Najnowsze mechanizmy w DHCP

BIND10 DHCP oraz prace w IETF

The Newest DHCP Mechanisms

BIND10 DHCP and IETF work

PLNOG10, Warsaw, Poland

1 March 2013

Tomek Mrugalski <tomasz(at)isc.org>

Agenda

1. About presenter and ISC

2. DHCP in BIND10 (codename Kea)
   - Reasons
   - Status
   - Roadmap

3. Performance

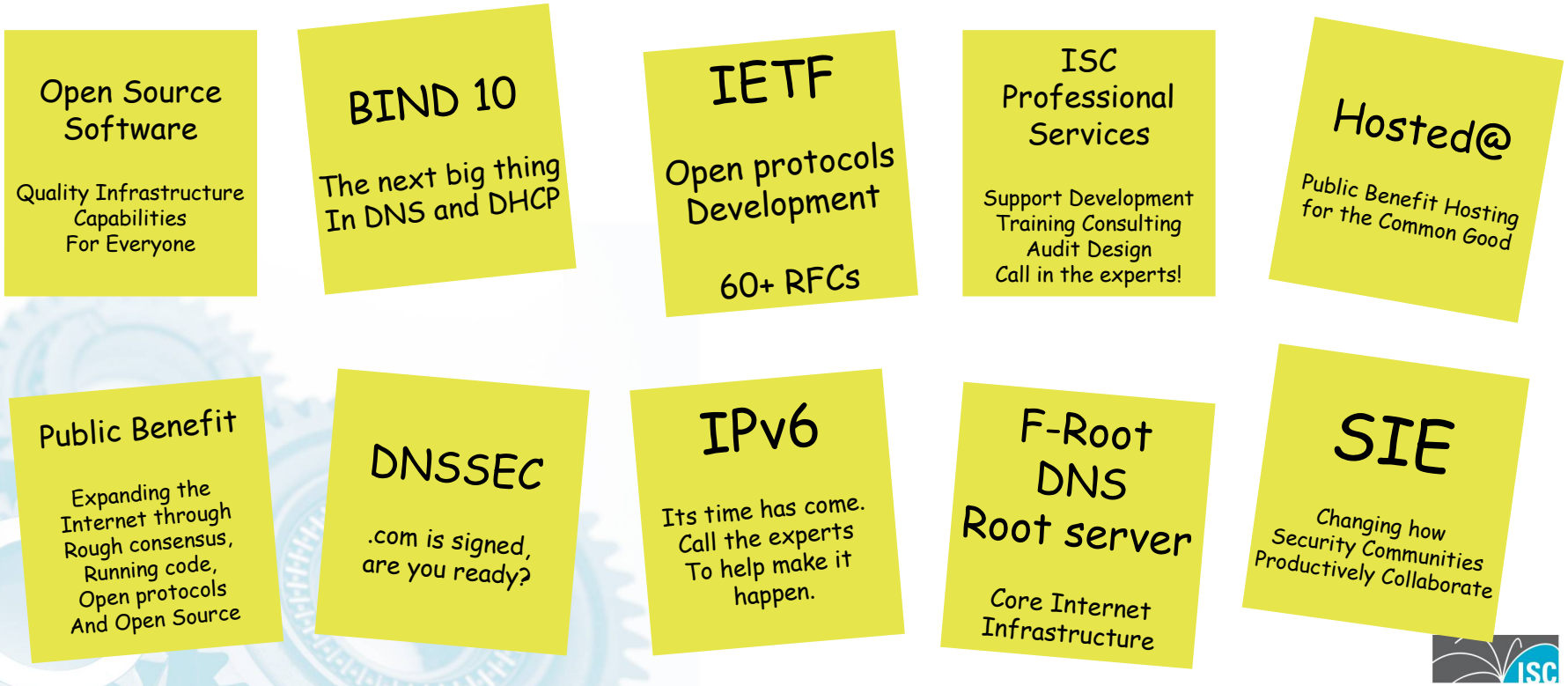
4. DHCP in IETF
   - DHCPv6 Failover

Who is Tomek?

- M.Sc., Ph.D from Gdansk University of Technology
- Primary author of Dibbler
  - Portable DHCPv6 implementation (srv, cli, relay)
  - Supports Win 2k-Win8, Linux, BSD, Solaris
  - Confirmed use in 34 countries
- 7 years at Intel (Network Quality Labs, chipsets group)
- 2 years at ISC
  - Lead Developer of BIND10 DHCP (Kea)
  - Occasional contributor to ISC-DHCP
- Active IETF participant since 2009
  - 2 RFCs, 15+ drafts

Who is ISC?

