
CERN

DNS and DHCP service evolution plans

Quentin Barrand
CERN IT / Communication Systems

Outline

- DNS & DHCP at CERN
  - Our setup
  - Pain points
  - Evolution plans

- Wrap-up and final thoughts

DNS evolution plans

Dynamic zones, Go software, anycast

Our DNS setup

ISC BIND 9.8 to 9.12
`cern.ch`: ~500k records

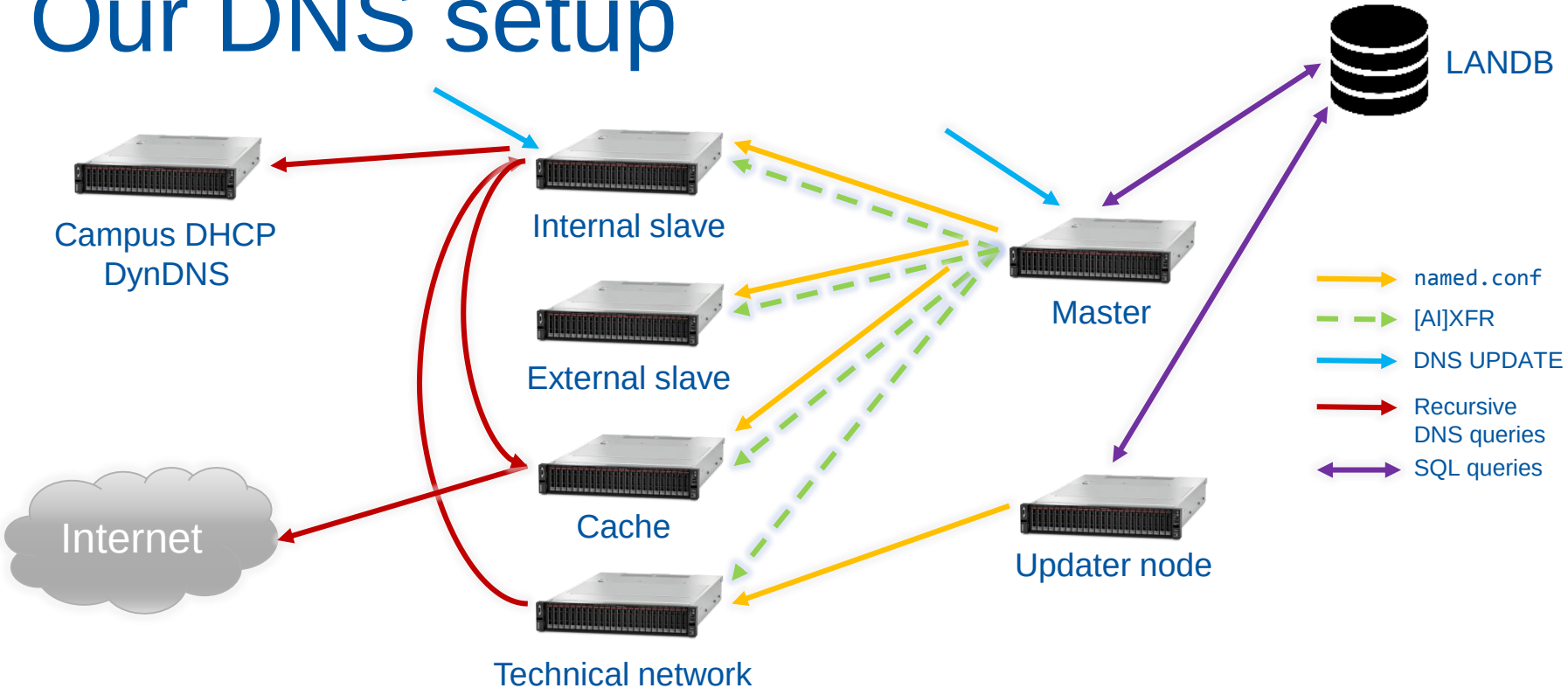
Master (hidden)
- Hidden: does not answer any query
- Updated every 10 minutes

Technical network
- Stability is the main concern
- IXFR for `cern.ch`
- Forwarding for dynamic zones
- Root servers to avoid uncontrolled recursion

Authoritative slaves
- Internal: clusters (corosync + pacemaker)
- External: no clustering
- IXFR for all zones

Cache
- Recursion (with DNSSEC)
- Response Policy Zones (RPZ)

Our DNS setup



Updating the configuration

- LANDB is our network database

- In-house Perl scripts
  - Run every 10 minutes
  - Generate `named.conf` and zone files using LANDB data
  - Check them with BIND tools (`named-check[conf|zone]`)
  - Deploy with `scp` and restart `named`
  - Version with CVS

