FRR Workshop

Donald Sharp - Stephen Worley
Donald Lee - Anuradha Karuppiah

Agenda

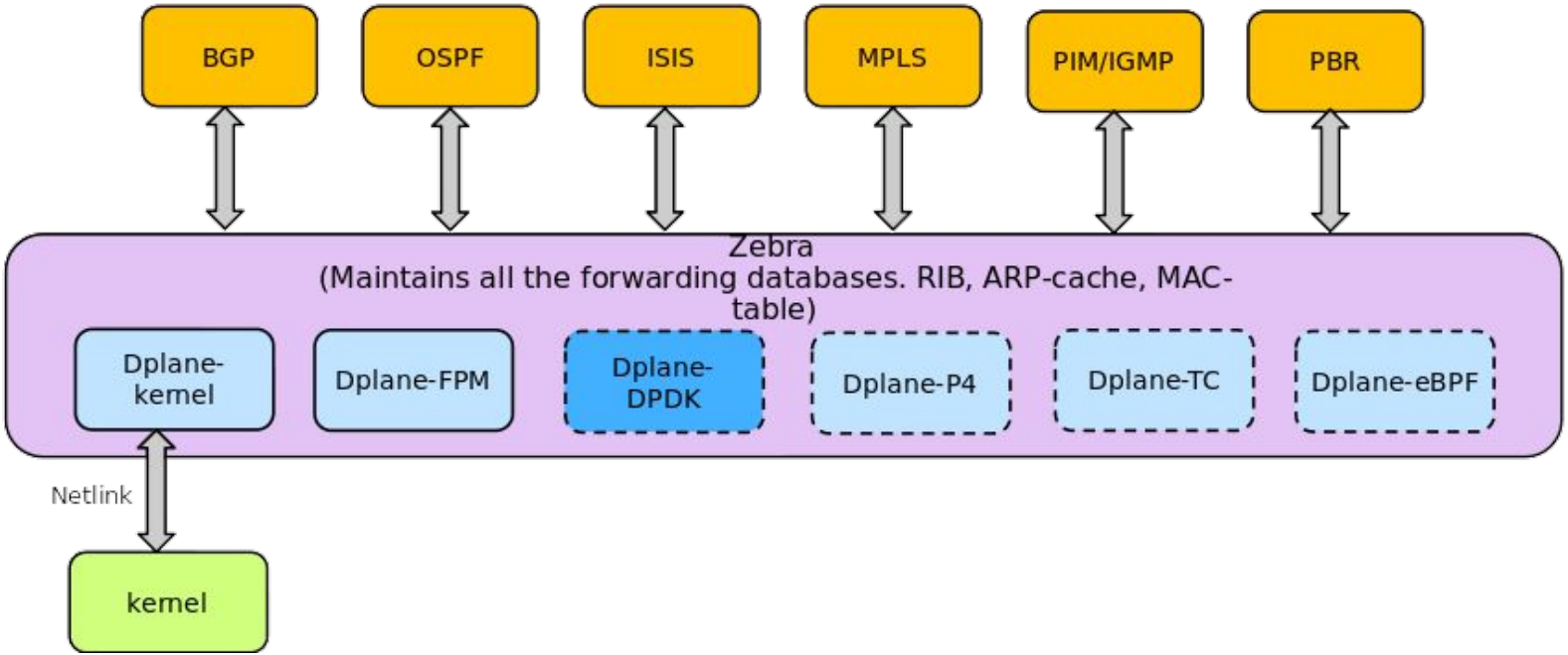
- FRR DPDK data plane plugin
- LUA Hook System
- EVPN Updates
- Backup Nexthops and FRR Integration
- FRR 8.0 Release
- Open Mic

FRR + DPDK plugin

Anuradha Karuppiah

FRR DATAPLANE PLUGINS

Modules for network-offload



DPLANE-DPDK PLUGIN

Validated on Bluefield2

Rise of DPUs have increased demand for routing-function offload

FRR dplane plugins can provide a batteries-included solution for line rate traffic forwarding on servers

- Targeted routing-function offload

What has been implemented

- PBR (with L3, L4 match and set) has been offloaded using the dplane-dpdk pugin
- Nvidia Bluefield2 platform was used for testing this offload

What is in the future

- Add L7 match to PBR for DPI offload
- Offload other routing-functions, primarily LPM, via the dplane-dpdk plugin
- Add other batteries-included dplane plugins for flow offload – P4, TC, eBPF

Lua Hook System

GSOC 2021 Student Project - Donald Lee

Introduction

Hi! I’m Donald Lee

Final year Computer Science student at the National University of Singapore

Tracks: Database Systems, Programming Languages (so networking is new for me)

