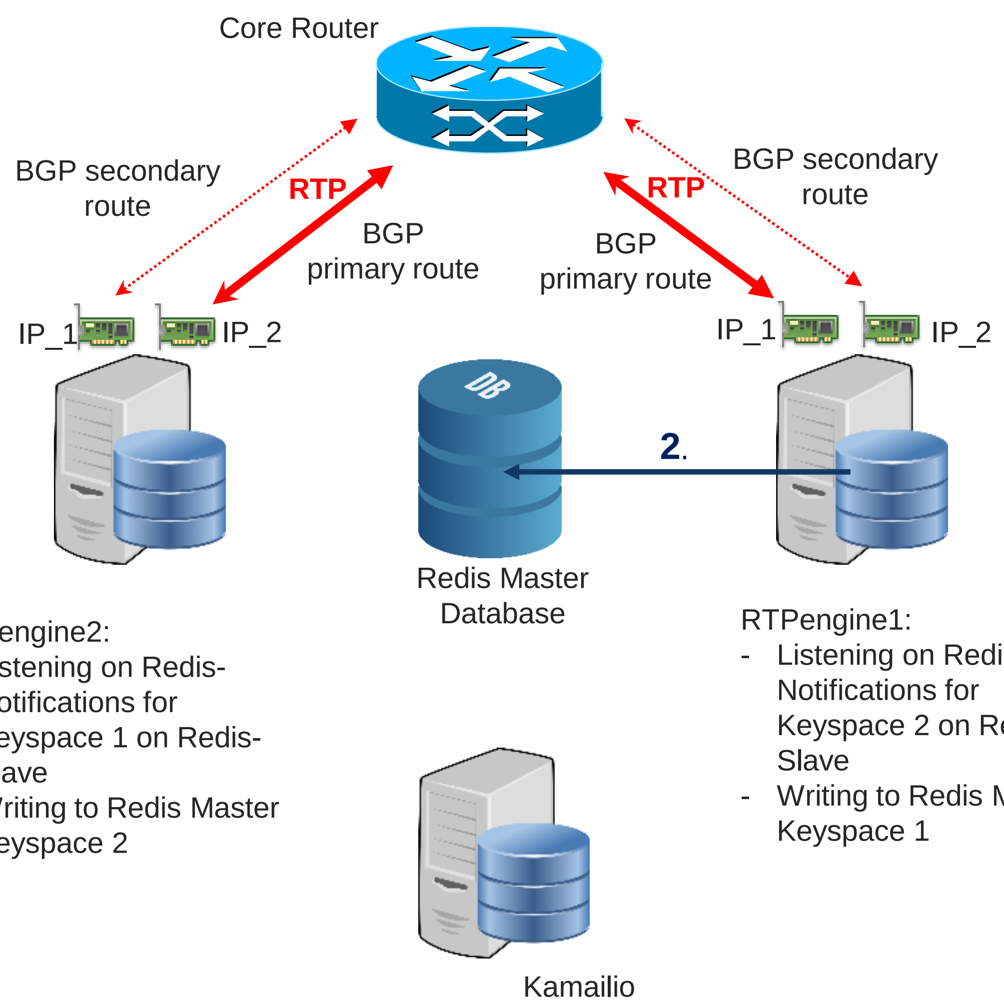
RTPengine1:
- Listening on Redis-Notifications for Keyspace 2 on Redis-Slave
- Writing to Redis Master Keyspace 1

