FRR Workshop
Oct 30

Donald Sharp
Dr Helen Chen
Dr Feng Xie

Agenda

- OSPFv3 over IPv4
- BGP Memory Optimizations
- FRR Infrastructure Changes
- FRR Releases

OSPFv3 over IPv4

I. Chen, F. Xie, and S. Abdallah

October 30, 2023

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD®

Evolution of OSPFv3

- OSPFv3 originally only supported IPv6 address family.
  - RFC5340 - OSPF over IPv6
- OSPFv3 standards enhanced OSPFv3 to carry both IPv4 and IPv6 address families over IPv6 transport.
  - RFC5838 - Support of Address Families in OSPFv3
- RFC7949 allows OSPFv3 to become the routing protocol of choice over both IPv4 and IPv6 transports.
  - RFC7949 - OSPFv3 over IPv4 for IPv6 Transition
  - An IP router can use OSPFv3 to replace OSPFv2 both in IPv4-only deployments and in mixed IPv4-IPv6 deployments

Motivations

- Many existing networks
  - Have an existing IPv4 core
  - Difficult to migrate to IPv6 due to the sizable task of renumbering
  - Rely upon relatively low-speed radio links
- Replacing OSPFv2 with OSPFv3 over IPv4 transport
  - Is an intermediate step for eventual migration to IPv6
  - Allows the use of new features standardized in OSPFv3
  - Enables reduction in overhead bandwidth when compared to OSPFv3 over IPv6 transport

Implementation

- Built a prototype implementation of RFC7949, based on FRR Release 7.3
- FRR provides an IETF standards-compliant implementation of OSPFv2/v3 with deployment experience both in commercial and tactical environments
- The modular design of OSPFv3 in FRR Release 7.3 makes the implementation of RFC7949 relatively straight forward.
- An indicator for the underlying IPv4 transport address was added to both the OSPFv3 interface and to the OSPFv3 neighbor structure.
- An AF_INET socket is added to handle OSPFv3 packets encapsulated in an IPv4 packet.
- When IPv4 is used as the transport mechanism, a different pseudo-header is used to calculate the OSPFv3 checksum and the corresponding AF_INET socket is used for transmitting and receiving OSPFv3 packets.
- The implementation improved FRR's sending of OSPFv3 packets by replacing the IPv6 link local address with an interface identifier. Replacing the address with the interface identifier increases modularity and reduces the potential for operational issues should an interface address change for any reason

Operational Considerations

- If the transport control is explicitly configured, either at the protocol/instance level or at the OSPFv3 interface level, the specified transport mechanism MUST be used.

- If an interface is not assigned an IP address required for the transport control configuration (except for unnumbered IPv4 interfaces), then the OSPFv3 interface becomes inactive.

- Removal of an IP address may result in de-activation of the corresponding OSPFv3 interface if the required transport configuration cannot be satisfied .

- Addition of an IPv4/IPv6 address may result in re-activation of the corresponding OSPFv3 interface if the required transport configuration can be satisfied.

- Changing the OSPFv3 transport at the process/instance level may result in activation/de-activation of the associated OSPFv3 interfaces depending on whether the new transport requirement on each interface can be satisfied.

- Changing the OSPFv3 transport at the OSPFv3 interface level may result in activation/de-activation of the OSPFv3 interface depending on whether the new transport requirement on the interface can be satisfied.

MITRE

Conclusions

- Use IPv4 as the transport for OSPFv3
- Facilitates the eventual migration
- Enables reduction in overhead bandwidth when compared to OSPFv3 over IPv6 transport
  - Extremely important for many mission-critical tactical deployments

Packet Analysis – PDU Byte Count Comparison

- n_1 is the number of LSAs
- n_2 is the number of LSAs requested
- n_3 is the number of LSAs acknowledged
- n_4 is the number of LSAs updated
- n_5 is the number of OSPF interfaces

Type	OSPFv2/IPv4	OSPFv3/IPv6	OSPFv3/IPv4
IP Header	20	40	20
OSPF Packet Header (OPH)	24	16	16
Authentication	Included in OPH	24	24
OSPF Hello Packet (OHP)	20	20	20
LSA Header (LH)	20	20	20
Database Description Packet (DBP)	OPH+8+n_1*LH	OPH+12+n_1*LH	OPH+12+n_1*LH
Link State Request Packet (LSRP)	OPH+ n_2*12	OPH+n_2*12	OPH+n_2*12
Link State acknowledgement Packet (LSAP)	OPH+n_3*LH	OPH+n_3*LH	OPH+n_3*LH
Link State Update Packet (LSUP)	OPH+n_4*LSA	OPH+n_4*LSA	OPH+n_4*LSA
Type-5 External LSAs	16	48	36
Type-4 Summary LSAs	8	12	12
Type-3 Summary LSAs	8	24	12
Type-2 Network LSAs	4	4	4
Type-1 Router LSAs	4 + n_5*8	4+n_5*16	4+n_5*16
OSPFv3 Link LSA	Not applicable	44	20
OSPFv3 Intra-Area-Prefix LSA	In Router LSA	52	12+8

MITRE

Six Node Topology Comparison

- Single area – No summary LSAs
- Packet capture from an OSPFv2 router of a network of six OSPFv2 routers
- Assumed the same sequence of events in both OSPFv2 and OSPFv3 over IPv4.
- Expect a 2% increase in protocol PDU total bytes for the duration of the run