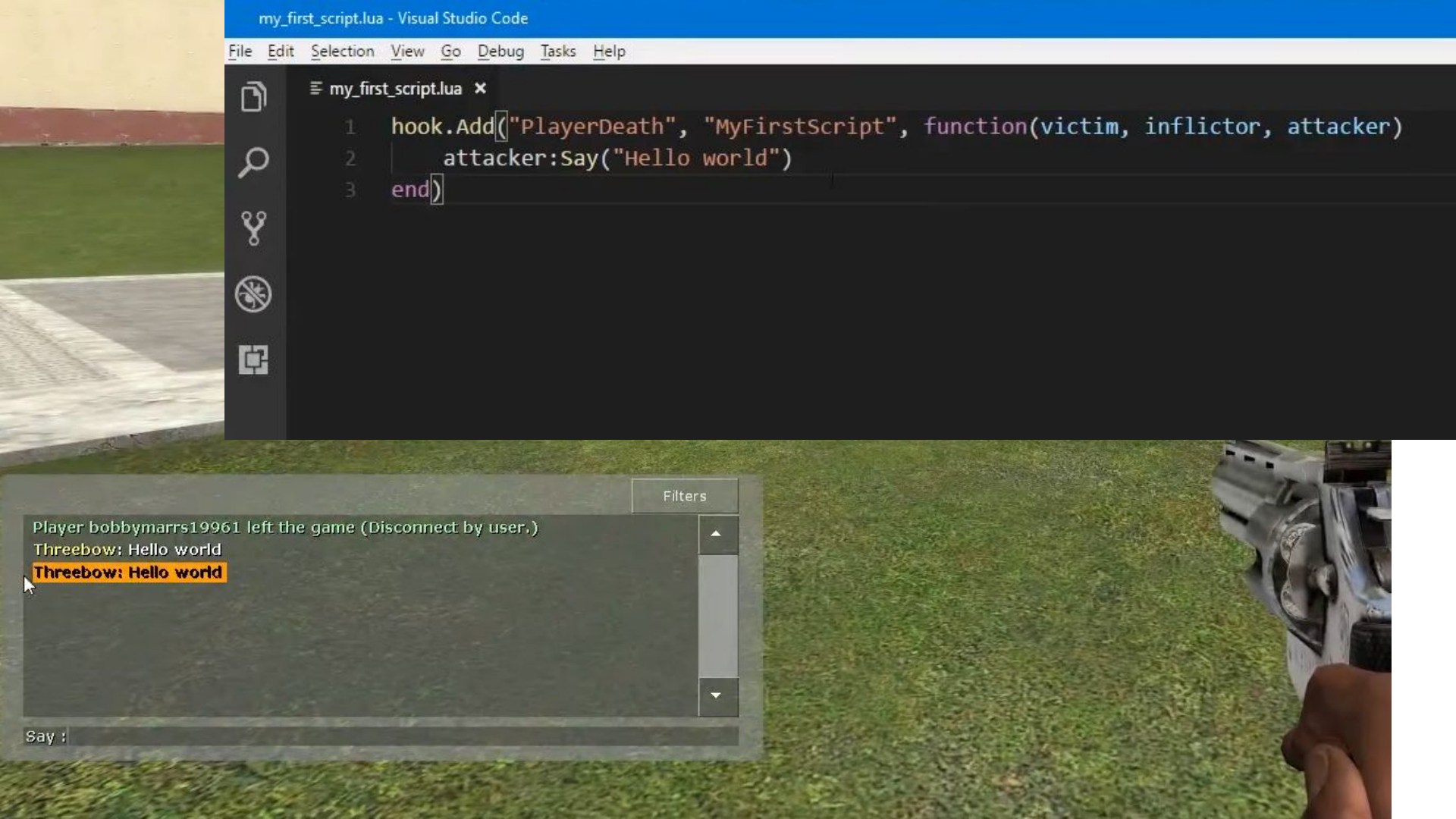
Why FRR: I wanted the chance to work on something low-level, related to PLs, systems programming.

Have you ever wanted to...

Dynamically implement logic for route maps?

Send a tweet when a route install fails?

Hook calls are your friend


my_first_script.lua - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
my_first_script.lua
1 hook.Add("PlayerDeath", "MyFirstScript", function(victim, inflictor, attacker)
2 attacker:Say("Hello world")
3 end)
Filters
Player bobbymarrs19961 left the game (Disconnect by user.)
Threebow: Hello world
Threebow: Hello world
Say :

Why a hook system?

Allows us to implement logic without changes to source code

Logging, accessing external services, extensibility

In a typesafe, transparent, configurable way

Why Lua?