Internet Systems Consortium, Inc. (ISC) is a non-profit 501(c)(3) public benefit corporation dedicated to supporting the infrastructure of the universal connected self-organizing Internet - and the autonomy of its participants - by developing and maintaining core production quality software, protocols, and operations.



BIND10 DHCP

Why DHCP rewrite?

- Existing code is 17 years old
- Hardware changed (many cores)
- Networks changed
- DHCP landscape changed
- New software development techniques
- Lacking performance
- Monolithic
- Documentation is lacking
- Complex code, difficult to extend

BIND10 DHCP
Codename Kea

- Common infrastructure with BIND10 DNS
  - On-line configuration
  - Logging
  - Statistics
- Performance is essential
- IPv6 is a first class citizen, not add-on
- C++ as a language of choice
- Multi-core support
- Switchable backends (mem+file, SQLite, MySQL, …)
- Hooks
- Modular
- Resiliency (fault isolation and recovery)

Kea: Current status (1)

DHCPv4 server
(b10-dhcp4)

- Supports DORA
- Relayed traffic only

DHCPv6 server
(b10-dhcp6)

- Supports SARR
- Direct traffic only

- Address assignment, renewal, release, expiry
- On-line configuration (common for all BIND10)
- Switchable backends: MySQL, memfile
- Custom option definitions
  - Standard options
  - Custom formats
  - Nested options
  - Option namespaces

Kea: Current status (2)

DHCPv4 server
(b10-dhcp4)

DHCPv6 server
(b10-dhcp6)

Perfdhcp
Performance Tool
(stand alone)

libdhcp++
- general purpose DHCP library
- v4/v6 packet parsing/assembly
- v4/v6 options parsing/assembly
- interface detection (Linux, other OSes planned)
- socket management

Kea: Work to Date (2)

- Documentation
  - BIND10 Guide
  - BIND10 Developer's Guide
  - Man pages
- Designs
  - Hooks
  - Lease/database design
  - Option Definition Design

http://bind10.isc.org/docs/bind10-guide.html

http://bind10.isc.org/wiki/Kea

DHCP Performance

Perfdhcp and BIND10 DHCP?

Why did we implement perfdhcp first?

Performance is essential
in BIND10 DHCP

DHCP Performance
Problem Space

Vendors often provide performance results, why measure it again?

- Marketing data is always trustworthy, right?
- Your HW may differ from reference HW(CPU, disk, fs, OS,…)
- Your traffic model may differ

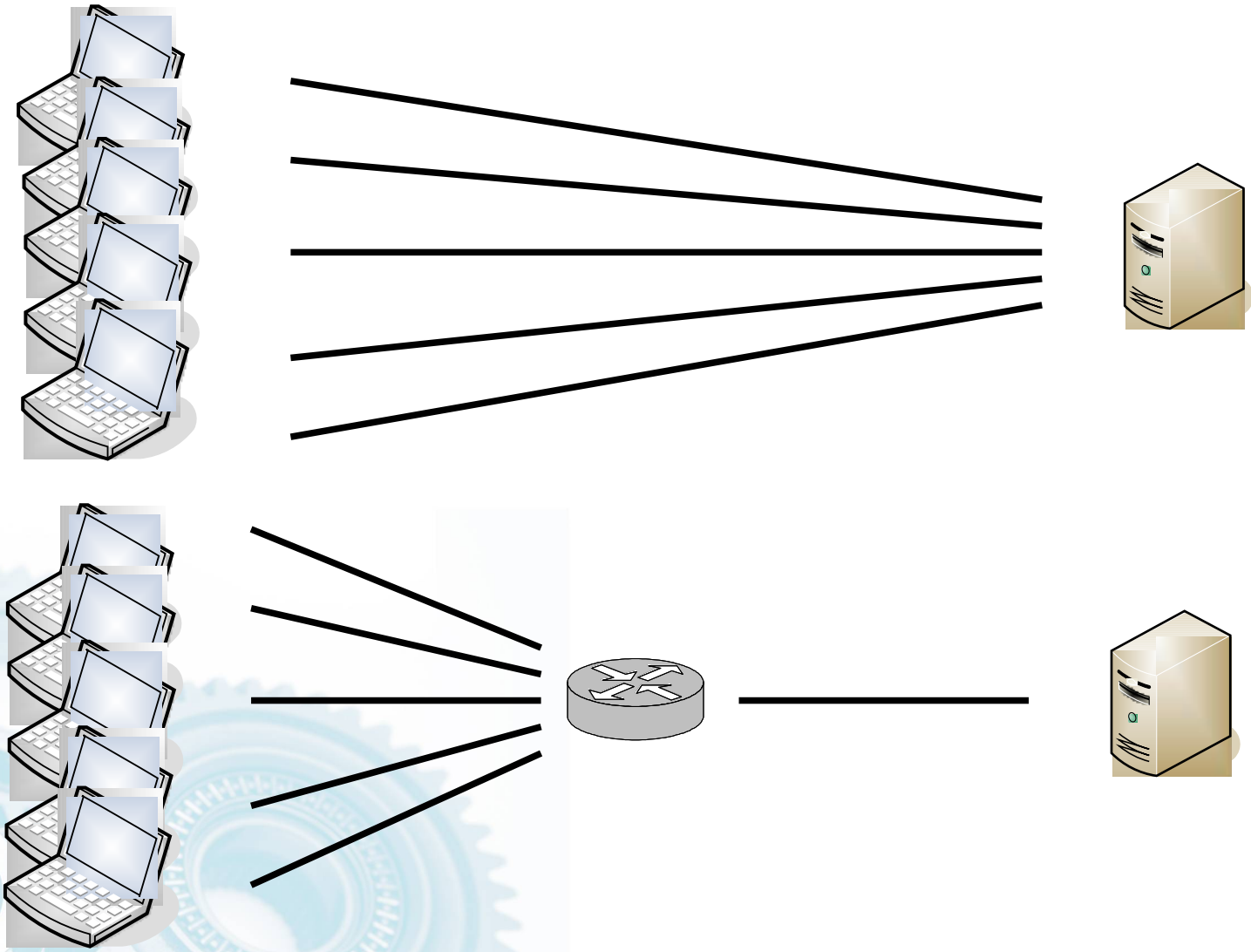
Conclusion:

The most reliable measurements

are ***your own***.

DHCP Performance Measurements (1)

DHCP Performance Measurements (2)



Perfdhcp :: Overview

- No feasible alternatives
  - are outdated (e.g. v4 only)
  - commercial (dedicated test HW is $$$$)
- Need a tool that is:
  - Flexible (lots of options and knobs)
  - Portable (Linux, BSD, perhaps Solaris)
  - Test any conformant implementation
  - Free (open source)
- Started project on our own

Perfdhcp :: Status

- Open source (ISC), currently Linux, but BSD and Solaris planned
- DHCPv4 & DHCPv6 (2-way & 4-way exchanges)
- Support for packet template files (optional)
- Server/interface selection (multicast/unicast)
- Parameterized traffic/test
  - # of clients,
  - # of transactions/sec,
  - best effort test,
  - test duration,
  - number of requests,
  - max number/% of drops …
- Diagnostics selector
- Measurements

```
Rate: 986.6 exchanges/second, expected rate:

***Statistics for: SOLICIT-ADVERTISE***
sent packets: 9866
received packets: 9866
drops: 0
orphans: 0

min delay: 0.168 ms
avg delay: 0.263 ms
max delay: 0.655 ms
std deviation: 0.039 ms
```

```
perfdhcp
 [-hv] [-4|-6] [-r<rate>] [-t<report>] [-R<range>] [-b<base>]
 [-n<num-request>] [-p<test-period>] [-d<drop-time>] [-D<max-drop>]
 [-l<local-addr|interface>] [-P<preload>] [-a<aggressivity>]
 [-L<local-port>] [-s<seed>] [-i] [-B] [-c] [-1]
 [-T<template-file>] [-X<xid-offset>] [-O<random-offset]
 [-E<time-offset>] [-S<srvid-offset>] [-I<ip-offset>]
 [-x<diagnostic-selector>] [-w<wrapped>] [server]
```

Perfdhcp :: Roadmap

2013: No specific plans (unfunded)