- Extensive use of dynamic zones for subdomains
  - Maintained directly by service managers via RFC2136 messages (DNS Update)

Pain points

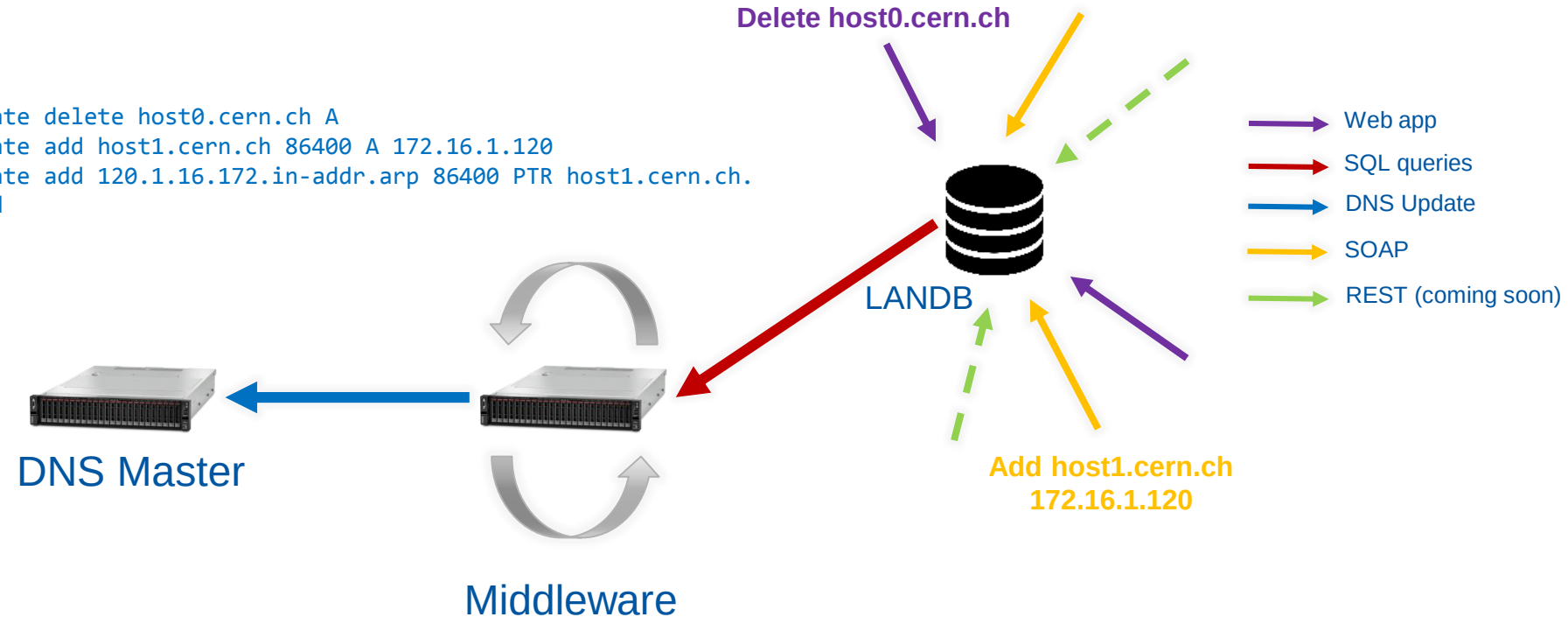
- **Master redundancy is hard**
  - What about dynamic zones?
  - BIND database backend: rarely deployed
- Our users want **faster updates**!
  - Cloud, device registration workflows and others would benefit
  - Generating the full configuration often is not scalable
  - CVS is slow and requires housekeeping
- DNS must be **rock solid**
  - How can we distribute the load and downtime risk?
- **Maintaining our software** is a concern
  - Perl is old-fashioned; developers are hard to find
  - CVS libraries are rare