Lightweight scripting language, cross platform

Several projects integrate Lua: Garry’s Mod, GTA V, DOTA 2, Apache HTTP

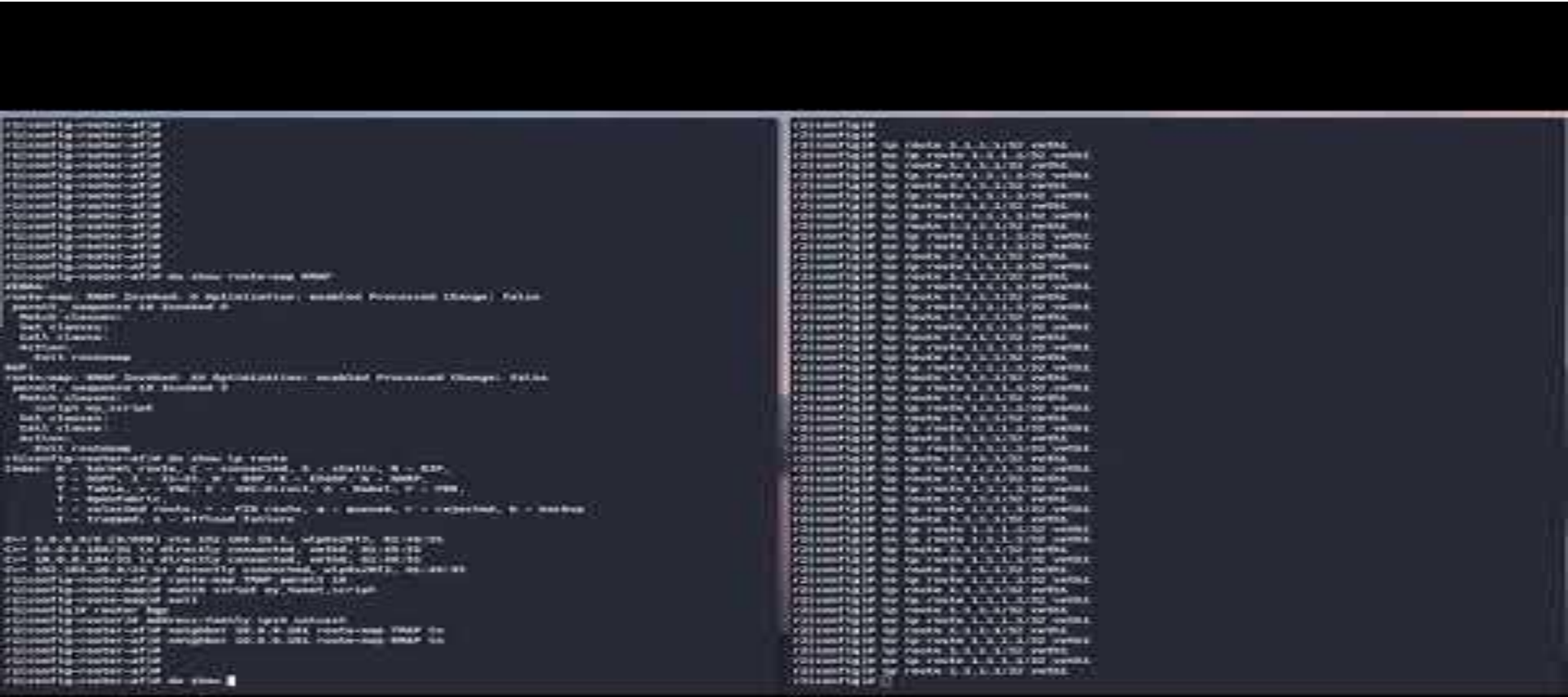
Where in FRR?

When a route is installed

BGP route map match

You decide!

Example: What can FRR users do with hook functions? (link)



Example hook call: BGP route map: when receiving a route from a peer

Want to match certain routes and set various parameters

Example:

route-map test permit 10

match script route_match

set local-preference 200

Lua:

```
function on_routematch(prefix, attrs, peer, ...)
  special_routes = { ["172.16.10.4/24"], … }

  if (special_routes[prefix.network]) then
    attrs[“metric”] = attrs[“metric”] + 7
  else ...
  return { action = }
end
```

Example: How can FRR devs implement hook calls?

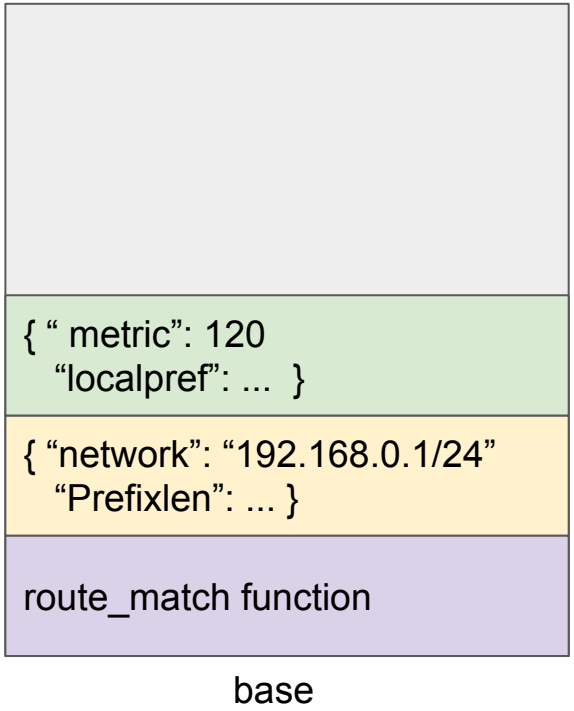
```
frrscript_new(“my_script”);

frrscript_load(fs, “route_match”, NULL)

frrscript_call(fs, “route_match”,
    ("prefix", prefix), ("attributes", &newattr) … )

frrscripte_get_result(fs, “action”, … )
```

1. Identify the file
2. Identify the function
3. Make the hook call*
4. Get results from hook function
5. Document

Example: How can FRR devs implement hook calls?

frrscript_call(fs, “route_match”,
("prefix", prefix), ("attributes", &newattr))

What really happens:
- Encode struct prefix * -> Lua table
- Encode struct attributes * -> Lua table
- Call …
- Decode Lua table -> struct attributes *
- Decode Lua table -> struct prefix *