RTPengine Redundancy – Simplified illustration

1. **Kamailio creates RTP session on RTPengine1 with active IP_1**



RTPengine Redundancy – Simplified illustration



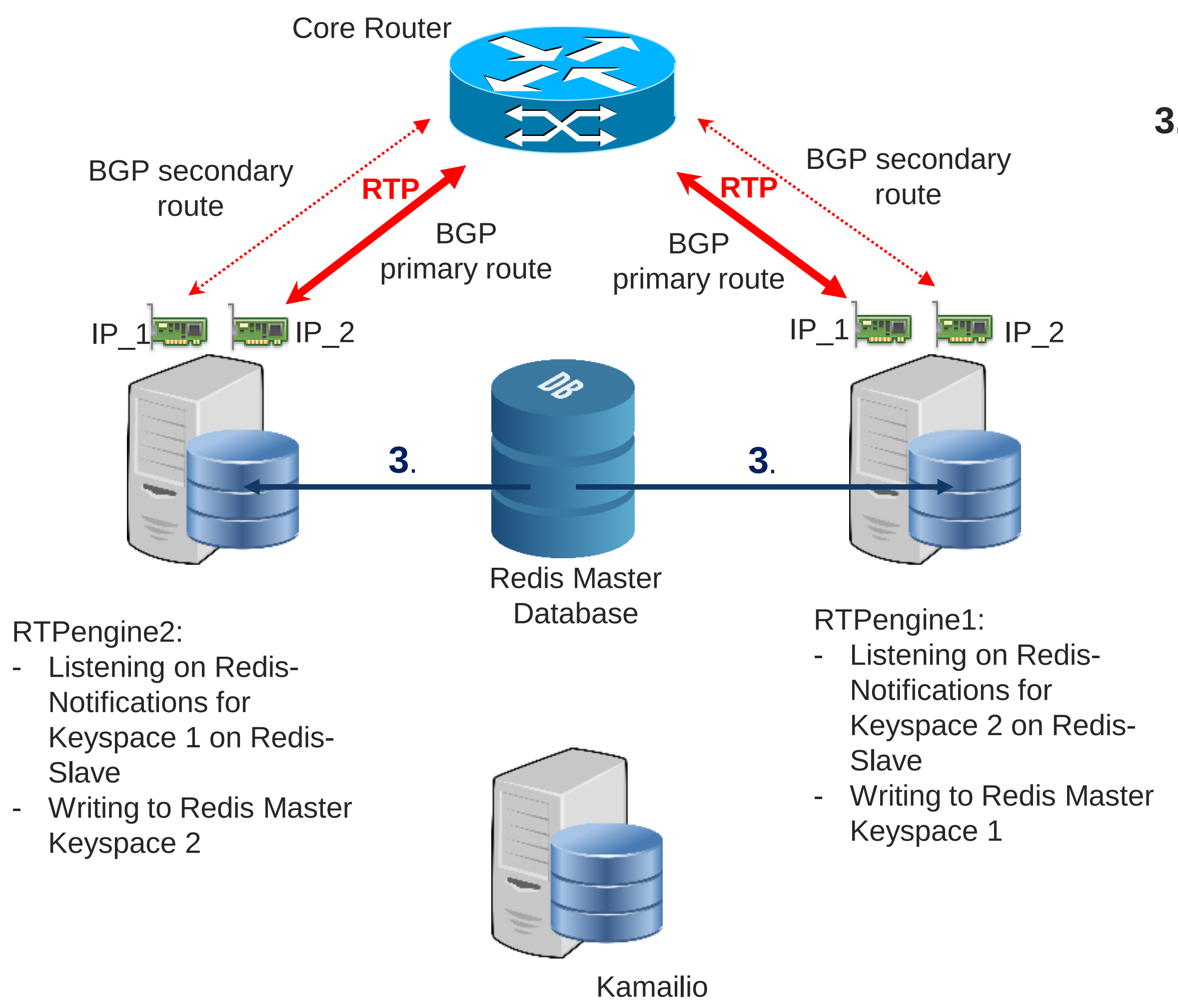
RTPengine2:
- Listening on Redis-Notifications for Keyspace 1 on Redis-Slave
- Writing to Redis Master Keyspace 2

RTPengine1:
- Listening on Redis-Notifications for Keyspace 2 on Redis-Slave
- Writing to Redis Master Keyspace 1