Type	OSPFv2/IPv4	OSPFv3/IPv4	Change over OSPFv2
Hello Packet (count)	349	349	Not applicable
Hello Packet (byte)	30634	36218	+2%
Database Description Packet (count)	11	11	Not applicable
Database Description Packet (byte)	1170	1390	+2%
Link State Request Packet (count)	3	3	Not applicable
Link State Request Packet (byte)	246	294	+2%
Link State Acknowledgement Packet (count)	8	8	Not applicable
Link State Acknowledgement Packet (byte)	716	844	+2%
Link State Update (count)	39	39	Not applicable
Link State Update (byte)	7086	9258	+3%
Total (byte)	39852	48004	+2%

MITRE

OSPFv3 over IPv4 for IPv6 Transition

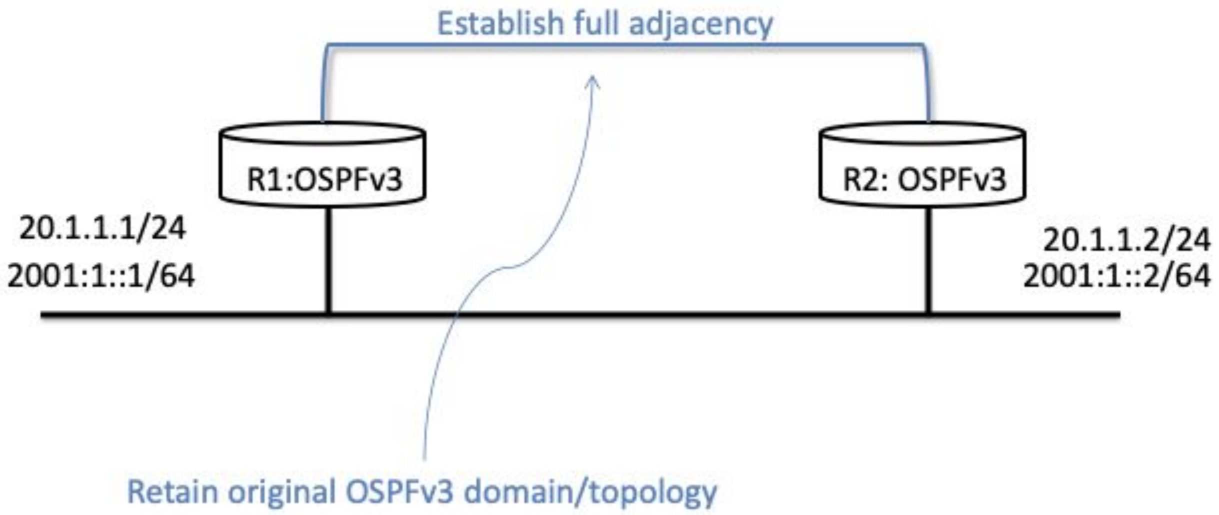
draft-chen-ospf-transition-to-ospfv3-00

I. Chen and A. Lindem

Applicability

- Existing approach
  - Carry IPv4 routes via OSPFv2
  - Carry IPv6 routes via OSPFv3
  - This increases operational complexity & cost because need to manage 2 IGPs at the same time for the same site
- OSPFv3 over IPv4 is an alternative for some
  - Use existing Address Family extension to carry both IPv4 & IPv6 prefixes in OSPFv3 at the same time
  - Start with OSPFv3/IPv4, later add IPv6 routes, and eventually transition to OSPFv3/IPV6
  - Lower operational complexity & cost for some sites
- This helps some sites with IPv6 transition

Transition to OSPFv3



OSPFv3 Packet in IPv4

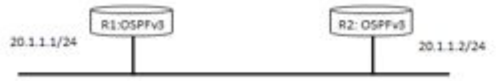
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   4   |  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live | Protocol 89   |         Header Checksum       |   IPv4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   Header
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       3       |     Type      |         Packet length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Router ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   OSPFv3
|                           Area ID                             |   Header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Checksum             |  Instance ID  |      0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         OSPFv3 Body ...                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Source Address

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   4   |  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live | Protocol 89   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

R1 sends OSPFv3 packets using source address 20.1.1.1

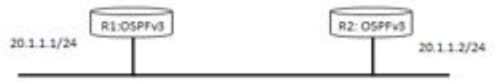
R2 sends OSPFv3 packets using source address 20.1.1.2



Destination Address

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   4   |  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live | Protocol 89   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Multicast packets
  - ALLSPFRouters (224.0.0.5) and ALLDRouters (224.0.0.6)
- Unicast packets
  - IPv4 address assigned to the receiving interface.
  - R1 sends to destination address 20.1.1.2.



Deployment

- Normally, IPv4 and IPv6 network topologies will be identical in an OSPF deployment
  a. OSPFv2 on IPv4 and OSPFv3 on IPv6
  b. OSPFv3 using single IP version for transport
- During transition
  – If IPv4 network is larger, then OSPFv3 on IPv4
  – If IPv6 network is larger, then OSPFv3 on IPv6
- Out of Scope
  – IPv4/OSPFv2 and IPv6/OSPFv3 islands
  – Other misconfigurations

BGP Memory Optimizations

- Why do we Care?
- BGP Peering Data Structure Reduction
- BGP Node -> dest breakup
- Input Queue/Output Queue for Peering
- bgp_path_info_extra cleanup
- ZAPI Buffer and BGP Announcement Cleanup

Why do we care?

