CUMULUS

Improving Route Scalability: Nexthops as Separate Objects

September 2019

David Ahern | Cumulus Networks

Agenda

Executive Summary

- If you remember nothing else about this talk …

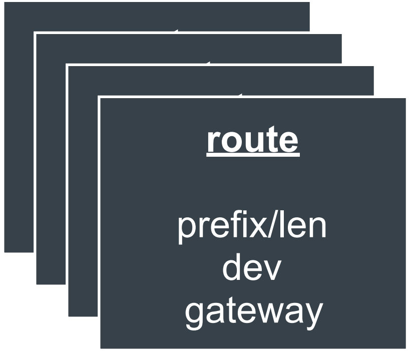
Driving use case

Review legacy route API

Dive into Nexthop API

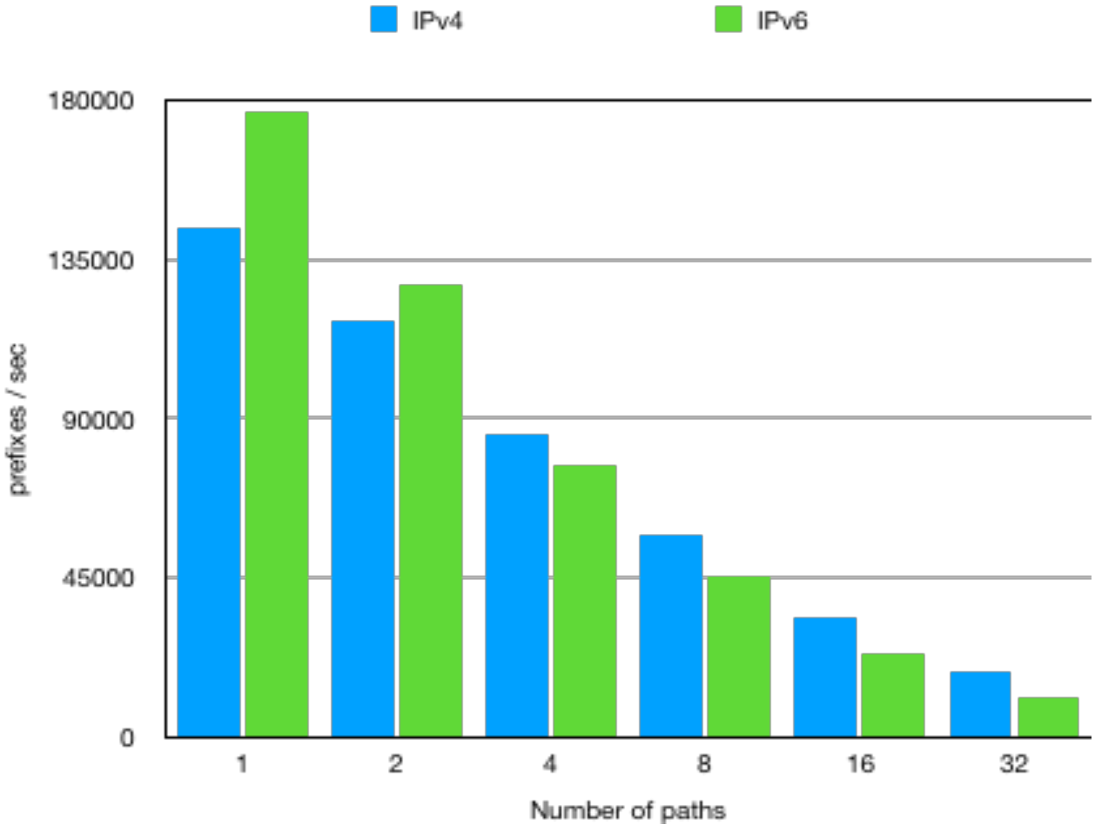
Benefits of the new API

Performance with the Legacy Route API

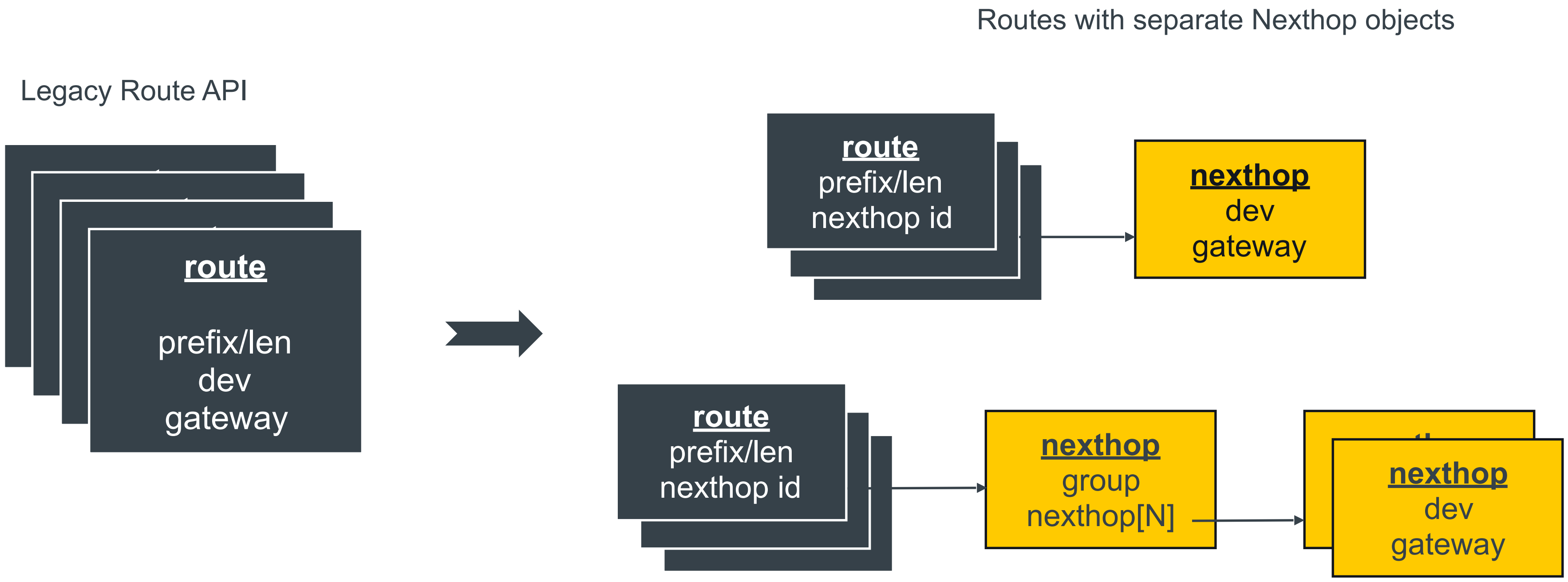


route

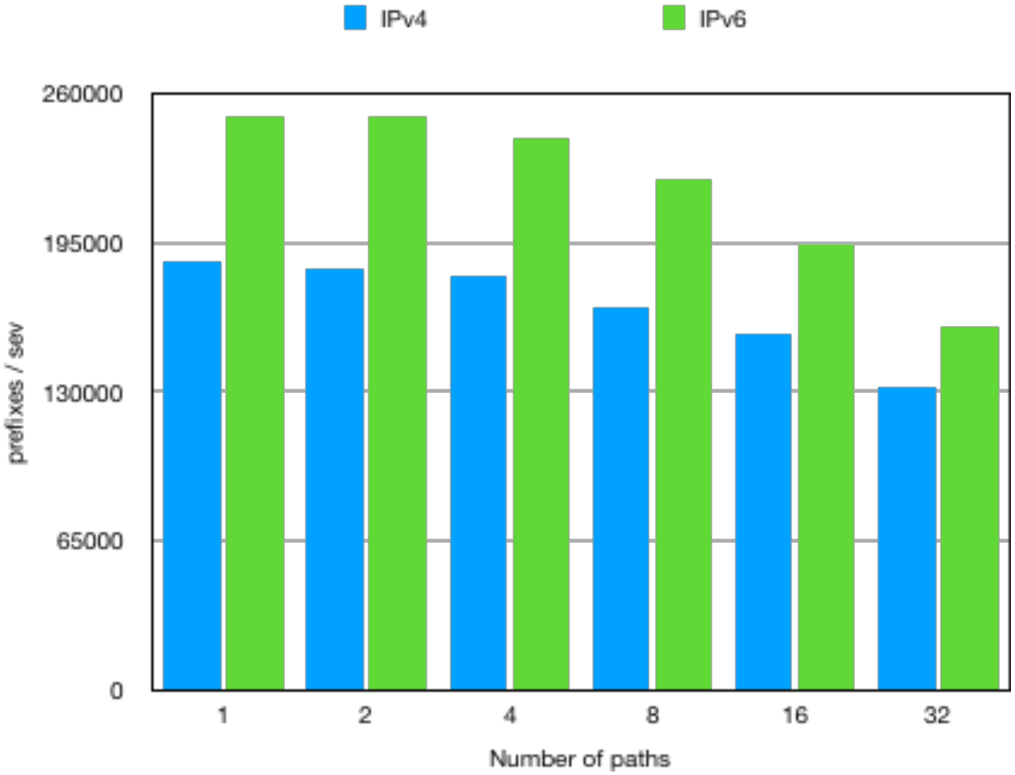
prefix/len
dev
