facebook

DHCP Infra @ Facebook

Evolution of the infrastructure and lessons learned

SRECon15 Europe - Dublin - 15th May 2015

Angelo “pallotron” Failla <pallotron@fb.com> - Cluster Infrastructure Dublin

facebook
INFRASTRUCTURE

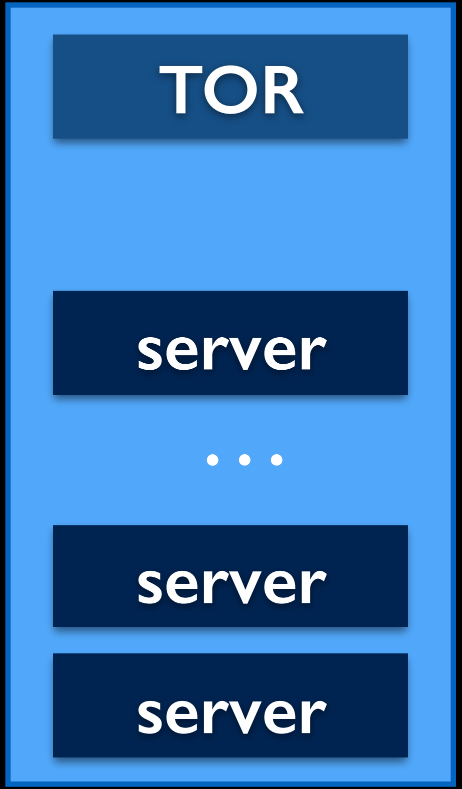
Production Engineering

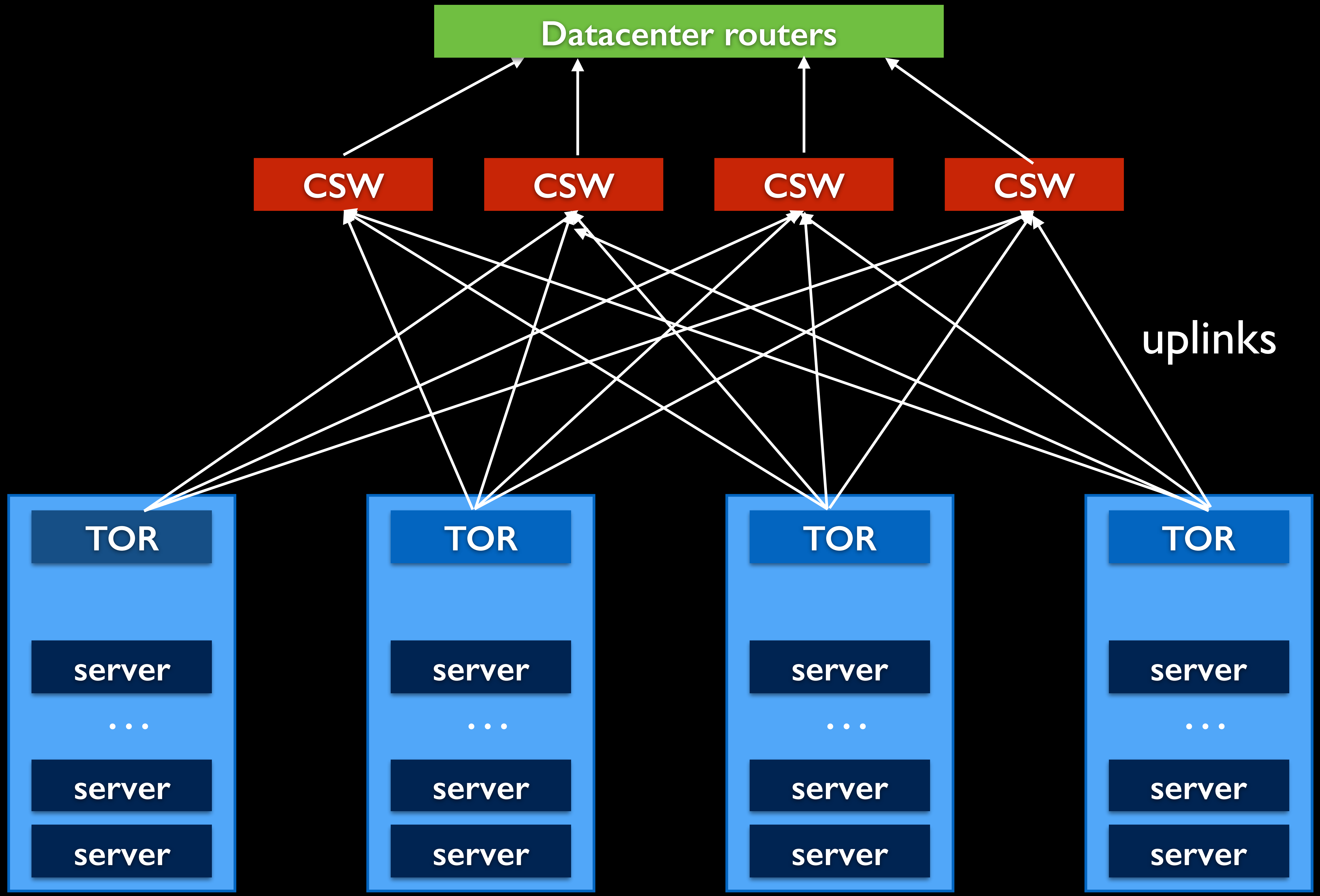
Cluster Infrastructure

Agenda

- Cluster overview
- DCHP: how and why it’s used
- Old architecture and its limits
- How we solved those limits
- Lesson learned and other takeaways

"Wedge" switch running FBOSS

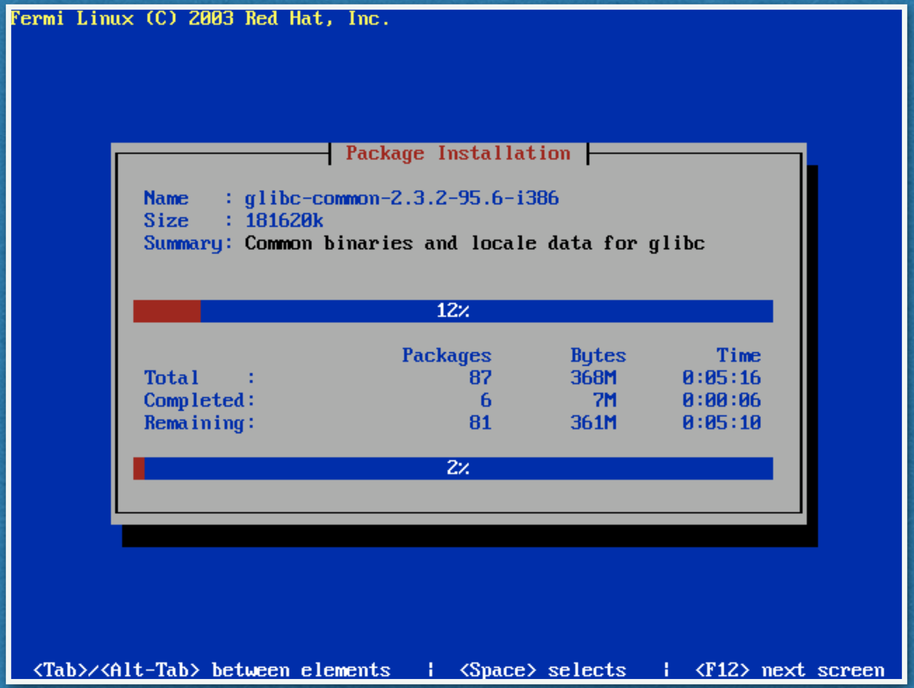

TOR
server
. . .
server
server


Datacenter routers
CSW
CSW
CSW
CSW
uplinks
TOR
server
. . .
server
server
TOR
server
. . .
server
server
TOR
server
. . .
server
server
TOR
server
. . .
server
server

DHCP: how and why?

For bare metal provisioning:

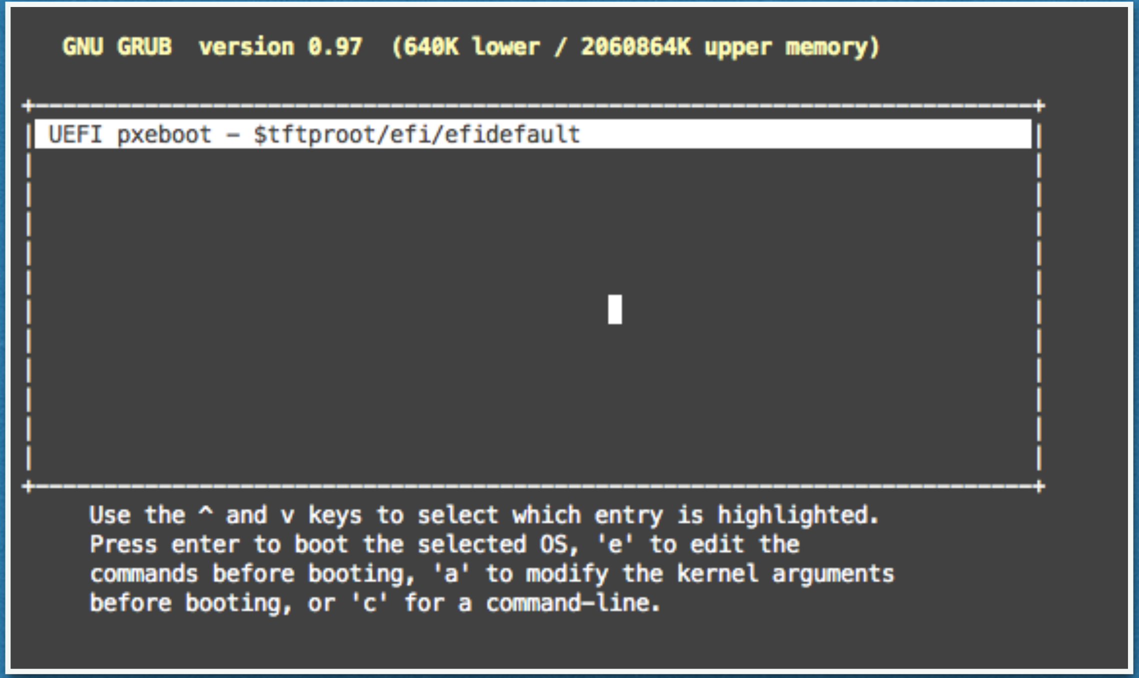
- At reboot
- Used to install OS on hosts
- Anaconda based
- iPXE



```
Fermi Linux (C) 2003 Red Hat, Inc.

                  ┤ Package Installation ├

   Name    : glibc-common-2.3.2-95.6-i386
   Size    : 181620k
   Summary: Common binaries and locale data for glibc

                             12%

                         Packages       Bytes        Time
   Total     :                 87        368M     0:05:16
   Completed:                   6          7M     0:00:06
   Remaining:                  81        361M     0:05:10

                              2%

 <Tab>/<Alt-Tab> between elements   |  <Space> selects   |  <F12> next screen
```