The plan

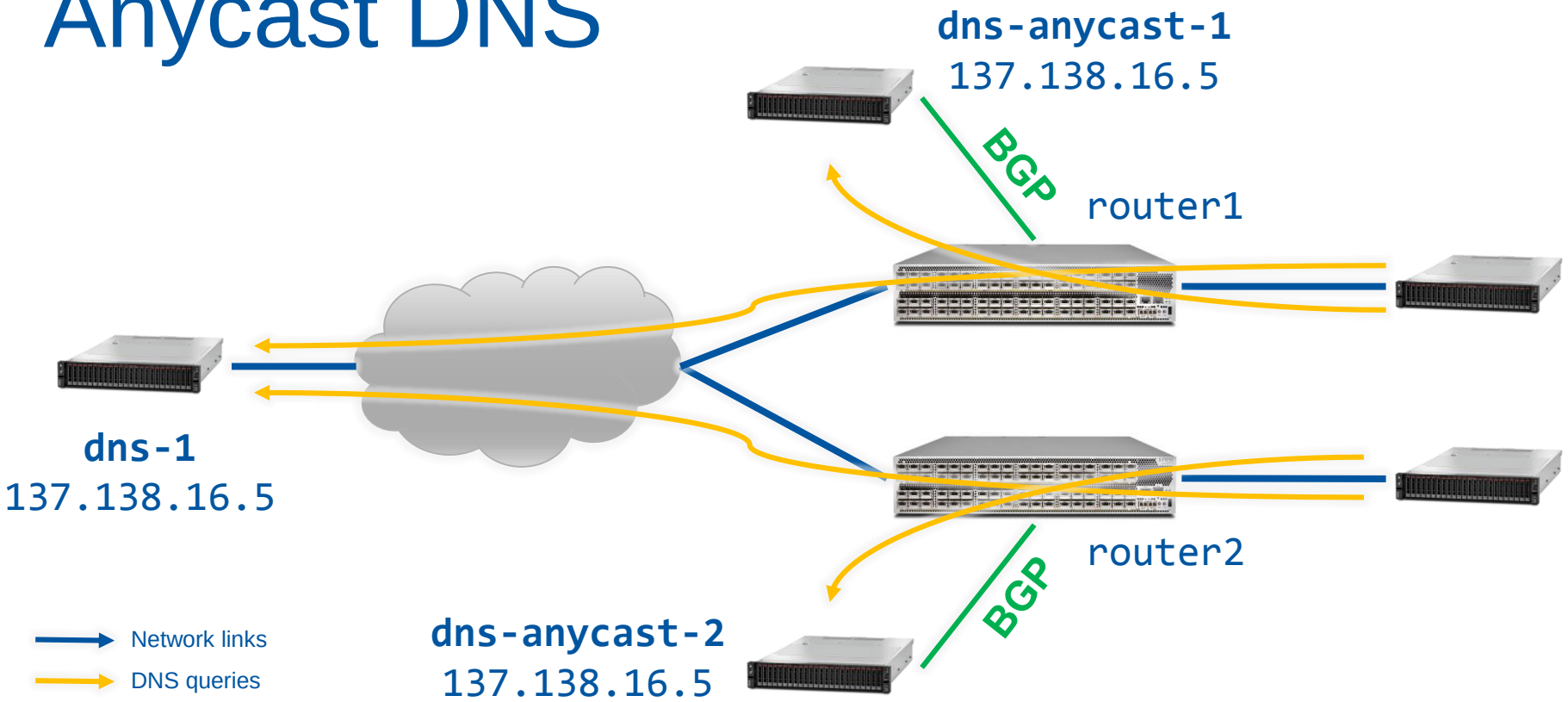
- **Short-term**: rewrite the software in Go, version with Git
  - We need this app anyway for recovery purposes
  - 5-10 times faster compared to Python
  - Safer, packageable and easy to learn
- **Dynamic `cern.ch`**
  - Convert LANDB updates into **DNS Update** messages
- Master redundancy
  - Some setups work with a set of `rsync` scripts
- Load distribution: **anycast DNS**
  - Spawn servers anywhere in the network
  - BGP peer with routers using BIRD
  - Write Puppet manifests for fast server provisioning

Dynamic cern.ch

```
update delete host0.cern.ch A
update add host1.cern.ch 86400 A 172.16.1.120
update add 120.1.16.172.in-addr.arp 86400 PTR host1.cern.ch.
send
```



Anycast DNS


dns-anycast-1
137.138.16.5
BGP
router1
dns-1
137.138.16.5
router2
BGP
dns-anycast-2
137.138.16.5
Network links
DNS queries

DHCP evolution plans

Kea, Go software, failover

Our DHCP setup

- ISC DHCP 4.3
- Configuration updated every 5 minutes by a Perl script
- OMAPI to get lease data
- Datacenter and technical network: static configuration
- Campus: split pools
  - Each server owns half of the addresses
  - IETF's DHCP Failover sounds quite complex