Future

- vtysh command to show hooks
- Lua packages
- Logic to conditionally run hooks

- Current in-progress implementation: https://github.com/FRRouting/frr/pull/8982

EVPN Updates

Stephen Worley

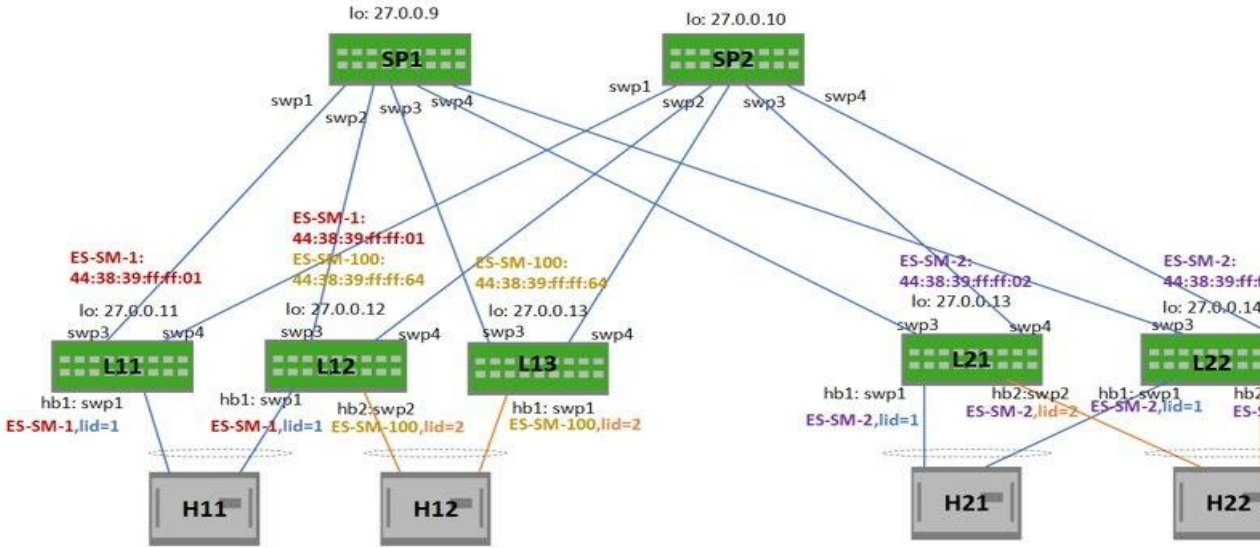
EVPN Multihoming

EVPN-MH - Motivation

- Replace MLAG in datacenter CLOS for active-active Vxlan use case
- MLAG:
  - Requires dedicated peer-link between switches
  - Proprietary control implementation (no standards)
  - Bond/link failures handled within rack
  -
- EVPN-MH:
  - Doesn’t require dedicated peerlink
  - Standards based, using BGP-EVPN route-types for control
  - Guarantee support >2 in a redundancy group (usually vendor specific)

EVPN-MH - Ethernet Segments

- Group of links connected to the same server (L11:swp1 L12:swp1)
- 10-byte Ethernet Segr ID (ESI)
- Each ES is a distinct redundancy group, not how H11 and H12 are multihomed to 3 differe switches/ ESI's