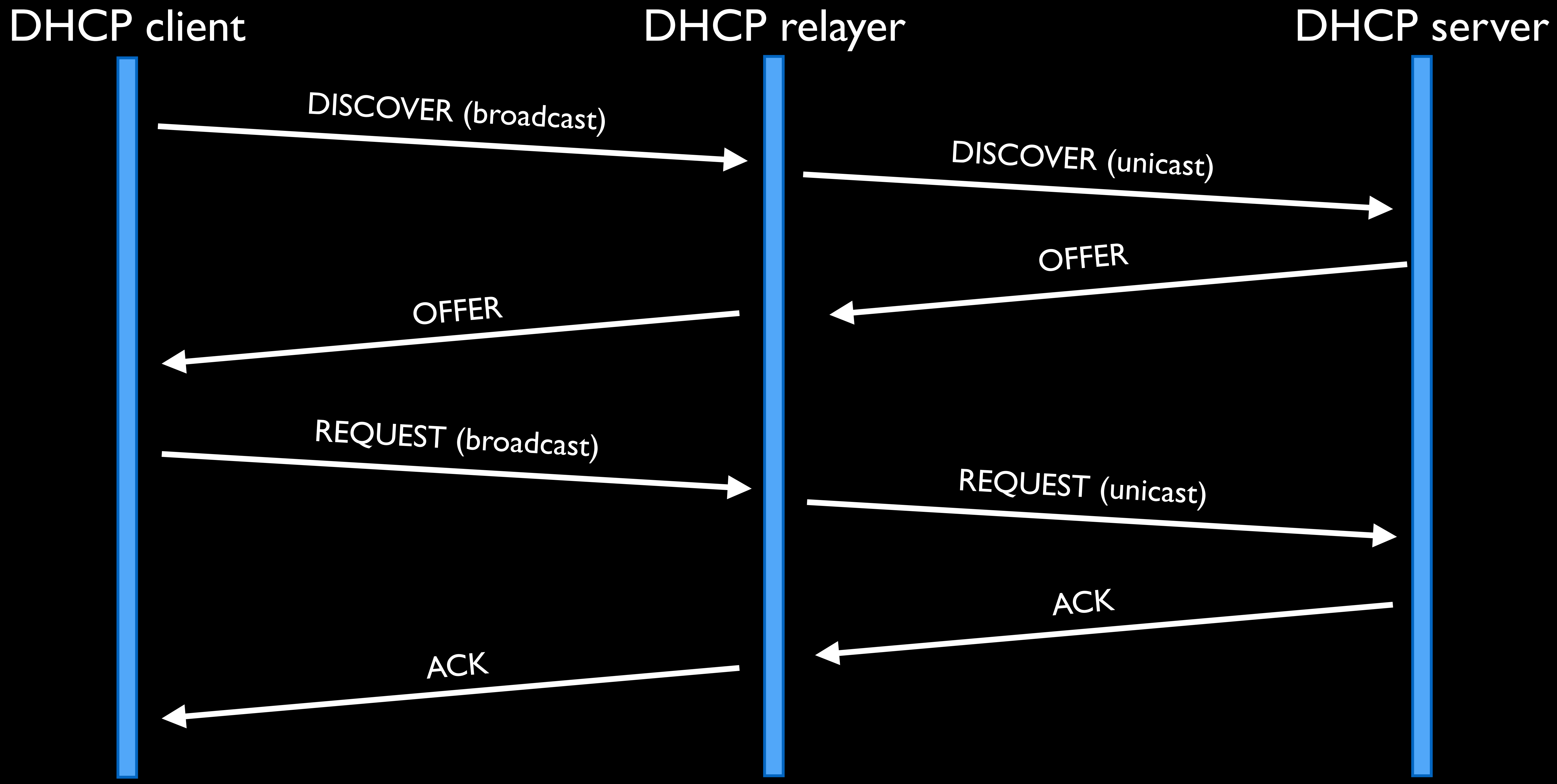
For Out Of Band management:

- To assign IPs to OOB interfaces
- Leases renewed typically once a day



```
    GNU GRUB  version 0.97  (640K lower / 2060864K upper memory)

 +-------------------------------------------------------------------+
 | UEFI pxeboot - $tftproot/efi/efidefault                           |
 |                                                                   |
 |                                                                   |
 +-------------------------------------------------------------------+

      Use the ^ and v keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the
      commands before booting, 'a' to modify the kernel arguments
      before booting, or 'c' for a command-line.
```

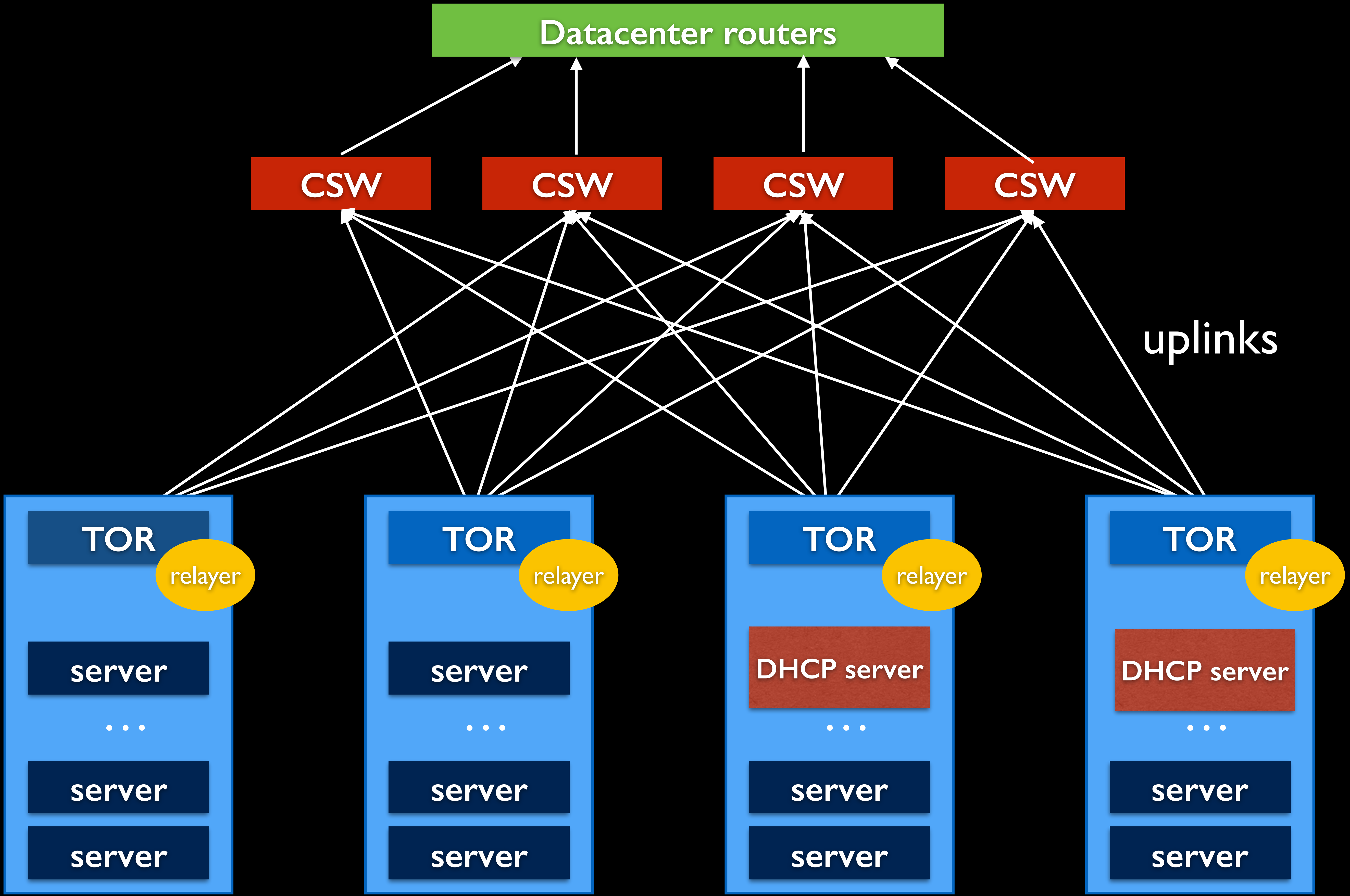
Anatomy of a DHCP4 handshake



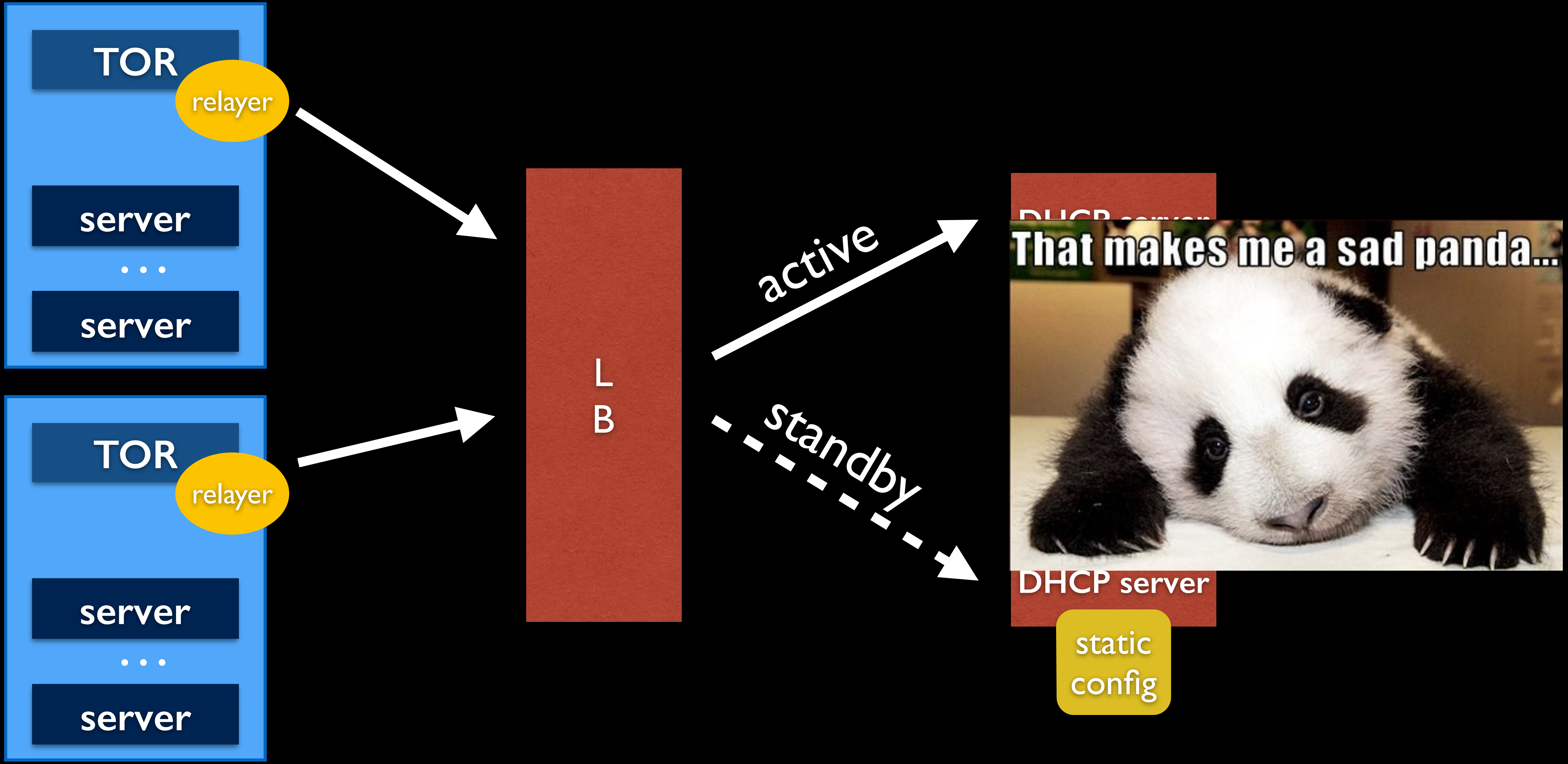
What about DHCPv6 (RFC3315)?