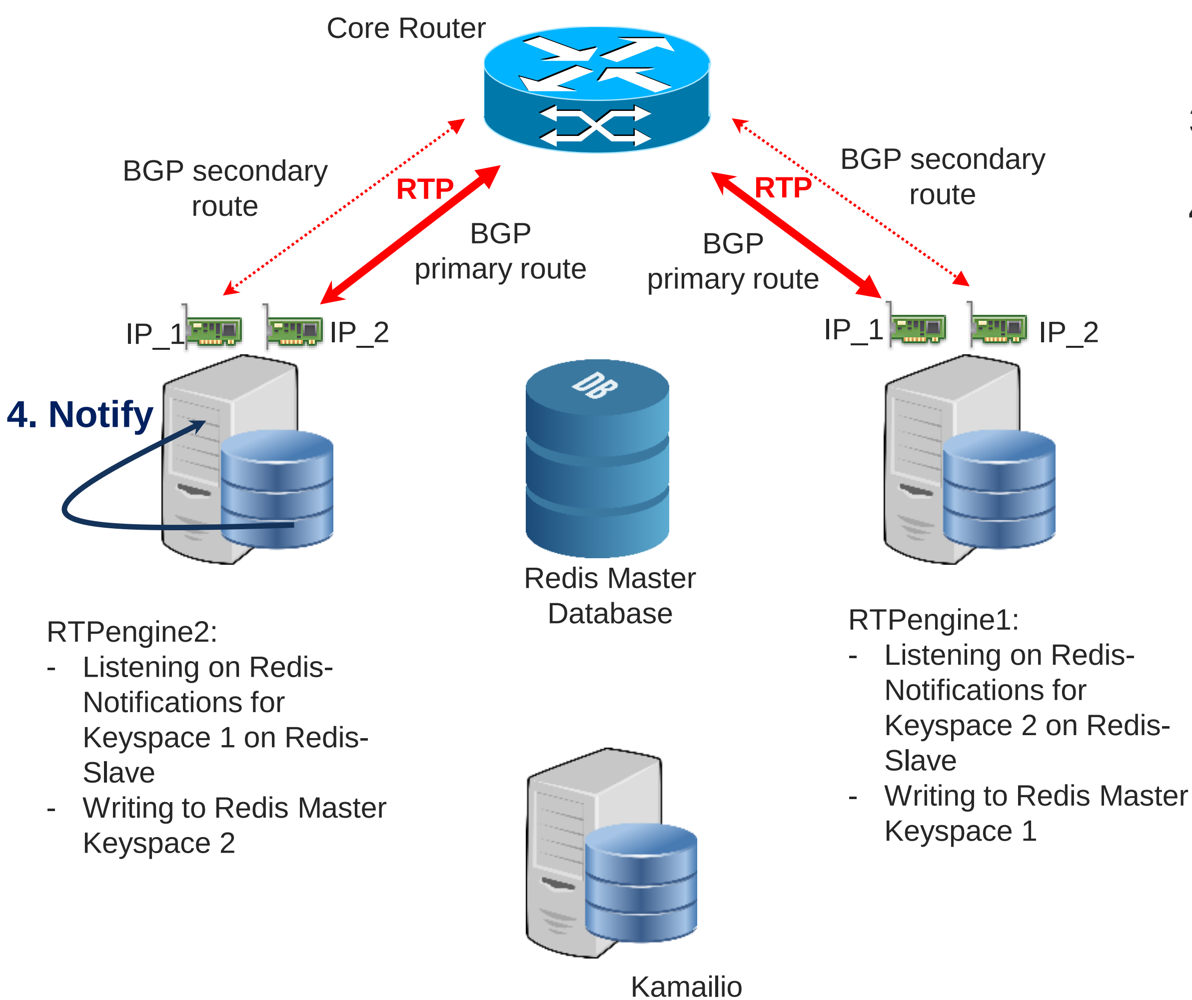
1. Kamailio creates RTP session on RTPengine1 with active IP_1
2. **RTP traffic arrives. RTPengine1 persists session information to Redis Master, Keyspace 1**

RTPengine Redundancy – Simplified illustration

1. Kamailio creates RTP session on RTPengine1 with active IP_1
2. RTP traffic arrives. RTPengine1 persists session information to Redis Master, Keyspace 1
3. **Redis Master replicates information to slaves**