- FRR runs on embedded systems
  - Limited memory in a bunch of cases
- No good way to give memory back
  - malloc_trim() has no good way to not block the entire program
- Good Progamatic hygiene

BGP Peering Data Structure Reduction

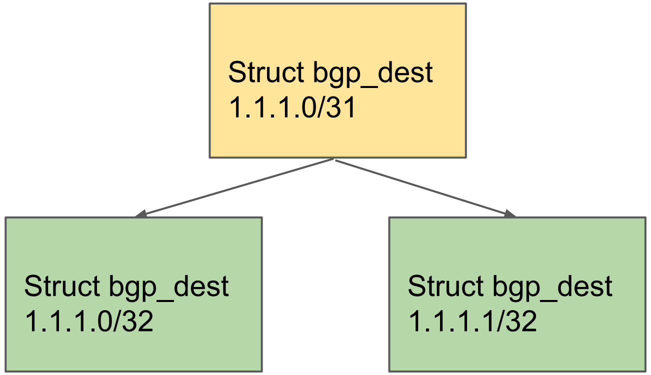
- peer->ibuf_scratch
  - Allocating 65535 * 10 for scratch space on a peer, can be bgp_read() private static data
- `struct bgp_synchronize`->withdraw_low just never used
- peer->sync[AFI_MAX][SAFI_MAX]
  - Never Used, 48 bytes * 4 * 8 per peer (1536 bytes)
- peer->obuf_work
  - Never Used, 66559 bytes
- peer->scratch
  - Never Used, 65535 bytes
- peer->last_reset_cause
  - Used but never approaches the 65535 allocated, allocate a stream structure when peer goes down instead just copy the stream data and use that.

BGP Peering Data Size Reduction

- 8.5 -> sizeof(struct peer) = 741976 bytes(!!!)
- 9.0 -> sizeof(struct peer) = 20600 bytes

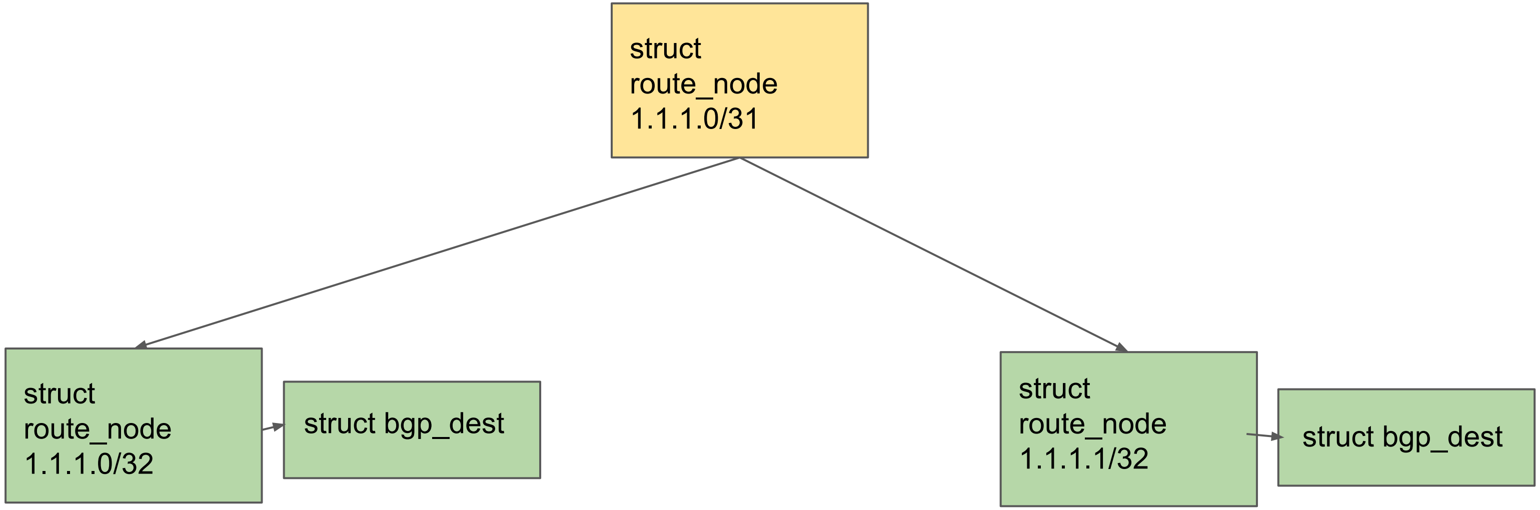
- Lesson Learned
  - Frequent review of what is used or not in our code base
- https://github.com/FRRouting/frr/pull/14075