It's similar but with few differences:

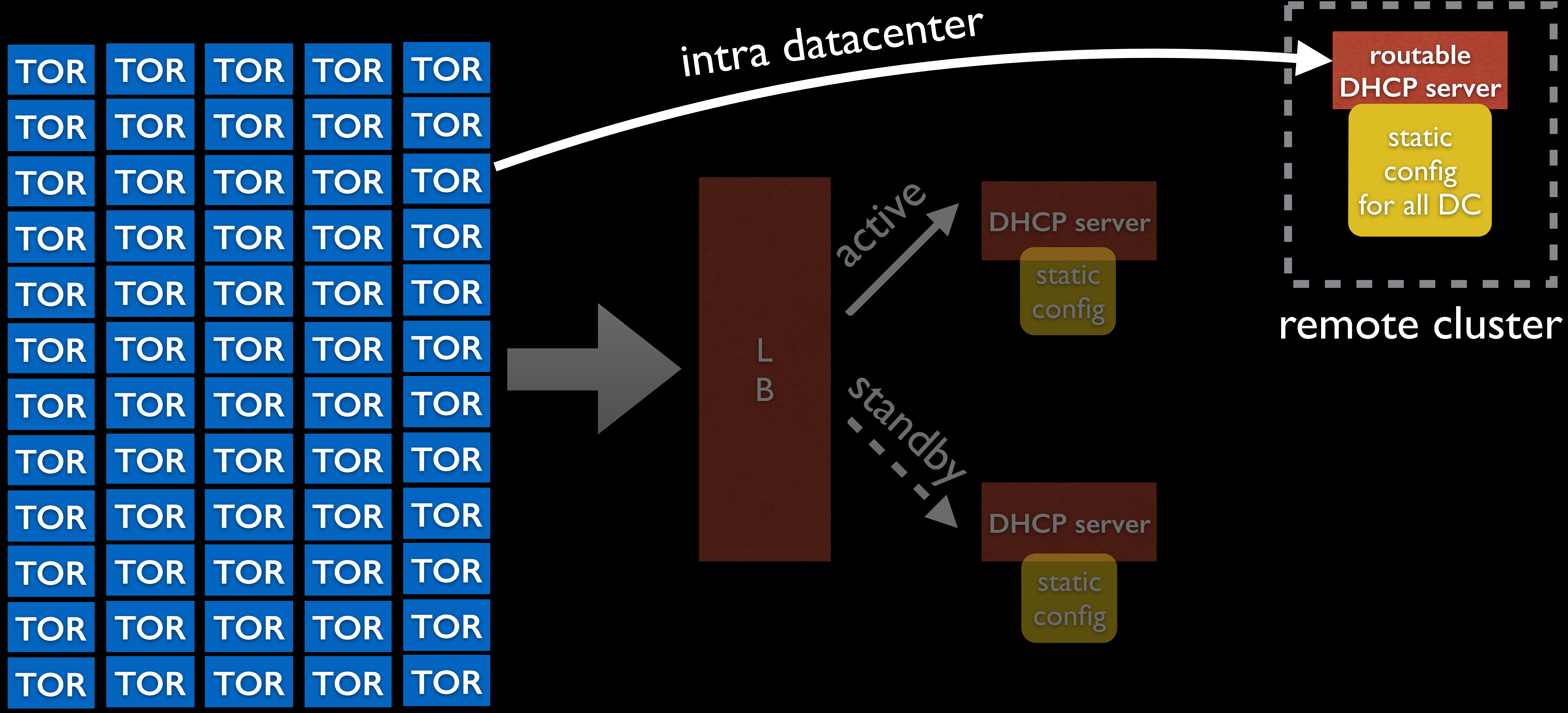
- Different option names and formats
- Doesn't deliver routing info (done by IPv6 via RA packets)
- 255.255.255.255 -> ff02::1:2 (special multicast IP) -> needs Relayer
- DUID ("***D***_hcp_ ***U***_nique_ ***ID***_entifier_") replaces MAC
  - we use DUID-LL[T]


Datacenter routers
CSW
CSW
CSW
CSW
uplinks
TOR
relayer
server
• • •
server
server
TOR
relayer
server
• • •
server
server
TOR
relayer
DHCP server
• • •
server
server
TOR
relayer
DHCP server
• • •
server
server

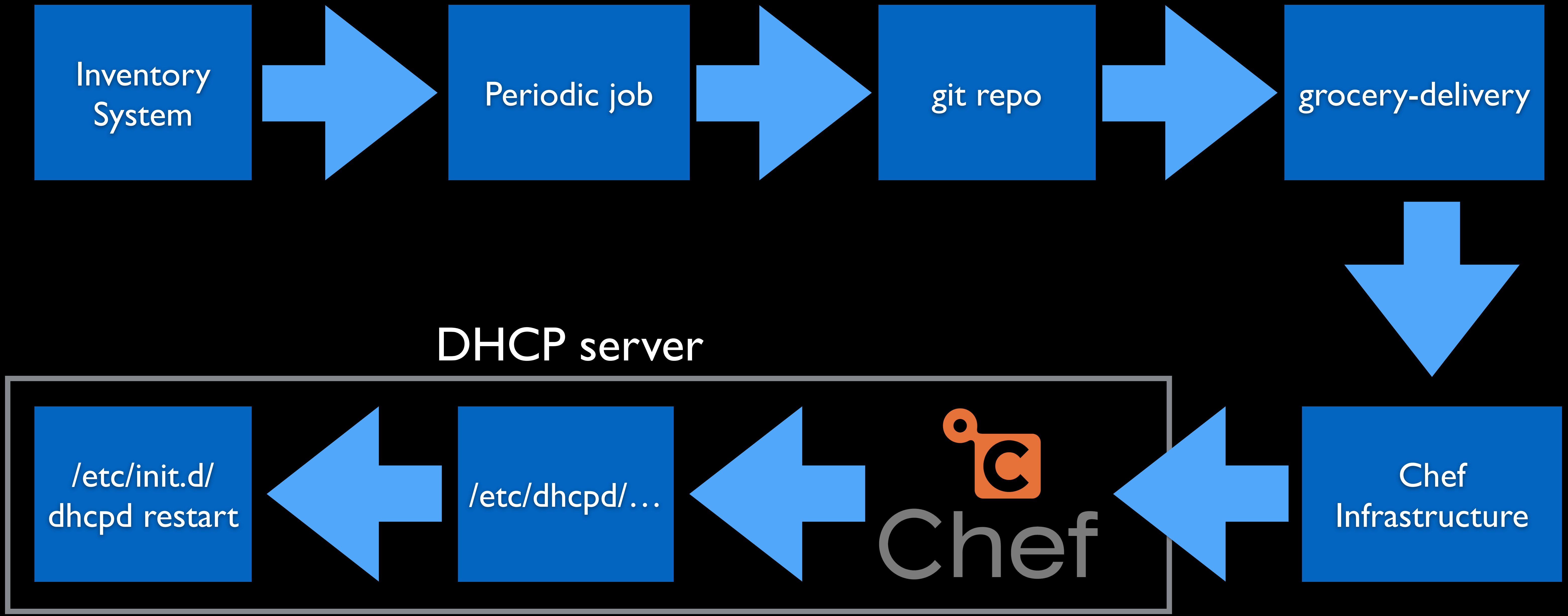
Problem: failure domain of the old architecture



Problem: bootstrapping a cluster

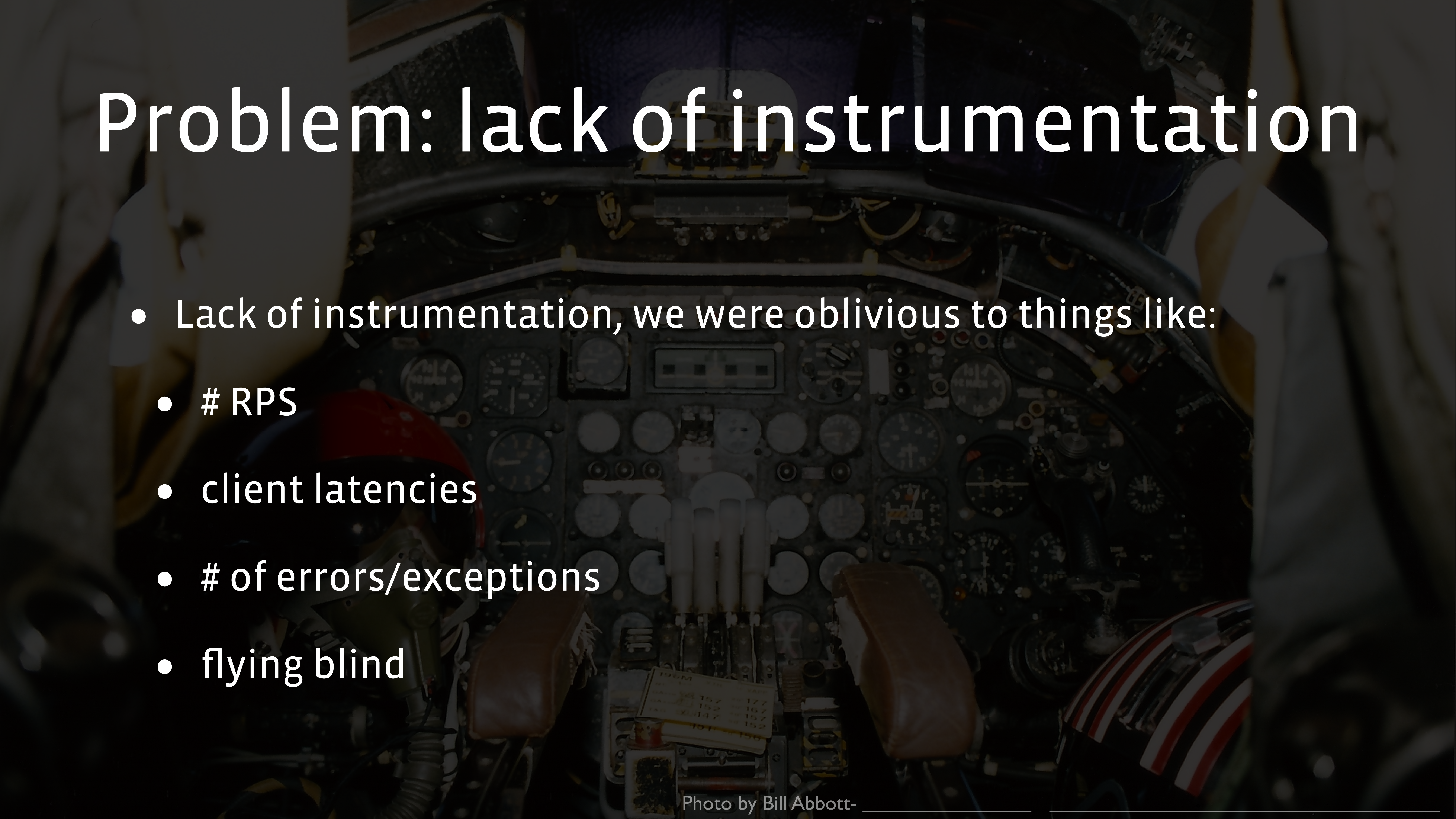


Problem: configuration distribution



facebook
facebook

THE SLOW SERVICE
IS UNBEARABLE
makeameme.org



Problem: lack of instrumentation

- Lack of instrumentation, we were oblivious to things like:
  - # RPS
  - client latencies
  - # of errors/exceptions
  - flying blind