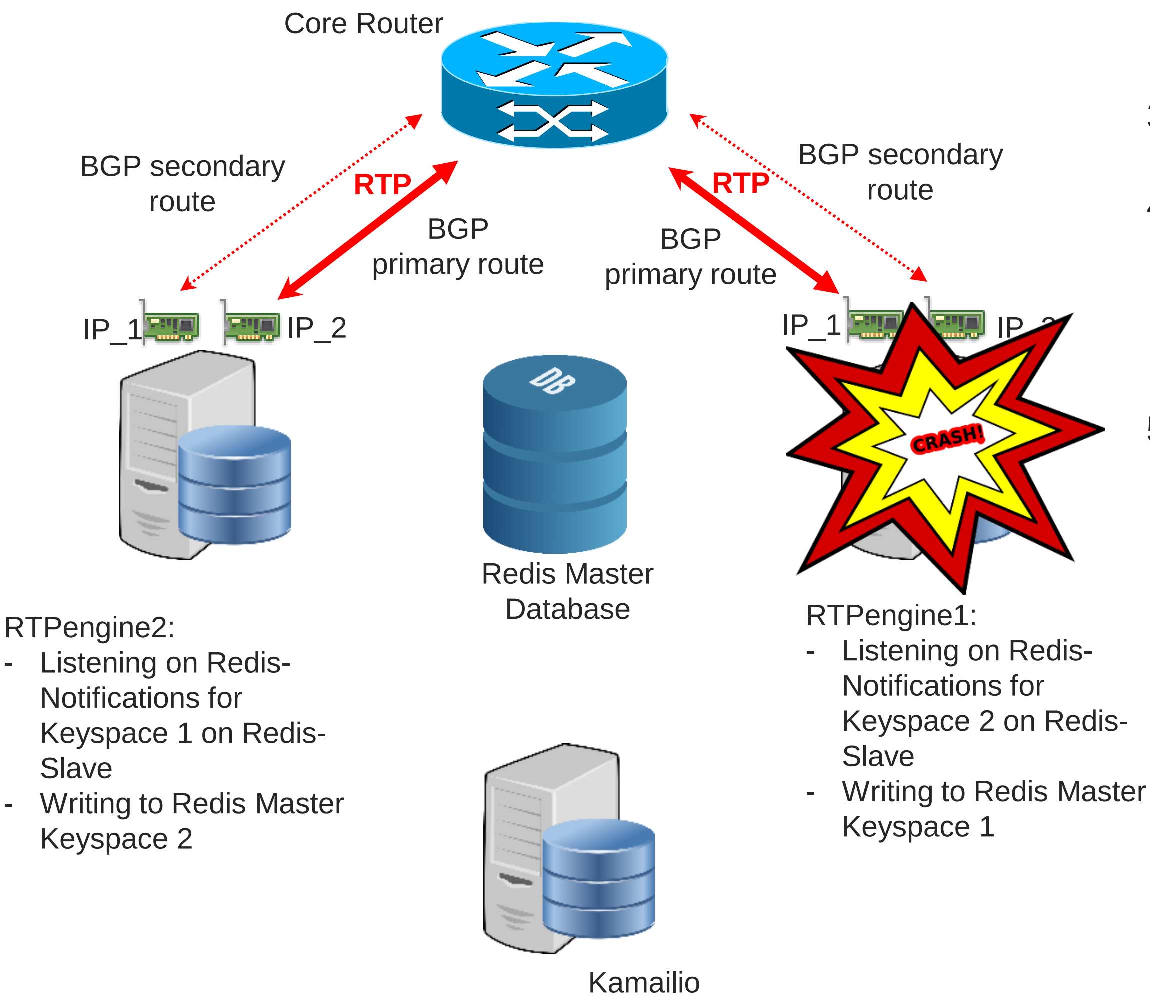


RTPengine Redundancy – Simplified illustration



1. Kamailio creates RTP session on RTPengine1 with active IP_1
2. RTP traffic arrives. RTPengine1 persists session information to Redis Master, Keyspace 1
3. Redis Master replicats information to slaves
4. **RTPengine2 (listening on Keyspace 1) handles Redis-Notification and creates session on IP_1**