gateway



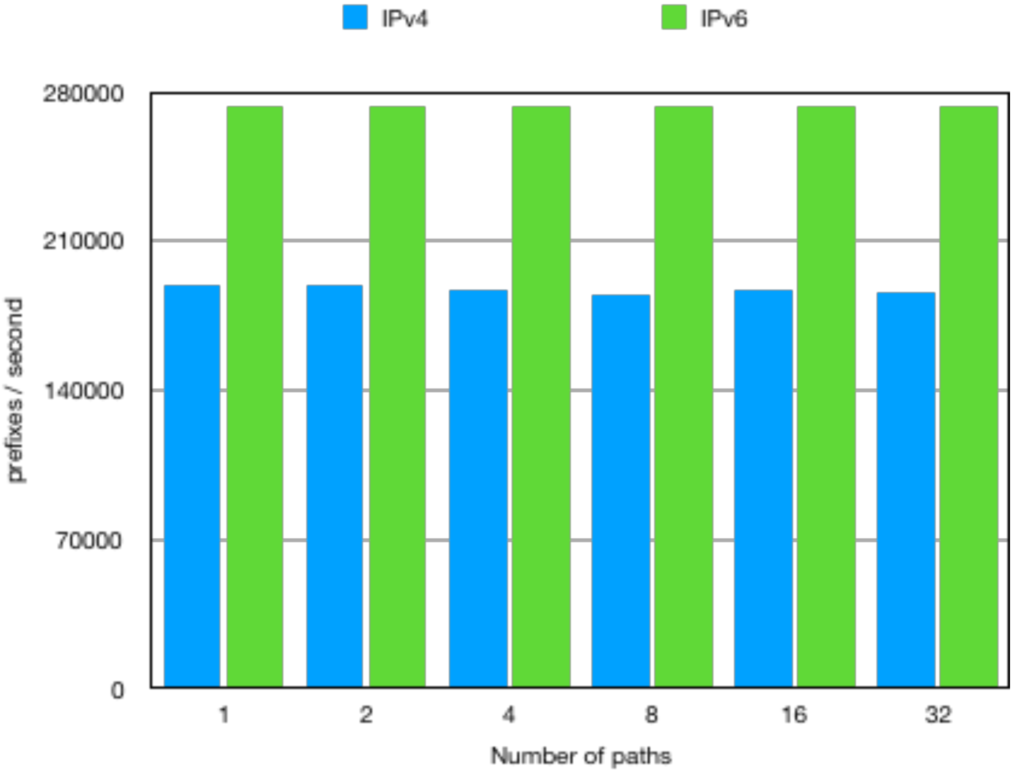
Splitting Next Hops from Routes



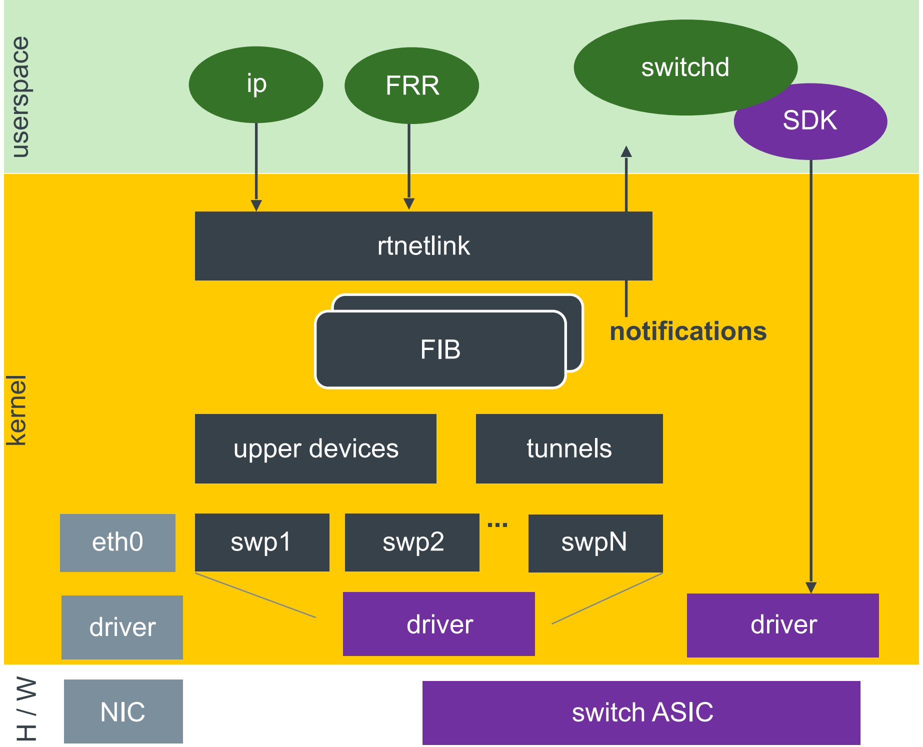
Dramatically Improves Route Scalability …



… with the Potential for Constant Insert Times



Networking Operating System Using Linux APIs



Routing daemon or utility manages entries in kernel FIBs via rtnetlink APIs

- Enables other control plane software to use Linux networking APIs

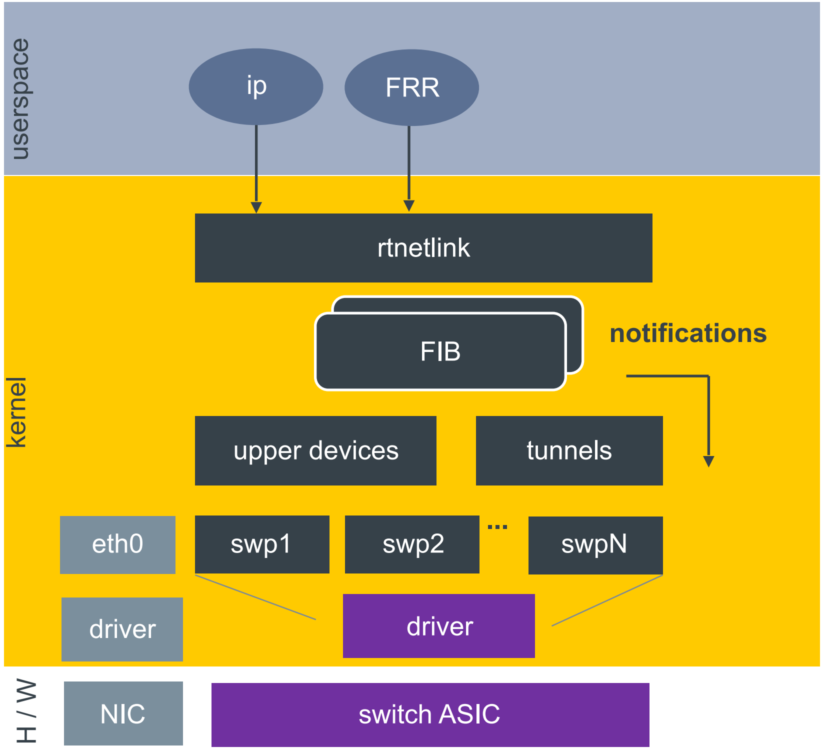
  Data path connections, stats, troubleshooting, …