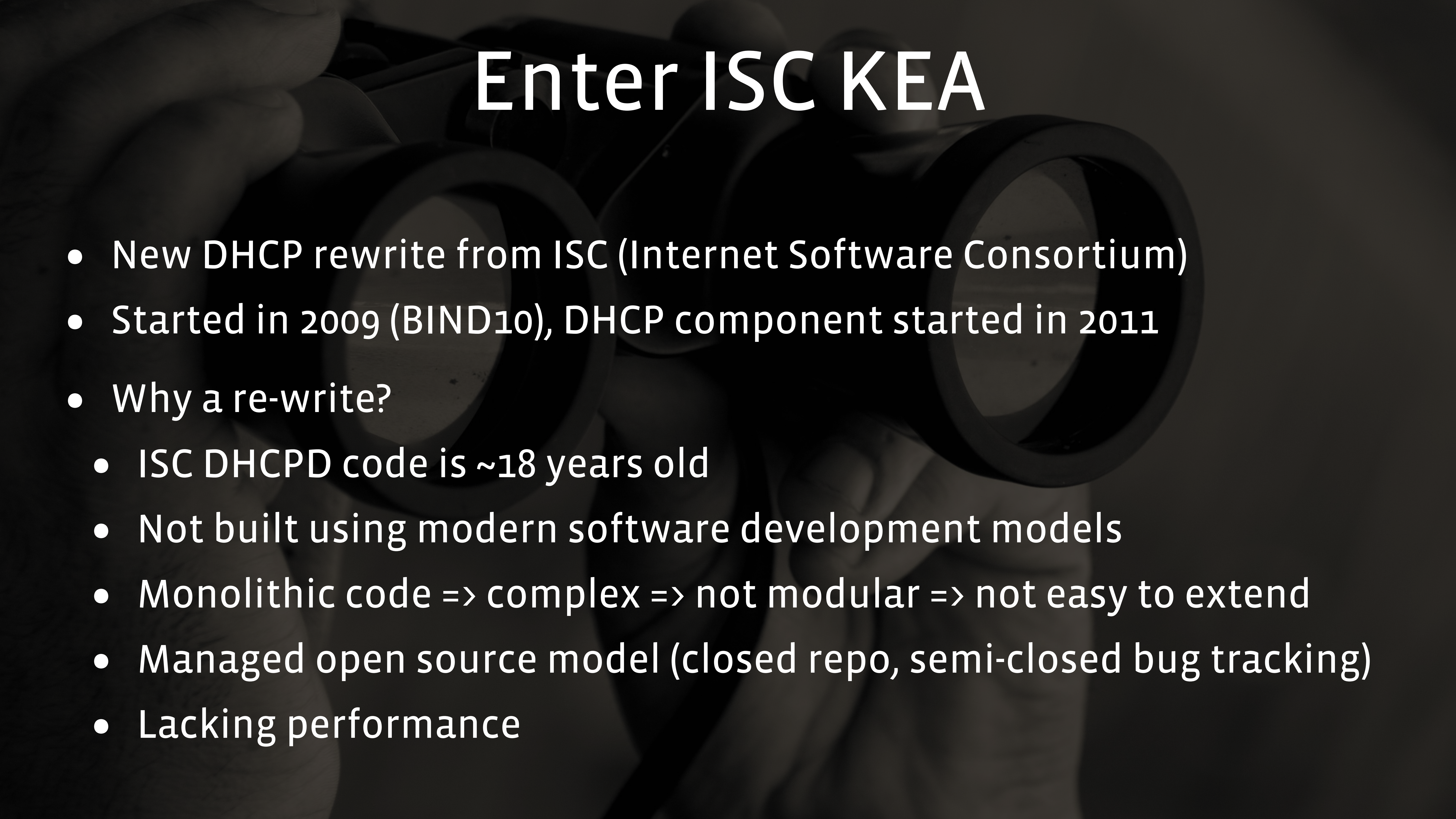
Photo by Bill Abbott-

Path to success

- Support both DHCPv4 and DHCPv6
- Stateless server
  - shipping config shouldn’t be required
  - host data should be pulled dynamically from inventory system
- Get rid of the hardware load balancers
- Must be easy to “containerize”
- Integrated with Facebook infrastructure