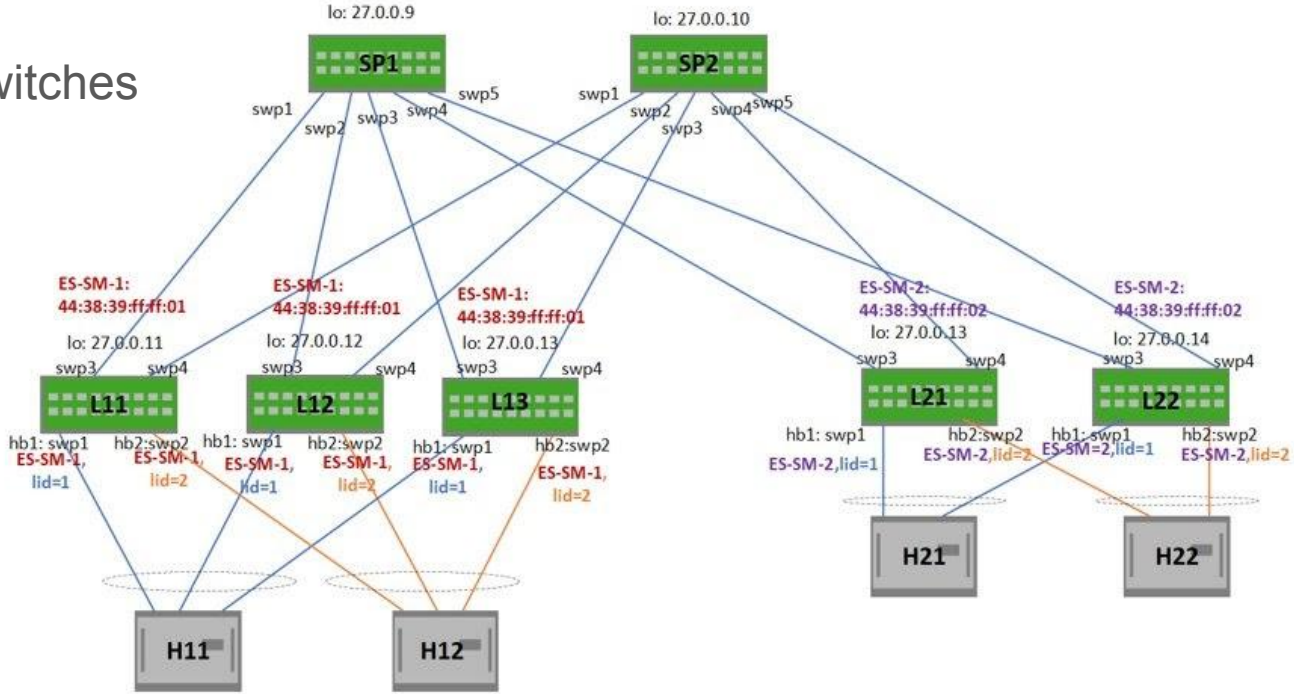


EVPN MH – peering per ES

Four Ethernet Segments: ES-SM1:1, ES-SM100:2, ES-SM2:1, ES-SM2:2

EVPN-MH - Ethernet Segments

- Spanning > 2 switches

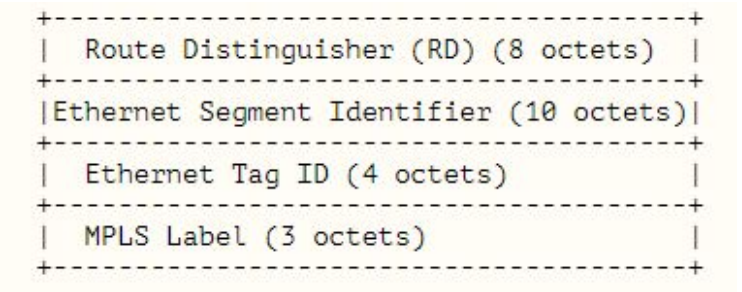


EVPN MH – three peers

Four Ethernet Segments: ES-SM1:1, ES-SM1:2, ES-SM2:1, ES-SM2:2

EVPN-MH - Route Type 1 Ethernet Auto Discover (EAD)

- Used for discovering all ESs
- Carries ES id
- Two types:
  - EAD-per-ES - one route is generated per-ES by the PEs attached to the ES
  - EAD-per-EVI - one route is generated per-ES, per-VNI by the PEs attached to the ES
- EAD-ES and EAD-EVI routes are imported by all PEs in the POD
  - Due to this, every PE maintains state about all ESs and can react to link failures on remote ESs



EVPN-MH - ES Configs

```
!
interface hostbond1
 evpn mh es-id 1
 evpn mh es-sys-mac 00:00:00:00:01:11
!
interface hostbond2
 evpn mh es-id 2
 evpn mh es-sys-mac 00:00:00:00:01:22
!
```

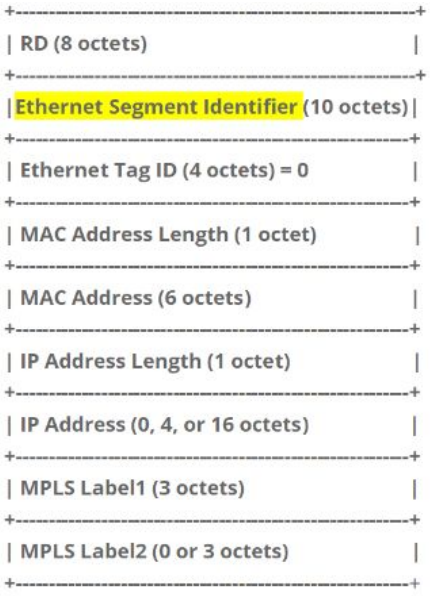
```
!
interface swp3
   alias to-spine1
  evpn mh uplink
!
!
interface swp4
   alias to-spine2
  evpn mh uplink
!
```

EVPN-MH - Route Type 2 as SYNC Routes

- Type-2 routes are used for FDB and neigh syncing between ES-peers
  - Every redundant PE processes the route to stay in sync
- Use ES-ID as destination
- Remote or SYNC routes
  - Remote: ES-ID is not local, fdb entry will point to network port
  - SYNC: ES-ID is local (from redundant PE also attached to this server), fdb entry will point to local access port
- SYNC neigh/fdb entries are proxy advertised to handle failures
  - Basically as long as one PE still has the ES, we will keep advertising even if it’s a SYNC entry

EVPN-MH - Route Type 2 as SYNC Routes

- This was always 0 before in type-2 routes, now it will be set with the ES-ID for multihomed networks



EVPN-MH - L2/L3 Kernel Nexthop Groups

- EVPN-MH is the first feature in FRR to make full use of kernel NHGs
- Its control plane process fits perfectly with the decoupling of routes and their nexthops (really just ESs in EVPN-MH’s case)

EVPN-MH - L2 Nexthop Groups

- A database of ESs in each PE is maintained from Type1 (EAD) routes from which we can infer vtep IPs from.
- We build L2 NHGs from this database, VNI agnostic
  - Every ES-ID is associated with a specific NHG
- Since Type 2 routes just use the ES-ID as a dest, from that we point to the appropriate L2 NHG / FDB entries.
  - MAC => ESI
  - ESI => NHG
  - NHG => [VTEP-1, VTEP-2 etc.]