Management of hardware offload is separate

- Keeps hardware in sync with kernel

Userspace driver with SDK leveraging kernel notifications

NOS with switchdev Driver

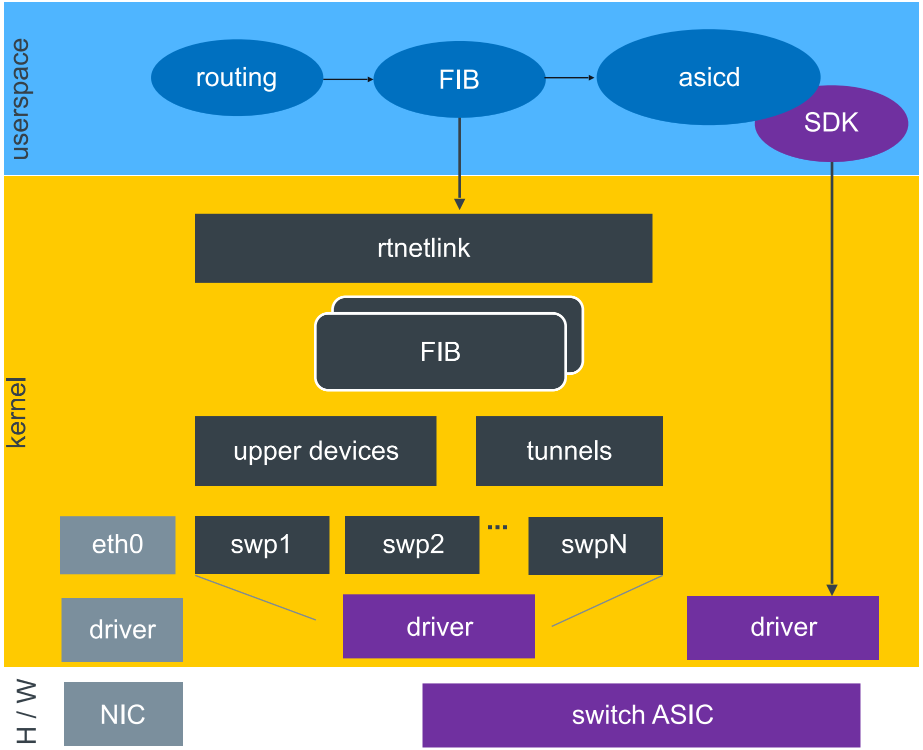


In-kernel switchdev driver

Leverages in-kernel notifications to keep hardware in sync

Minus the hardware offload and this is the same architecture for RoH

Alternative NOS with SDK Based ASIC Driver



No reliance on kernel notifiers

Kernel is treated like hardware

- Another entity to "program" based on its networking model

Key points

- Limited number of front panel ports
- Large route capacity in ASIC
- Forwarding data is pushed to kernel
- Scalability for the future

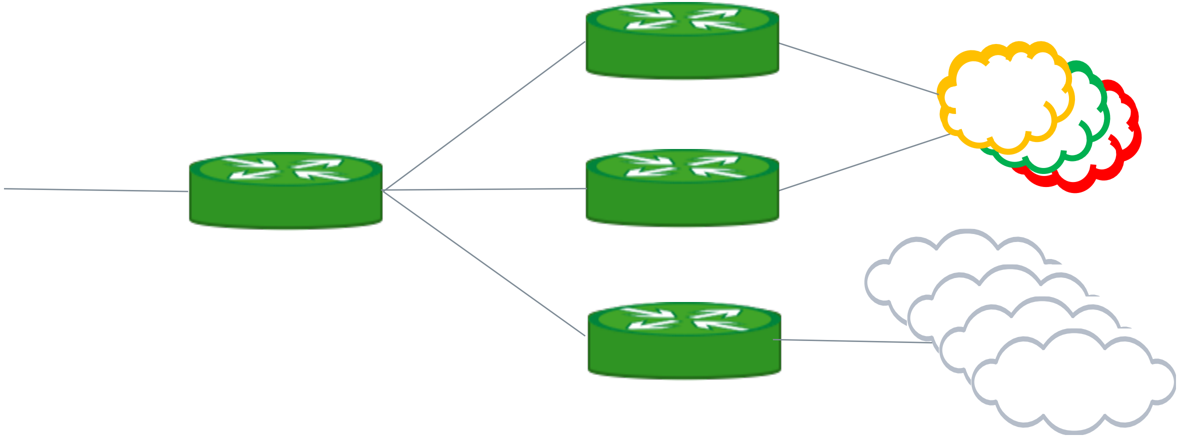
Next hops for Routes are Repetitive

Network path typically has many networks behind it

Result is prefixes out number unique nexthops by large factor

- Depending on route scale of a node, it could be 100k’s of routes with 10’s to 100’s of unique paths (nexthops and nexthop groups)

Redundant information in the forwarding configuration

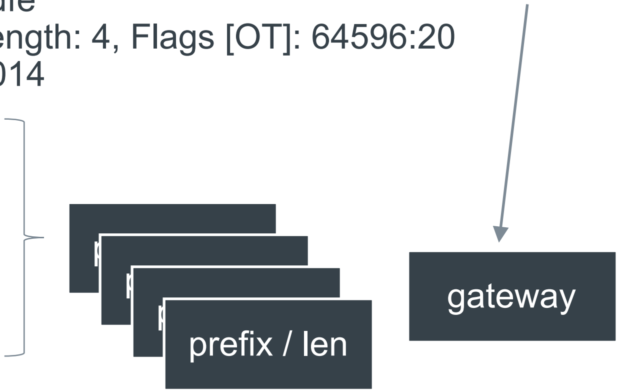


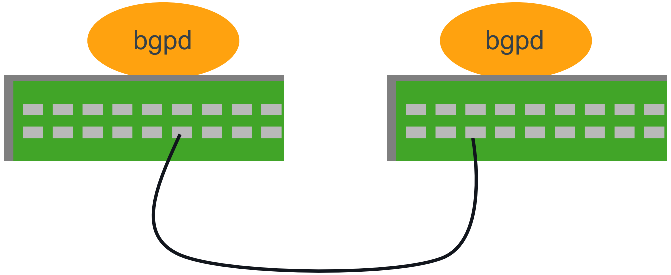
Routing Suites

Nexthop information typically separate from prefixes

- Varies by daemon (bgp, ospf, etc)

```
Update Message (2), length: 470
    Origin (1), length: 1, Flags [T]: EGP
      0x0000:  01
    AS Path (2), length: 38, Flags [T]: 65534 …
          …
    Next Hop (3), length: 4, Flags [T]: 10.203.253.254
      0x0000:  0acb fdfe
    Community (8), length: 4, Flags [OT]: 64596:20
      0x0000:  fc54 0014
    Updated routes:
      10.118.182.0/20
      10.158.166.0/20
      10.158.150.0/20
      10.158.134.0/20
      10.158.108.0/20
      10.158.102.0/20
      <more prefixes>
```





Pushing Routes to the Kernel

Netlink message per prefix to add route to kernel FIB

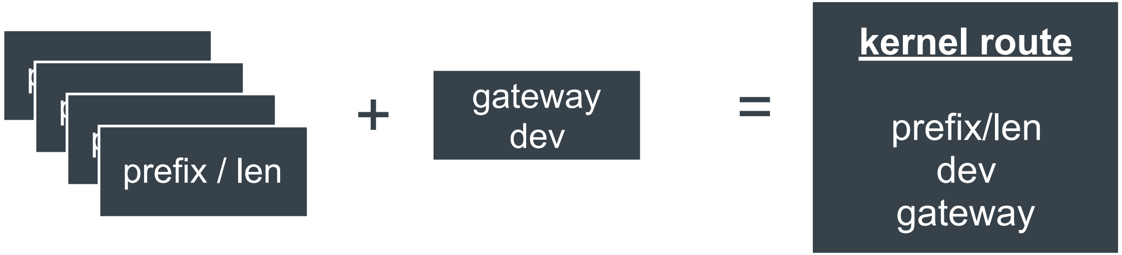
- RTM_NEWROUTE, RTM_DELROUTE

Each route expected to contain nexthop data

- RTA_OIF, RTA_GATEWAY, …

Example using iproute2:

- ip route add <prefix> via [<gw>] dev <device>
- ip route add <prefix> nexthop via [<gw>] dev <device> …



Kernel Handling

Data in each route message needs to be validated

- Gateway lookup based on current FIB data
- Verify egress device matches lookup

Nexthop specs are integrated into route structs

- ipv4: fib_nh at the end of fib_info, fib entries point to fib_info
- ipv6: fib6_nh in a fib6_info (after refactoring in early 2018)
- mpls: mpls_nh at the end of mpls_route