Pain points

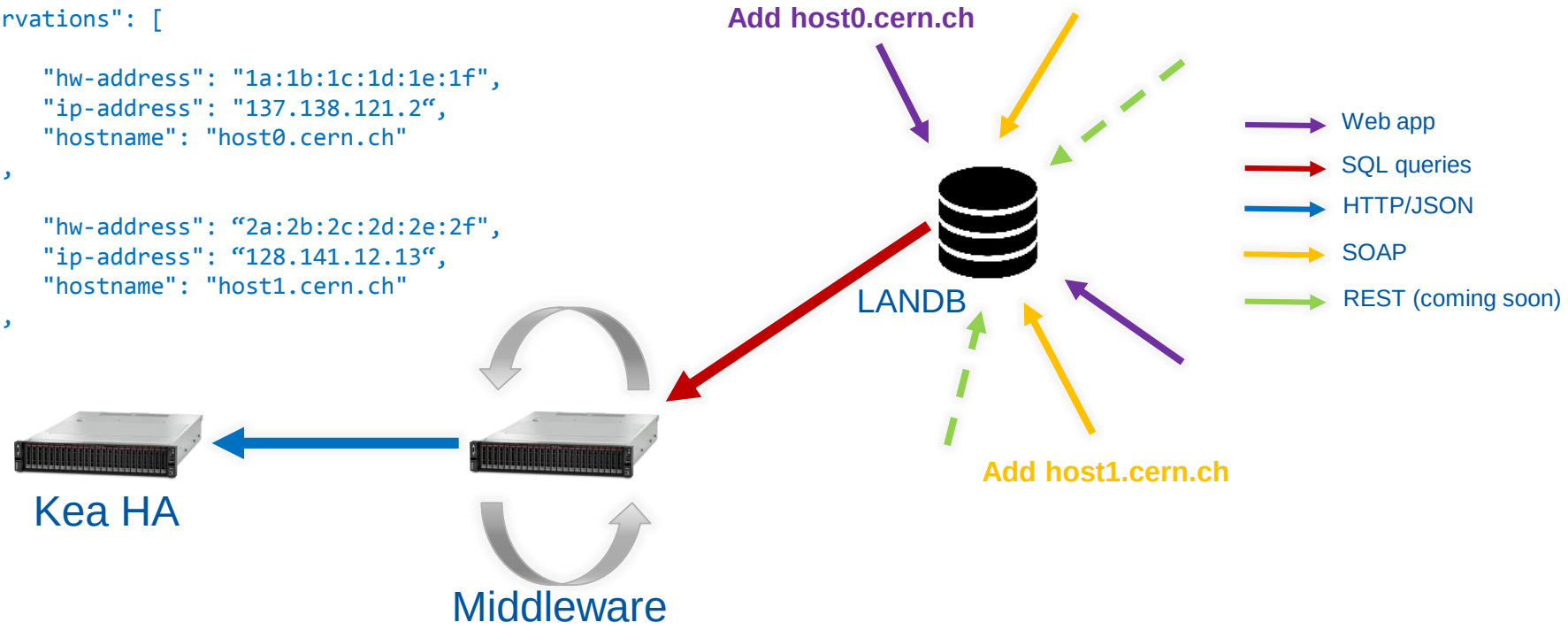
- **No real redundancy** on the campus
  - Losing one server halves the number of addresses available
  - Some pools are already more than 50% used
- Updates **every five minutes**
  - Similar workflow to that of DNS
  - Same concerns regarding Perl and CVS
- **OMAPI** is outdated
  - Not many client libraries available
  - Does not work with IPv6

ISC Kea

- Modern, extensible successor to `dhcpd`
- Dynamic JSON configuration
  - No restart of the daemon required!
- Several backends available:
  - Memfile, MySQL, PostgreSQL, Cassandra
  - Multiple replication strategies available
- REST API
  - Get / update the server configuration
  - Premium (paid) hooks bring lease and reservations management, and much more
- Simpler, non-IETF HA protocol
  - Uses MAC/DUID hashing to assign a server
  - Supports DHCPv6

Dynamic reservations with Kea

```
"reservations": [
    {
        "hw-address": "1a:1b:1c:1d:1e:1f",
        "ip-address": "137.138.121.2“,
        "hostname": "host0.cern.ch"
    },
    {
        "hw-address": “2a:2b:2c:2d:2e:2f",
        "ip-address": “128.141.12.13“,
        "hostname": "host1.cern.ch"
    },
]
```



Wrap-up

- **DNS**
  - Dynamic update of `cern.ch` via DNS Update
  - Load distribution by using anycast resolution
  - Master redundancy: research in progress
- **DHCP**
  - Kea: database back-end, working HA, REST API
  - Updated on-the-fly
- **Work in progress!**
  - Go is a good candidate for our software
- **We would like to hear from you!**
  - Are you doing dynamic updates on your organization's main domain?
  - How are you achieving DNS / DHCP redundancy?


CERN