struct bgp_dest breakup



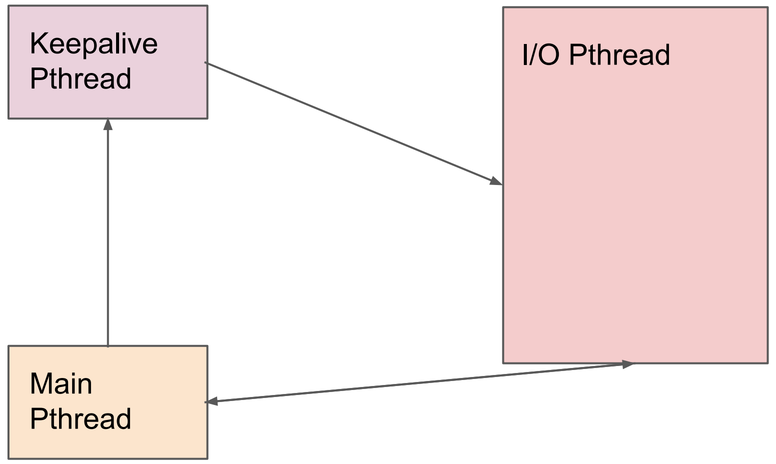
- Table data structure in FRR is a specialized binary tree that allows for handling route prefixes
- Parent node is always a shorter prefix that contains all the child nodes
- Top of the tree is always 0.0.0.0/0 or the v6 equivalent
- 200 bytes for each node in BGP
- Full BGP Feed has 1.4 million nodes, of which 900k are actual routes

struct bgp_dest breakup


struct route_node 1.1.1.0/31
struct route_node 1.1.1.0/32
struct bgp_dest
struct route_node 1.1.1.1/32
struct bgp_dest

- Each route_node info pointer points at the `struct bgp_dest` which contains the bgp specific data
- Place Holder Nodes now just use a `struct route_node`
- `struct route_node` is 120 bytes
- `struct bgp_dest` is 96 bytes
- Saves about 30mb per full BGP feed
- https://github.com/FRRouting/frr/pull/14118/

BGP Input/Output Queue Memory Optimizations



- BGP attempts to shove data in/out as fast as possible
- Full BGP feeds or interfaces that flap were causing millions of streams, Limit Incoming data to 10k packets at a time
- Outgoing Packets could also grow unbounded with any type of interface flapping going on
- connection->obuf_fifo for data output
  - Both Keepalive Pthread and Main Pthread generate packets
- connection->ibuf_fifo for data input
  - I/O Pthread reads data for processing in the master pthread

BGP Packet I/O Queue Optimizations

- Simple!
  - For Input Stop reading packets after X packets in read_ibuf_work(), then go do other work
  - For Output Stop generating packets in bgp_generate_updgrp_packets when limit is hit and go do other work

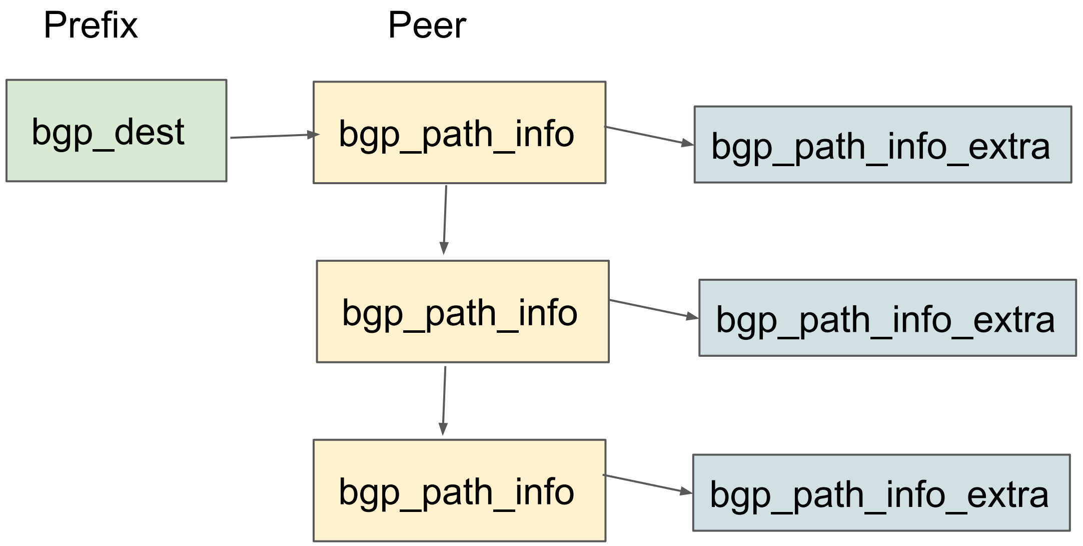
- https://github.com/FRRouting/frr/pull/12176 -> Input Queue
- https://github.com/FRRouting/frr/pull/12666 -> Output Queue

BGP I/O Packet Reduction Continued

```
janelle.pinkbelly.org# show bgp ipv4 uni summ
BGP router identifier 192.168.44.1, local AS number 64539 vrf-id 0
BGP table version 151714
RIB entries 286307, using 52 MiB of memory
Peers 4, using 2896 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
100.99.229.142  4      65011         4      2783        0    0    0 00:00:19            2   151714 N/A
192.168.119.205 4      23952         4      2690        0    0   55 00:00:19            3   151714 N/A
194.147.139.1   4     207465      5492         4        0 179927    0 00:00:19       151733        0 Martins feed
2a07:e480:2::2  4     207465      5708         5        0 314272    0 00:00:19            0        0 Martins feed

Total number of neighbors 4
```