Notifiers in turn pass integrated data in notifier

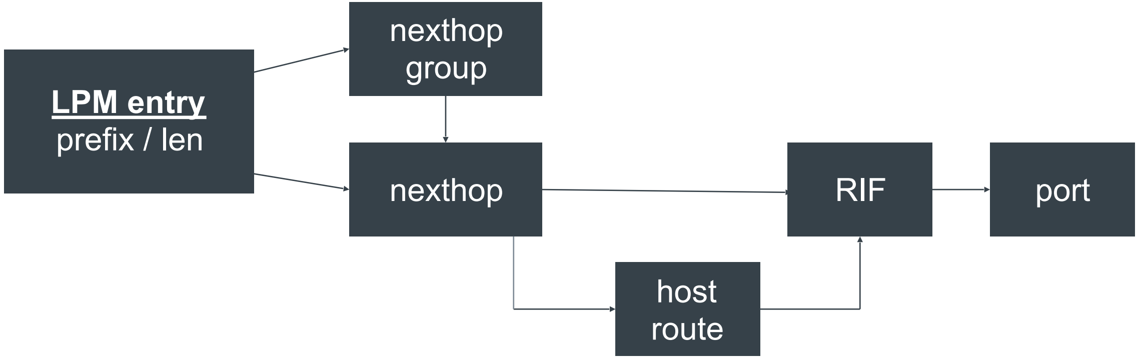
- userspace notifications and in-kernel notifiers

ASIC Programming

Map kernel model to asic resources

- Route egress netdevice = RIF on front panel port
- Gateway resolved to neighbor entry
- Add host route for gateway pointing to RIF
- Nexthop entry created pointing to RIF and host route
- Nexthop group created for multipath routes

LPM entry references nexthop or nexthop group

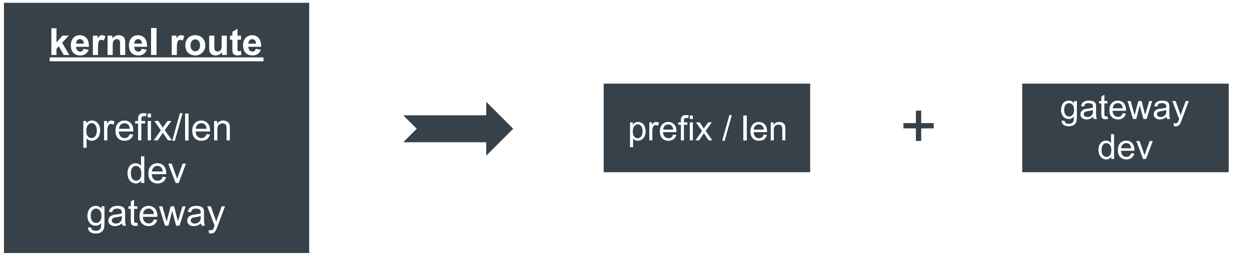


Notifier Handling - Kernel or Userspace Driver

Separate prefix / length from nexthop data

Find unique nexthop / nexthop group entry in hardware

- Lookup to see if entry already exists
- Create logically in s/w and allocate in backend RIF created for Layer 3 routing
- Reference to port and VRF



End to End – Lot of Wasted Cycles

Redundant processing adding routes

- Lookups to validate gateway addresses
- Validating lwtunnel state (e.g., MPLS encapsulation)
- Comparison of nexthop specs
- Memory allocations (e.g., pcpu for route caches)

All of it affects convergence time following a link event

- critical benchmark for a NOS

Relevant as scaling goes into the millions of routes

Nexthops as Standalone Objects

Nexthops as separate object

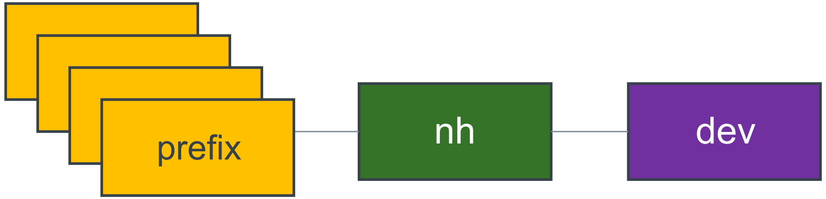
- Separate add/create/modify lifecycle from route entries
- Validation is done once

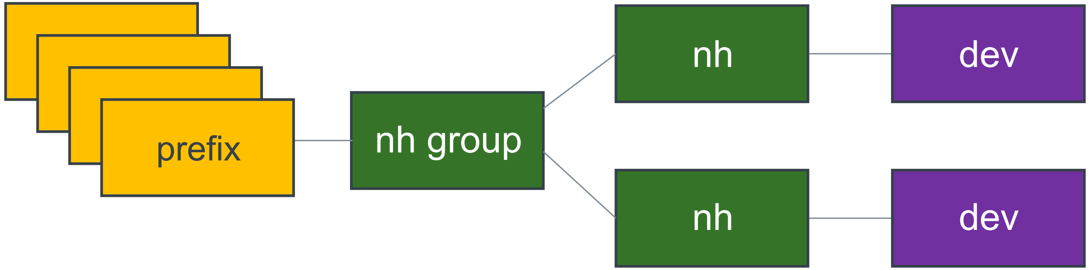
Nexthop group references one or more 'basic' nexthops

- Multipath routes

FIB entries reference nexthop by id

Simple idea; huge implications





Nexthop API

New objects with own commands and lifecycle

RTM_{NEW,DEL,GET}NEXTHOP with NHA_ attributes

- Attributes and header struct defined in include/uapi/linux/nexthop.h
- NHA_ attributes are direct parallels to RTA_ versions

Two kinds of nexthop objects: ‘basic’ nexthop or group

- Id for both can be specified (NHA_ID) or assigned by kernel
- Id (NHA_ID or nexthop->id) is unique; ASIC drivers can leverage the id to manage cache

Basic nexthop

- Device (NHA_OIF) + gateway (NHA_GATEWAY) OR blackhole (NHA_BLACKHOLE)
- Requires address family to be set

Nexthop API, cont'd

Nexthop groups reference one or more basic nexthops

- References existing nexthop by id and weight
- Address family is AF_UNSPEC
- Group can reference any 'basic' nexthops (groups with mix of address family supported)

Nexthop objects can be updated

- RTM_NEWNEXTHOP with NLM_F_REPLACE

Constraints on Nexthops

Multipath groups can not be a nexthop within a group

- No nested groups

Blackhole in a group – only 1 nexthop allowed in group

Same nexthop id can not be in a group more than once

- Limitation in how the kernel tracks nexthop references

Updates can not change nexthop ‘type’ for the id

- Basic can not become a group and vice versa

Routes with Nexthop Objects

Add routes referencing nexthop (or nexthop group) by id

- RTA_NH_ID attribute for routes
- RTA_NH_ID means RTA_OIF, RTA_GATEWAY, RTA_ENCAP can not be given

Minimal kernel checks on route add

- Nexthop id is valid
- Nexthop type is valid for route

  IPv4: scope check

  IPv6: route can not reference v4 nexthop

Co-existence of Models

If you like your current route model, you can keep it – really

- Backwards compatibility for legacy software

Userspace (e.g., routing daemons) opts in to new API

Route notifications expand nexthop

- New RTA_NH_ID attribute plus nexthop (RTA_OIF, RTA_GATEWAY)

Userspace Notifications

Usual notifications for add / delete / update of nexthop object

Intent is to minimize userspace notifications

- No notifications for link events
- Carrier down, admin down or device delete
  - Nexthop object removed
  - Routes referencing it are removed
  - Userspace expected to respond to link event

Backwards compatibility for legacy apps

- Route notifications have nexthop id and expansion of nexthop data
- Updates to nexthop generate notifications for linked routes

Nexthop Kernel Code

Code is in net/ipv4/nexthop.c, include/net/nexthop.h

- Expectation is future extensions / features with nexthop code does not require any changes to core IPv4 and IPv6

Nexthops stored in per network namespace rbtree

- Index is nexthop id

Leverages core code as much as possible

- One of the objectives of all the refactoring: move to fib_nh_common, exporting init and release for fib{6}_nh management, etc