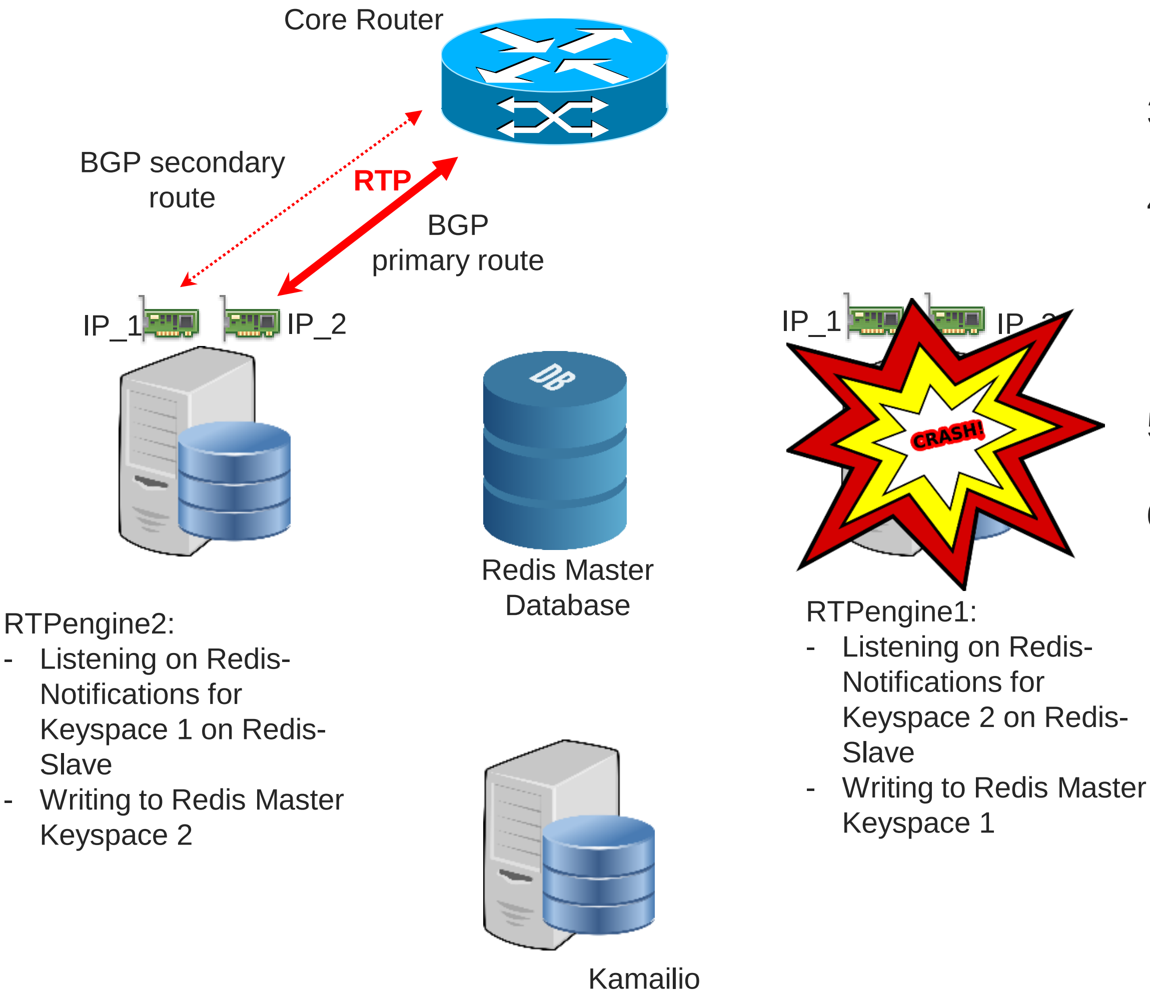
RTPengine Redundancy – Simplified illustration



1. Kamailio creates RTP session on RTPengine1 with active IP_1
2. RTP traffic arrives. RTPengine1 persists session information to Redis Master, Keyspace 1
3. Redis Master replicats information to slaves
4. RTPengine2 (listening on Keyspace 1) handles Redis-Notification and creates session on IP_1