EVPN-MH - L2 Nexthop Groups

- FDB

```
root@mlx-3700c-07:mgmt:~# bridge -d fdb |grep 00:21:
00:21:00:00:00:01 dev vx-1000 vlan 1000 extern_learn master bridge
00:21:00:00:00:01 dev vx-1000 nhid 536870918 self extern_learn
root@mlx-3700c-07:mgmt:~#
```

- L2 NHG

```
root@mlx-3700c-07:mgmt:~# ip nexthop show id 536870918
id 536870918 group 268435461/268435467/268435465 fdb
```

```
root@mlx-3700c-07:mgmt:~# ip nexthop show id 268435461
id 268435461 via 27.0.0.24 scope link fdb
root@mlx-3700c-07:mgmt:~# ip nexthop show id 268435467
id 268435467 via 27.0.0.25 scope link fdb
root@mlx-3700c-07:mgmt:~# ip nexthop show id 268435465
id 268435465 via 27.0.0.26 scope link fdb
root@mlx-3700c-07:mgmt:~#
```

EVPN-MH - L3 Nexthop Groups (Symmetric Routing)

- Two Levels of ECMP at L3
- Overlay (in tenant VRF):

```
11.0.101.21 nhid 75000028 table vrf1 proto bgp metric 20

id 198 via 27.0.0.21 dev vlan4001 scope link proto bgp onlink
id 216 via 27.0.0.22 dev vlan4001 scope link proto bgp onlink
id 75000028 group 198/216 proto bgp
```

- Underlay (in default VRF):

```
27.0.0.22 nhid 209 proto bgp metric 20
```

- An L3 NHG is created per-ES/per-tenant VRF to handle the fast failover requirements

EVPN-MH - Fast Failover

- Redundancy and speed
- Triggers:
  - Access port (link between TOR and host) failure
  - TOR/VTEP reboot (including planned switch upgrades)
  - Uplink failure
- Fast Failover via:
  - L2/L3 NHG use
  - ***ES bond redirect for rack-local traffic failover via the VxLAN overlay
    - Coming soon to a kernel near you?

EVPN-MH - Remote Bridge Traffic Failover

- Reminder: L2 NHG is maintained per ES
- On access port/ES-link failure detection by a PE, it sends a single EAD-ES route withdraw
- Withdraw is processed by all other PEs and they update their associated L2 NHG:
  - NHG-ES-2:1 [~~27.0.0.21~~, 27.0.0.22]
- All PEs have failed over to new group
- Note: 1 update into the fabric and 1 update into each PEs' dataplane. Always, no matter the # of Hosts connected to that failed link. No individual MACs updated

EVPN-MH - Remote Routed Traffic Failover

- Reminder: A L3-NHG is maintained per-ES/per-tenant-VRF
  - id 198 via 27.0.0.21 dev vlan4001 scope link proto bgp onlink
  - id 216 via 27.0.0.22 dev vlan4001 scope link proto bgp onlink
  - id 75000028 group 198/216 proto bgp
- On access port/ES-link failure detection by a PE, it sends a single EAD-ES route withdraw
- Withdraw is processed by all other PEs and they update their associated L3 NHG:
  - NHG-75000028 [~~27.0.0.21~~, 27.0.0.22]
- All PEs have failed over to new group
- Note: 1 update into the fabric and 1 update into each PE's dataplane. Always, no matter the # of Hosts connected to that failed link. No individual routes updated

Enhancements to L3 NHG API in FRR

- EVPN MH first to make full use of it but could be extended to any protocol
- Added new ZAPI messaging for NHG_ADD and NHG_DEL to handle add/del/replace and NHG_ID for ROUTE_ADD.
- Each daemon “owns” a cut-out ID space to provision NHGs with. It is their job to create, modify, and delete those. Zebra doesn’t delete or add them without ZAPI.
- A callback is sent to the daemon on creation.
- Daemon can then send a route down with the NHG_ID only rather than a full group. Zebra installs into kernel with this ID.
- Single NHG_ADD to replace group with new one and send single update to kernel for N routes.

SHARP Demo L3 NHG Failover

- https://asciinema.org/a/VHzxMjE6BCYE0qa6h8BdVnZMi

```
!
nexthop-group red
 nexthop 1.1.1.1 dummy1
 nexthop 1.1.1.2 dummy2
 nexthop 1.1.1.3 dummy3
 nexthop 1.1.1.4 dummy4
 nexthop 1.1.1.5 dummy5
 nexthop 1.1.1.6 dummy6
 nexthop 1.1.1.7 dummy7
 nexthop 1.1.1.8 dummy8
 nexthop 1.1.1.9 dummy9
 nexthop 1.1.1.10 dummy10
 nexthop 1.1.1.11 dummy11
 nexthop 1.1.1.12 dummy12
 nexthop 1.1.1.13 dummy13
 nexthop 1.1.1.14 dummy14
 nexthop 1.1.1.15 dummy15
 nexthop 1.1.1.16 dummy16
!
```

