Scaling bridge forwarding database

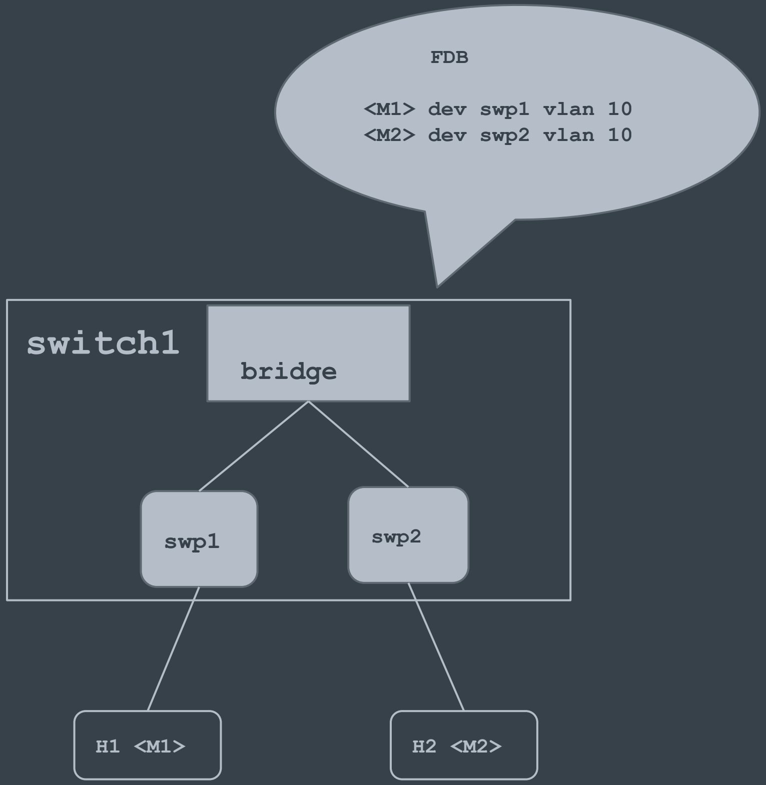
Roopa Prabhu, Nikolay Aleksandrov

Agenda

- Linux bridge forwarding database (FDB): quick overview
- Linux bridge deployments at scale: focus on multihoming
- Scaling bridge database: challenges and solutions

Bridge FDB entries

- Flood and learn (most basic case)
- End point Orchestrator/provisioning controller based FDB programming
- Control plane learning:
  - Local or distributed
- [**<Mac> <vlan> <dst_port>**]

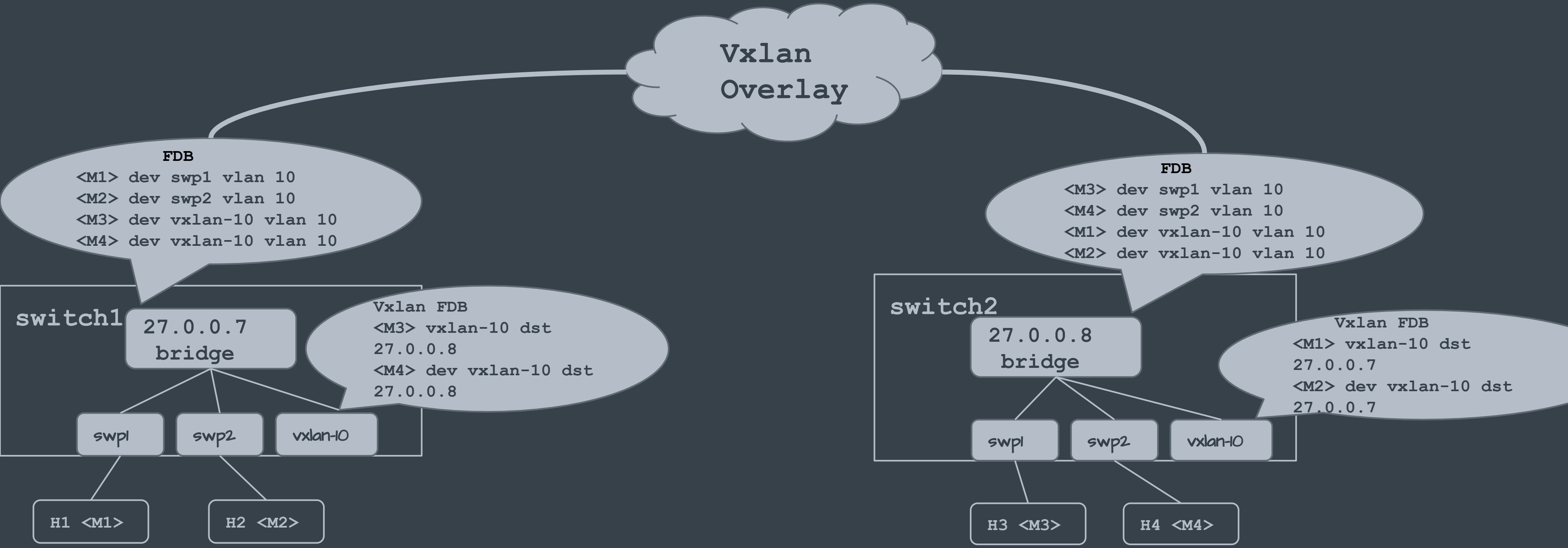


Bridge FDB entries: network virtualization (overlay: eg vxlan)

- **Overlay macs point to overlay termination end-points**
- **Eg Vxlan tunnel termination endpoints (VTEPS)**
  - Vxlan FDB extends bridge FDB
  - Vxlan FDB carries remote dst info
  - [**<mac> <vni> <remote_dst list>**]
    - Where remote_dst_list = remote overlay endpoint ip’s
    - Pkt is replicated to list of remote_dsts