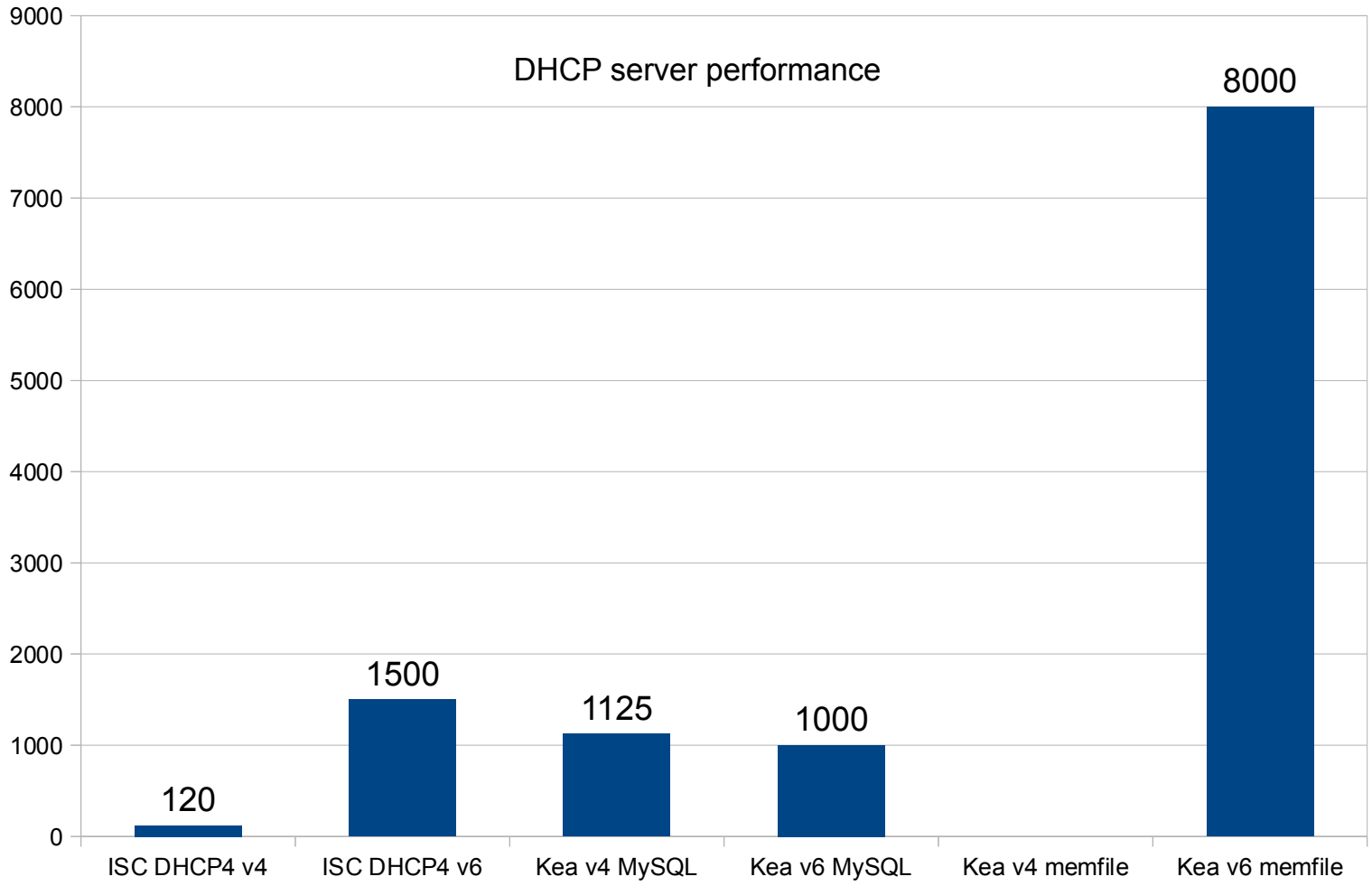
- Implement support for Prefix Delegation
- Relays
  - Traffic via relays
  - Relay options (subscriber-id, remote-id,...)
  - DOCSIS3.0 options
- Expand customization
- Improve response validation

Long term: maintain and develop

Kea :: Performance Results(1)

- Server run on a beefy server
  - HP ProLiant DL360 G7
  - Intel(R) Xeon(R) CPU E5649 (24 logical CPUs)
  - 72GB ram
  - HP Smart Array P410i + 10k rpm disks
- Client traffic generated by perfdhcp
- Performance may go...
  - ...up (optimizations, multi-core)
  - ...down (new features)

Kea :: Performance Results (2)