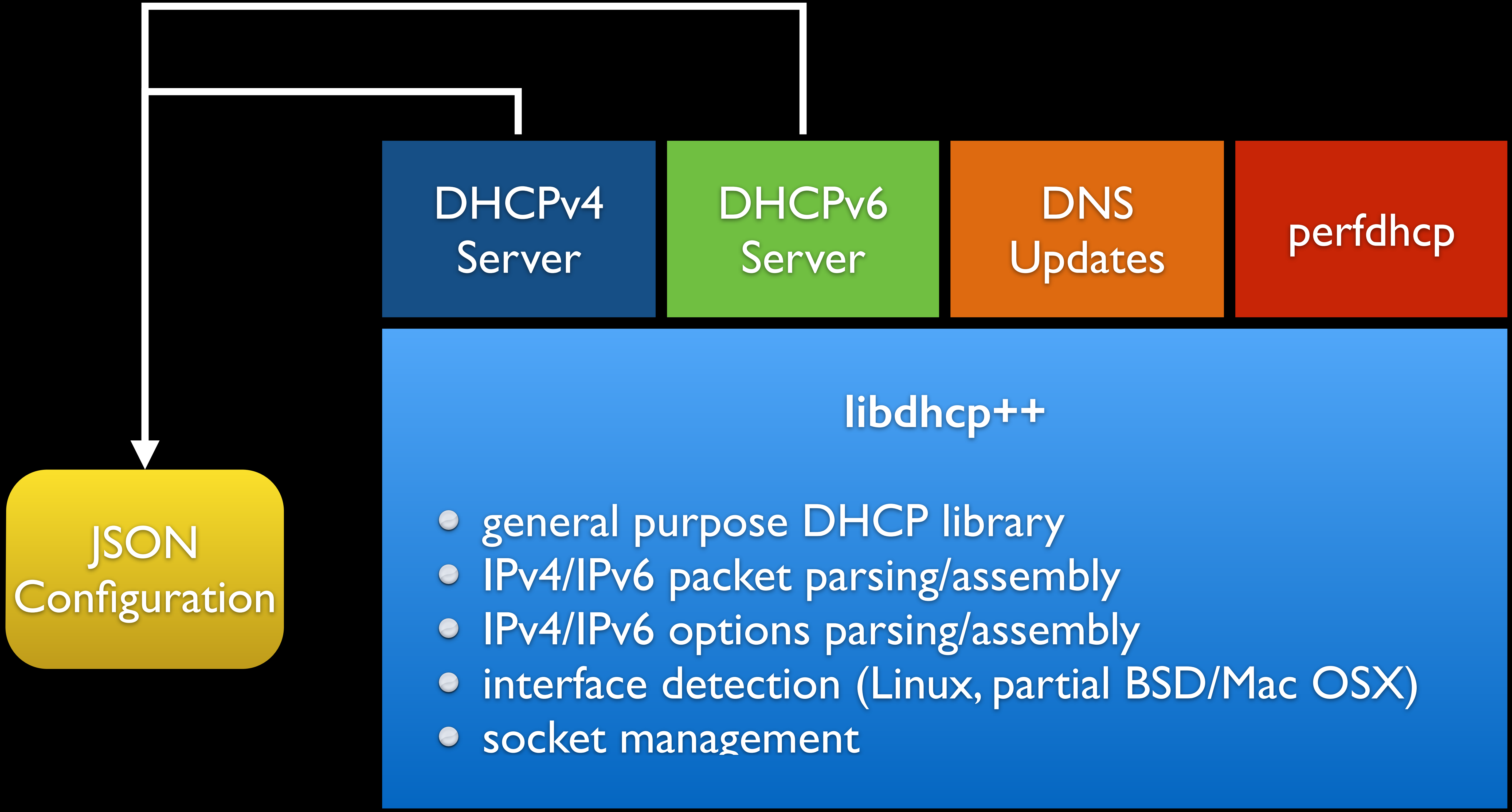
Photo by Angelo Failla -

Photo by Edith Soto -



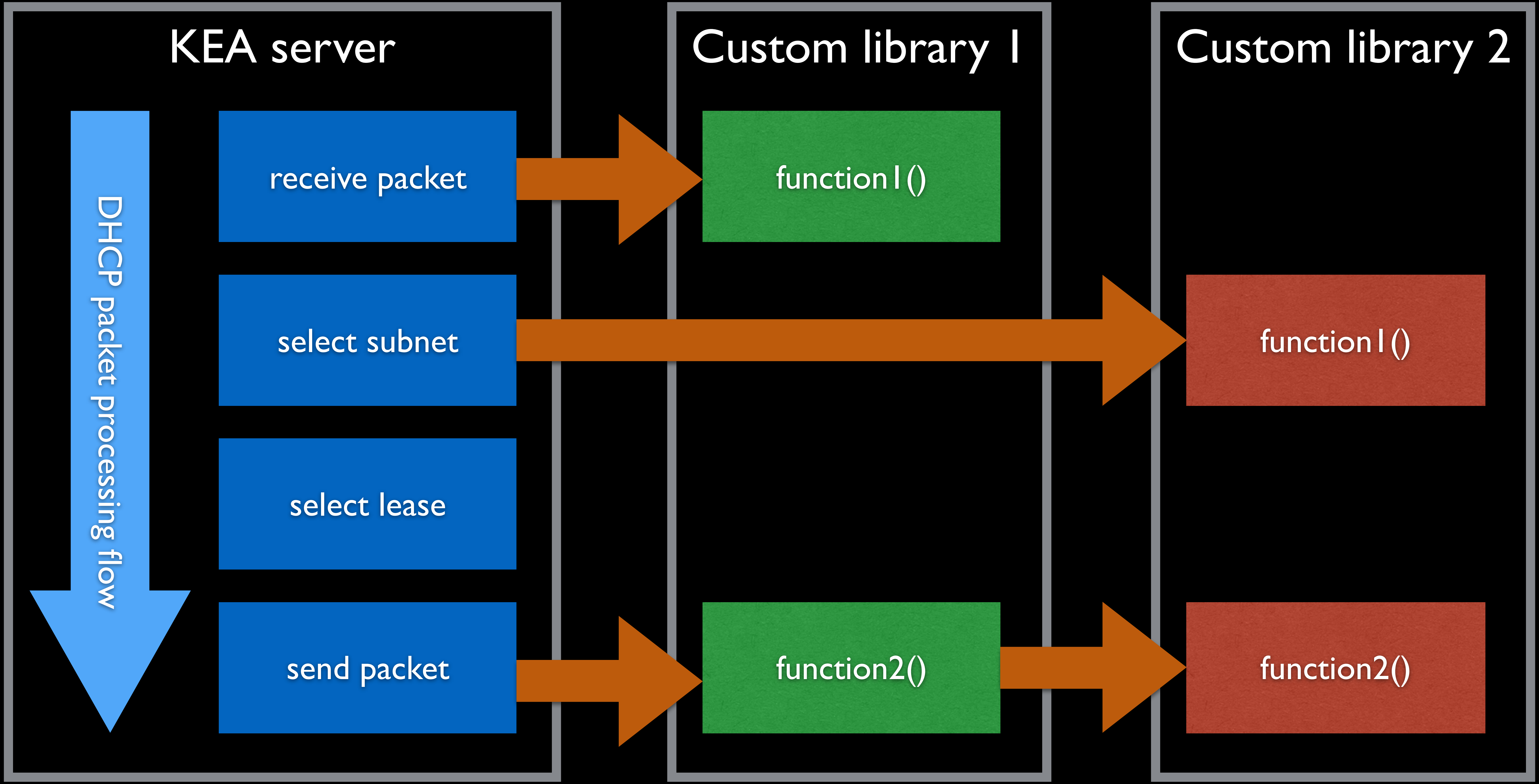
Enter ISC KEA

- New DHCP rewrite from ISC (Internet Software Consortium)
- Started in 2009 (BIND10), DHCP component started in 2011
- Why a re-write?
  - ISC DHCPD code is ~18 years old
  - Not built using modern software development models
  - Monolithic code => complex => not modular => not easy to extend
  - Managed open source model (closed repo, semi-closed bug tracking)
  - Lacking performance



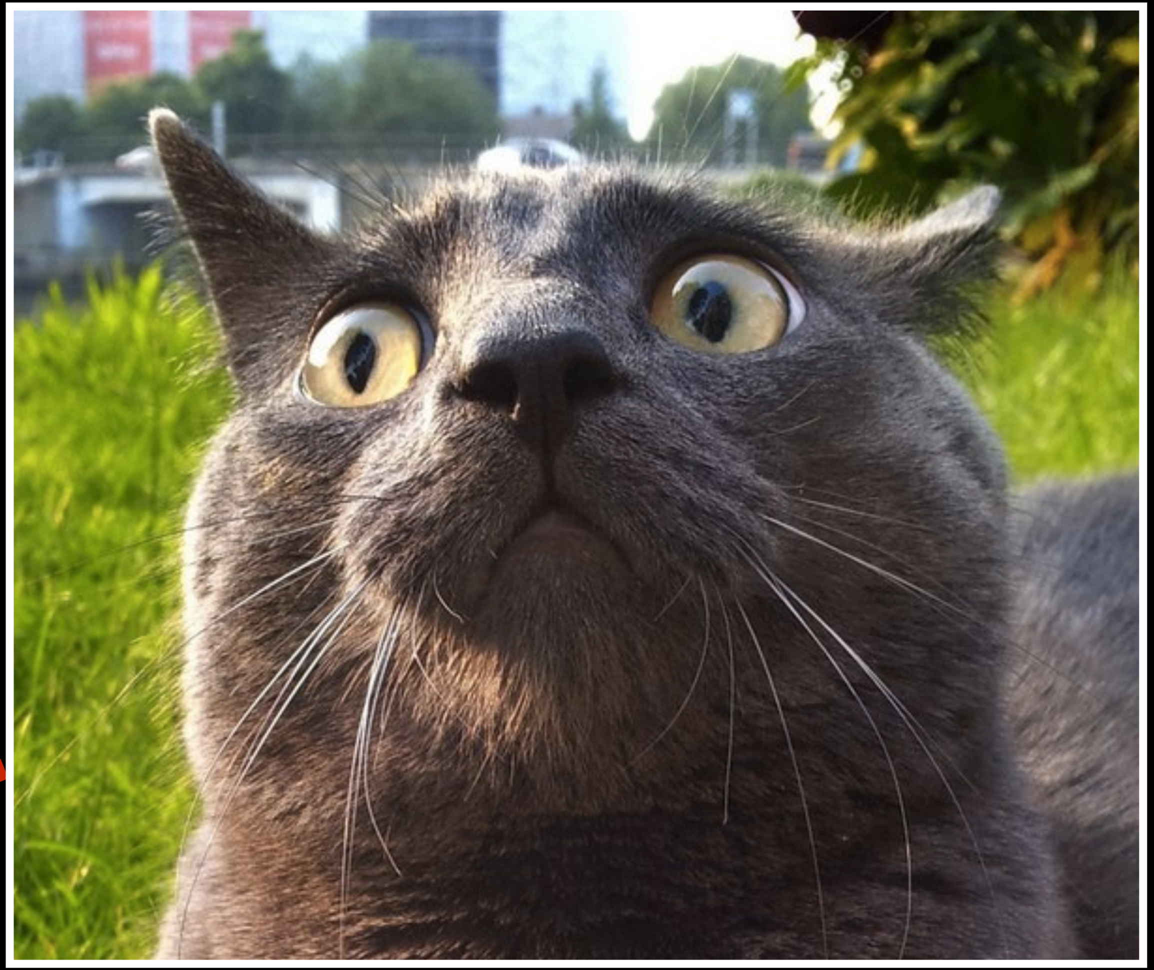
- general purpose DHCP library
- IPv4/IPv6 packet parsing/assembly
- IPv4/IPv6 options parsing/assembly
- interface detection (Linux, partial BSD/Mac OSX)
- socket management

Extending KEA: the Hook API

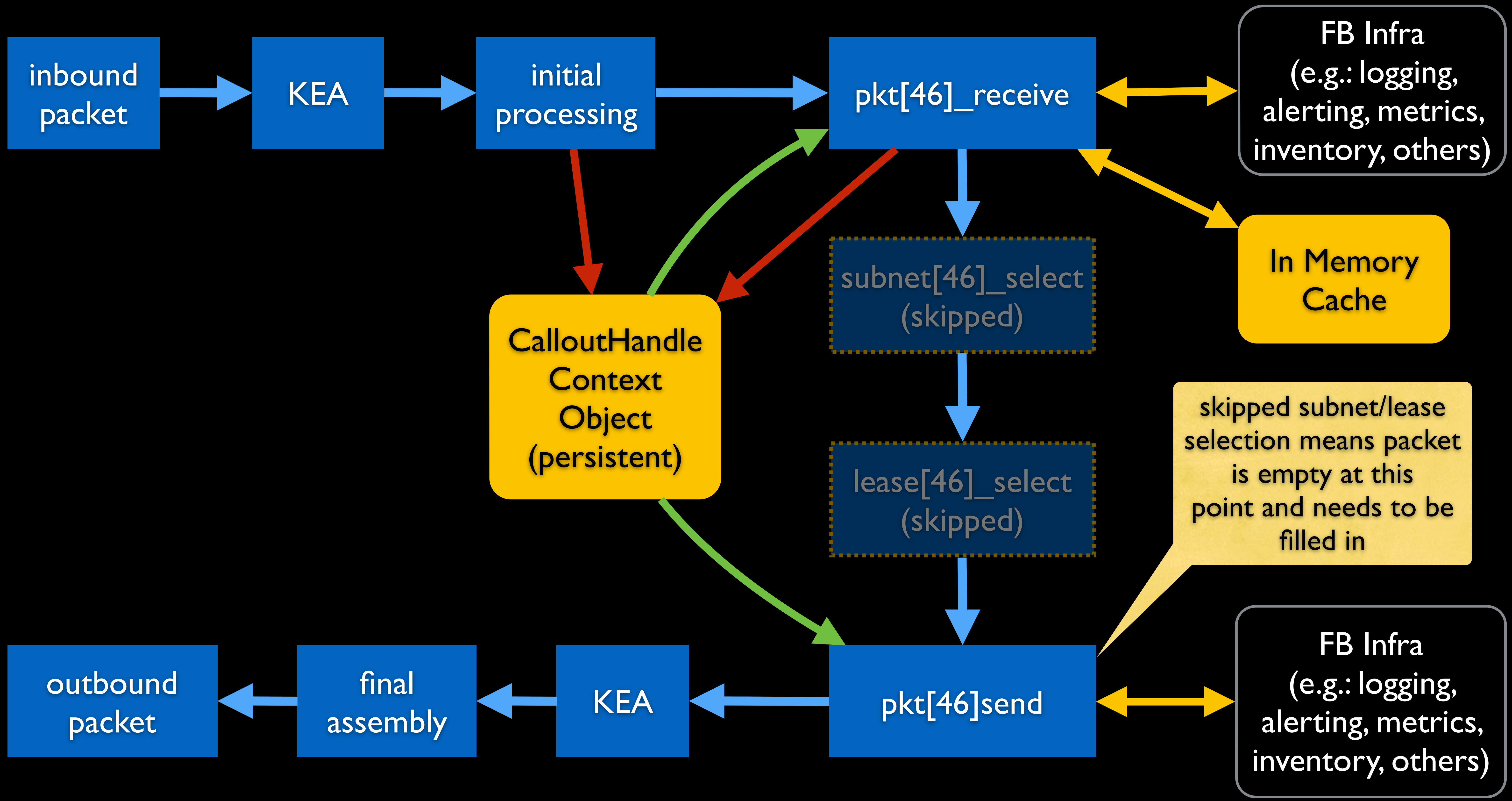


KEA JSON config file looks like this:

```
{
  "Logging": {
    "loggers": [{
        "severity": "DEBUG",
        "name": "*",
        "debuglevel": 0
      }]
  },
  "Dhcp4": {
    "hooks-libraries": [
      "/path/to/your/library.so"
    ],
    "interfaces": [
      "eth0"
    ],
    "valid-lifetime": 4000,
    "renew-timer": 1000,
    "rebind-timer": 2000,
    "subnet4": [{
        "subnet": "0.0.0.0/0",
        "pools": [{
            "pool": "0.0.0.0-255.255.255.255"
          }]
    }]
  }
}
```