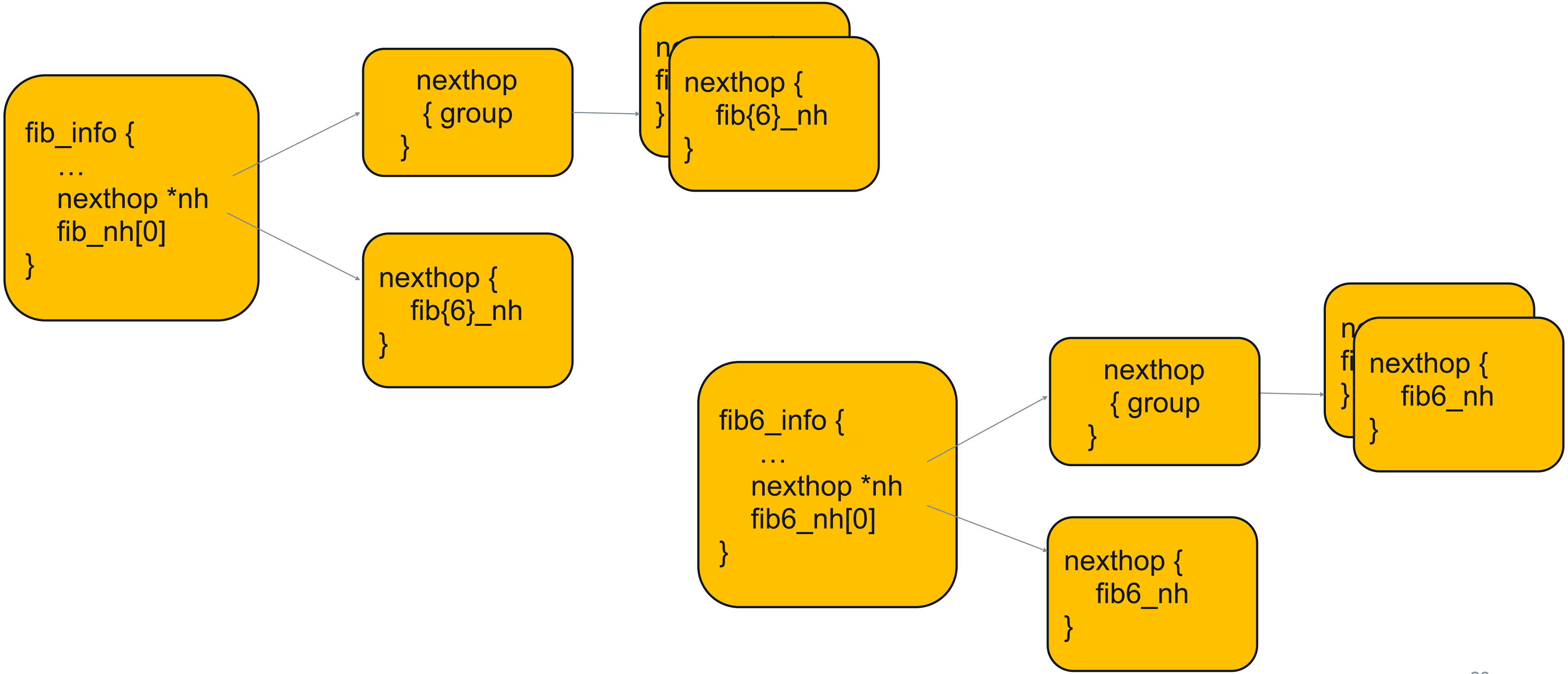
Nexthop Kernel Code

struct nexthop

- lists for tracking which FIB entries reference nexthop
- list for tracking which groups reference nexthop
- hash table tracking netdevice to nexthop objects

All of it is intended to be able to quickly correlate an event to a nexthop or vice versa

Kernel Data Structures



```
fib_info {
    …
    nexthop *nh
    fib_nh[0]
}
```

```
nexthop
  { group
}
```

```
nexthop {
    fib{6}_nh
}
```

```
nexthop {
    fib{6}_nh
}
```

```
fib6_info {
    …
    nexthop *nh
    fib6_nh[0]
}
```

```
nexthop
  { group
}
```

```
nexthop {
    fib6_nh
}
```

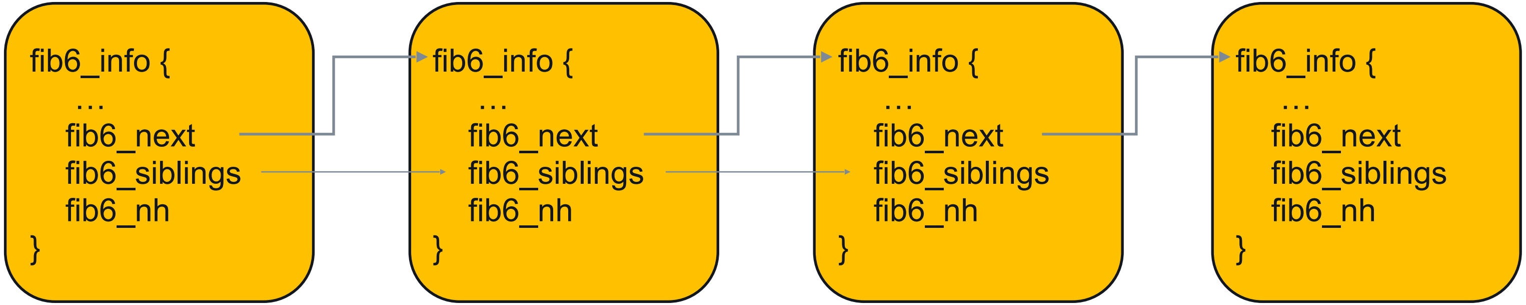
```
nexthop {
    fib6_nh
}
```

Nexthop Integration into IPv6

Code iterates over fib6_info

IPv6 multipath routes implemented as series of linked fib6_info

- Different from IPv4 where fib_info references an array of fib_nh (paths)



```
fib6_info {
    …
    fib6_next
    fib6_siblings
    fib6_nh
}
```

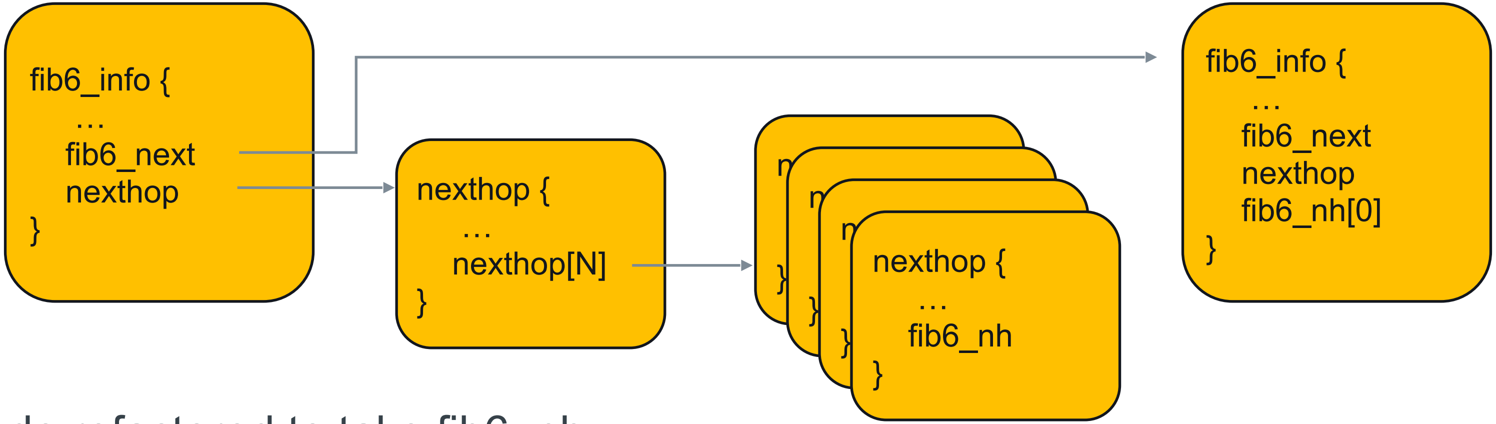
```
fib6_info {
    …
    fib6_next
    fib6_siblings
    fib6_nh
}
```

```
fib6_info {
    …
    fib6_next
    fib6_siblings
    fib6_nh
}
```

```
fib6_info {
    …
    fib6_next
    fib6_siblings
    fib6_nh
}
```

Nexthop Integration into IPv6

With nexthop objects, IPv6 multipath routes effectively become:



Code refactored to take fib6_nh

Updated to iterate over fib6_nh within a fib6_info

IPv6 does not quite align with IPv4 due to legacy implementation, but it is closer

Example Using iproute2

Basic nexthops

- ip nexthop add id 1 via 172.16.1.1 dev eth1
- ip nexthop add id 2 via 2001:db8::1 dev eth2

Example Using iproute2

Basic nexthops

- ip nexthop add id 1 via 172.16.1.1 dev eth1
- ip nexthop add id 2 via 2001:db8::1 dev eth2

Blackhole nexthop

- ip nexthop add id 3 blackhole

Example Using iproute2

Basic nexthops

- ip nexthop add id 1 via 172.16.1.1 dev eth1
- ip nexthop add id 2 via 2001:db8::1 dev eth2