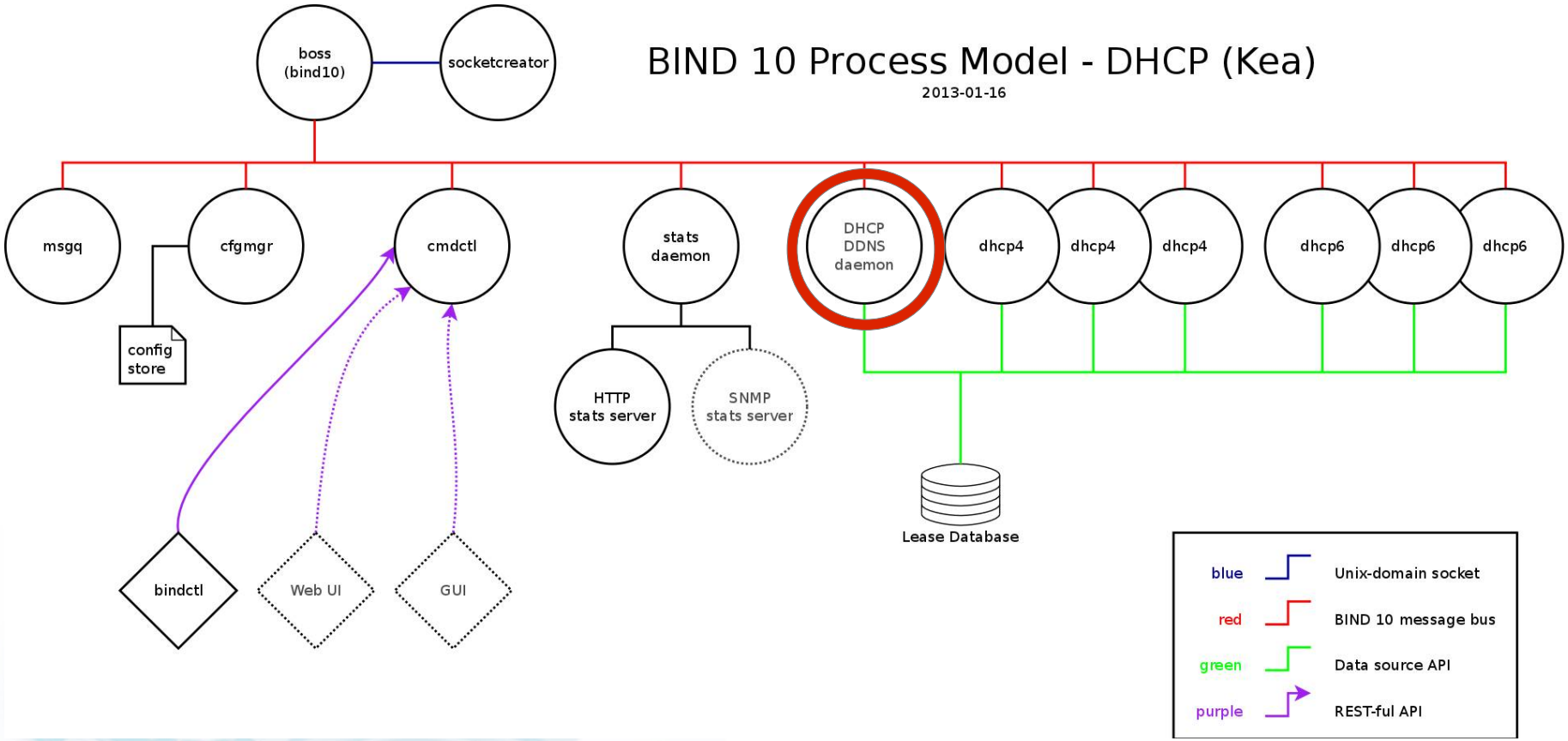
* initial data. Your mileage may vary.

Kea Plans

2013 Kea Roadmap

- Support for directly-connected IPv4 clients
- Support for IPv6 relays
- DDNS daemon
- Hooks
- Extend OS support to BSD, Solaris (?)