Life of a packet in the FB Hook library


inbound packet
KEA
initial processing
pkt[46]_receive
FB Infra (e.g.: logging, alerting, metrics, inventory, others)
In Memory Cache
CalloutHandle Context Object (persistent)
subnet[46]_select (skipped)
lease[46]_select (skipped)
skipped subnet/lease selection means packet is empty at this point and needs to be filled in
pkt[46]send
FB Infra (e.g.: logging, alerting, metrics, inventory, others)
KEA
final assembly
outbound packet

```
#include <hooks/hooks.h>
#include <dhcp/pkt4.h>
#include <dhcp/hwaddr.h>
#include "yourlibs.h"

using namespace isc::hooks;

extern "C" {

  int version() {
    return KEA_HOOKS_VERSION;
  }

  int load(LibraryHandle& libhandle) {
    // initialize needed objects
    // (logging, cache, config, etc)
    return 0;
  }

  int unload() {
    // destroy the objects
    return 0;
  }

. . . . . . .
```

```
. . . . . . .

  int subnet4_select(CalloutHandle& handle) {
    handle.setSkip(true);
    return 0;
  }

  int lease4_select(CalloutHandle& handle) {
    handle.setSkip(true);
    return 0;
  }

. . . . . . .
. . . . .

 int pkt4_receive(CalloutHandle& handle) {











 }

. . . . .
```

```
  . . . . .

  int pkt4_send(CalloutHandle& handle) {
















}
```

```
. . . . .

 int pkt4_receive(CalloutHandle& handle) {

   Pkt4Ptr query4_ptr;
   handle.getArgument("query4", query4_ptr);

 }

. . . . .
```

```
  . . . . .

  int pkt4_send(CalloutHandle& handle) {

}
```

```
. . . . .

 int pkt4_receive(CalloutHandle& handle) {

   Pkt4Ptr query4_ptr;
   handle.getArgument("query4", query4_ptr);

   HWAddrPtr hwaddr_ptr = query4_ptr->getHWAddr();















 }

. . . . .
```

```
  . . . . .

  int pkt4_send(CalloutHandle& handle) {

















}
```

```
. . . . .

int pkt4_receive(CalloutHandle& handle) {

  Pkt4Ptr query4_ptr;
  handle.getArgument("query4", query4_ptr);

  HWAddrPtr hwaddr_ptr = query4_ptr->getHWAddr();

  HostStateObject hostInfo;
  if (!getHostInfo(&hostInfo, hwaddr_ptr)) {
    LOG(ERROR) << "Something went wrong!";
    handle.setSkip(true);
    return 0;
  }






}

. . . . .
```

```
  . . . . .

  int pkt4_send(CalloutHandle& handle) {


















}
```

```
. . . . .

int pkt4_receive(CalloutHandle& handle) {

  Pkt4Ptr query4_ptr;
  handle.getArgument("query4", query4_ptr);

  HWAddrPtr hwaddr_ptr = query4_ptr->getHWAddr();

  HostStateObject hostInfo;
  if (!getHostInfo(&hostInfo, hwaddr_ptr)) {
    LOG(ERROR) << "Something went wrong!";
    handle.setSkip(true);
    return 0;
  }

  logStuff(query4_ptr);

  handle.setContext("hostInfo", hostInfo);

  return 0;
}

. . . . .
```

```
. . . . .

int pkt4_send(CalloutHandle& handle) {










}
```

```
. . . . .

 int pkt4_receive(CalloutHandle& handle) {

   Pkt4Ptr query4_ptr;
   handle.getArgument("query4", query4_ptr);

   HWAddrPtr hwaddr_ptr = query4_ptr->getHWAddr();

   HostStateObject hostInfo;
   if (!getHostInfo(&hostInfo, hwaddr_ptr)) {
     LOG(ERROR) << "Something went wrong!";
     handle.setSkip(true);
     return 0;
   }

   logStuff(query4_ptr);

   handle.setContext("hostInfo", hostInfo);

   return 0;
 }

. . . . .
```

```
 . . . . .

 int pkt4_send(CalloutHandle& handle) {

   Pkt4Ptr response4_ptr;
   HostStateObject hostInfo;

   // at this point response4 is empty so we have
   // to fill all the things ourselves
   handle.getArgument("response4", response4_ptr);
   handle.getContext("hostInfo", hostInfo);










}
```

```
. . . . .

int pkt4_receive(CalloutHandle& handle) {

  Pkt4Ptr query4_ptr;
  handle.getArgument("query4", query4_ptr);

  HWAddrPtr hwaddr_ptr = query4_ptr->getHWAddr();

  HostStateObject hostInfo;
  if (!getHostInfo(&hostInfo, hwaddr_ptr)) {
    LOG(ERROR) << "Something went wrong!";
    handle.setSkip(true);
    return 0;
  }

  logStuff(query4_ptr);

  handle.setContext("hostInfo", hostInfo);

  return 0;
}

. . . . .
```

```
. . . . .

int pkt4_send(CalloutHandle& handle) {

  Pkt4Ptr response4_ptr;
  HostStateObject hostInfo;

  // at this point response4 is empty so we have
  // to fill all the things ourselves
  handle.getArgument("response4", response4_ptr);
  handle.getContext("hostInfo", hostInfo);

  // set all relevant options (e.g. default gw,
  // boot options, subnet, DNS, domain search,
  // hostname, lease time, etc)
  fillUpResponsePacket(response4_ptr, hostInfo);



}
```

```
. . . . .

int pkt4_receive(CalloutHandle& handle) {

  Pkt4Ptr query4_ptr;
  handle.getArgument("query4", query4_ptr);

  HWAddrPtr hwaddr_ptr = query4_ptr->getHWAddr();

  HostStateObject hostInfo;
  if (!getHostInfo(&hostInfo, hwaddr_ptr)) {
    LOG(ERROR) << "Something went wrong!";
    handle.setSkip(true);
    return 0;
  }

  logStuff(query4_ptr);

  handle.setContext("hostInfo", hostInfo);

  return 0;
}

. . . . .
```

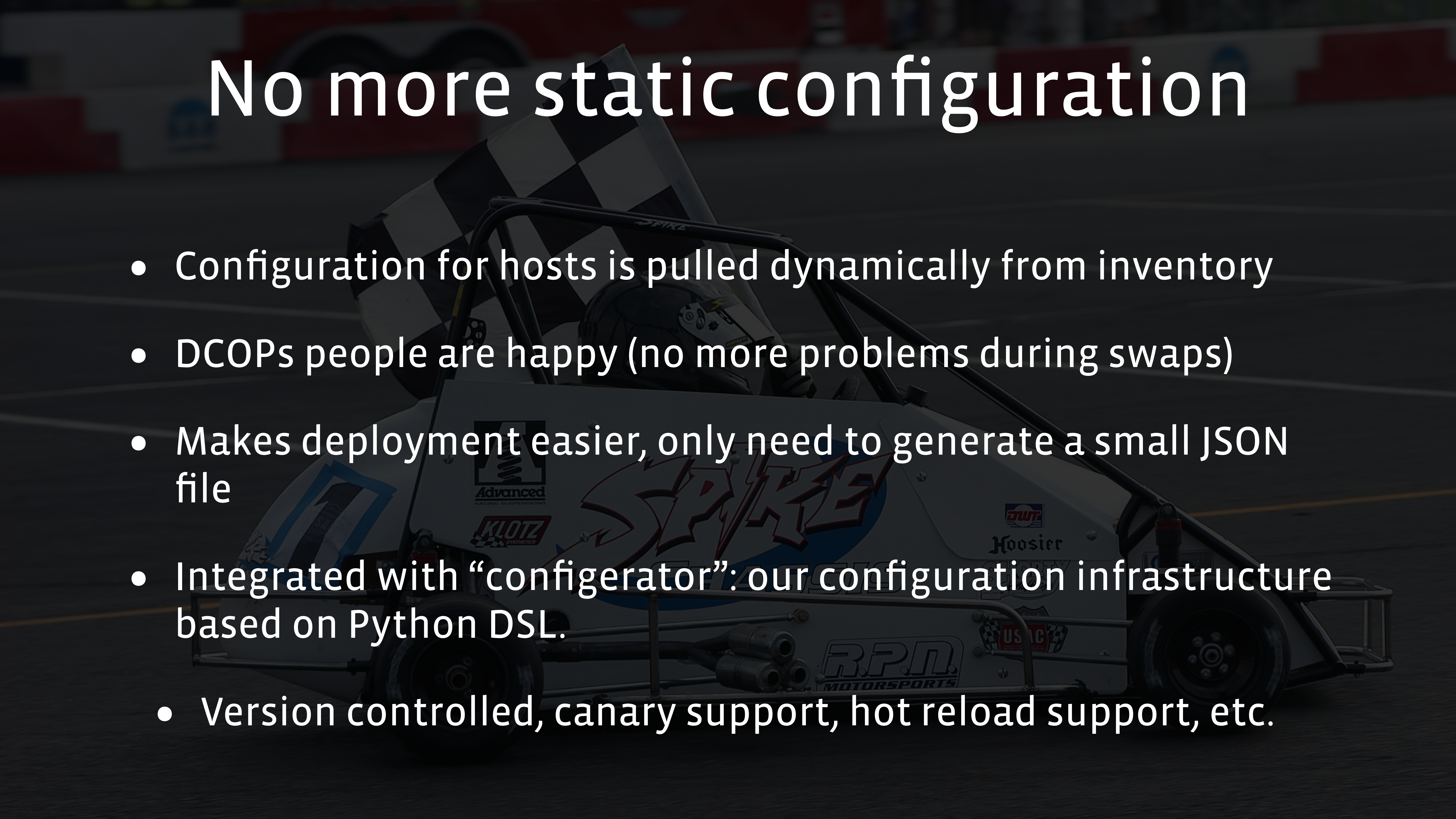
```
. . . . .

int pkt4_send(CalloutHandle& handle) {

  Pkt4Ptr response4_ptr;
  HostStateObject hostInfo;

  // at this point response4 is empty so we have
  // to fill all the things ourselves
  handle.getArgument("response4", response4_ptr);
  handle.getContext("hostInfo", hostInfo);

  // set all relevant options (e.g. default gw,
  // boot options, subnet, DNS, domain search,
  // hostname, lease time, etc)
  fillUpResponsePacket(response4_ptr, hostInfo);

  logStuff(response4_ptr);
  return 0;
}

}
```

You can compile your code using something like this:

```
$ g++ -I /usr/include/kea -L /usr/lib/kea/lib -fpic -shared -o ${your_lib}.so \
      ${your_hook_lib_files} -lkea-dhcpsrv -lkea-dhcp++ -lkea-hooks -lkea-log \
      -lkea-util -lkea-exceptions
```