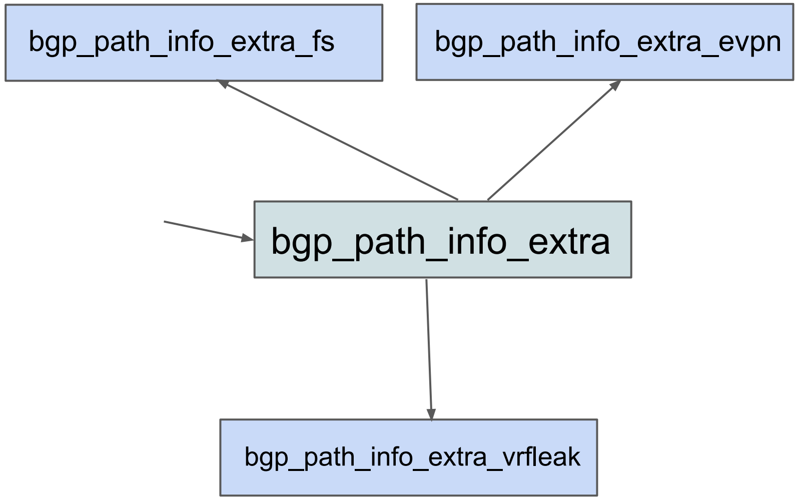
```
janelle.pinkbelly.org# show bgp ipv4 uni summ
BGP router identifier 192.168.44.1, local AS number 64539 vrf-id 0
BGP table version 479851
RIB entries 892018, using 82 MiB of memory
Peers 4, using 80 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
100.99.229.142  4      65011         5     25560   479851    0    0 00:01:14            2   479850 N/A
192.168.119.205 4      23952         5     25478   479851    0    0 00:01:15            3   479850 N/A
194.147.139.1   4     207465     51076         5   479851 10000    0 00:01:14       480418        0 Martins feed
2a07:e480:2::2  4     207465     51116         6   479851 10000    0 00:01:14            0        0 Martins feed
```

`struct bgp_path_info_extra` breakup



- `struct bgp_path_info` contains v4 basic path data
- `struct bgp_path_info_extra` is for other afi/safi's that BGP can talk
  - IPv6
  - EVPN
  - Flowspec
  - Labeled-Unicast
- It's a dumping ground for new NLRI data
- Everyone Pays for v4 data now
- A bgp_path_info will only ever describe one type of bgp route.

`struct bgp_path_info_extra`



- Break up different AFI/SAFI combinations into sub data and only allocate those pointers when needed.
- Add some pointers and extra memory allocation but savings is significant when not using those AFI/SAFI's
- 432 bytes for `struct bgp_path_info_extra` prior to this change
- 176 bytes for `struct bgp_path_info_extra` after this change
- https://github.com/FRRouting/frr/pull/14099
- There are more opportunities for optimization if needed

Zapi Buffer and BGP Announcement cleanup

- Zebra currently reads all the data it can over ZAPI and stores the individual ZAPI commands as streams.
- Processing this data is much more expensive than reading and storing in memory
- Full BGP feed dumping into zebra causes it to allocate huge amounts of memory
- This is exclusively a BGP/Zebra Communication issue in FRR

```
Memory statistics for zebra:
System allocator statistics:
  Total heap allocated:  241 MiB
  Holding block headers: 4100 KiB
  Used small blocks:     0 bytes
  Used ordinary blocks:  107 MiB
  Free small blocks:     57 KiB
  Free ordinary blocks:  134 MiB
  Ordinary blocks:       96552
  Small blocks:          618
  Holding blocks:        1
(see system documentation for 'mallinfo' for meaning)
--- qmem libfrr ---
Stream                        :       44 variable   3432352    15042 161243800
Stream FIFO                   :       45     72        3240       48      3456
```

ZAPI Buffer and BGP Announcement Cleanup Current

Zserv Pthread/Client

```
While Incoming Data:
    Grab packet
    Stuff in Stream
    Lock Mutex
    Place in zserv->ibuf_fifo
    Unlock Mutex
Signal Main Pthread to wake up
```

Main Pthread

```
Foreach zserv:
    Lock Mutex
    Grab 1000 streams on
    zserv->ibuf_fifo
    Unlock Mutex

Foreach stream Grabbed:
    Process each Stream
```

Zebra Fix

Zserv Pthread/Client

```
Look at count(zserv->ibuf_fifo)
to_process = 1000 -
count(zserv->ibuf_fifo)
While to_process || Incoming Data:
        Grab packet
        Stuff in Stream
        Lock Mutex
        Place in zserv->ibuf_fifo
        Unlock Mutex
        To_process -= 1
Signal Main Pthread to wake up
```

Main Pthread

```
Foreach zserv:
        Lock Mutex
        Grab up to 1000 streams on
        zserv->ibuf_fifo
        Unlock Mutex

Foreach stream Grabbed:
        Process each Stream
```

Zapi Message Buffering (BGP/Zebra Memory Bloat)

Zclient_send_message
- Call buffer_write
- On BUFFER_PENDING
  - Schedule wakeup via event_add_write
- On BUFFER_EMPTY
  - Data was sent

Buffer_write
- If WOULDBLOCK
  - Save data on buffer structure
  - Return BUFFER_PENDING
- Else Write data to zebra

```
Memory statistics for bgpd:
System allocator statistics:
  Total heap allocated:  1132 MiB
  Holding block headers: 35 MiB
  Used small blocks:     0 bytes
  Used ordinary blocks:  1128 MiB
  Free small blocks:     5008 bytes
  Free ordinary blocks:  3359 KiB
  Ordinary blocks:       31615
  Small blocks:          87
  Holding blocks:        9
(see system documentation for 'mallinfo' for meaning)
--- qmem libfrr ---
Type                          : Current#   Size       Total     Max#  MaxBytes
Buffer                        :        7     24         184        8       224
Buffer data                   :        1 variable      4120    20904  86124480
```