Blackhole nexthop

- ip nexthop add id 3 blackhole

Multipath nexthop

- ip nexthop add id 101 group 1/2

Example Using iproute2

Basic nexthops

- ip nexthop add id 1 via 172.16.1.1 dev eth1
- ip nexthop add id 2 via 2001:db8::1 dev eth2

Blackhole nexthop

- ip nexthop add id 3 blackhole

Multipath nexthop

- ip nexthop add id 101 group 1/2

Route referencing nexthop object

- ip route add 192.168.1.0/24 nhid 101

Old to New API

Route vs nexthop

- ip route add 192.168.1.0/24 nexthop via 172.16.1.1 dev eth1 nexthop via 172.16.2.1 dev eth2

Old to New API

Route vs nexthop

- ip route add 192.168.1.0/24 **nexthop via 172.16.1.1 dev eth1** nexthop via 172.16.2.1 dev eth2

- **ip nexthop add id 1 via 172.16.1.1 dev eth1**

Old to New API

Route vs nexthop

- ip route add 192.168.1.0/24 nexthop via 172.16.1.1 dev eth1 **nexthop via 172.16.2.1 dev eth2**

- ip nexthop add id 1 via 172.16.1.1 dev eth1
- **ip nexthop add id 2 via 172.16.2.1 dev eth2**

Old to New API

Route vs nexthop

- ip route add 192.168.1.0/24 nexthop via 172.16.1.1 dev eth1 nexthop via 172.16.2.1 dev eth2

- ip nexthop add id 1 via 172.16.1.1 dev eth1
- ip nexthop add id 2 via 172.16.2.1 dev eth2
- **ip nexthop add id 101 group 1/2**

Old to New API

Route vs nexthop

- **ip route add 192.168.1.0/24** nexthop via 172.16.1.1 dev eth1 nexthop via 172.16.2.1 dev eth2

- ip nexthop add id 1 via 172.16.1.1 dev eth1
- ip nexthop add id 2 via 172.16.2.1 dev eth2
- ip nexthop add id 101 group 1/2
- **ip route add 192.168.1.0/24 nhid 101**

Benefits

Removes redundant processing on route add

- Already validated the nexthop gateway, device and LWT config

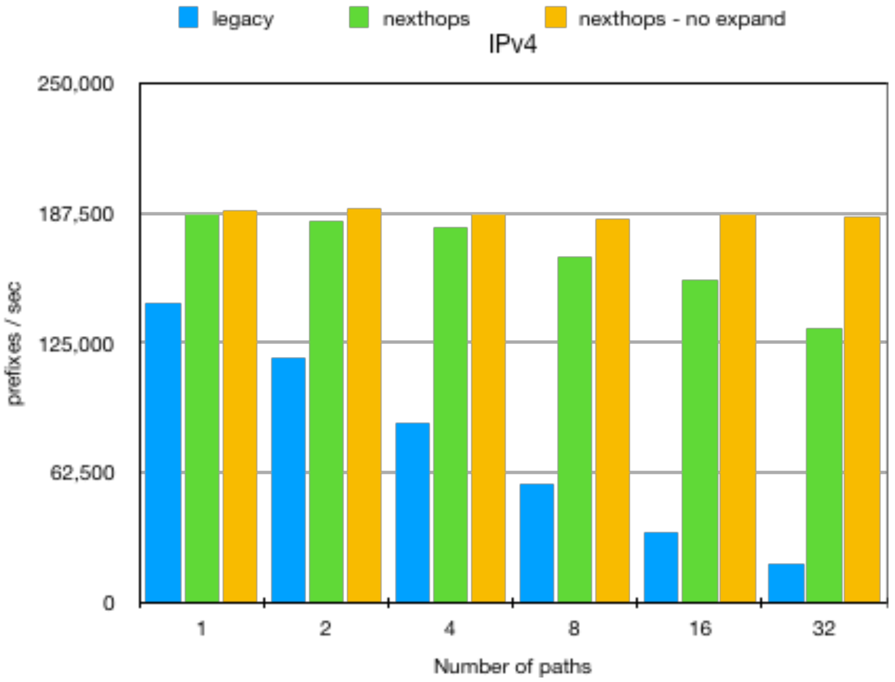
Opportunity to have better alignment across protocols

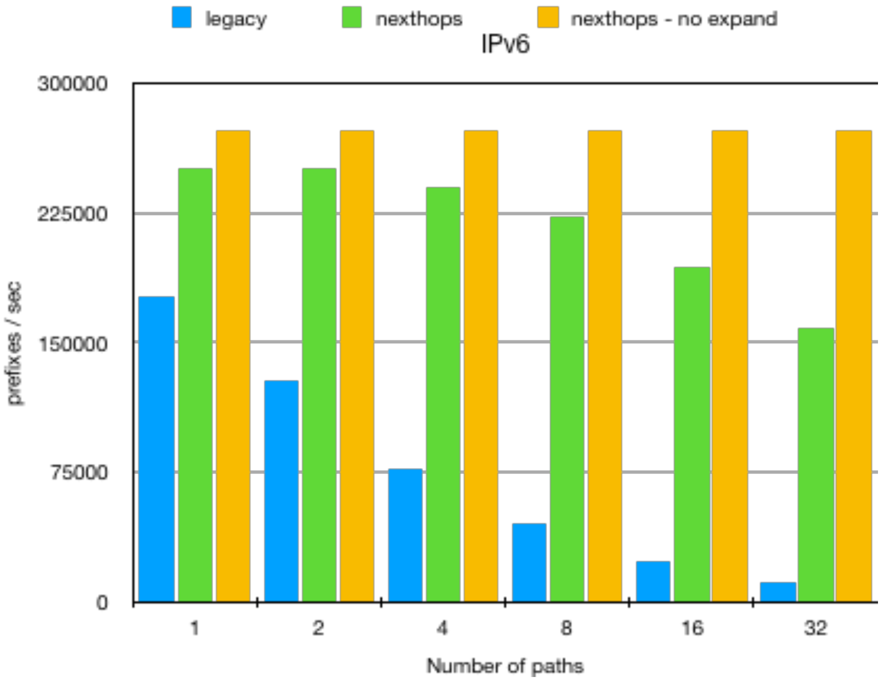
- Bring fib_info type efficiencies to IPv6 and MPLS
  - Better memory utilization
  - No duplicate nexthop checking

Alignment with hardware offload

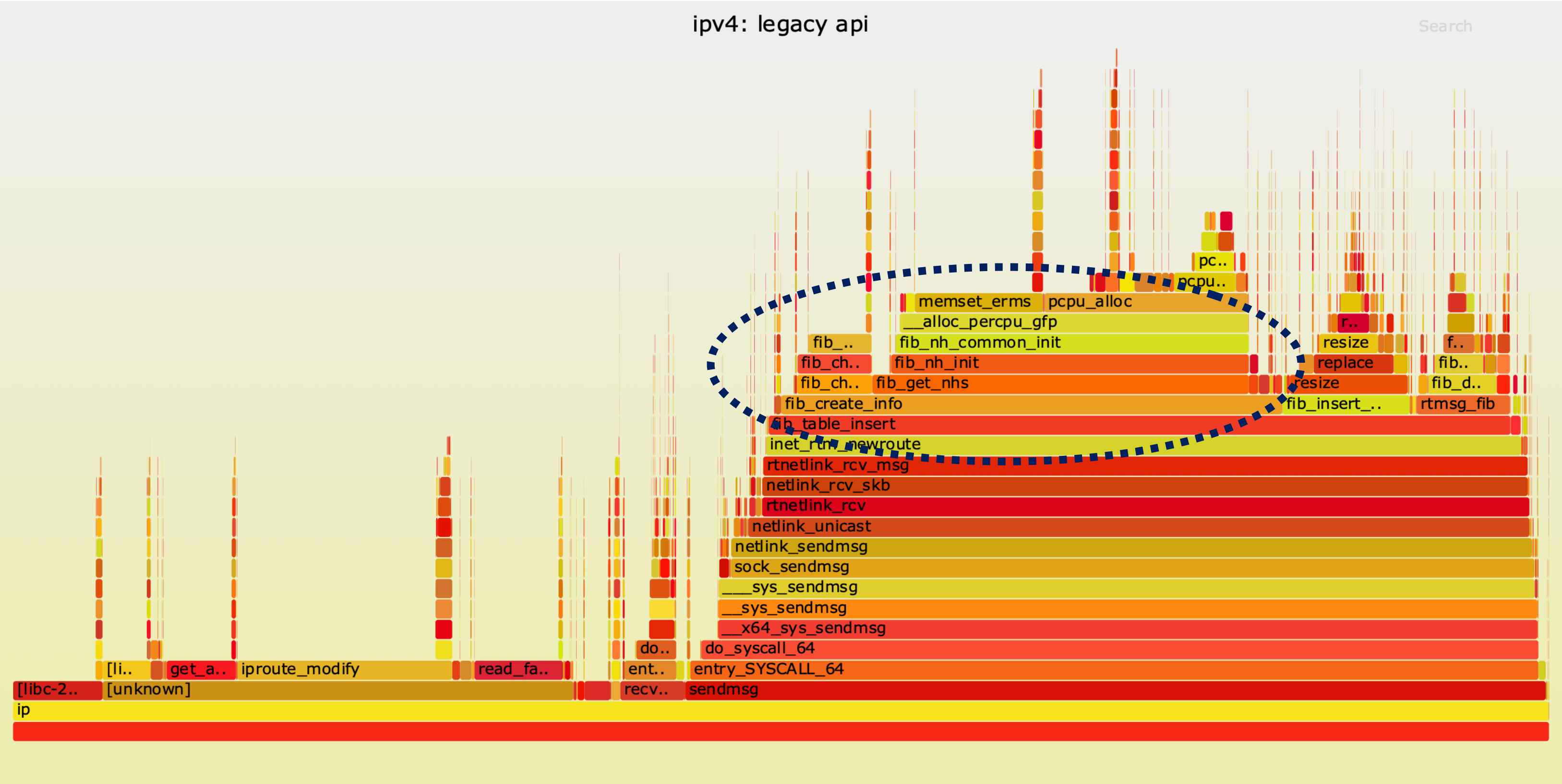
- Reduced burden on asic driver to map Linux objects to ASIC