5. **HW Error / Network Error / …**

RTPengine Redundancy – Simplified illustration

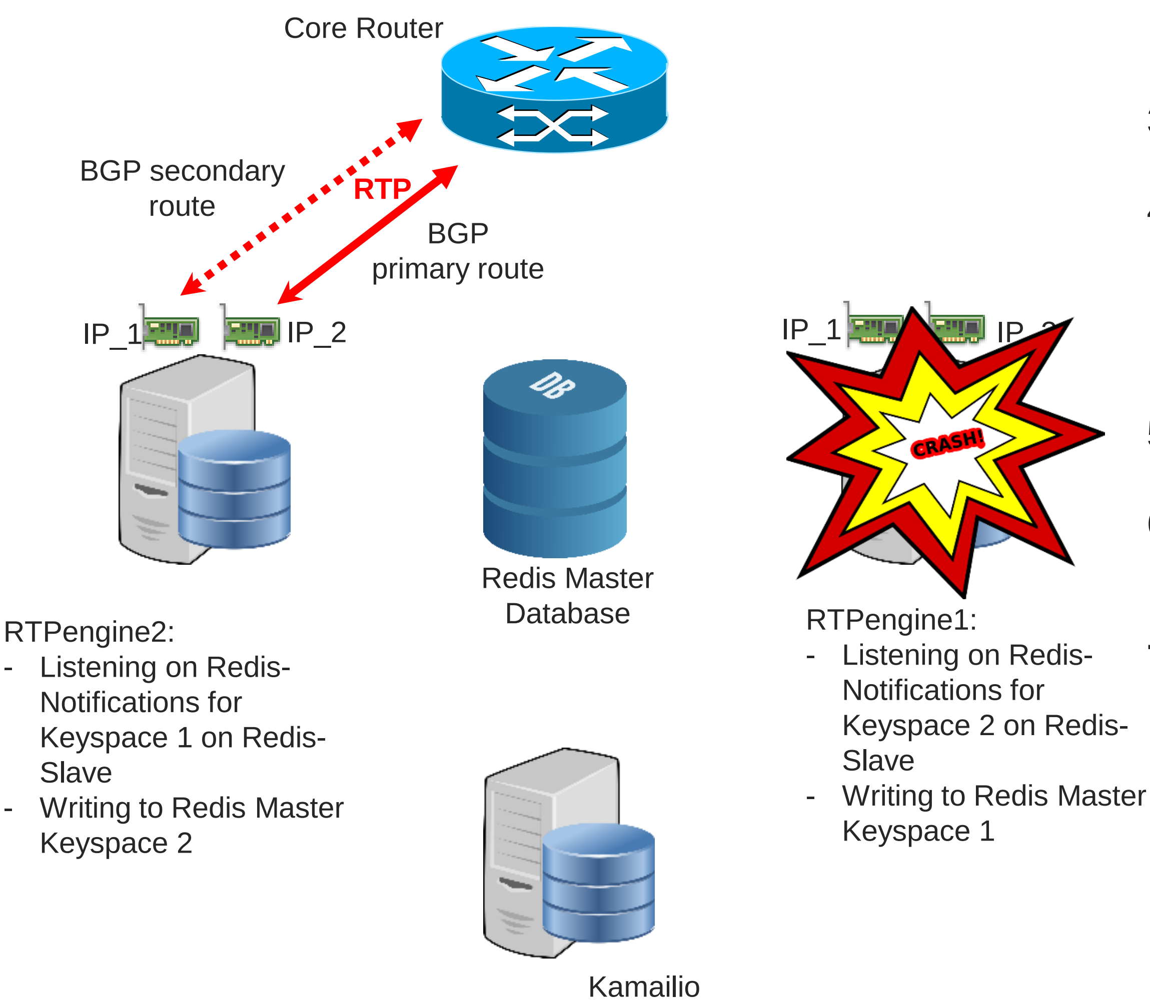


1. Kamailio creates RTP session on RTPengine1 with active IP_1
2. RTP traffic arrives. RTPengine1 persists session information to Redis Master, Keyspace 1
3. Redis Master replicats information to slaves
4. RTPengine2 (listening on Keyspace 1) handles Redis-Notification and creates session on IP_1

5. HW Error / Network Error / …

6. **Core Router: Withdraw BGP announcements for RTPengine1 (< 1s delay)**

RTPengine Redundancy – Simplified illustration



1. Kamailio creates RTP session on RTPengine1 with active IP_1
2. RTP traffic arrives. RTPengine1 persists session information to Redis Master, Keyspace 1
3. Redis Master replicats information to slaves
4. RTPengine2 (listening on Keyspace 1) handles Redis-Notification and creates session on IP_1

5. HW Error / Network Error / …

6. Core Router: Withdraw BGP announcements for RTPengine1 (< 1s delay)
7. **Traffic routed via alternative route**

Roundup

- Additional VoIP server in 2 new data centers
- Redundant active-active setup
- RTPengine software running on media relay servers
- Invoking RTPengine via Kamailio‘s rtpengine module
- Tuning IRQ handling
- Synchronizing RTPengine session information via Redis and keyspace notifications
- Using BGP mechanisms for redundancy

Interested?

➔ jobs.1und1.de