Photo by Greg Hildebrand -

Big wins

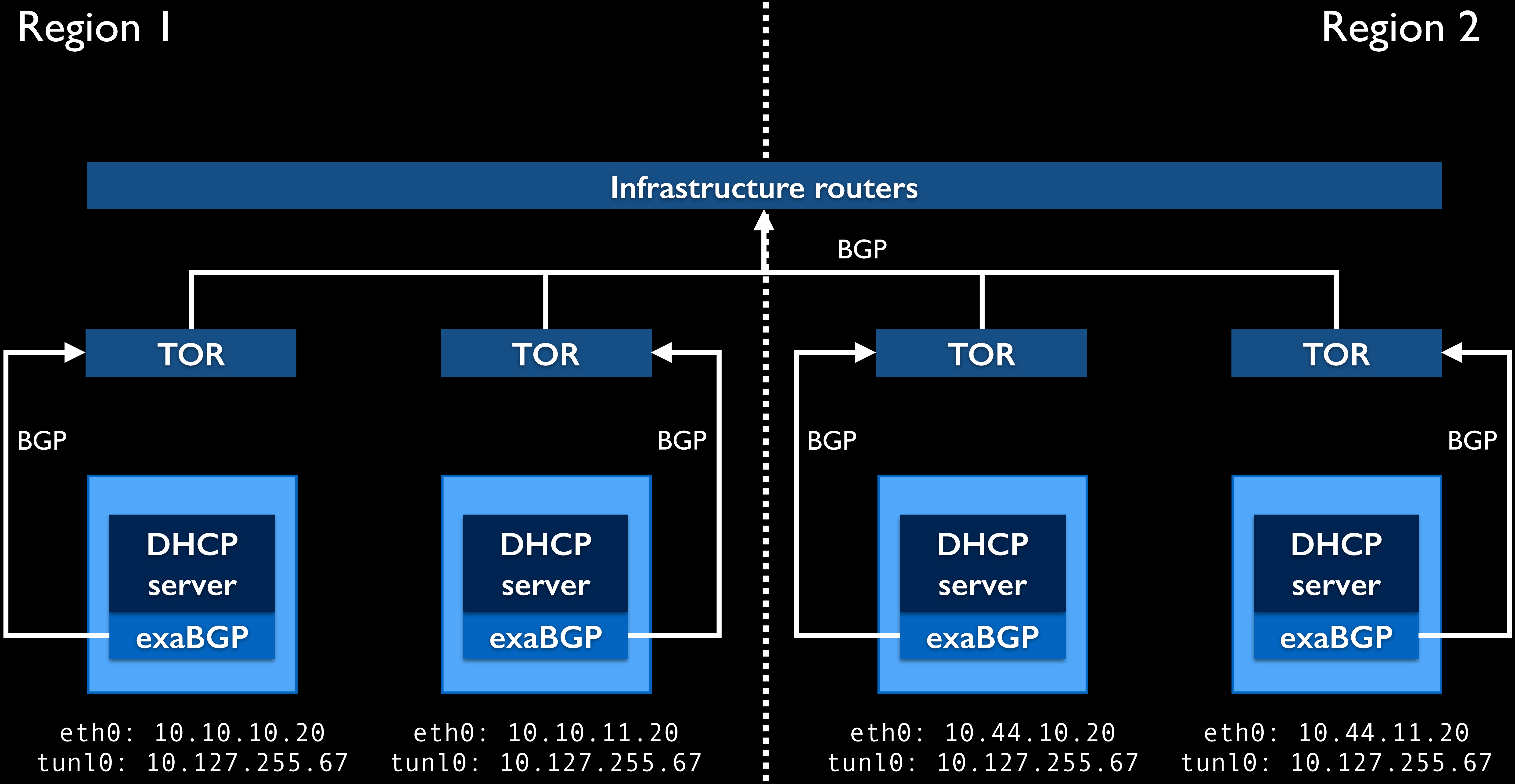


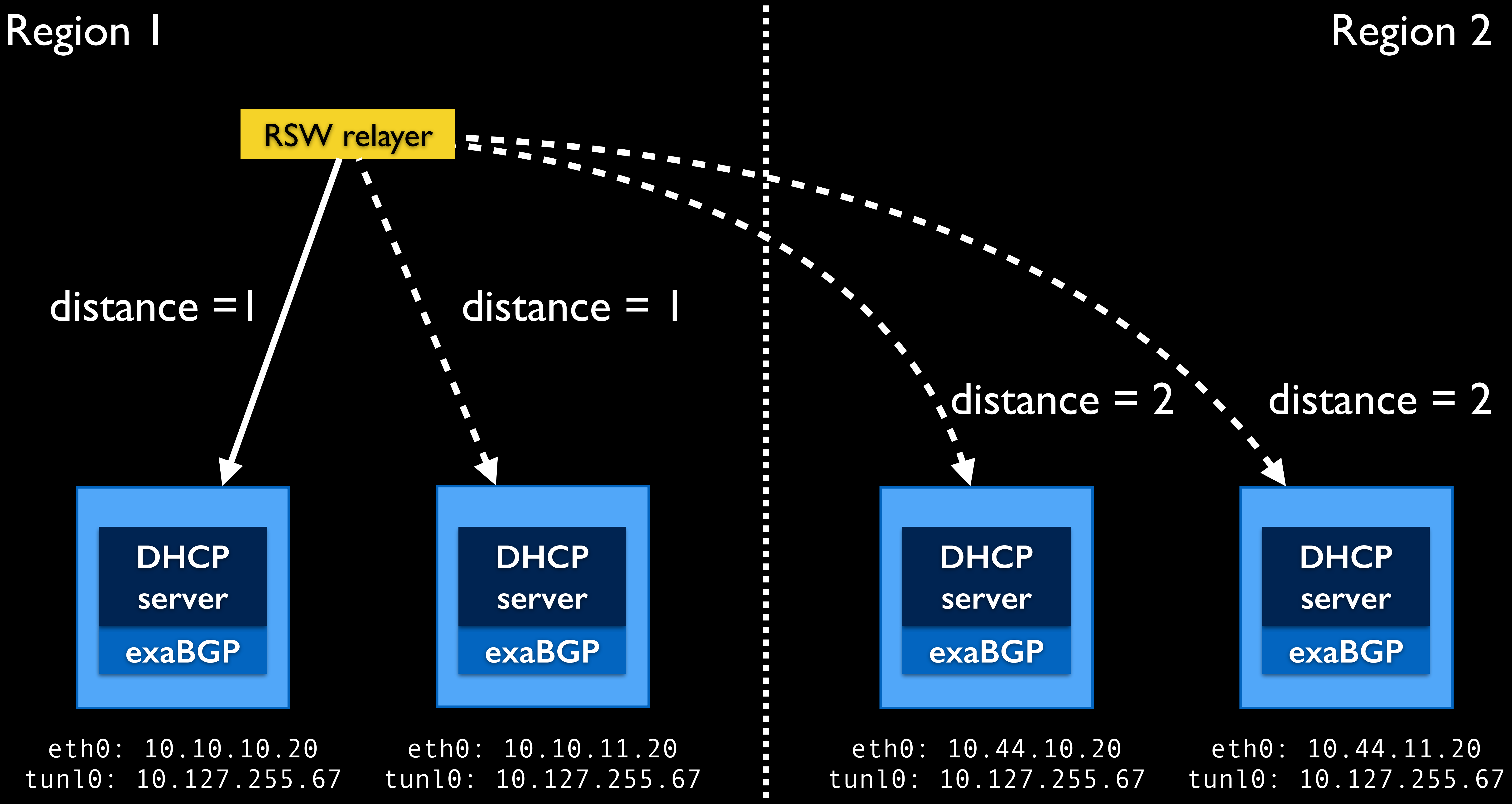
No more static configuration

- Configuration for hosts is pulled dynamically from inventory
- DCOPs people are happy (no more problems during swaps)
- Makes deployment easier, only need to generate a small JSON file
- Integrated with “configerator”: our configuration infrastructure based on Python DSL.
  - Version controlled, canary support, hot reload support, etc.

No more hardware load balancing!

- Switched to Anycast/ECMP
- Packets sent to the anycast address are delivered to the nearest server
- Same fleet-wide “anycast” IP is assigned to all DHCP servers (to `ip6tnl0/tunl0` interfaces)
- ExaBGP is used to advertise the anycast IP
- Servers become routers and part of the network infrastructure


Region 1
Region 2
Infrastructure routers
BGP
TOR
TOR
TOR
TOR
BGP
BGP
BGP
BGP
DHCP server
exaBGP
DHCP server
exaBGP
DHCP server
exaBGP
DHCP server
exaBGP
eth0: 10.10.10.20
tunl0: 10.127.255.67
eth0: 10.10.11.20
tunl0: 10.127.255.67
eth0: 10.44.10.20
tunl0: 10.127.255.67
eth0: 10.44.11.20
tunl0: 10.127.255.67



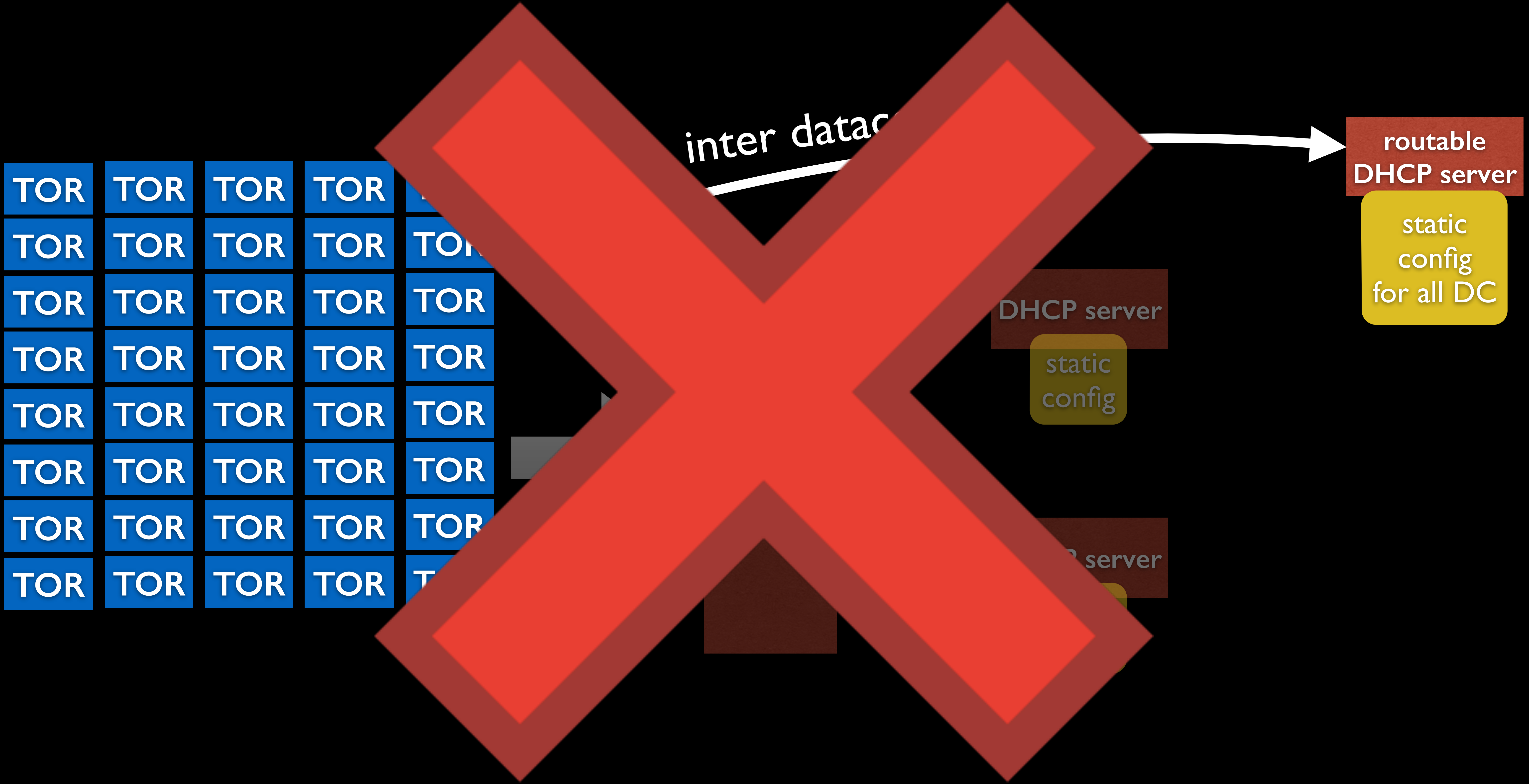
```
eth0: 10.10.10.20
tunl0: 10.127.255.67
```

```
eth0: 10.10.11.20
tunl0: 10.127.255.67
```