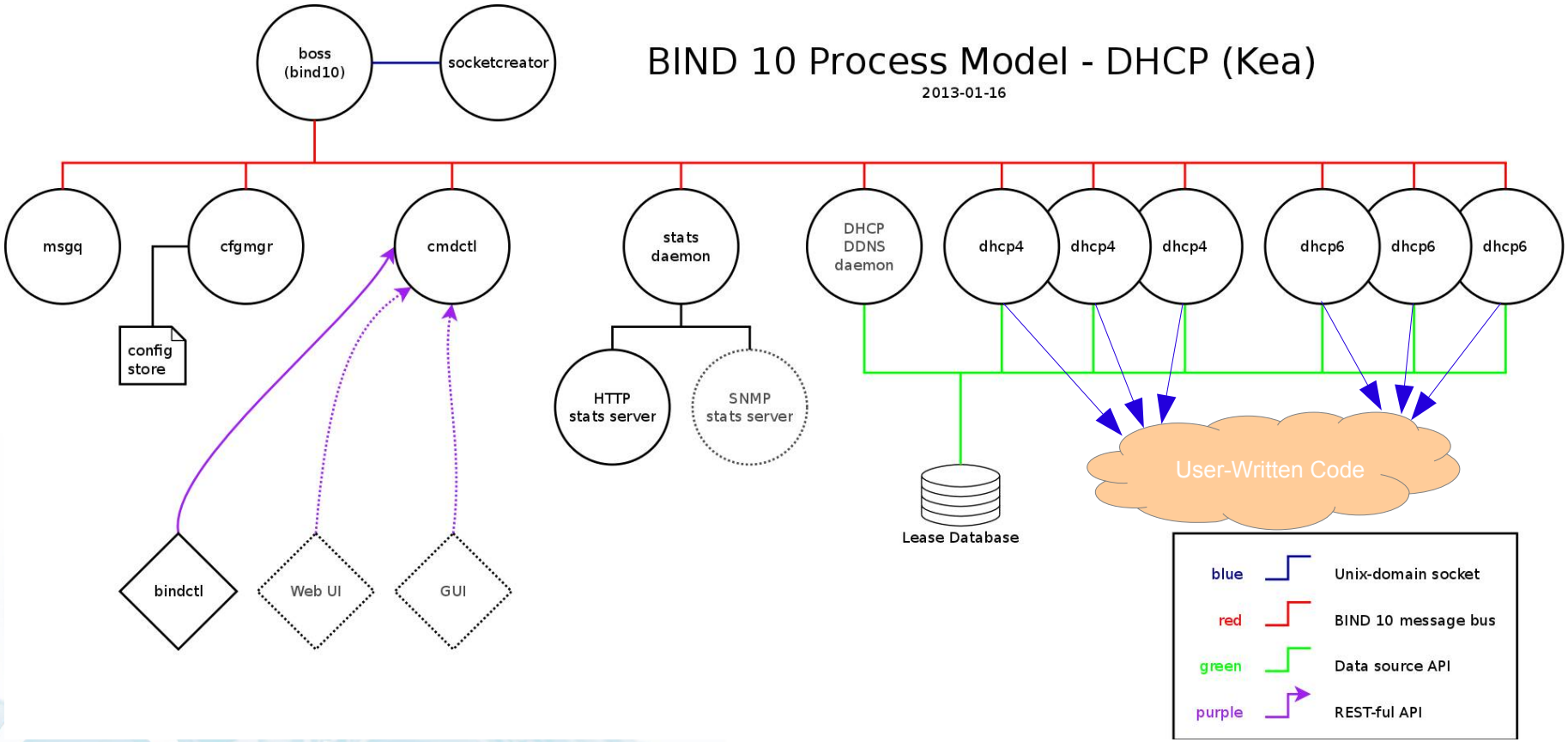
DHCP DDNS Daemon (1)



DHCP DDNS Daemon (2)

- Will handle addition/removal of name/address translations from forward and reverse DNS zones
- Separate process in keeping with BIND 10 philosophy
- Approach to be decided during design stage – some prototyping needed.

DHCP Hooks (1)


BIND 10 Process Model - DHCP (Kea)
2013-01-16
boss (bind10)
socketcreator
msgq
cfgmgr
config store
cmdctl
stats daemon
HTTP stats server
SNMP stats server
DHCP DDNS daemon
dhcp4
dhcp4
dhcp4
dhcp6
dhcp6
dhcp6
Lease Database
User-Written Code
bindctl
Web UI
GUI
blue
Unix-domain socket
red
BIND 10 message bus
green
Data source API
purple
REST-ful API

DHCP Hooks (2)

- Set of hooks to be included in the code:
  - Call out to user code at defined points in packet processing
  - Replaces “conditional” configuration processing in DHCP4
- API designed
- Comments are more than welcome