echo 0 > /proc/sys/net/ipv4/nexthop_compat_mode

Coming Soon

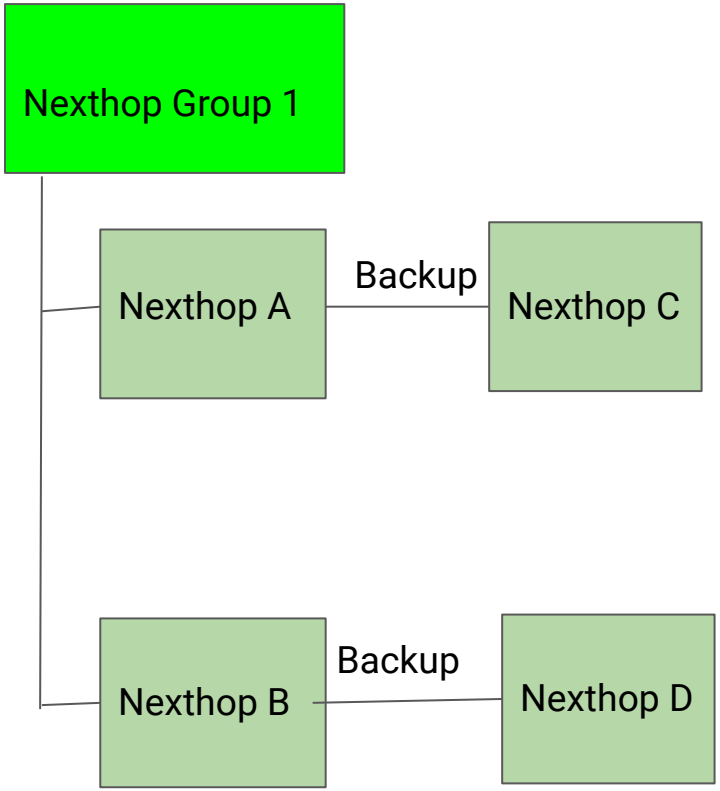
- Single Vxlan Device Support
  - Set of vnis are encapsulated in a single device model
  - With single vxlan device there would be only a single vxlan device say for example vxlan0 which would have set of attributes pertaining the vxlan construct.
  - Individual vnis now are represented as a vlan-vni mapping where user can specify what vlans map to its associated vnis
- Downstream VNI
  - Enables you to assign a VNI from a downstream remote VTEP through EVPN routes instead of configuring layer 3 VNIs globally across the network.
  - Primary use case: access to shared services
  - 50.1.2.31 **encap ip id 204003** src 0.0.0.0 dst **110.0.0.3** ttl 0 tos 0 dev vxlan0 table vrf-BLUE **110.0.0.3** dev vxlan0 lladdr 00:02:00:00:00:23

Backup Nexthops and FRR Integration

Donald Sharp

Motivation

- Backup Paths are used in Several domains
  - IP-FRR
  - TI-LFA
  - BGP PIC
  - MPLS-FRR
- Precompute what to do when something `bad` happens




```
eva# show ip route
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

K>* 0.0.0.0/0 [0/100] via 192.168.1.1, enp39s0, 5d23h10m
K>* 0.0.0.0/1 [0/0] via 172.27.234.1, tun0, 19:46:27
C>* 1.2.3.4/32 is directly connected, lo, 5d23h10m
D>* 4.5.6.7/32 [150/0] via 192.168.1.1, enp39s0, weight 1, backup 0, 00:00:03
  b                    via 192.168.1.40, enp39s0, weight 1
```

Design

- IP Routes and MPLS LSP’s will support backup nexthops
- Individual nexthops can have 1 to 8 backup nexthops, with a limit of 256 backup nexthops in total
- ZAPI is extended to allow upper level protocols to pass backup nexthops
- Zebra resolves each backup nexthop
- Reachability (NHT/Redistribution) will include sending backup nexthop information to interested daemons
- LSP Struct is extended to include a backup NHLFE list
- Backup Nexthops are presented to the dataplane for processing

https://github.com/mjstapp/frr/wiki/Backup-nexthop-design

What Code Uses It?

- Base Implementation
  - https://github.com/FRRouting/frr/pull/6765
- OSPF TI-LFA
  - https://github.com/FRRouting/frr/pull/7127
- This code will be in 8.0
- How does it work without a linux kernel implementation?

What is next?

- Need Kernel Implementation
  - https://lore.kernel.org/netdev/20200610034953.28861-1-dsahern@kernel.org/
- Just need to tell FRR to program the kernel when backup nexthops become available

FRR 8.0 Release

What is FRR 8.0

- Approximately ⅔ of a year of active development
- 2200 commits
- 91 different Contributors
- Tracepoint Support
- All new Features have Topology Tests
  - Continued expansion of current topologies
- Coming in the next couple of days
  - https://deb.frrouting.org
  - https://rpm.frrouting.org
- Libyang 2.0 upgrade
- New Daemon `pathd`
  - http://docs.frrouting.org/en/latest/pathd.html

Tracepoints

- Internal support for defining tracepoints
  - Good support for LTTNG
  - Limited support for USDT probes
- vs. logging:
  - Machine parseable unlike logs
  - Cheaper and more data-rich than log messages
  - Granularity easily filtered post facto
  - “Flight recorder” mode + low overhead = OK to leave tracepoints on all the time, even debug-level ones
- Usable today!
  - All existing log messages are exposed as LTTng tracepoints
- bgpd has an initial set of useful ones
  - input/output filter results
  - I/O events
  - Message processing
  - BMP events
- Adding more as needed over time
  - If you have something you’d like to collect detailed metrics on, consider adding a tracepoint
  - Chances are somebody else could use it too!
  - See developer docs for details on how