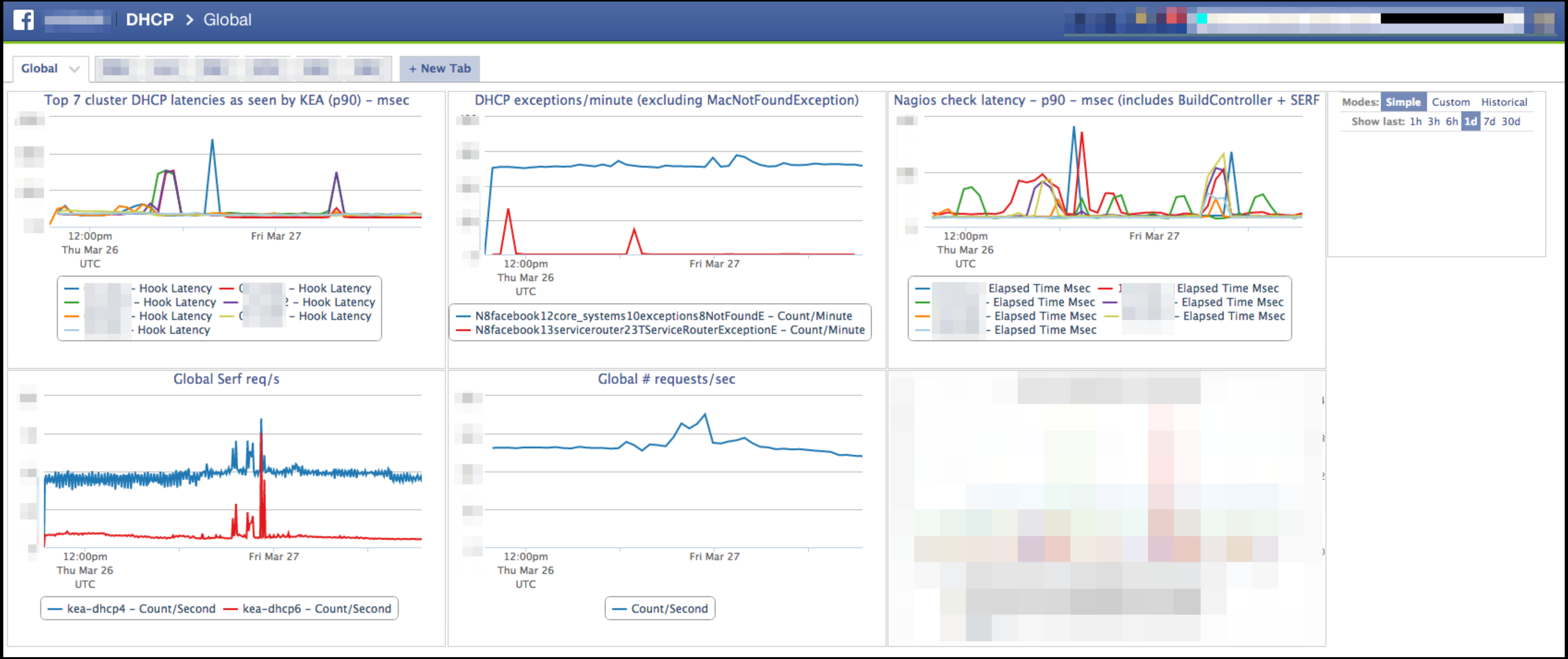
```
eth0: 10.44.10.20
tunl0: 10.127.255.67
```

```
eth0: 10.44.11.20
tunl0: 10.127.255.67
```

Seamless cluster/datacenter turnups



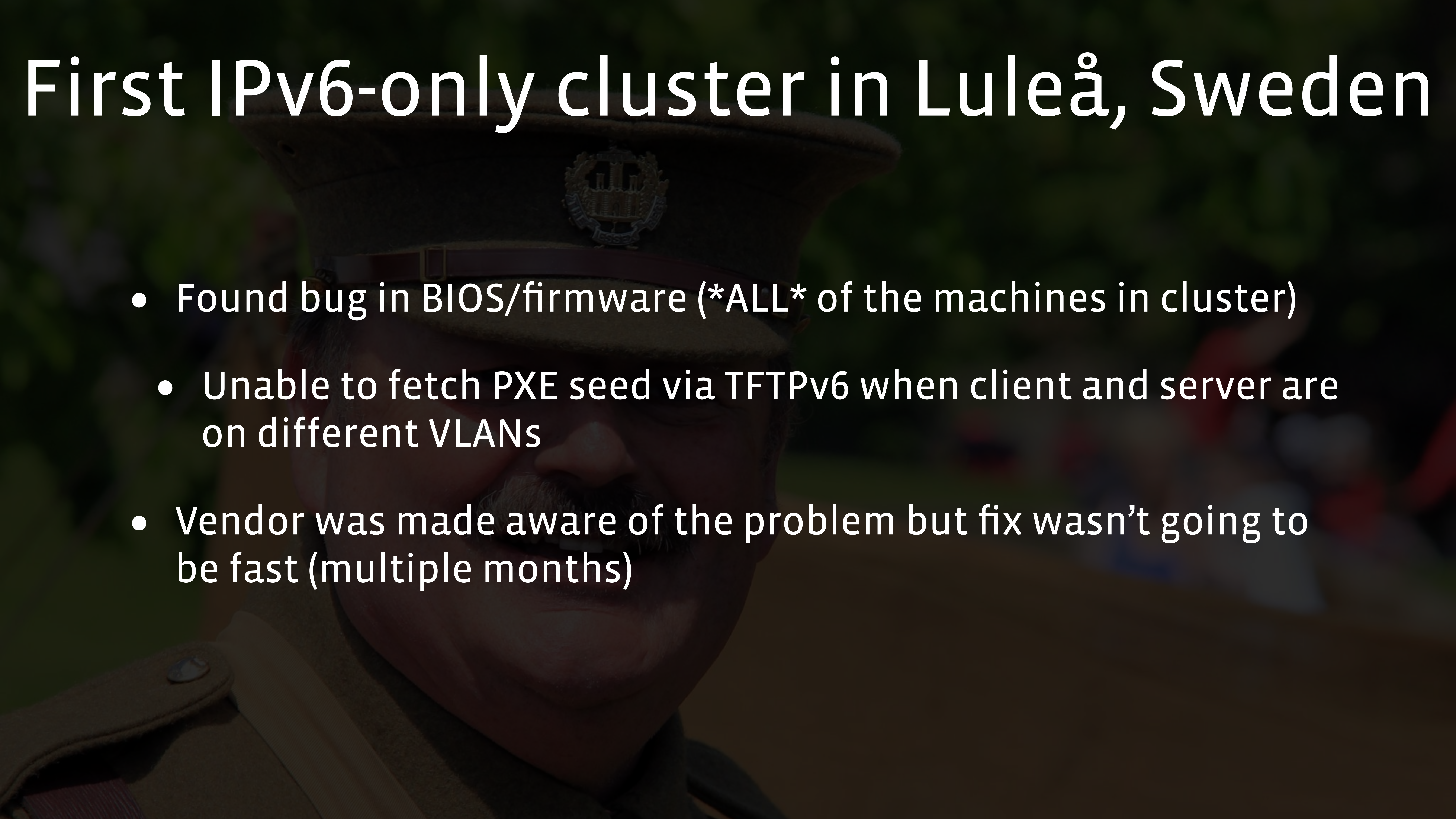
Metrics!



DHCP > Global
Global
+ New Tab
Top 7 cluster DHCP latencies as seen by KEA (p90) – msec
12:00pm Thu Mar 26 UTC
Fri Mar 27
- Hook Latency
- Hook Latency
- Hook Latency
- Hook Latency
- Hook Latency
- Hook Latency
- Hook Latency
DHCP exceptions/minute (excluding MacNotFoundException)
N8facebook12core_systems10exceptions8NotFoundE – Count/Minute
N8facebook13servicerouter23TServiceRouterExceptionE – Count/Minute
Nagios check latency – p90 – msec (includes BuildController + SERF
- Elapsed Time Msec
- Elapsed Time Msec
- Elapsed Time Msec
- Elapsed Time Msec
- Elapsed Time Msec
- Elapsed Time Msec
- Elapsed Time Msec
Modes: Simple Custom Historical
Show last: 1h 3h 6h 1d 7d 30d
Global Serf req/s
kea-dhcp4 – Count/Second
kea-dhcp6 – Count/Second
Global # requests/sec
Count/Second

Photo by Karen Roe - ______________ ______________________________

First IPv6-only cluster in Luleå, Sweden



- Found bug in BIOS/firmware (*ALL* of the machines in cluster)
  - Unable to fetch PXE seed via TFTPv6 when client and server are on different VLANs
- Vendor was made aware of the problem but fix wasn't going to be fast (multiple months)

The workaround

- Realized proprietary TORs could run Python scripts
- Wrote quick and dirty Python TFTP relayer
- Deployed into all TORs in the cluster
- Modify KEA hook lib to override the IP of the cluster TFTP endpoint with the IP of the machine in the rack (which is in same VLAN)

Photo by Ollie Richardson -

Some takeaways...

Stateless is good! → Keep data (state and config) remotely → It simplifies configuration management → and deployment!

The “Not Invented Here” syndrome

Resources

- **KEA:** http://kea.isc.org
- **exaBGP:** https://github.com/Exa-Networks/exabgp
- **OpenCompute: FBOSS and wedge rack switch:**
    - https://code.facebook.com/posts/843620439027582/facebook-open-switching-system-fboss-and-wedge-in-the-open/
    - https://code.facebook.com/posts/681382905244727/introducing-wedge-and-fboss-the-next-steps-toward-a-disaggregated-network/
    - https://code.facebook.com/posts/717010588413497/introducing-6-pack-the-first-open-hardware-modular-switch/
    - https://github.com/facebook/fboss

Questions?

facebook