BGP Current Process For Route Installation

- Receives packet of data on bgp I/O pthread
  - Saved in a Fifo and Passed to BGP Master Pthread
- Master Pthread grabs stream data from Fifo, decodes the NLRI and prefix data to created basic data structures and saves data in appropriate tables
- Schedules Bestpath Running for prefixes read in
- BestPath processing wakes up and processes each prefix
- Once BestPath is run the routes are immediately installed into zebra

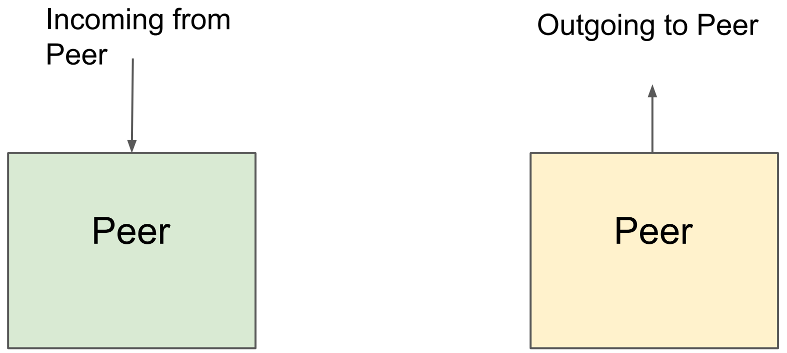
BGP zebra Install event creation

- Create a new fifo via typesafe node in `struct bgp_dest` called zi(short for zebra_item)
  - Head should be off of the `struct bgp_master`->zebra_announce
  - Can this be shared with the pq? < I am not sure >
- Create a new function bgp_zebra_schedule_announce
  - Insert bgp_dest into bm->zebra_announce fifo, lock the bgp_dest, if not already on the queue
    - If withdrawal flag is set, unset it
  - Set a new flag on the node `BGP_NODE_SCHEDULED_FOR_INSTALLATION_ZEBRA`
  - Schedule an event to handle the installations/withdrawals if not already scheduled
    - Should be cognizant of zclient->zebra_buffer_write_ready in that don’t schedule if bgp is waiting for this callback
- Create a new function bgp_zebra_schedule_withdrawal
  - Insert bgp_dest into bm->zebra_announce fifo
  - Set a new flag on the node `BGP_NODE_SCHEDULED_FOR_WITHDRAWAL_ZEBRA`
    - If installation flag is set, unset it
  - Schedule an event to handle the installations/withdrawals if not already scheduled
    - Should be cognizant of zclient_zebra_buffer_write_ready in that don’t schedule if bgp is waiting for this callback
- Add a new function bgp_zebra_buffer_write_ready to zclient->zebra_buffer_write_ready
  - Calls bgp_zebra_handle_announcements
- Create a new function bgp_zebra_handle_announcements
  - Pulls bgp_dest’s off of the bm->zebra_announce fifo
  - Sends the appropriate announce/withdrawal to zebra
  - UNSET’s appropriate flags and unlocks the nodes
  - If BUFFER_PENDING is received, note we are waiting for zebra_buffer_write_ready callback and stops processing of new nodes at this point in time else process another node, bgp_zebra_buffer_write_ready will be called when it’s time to proceed again
- On shutdown of bgp table, ensure that the node(s) are removed from the bm->zebra_announce FIFO, flags unset and unlocked

FRR Infrastructure Changes Coming

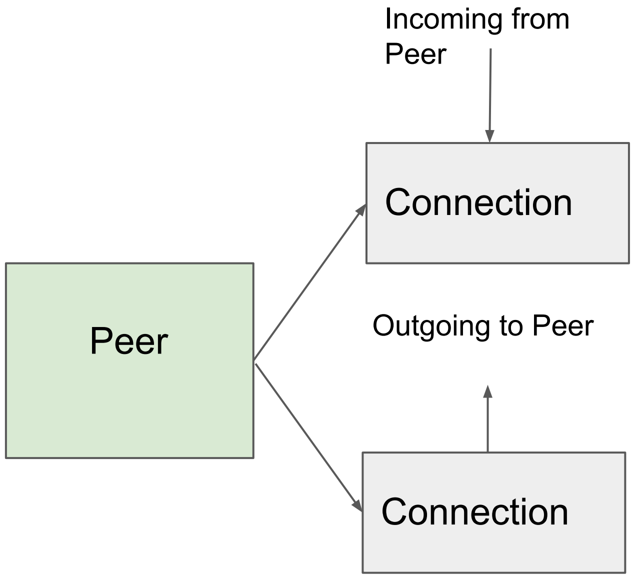
- BGP Connection/Peer Breakup
- BGP Bestpath Reorganization
- Zebra Dataplane Reorganization

BGP Connection / Peer Breakup



- Peer structure per incoming and outgoing connection at peer startup
- BGP has a state machine that dictates which connection wins
- Consolidate down to one peer once winner is decided
- Throws away loser data structure
- Problem
  - CLI consolidation
  - Connection data consolidation
  - Peer_xfer_conn
  - peer_xfer_config
  - BUGS BUGS BUGS