BGP

- RFC 4271 Delay Open Timer
- RFC 8050 MRT Add Path
- RFC 4273 SNMP Trap Notifications
- RFC 8654 Extended Message Support
- TCP-MSS for Neighbors
- BGP Table Version Filtering
- Alias Support for Communities
- EVPN MH Support
- Improved RPKI reporting
- Interface based LL reachability tracking
- MPLSVPN SNMP Support
- Wait For Installation
- Reduce route updates when not needed
- Conditional Advertisement Support
- Fix non-deterministic locally originated bestpath

ZEBRA

- Various EVPN support and cleanup
- Scale improvement for large number’s of VRF’s
- Improve asic-offload handling
- BSD interface and route handline improvements
- Human Readable netlink dumps
- Nexthop Group Handling
- Cleanup rule encoding via ZAPI
- Allow `set src X` to work on startup
- Deny when route-map does not exist yet
- More JSON support
- L2 NHG’s
- Opaque Data from upper level protocols
- Improved debugability of routes and vrf’s
- Improve route-map processing

OSPF

- TI-LFA
- OSPF GR Support
- BFD Profile Support
- Traffic Engineering Database Support
- Add ‘summary-address A.B.C.D/M ..’ commands
- JSON support for various commands
- Add ‘clear ip ospf neighbor’ command
- Add ‘area X nssa suppress-fa` Support
- Support of DMVPN
- SNMP Updates
- NSSA Cleanups

OSPFv3

- More JSON support
- Add control for maximum-paths
- Don’t send hello’s on loopbacks
- Cleanup area handling around interfaces
- YANG support for route-maps
- Add `show ipv6 ospf6 vrf` commands
- BFD Profile Support
- VRF Support

BABEL

- Add `distribute-list` commands
- Cleanup memory leak in connected route handling

EIGRP

- Add `distribute-list` commands
- Ensure received AS Number is the same as ours
- Properly validate TLV lengths

RIP and RIPNG

- Cleanup of print formatters
- Fixup of interface wakeup after shutdown

NHRP

- Make vici socket path configurable
- Retruy IPsec under some conditions
- Use MTU properly
- Handle NAT extension
- Add support for forwarding multicast packets
- Introduce `nhrp multicast-nflog-group ..` command

PIM

- YANG Integration
- Various Prune and Prune-Pending issues cleaned up
- IGMP conformance cleanups
- Various JSON Support
- BFD profile Support

Topology Code Coverage

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level	**Lines:**	138575	269587	51.4 %
Test:	coverage.info	**Functions:**	12430	20128	61.8 %
Date:	2021-07-13 08:05:37				

Directory	Line Coverage		Functions	
/usr/include/libyang	75.0 %	21 / 28	100.0 %	4 / 4
/usr/include/python3.9	100.0 %	12 / 12	100.0 %	4 / 4
/usr/include/x86_64-linux-gnu/bits	0.0 %	0 / 6	0.0 %	0 / 3
babeld	14.2 %	549 / 3872	24.1 %	71 / 294
bfdd	43.1 %	2232 / 5173	53.1 %	226 / 426
bgpd	56.3 %	30855 / 54795	66.2 %	2354 / 3557
bgpd/rfapi	39.2 %	4596 / 11726	57.0 %	339 / 595
bgpd/rfp-example/librfp	70.2 %	59 / 84	81.8 %	9 / 11
eigrpd	0.8 %	38 / 5048	7.4 %	30 / 404
isisd	56.1 %	10196 / 18161	62.2 %	998 / 1604
ldpd	55.0 %	6659 / 12118	75.2 %	663 / 882
lib	55.4 %	20225 / 36523	67.5 %	2358 / 3492
lib/printf	40.6 %	336 / 827	43.8 %	14 / 32
mlag	0.0 %	0 / 160	0.0 %	0 / 48
nhrpd	42.4 %	1716 / 4047	60.1 %	218 / 363
ospf6d	54.5 %	6613 / 12128	67.8 %	474 / 699
ospfd	51.1 %	13787 / 26989	62.7 %	1113 / 1775
pbrd	65.3 %	1524 / 2335	77.4 %	178 / 230
pimd	46.4 %	9097 / 19599	57.0 %	804 / 1411
ripd	39.4 %	1690 / 4294	43.5 %	177 / 407
ripngd	48.6 %	1555 / 3202	54.6 %	167 / 306
sharpd	27.4 %	419 / 1530	47.9 %	56 / 117
staticd	61.5 %	1615 / 2624	71.5 %	138 / 193
vrrpd	7.5 %	169 / 2258	17.8 %	34 / 191
vtysh	83.8 %	5070 / 6052	49.6 %	127 / 256
watchfrr	5.3 %	40 / 752	18.4 %	9 / 49
yang	100.0 %	105 / 105	100.0 %	35 / 35
yang/ietf	100.0 %	9 / 9	100.0 %	3 / 3
zebra	55.2 %	19388 / 35130	66.8 %	1827 / 2737

http://www.pinkbelly.org/frr/8.0-code-coverage/

Open Microphone

Questions?

Thanks!