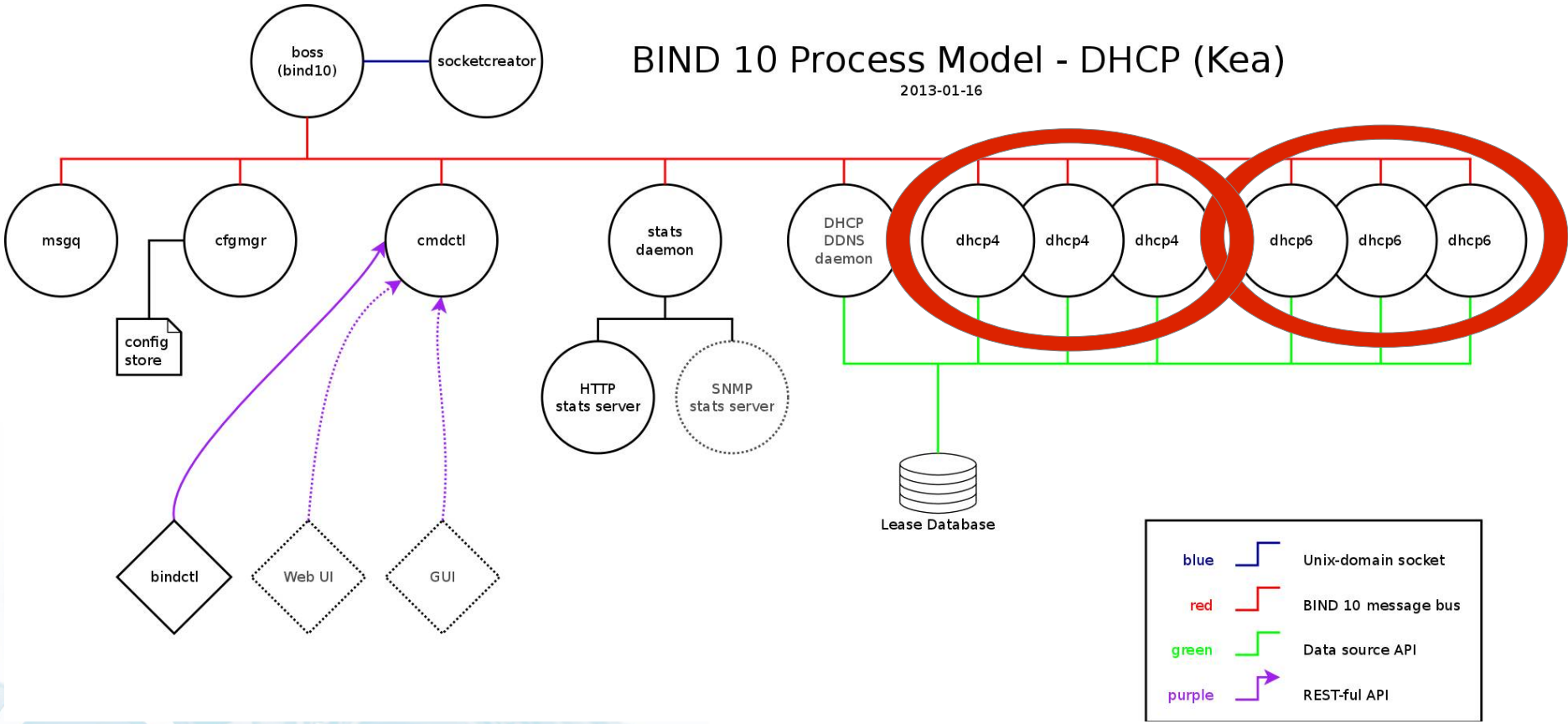
http://bind10.isc.org/wiki/DhcpHooks

Possible features

Unfunded ideas

- Multi-core support
- Prefix delegation in DHCPv6
- DHCPv6 failover
- DHCPv4 failover
- Different backends (Postgres? Cassandra?)
- CPE market
- …

Multi-Core Capability (1)



Multi-Core Capability (2)

- Aim to prototype different solutions before choosing one
- E.g. possible solution for scalability:
  - Divide queries between multiple processes
  - Receptionist process to route packets from a given client to the same daemon process to cope with state issues.

Interested?

Fully open source model

- Available for free (no strings attached)
- GIT repo, trac tickets available
- Test, report bugs
- Submit patches