Route Insertion Comparison

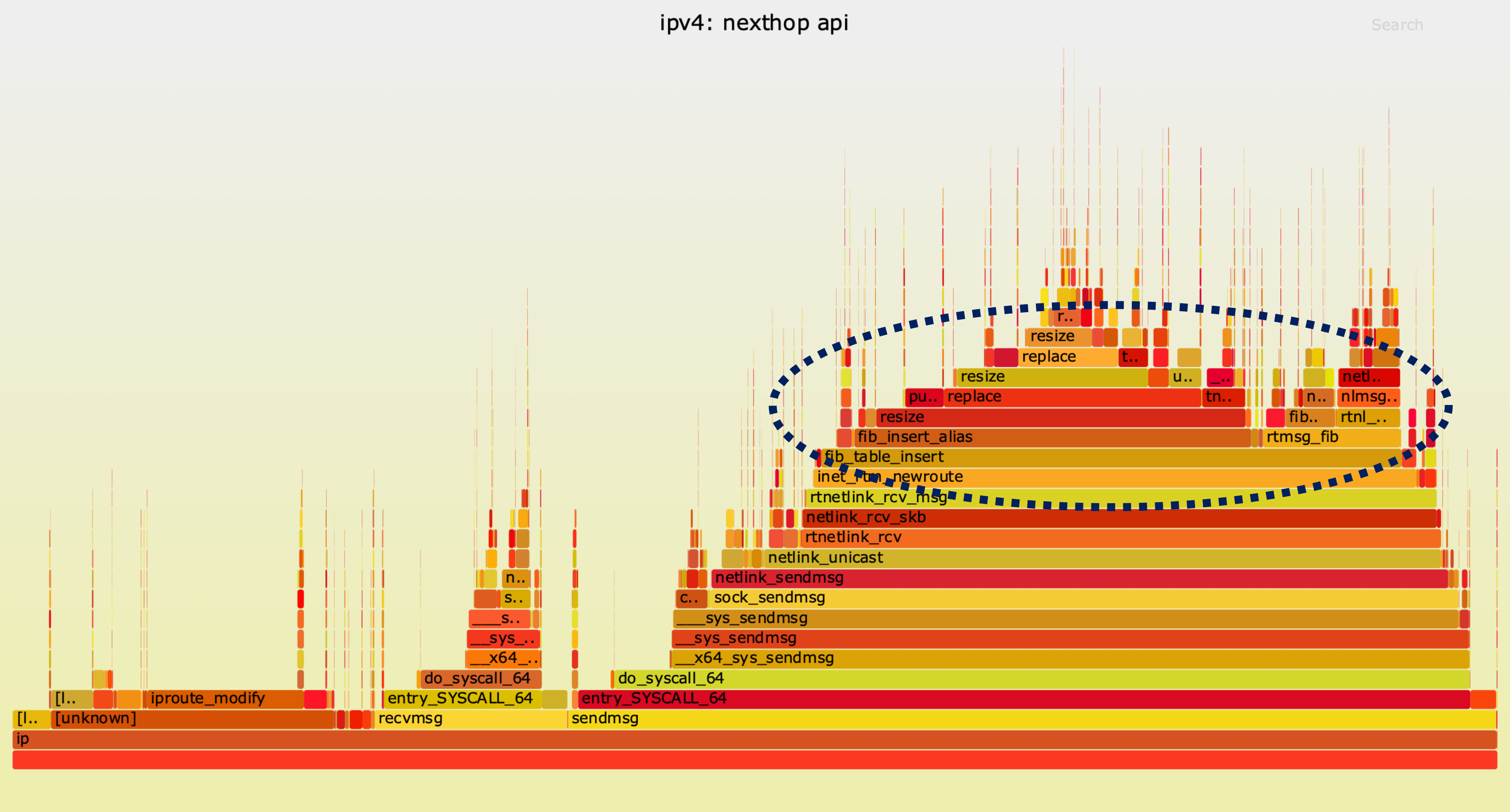




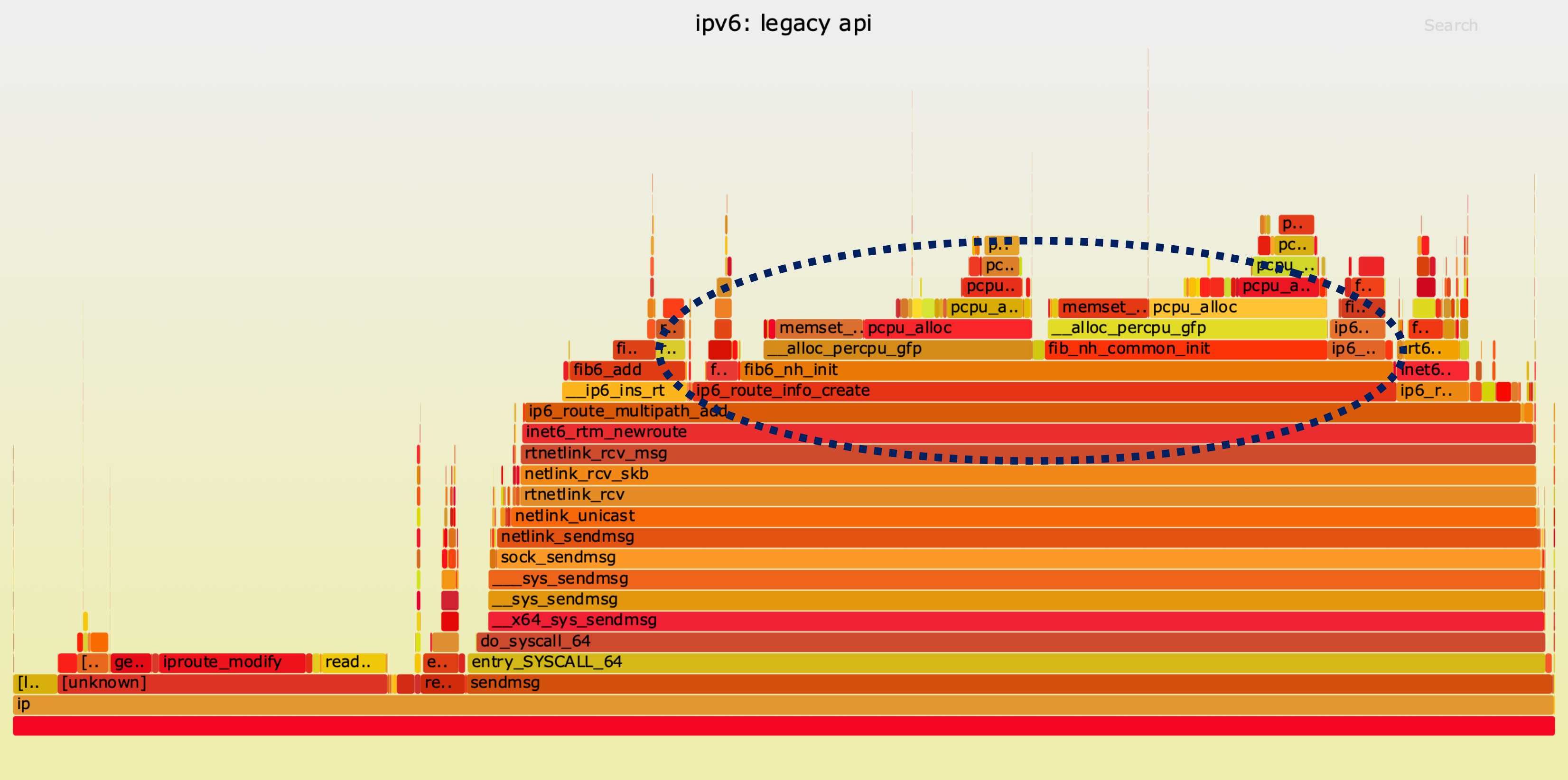
Flame Graph: IPv4 Legacy API



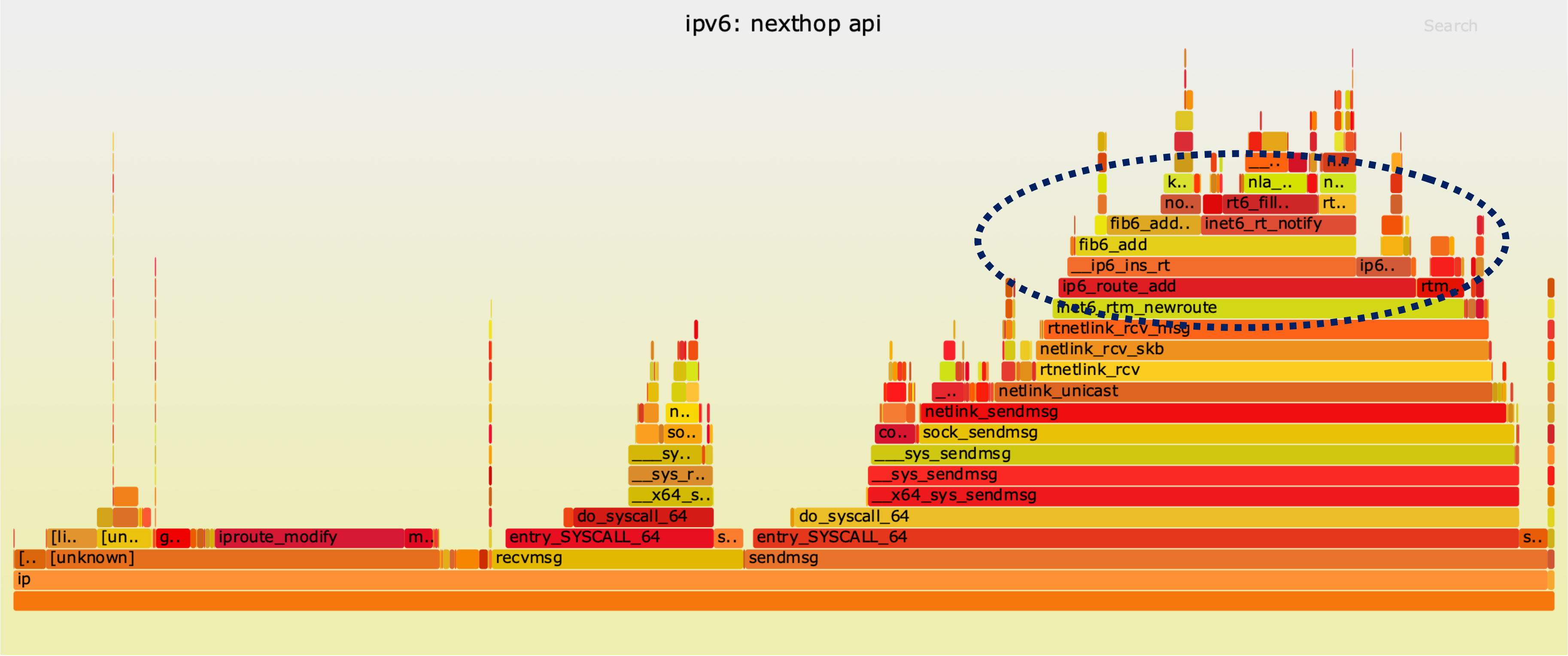
Flame Graph: IPv4 Nexthop API



Flame Graph: IPv6 Legacy API



Flame Graph: IPv6 Nexthop API



Faster Route Updates after Link Event

Legacy API routes have to be deleted/inserted or replaced one at a time

- N routes == N updates

Nexthop object can be updated without touching route entries

- Device, gateway, encap updated atomically
- Instantly updates all routes using nexthop
- 1 message to update N routes

RFC 5549

One objective of nexthop feature was to enable IPv4 routes with IPv6 nexthops

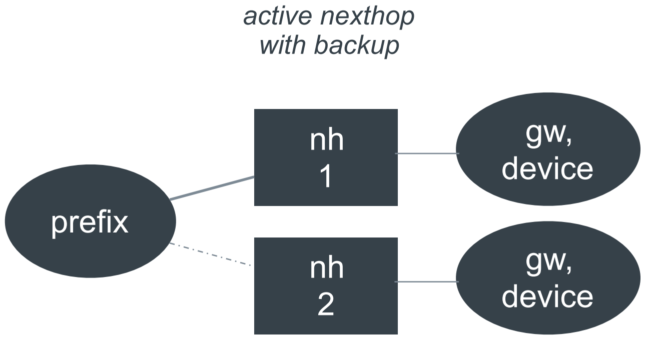
- Simplest implementation for BGP unnumbered

Objective of refactoring to use fib_nh_common

RTA_VIA instead of RTA_GATEWAY

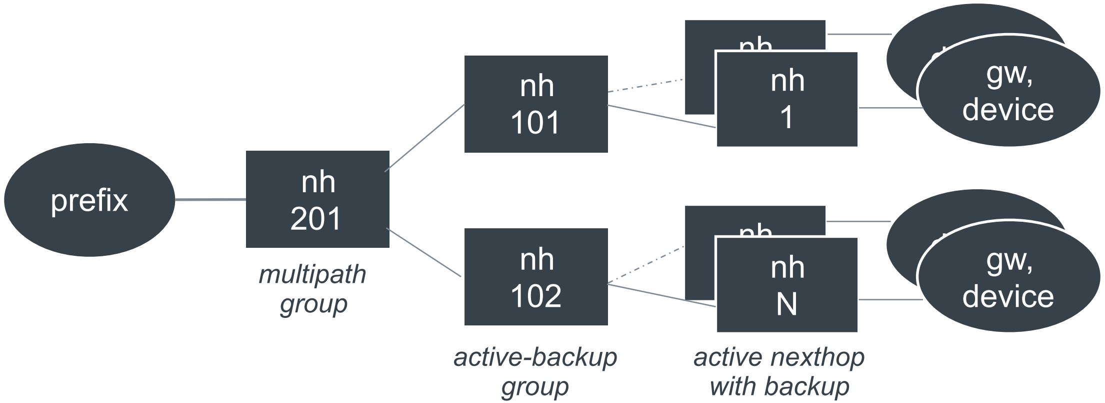
- ‘struct rtvia’ for the data; rtvia has address family followed by address
- this applies to IPv6 nexthop object with IPv4 route as well
- example: ip route add <prefix/len> nexthop via **inet6** <gw> dev <device>

Backup nexthop - aka, Fast Re-Routing



Backup nexthops

- Routing will use preferred nexthop if available
- per lookup atomic failover to backup



Status

Kernel version 5.2

- start of the refactoring for properly integrating nexthop objects
- IPv6 gateways with IPv4 routes (a.k.a., RFC 5549)

Kernel version 5.3

- remaining refactoring
- nexthop API

FRR

- initial support is in final testing – upstream soon
- initial support focused on correctness; room to improve
- 30% memory reduction

What's Next

Send patch for sysctl to opt out of backwards compatibility overhead

- Do not expand nexthop in route notifications

  Userspace relies on RTA_NH_ID

  Enables truly constant route management times

- Do not send route notifications on nexthop updates

  Nexthop notification should suffice for userspace

Add support for nexthop objects to MPLS code

Fast Re-Routing

- Someone with the time and interest should be able to add support for this fairly quickly

CUMULUS

Thank you!

Visit us at cumulusnetworks.com or follow us @cumulusnetworks