Solution for peer data structure



- Create a data structure for the connection
  - `struct connection`
  - FSM has to be transformed to connection oriented
    - Peer oriented to connection oriented
- Everything needs to be touched in BGP
- https://github.com/FRRouting/frr/pull/8790
  - Create `struct connection` and directly place in peer
  - Abstract alloc/free memory
  - Start passing connection around instead of peer
- https://github.com/FRRouting/frr/pull/14379
  - Continued finding of connection data move over
  - Abstract FSM to use connection
  - Convert Events for bgp_io.c bgp_packet.c and bgp_fsm.c to use a connection
- What’s Left?
  - Removal of peer_xfer_conn and peer_xfer_config functions after decision is made on which connection to keep

BGP Bestpath Reorganization

```
janelle.pinkbelly.org# show bgp ipv4 uni 40.30.10.0
BGP routing table entry for 40.16.0.0/12, version 1376540
Paths: (3 available, best #2, table default)
  Advertised to non peer-group peers:
  100.99.229.142 192.168.119.205
  23952 5 6 7 8 9 10 111 12
    192.168.119.205 from 192.168.119.205 (27.0.0.16)
      Origin IGP, metric 0, valid, external, rpki validation-state: not found
      Last update: Sun Oct 29 08:17:07 2023
  207465 6939 32787 4249, (aggregated by 4249 40.15.254.160)
    194.147.139.1 from 194.147.139.1 (194.147.139.1)
      Origin IGP, valid, external, atomic-aggregate, otc 6939, best (AS Path), rpki validation-state: not found
      Last update: Sat Oct 28 16:13:55 2023
  207465 6939 32787 4249, (aggregated by 4249 40.15.254.160)
    194.147.139.2 (inaccessible, import-check enabled) from 2a07:e480:2::2 (194.147.139.2)
      Origin IGP, invalid, external, atomic-aggregate, otc 6939, rpki validation-state: not found
      Community: 55:66
      Last update: Sat Oct 28 16:15:31 2023
```

BGP Bestpath Ordering

- Current Algorithm:
    - Receive new path information, push in new `struct bgp_path_info` to front of bgp_dest->info pointer, schedule best path selection via bgp_process()
    - BestPath calculation:(See bgp_best_selection() and bgp_path_info_cmp())
        - Set best = NULL
        - Foreach `struct bgp_path_info`:
            - Run bgp_path_info_cmp() If current is better path then best
                - Best = current
        - If Multipath is enabled
            - Foreach `struct bgp_path_info`:
                - If best == current continue;
                - Run bgp_path_info_cmp() if current can be used in mpath
                    - Save mpath data pointers for current

Changes to Best Path

- New Algorithm:
  - Receive new path information, store new `struct bgp_path_info` to front of bgp_dest->info pointer. Set UNSORTED flag to the new path info
  - Bestpath Calculation
    - While UNSORTED options
      - Grab first UNSORTED flag set equal to unsorted, pull off list
    - Find First `struct bgp_path_info` without UNSORTED flag set current equal to it
    - Foreach `struct bgp_path_info` starting at current:
      - Run bgp_path_info_cmp(unsorted, current)
      - If current is better
        - Current = current->next and continue
      - If unsorted is better
        - Place unsorted before current in list, store mpath info for this comparison

Changes to Best Path Continued

- For Mpath comparison:
  - Walk sorted list of best paths, stopping when mpath is no longer possible with comparisons, since the bestpath ordering reasons have been kept

- Discussion/Problems
  - Sorted Ordering Only really matters for any type of scaled peering
    - I'm looking at you Data Centers
  - Break up into series of 2 commits, Get bestpath sorted order right first, then come back through and get Multipath Working
  - Easy to prove correctness
  - Very hard to fix all the tests

Zebra DataPlane Cleanups

- Goal of adding the Dataplane api to zebra itself
  - FRR can talk to a variety of underlying data planes
    - Each can have their own requirements, should be able to pick up and place down a new one
    - Dataplanes can have different functionality, should be able to abstract to a generic way of talking about that functionality
  - FRR should have a pthread that is in charge of talking to the dataplane
    - Use the concept of a data plane context to allow for data communication such that memory locking is not necessary
  - FRR should be able to receive Asynchronous Data Plane updates about anything and process it accordingly

What zebra ended up with

- Dplane works with `struct dplane_contexts` that allow asynchronous discussion between the dataplane and Zebra
- 4 netlink sockets in zebra (see kernel_init())
    - Netlink -> Receives netlink notifications from the kernel based upon `groups`
    - Netlink_cmd -> Sends netlink commands as non batched data
    - Netlink_dplane_out -> Sends batching data to the kernel
    - Netlink_dplane_in -> Receives information based upon `dplane_groups`

```
groups = RTMGRP_IPV4_ROUTE | RTMGRP_IPV6_ROUTE | RTMGRP_IPV4_MROUTE |
         RTMGRP_NEIGH | ((uint32_t)1 << (RTNLGRP_IPV4_RULE - 1)) |
         ((uint32_t)1 << (RTNLGRP_IPV6_RULE - 1)) |
         ((uint32_t)1 << (RTNLGRP_NEXTHOP - 1)) |
         ((uint32_t)1 << (RTNLGRP_TC - 1));

dplane_groups = (RTMGRP_LINK            |
                 RTMGRP_IPV4_IFADDR     |
                 RTMGRP_IPV6_IFADDR     |
                 ((uint32_t) 1 << (RTNLGRP_IPV4_NETCONF - 1)) |
                 ((uint32_t) 1 << (RTNLGRP_IPV6_NETCONF - 1)) |
                 ((uint32_t) 1 << (RTNLGRP_MPLS_NETCONF - 1)));
```