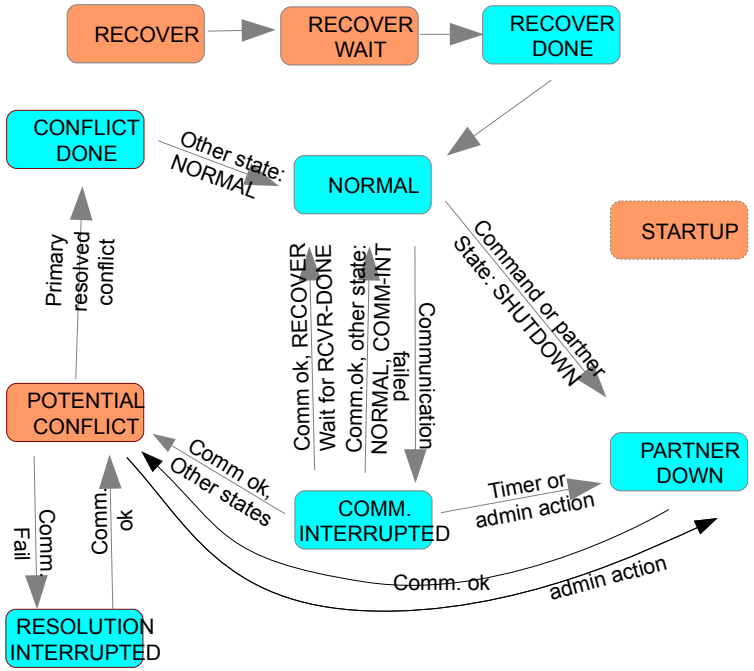
Contribute

- We are looking for sponsors (money and developers)
- Development contracts
- Review design documents (e.g. requirements)

DHCP in IETF

DHCPv6 Failover :: Overview

- Based on v4 failover draft, but simplified
- Hot standby (Active-passive only)
- No load balancing in design spec (likely extension, some provisioning ready, trying to have common state machine for base and LB)
- Major topics:
  - MCLT concept, Lazy Updates
  - state machines
  - Binding updates
    + conflict resolution
  - Connection management
  - 2 Allocation Algorithms
    (Proportional and Independent)
  - DDNS considerations
  - Lease reservation



DHCPv6 Failover Grand Plan

- **Step 0:** Redundancy considerations
  - Published as RFC6853 (Feb. 2013)
- **Step 1:** Requirements document (info)
  - WGLC done, to be published soon
  - Comments welcome
- **Step 2:** Design document (std)
  - WG item, published -02
  - Text complete (no major missing parts)
  - Comments welcome
- **Step 3:** Protocol document (std)
  - TBD

- Possible extension drafts

IPv4 provisioning in IPv6-only network

- **MAP** (Mapping Address and Port, DS-Lite successor)
  - Fully stateless (does not require per-session or per-subscriber state)
  - Draft-ietf-softwire-map-dhcp
- **LW4o6**
  - draft-cui-softwire-b4-translated-ds-lite
- **DHCPv4-over-IPv6**
  - draft-ietf-dhc-dhcpv4-over-ipv6

Attempts to unify/clarify:

- draft-rajtar-dhc-v4configuration,
- draft-bfmk-softwire-unified-cpe

DHCPv6 in IETF

- **DHCPv6 Stateful Issues**
  - draft-ietf-dhc-dhcpv6-stateful-issues
  - RFC3315bis planned
- **MAC vs DUID issue** (dual-stack clients parity)
  - draft-ietf-dhc-dhcpv6-client-link-layer-addr-opt
- **DHCPv6 Radius Option**
  - Draft-ietf-dhc-dhcpv6-radius-opt
- **DHCPv6 Load Balancing**
  - draft-ietf-dhc-dhcpv6-load-balancing
- **Multiple Provisioning domains**
  - Whole Homenet WG
- **Routing configuation over DHCPv6**
  - draft-ietf-mif-dhcpv6-route-option
  - dying slowly...

Questions?

Thank you

isc.org

backup

Failover Design :: Communication

1. Communication over TCP
2. Reusing bulk leasequery framing, but with new FO-specific message types
3. TLS usage (optional)
4. Connection management (CONNECT, CONNECTACK, DISCONNECT)
5. State notifications
6. Lease updates (BNDUPD, BNDUPDALL, BNDACK, UPDDONE)
7. Pool requests (POOLREQ, POOLRESP)
8. Keep alive (CONTACT)

Failover Design :: Resource Allocation

1. Proportional allocation (“IPv4 failover-style”)
    1. Useful for limited resources (e.g. prefixes)
    2. Pool may need to be rebalanced.
    3. Only unleased resources are owned by specific server.
    4. Released/expired resources return to primary

2. Independent allocation (“simple split”)
    1. Useful for vast resources (e.g. /64 address pool)
    2. All resources are owned by specific server.
    3. Pools are never rebalanced.
    4. Released/expired resources return to its owner.
    5. Simpler, but MCLT restrictions still apply.

Failover Design :: MCLT concept & Lazy update

1. Lazy Update:
    1. Server assigns a lease and responds to a client
    2. Server updates its partner at a later time (lockstep would introduce too much delay)

    Problem: failure between 1. and 2.
2. Maximum Client Lead Time
    - The maximum difference between lease time known by a client and acknowledged by its partner.
3. Useful in communications-interrupted
    - Server does not know if its partner extended any lease;
    - It knows that its parter could extend by at most MCLT;
    - To be on the safe side, server assumes that ALL leases were extended by MCLT.

Failover Endpoint (partner) State Machine