What is needed to be done

- The Problem
  - Under heavy load event processing would handle context data from the dplane sockets before the reading of data of the master pthread sockets
    - https://github.com/FRRouting/frr/issues/13288 Is a great Example
- The Solution
  - Move all groups received via the Zebra Master Pthread to parsed into a context and passed up through the data plane
    - https://github.com/FRRouting/frr/pull/13396 Is the fix for #13288

FRR Releases

Since Last Workshop

FRR 8.4

- BGP SoO support
- BGP Prefix Origin Validation State Extended Community RFC 8097
- BGP Route Leak Prevention w/ Roles RFC 9234
- BGP BMP l3VPN support
- PIMv6 Support w/ MLD

https://frrouting.org/release/8.4/

https://frrouting.org/release/8.4.1/

https://frrouting.org/release/8.4.2/

FRR 8.5

- PIMv6 BSM support
- Deprecation of shell commands
- BFD integration with static routes
- TC Zapi implementation
- BGP Graceful shutdown per neighbor
- BGP Accumulated IGP Metric RFC 7311
- BGP Accept Own community Attr RFC 7611
- BGP SNMP BGP4v2-MIB support

https://frrouting.org/release/8.5/

https://frrouting.org/release/8.5.1/

https://frrouting.org/release/8.5.2/

https://frrouting.org/release/8.5.3/

FRR 9.0

- Mgmtd
- BGP ASN dot notation RFC 5396
- BGP Software Version
- BGP Peering over 127.0.0.0/8
- `match source-protocol` for BGP Routemaps
- Flex-Algorithm for SR-MPLS RFC 9350
- BFD Support for RIP
- Upgrade to libyang2.1.80
- BGP Memory Reduction

https://frrouting.org/release/9.0/

https://frrouting.org/release/9.0.1/

MGMTd Overview

- Consolidation and management of all Management data by a Single Entity
  - Can interface with CLI, Netconf, Restconf, Grpc etc.
- Better control over configuration validation, commit and rollback
- Faster Collection of configuration data
- Offload computation burden of YANG data parsing and validations of new configuration data away from individual daemons
- Improve performance of individual component daemons while loading large configurations or retrieving operational dataset

- For 9.0 release staticd is the converted daemon

Mgmtd Example

```
eva# conf t file-lock
eva(config)# ip route 1.2.3.4/32 192.168.119.1
eva(config)# do show ip route static
eva(config)# do show mgmt datastore-contents candidate
json
{
  "frr-routing:routing": {
    "control-plane-protocols": {
      "control-plane-protocol": [
        {
          "type": "frr-staticd:staticd",
          "name": "staticd",
          "vrf": "default",
          "frr-staticd:staticd": {
            "route-list": [
              {
                "prefix": "1.2.3.4/32",
                "afi-safi": "frr-routing:ipv4-unicast",
                "path-list": [
                  {
                    "table-id": 0,
                    "distance": 1,
                    "frr-nexthops": {
                      "nexthop": [
                        {
                          "nh-type": "ip4",
                          "vrf": "default",
                          "gateway": "192.168.119.1",
                          "interface": "(null)"
                        }]}}]}]}}]}}}
```

```
eva(config)# mgmt commit apply
eva(config)# do show ip route static
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

S>* 1.2.3.4/32 [1/0] via 192.168.119.1, enp13s0, weight 1, 00:00:03
```

BGP Software Version Capability

```
router bgp 64539
 neighbor 192.168.119.205 remote-as external
 neighbor 192.168.119.205 capability software-version
 ...
```

```
janelle.pinkbelly.org# show bgp ipv4 uni summ
BGP router identifier 192.168.44.1, local AS number 64539 vrf-id 0
BGP table version 1787622
RIB entries 1724972, using 158 MiB of memory
Peers 4, using 80 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
100.99.229.142  4      65011        46    150874  1787622    0    0 00:42:35            2   943657 N/A
192.168.119.205 4      23952        17    282465  1787622    0    0 00:00:14            0        0 FRRouting/9.2-dev
194.147.139.1   4     207465    219986        46  1787622  153    0 00:42:35       943655        0 Martins feed
2a07:e480:2::2  4     207465    353506        47  1787622  154    0 00:42:35       943655        0 Martins feed

Total number of neighbors 4
janelle.pinkbelly.org#
```

FRR 9.1 Upcoming

- BGP MAC-Vrf SoO support
- BGP `neighbor addpath-tx-best-selected X`
- BGP MPLS vpn LSR Redistribution
- PBR - pcp/vlan-id/vlan-tags filters
- ISIS - redistribute table
- ISIS - SRv6 Support
- OSPF prefix-suppression

How to Communicate with FRR Developers

- Frrouting.slack.com
  - Most developers are on slack and willing to talk
- dev@lists.frrouting.org
  - Email alias where development questions can be asked
- frog@lists.frrouting.org
  - Email alias to discuss operational concerns
- security@lists.frrouting.org
  - Email alias for discussion of security concerns

Open Microphone

Questions?

Thanks!