Bridge FDB entries: overlay example

- switch1: M1 and M2 are local macs. M3 and M4 are remote macs



Bridge FDB database scale

Bridging scale on a data center switch

- layer-2 gateway
- Bridging accelerated by hardware
  - HW support for more than 100k entries
  - Learning in hardware at line rate
  - Flooding in hardware and software
- IGMP snooping + optimized multicast forwarding
- Bridging larger L2 domains with overlays (eg vxlan)
- Multihoming: Bridging with distributed state

Layer-2 gateway in a datacenter architecture

Bridge FDB performance parameters at scale

- Learning
- Adding, deleting and updating FDB entries
- Reduce flooding
- Optimized Broadcast-Multicast-Unknown unicast handling
- Network convergence on link failure events
- Mac moves

Multihoming

Multihoming

- Multihoming is the practice of connecting host or a network to more than one network (device)
  - To increase reliability and performance
- For the purpose of this discussion, let’s just say its a “Cluster of switches running Linux” providing redundancy to hosts

Common functions of a multihoming solution

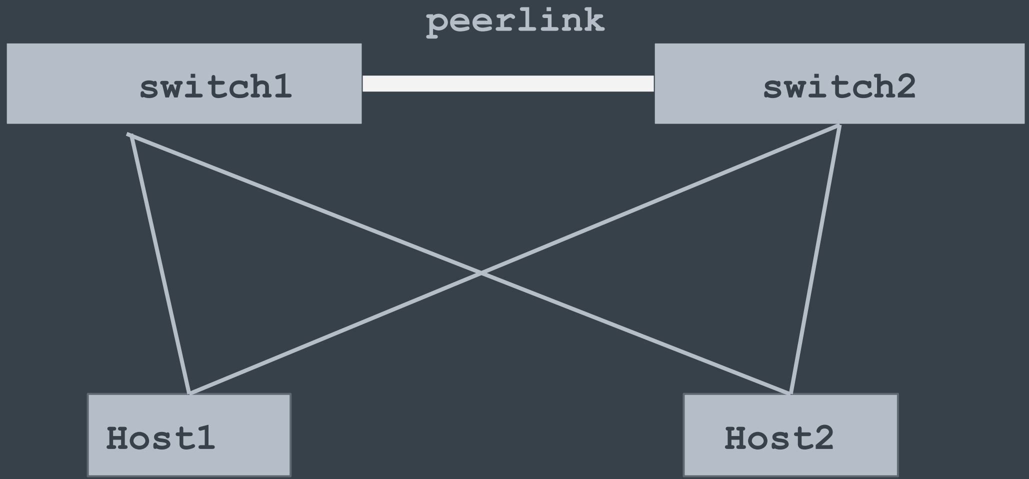
- Provide redundant paths to multihomed end-points
- Faster network convergence in event of failures:
  - Establish alternate redundant paths and move to them faster
- Distributed state:
  - Reduce flooding of unknown unicast, broadcast and multicast traffic regardless of which switch is active:
    - By keeping forwarding database in sync between peers
    - By Keeping multicast forwarding database in sync between peers

Multihoming: dedicated link

- Dedicated physical link (peerlink) between switches to sync multihoming state

- Hosts are connected to both switches

- Non-standard multihoming control plane

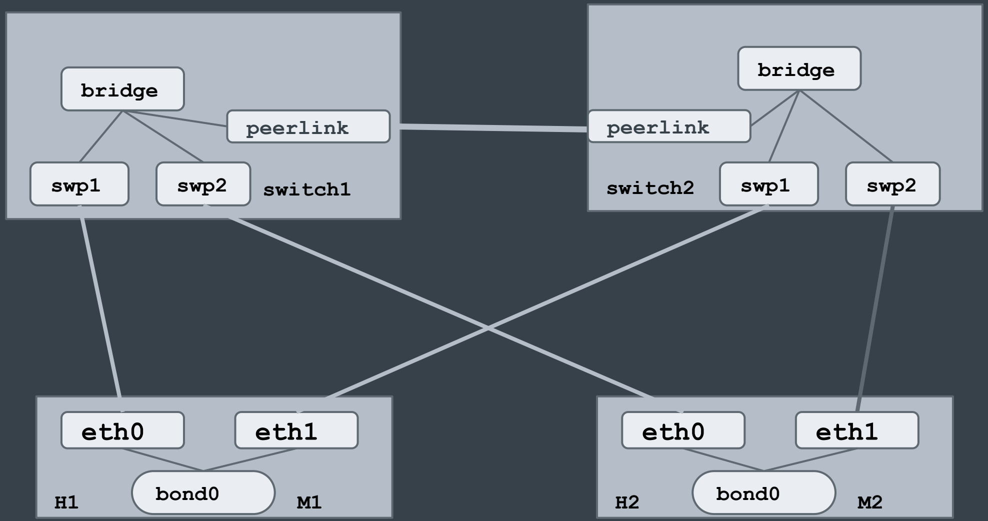


Multihoming: bridge: dedicated link

- Peerlink is a bridge port

- FDB entries to host point to host port
  <M1> dev swp1

- FDB entry on swp1 failure, moved to peerlink:
  <M1> dev peerlink



Network convergence during failures

- Multihoming Control plane reprogrames the FDB database:
  - Update FDB entries to point to peer switch link
  - Uses bridge FDB replace
  - Restore when network failure is fixed
- Problems:
  - Too many FDB updates and netlink notifications
  - Affects convergence

Bridge port backup port

- For Faster network convergence:
  - peer link is the static backup port for all host bridge ports
  - Make peer link the backup port at config time:
    - bridge seamlessly redirects traffic to backup port
  - Patch [1] does just that

Per Bridge backup port [1]

Before:

```
$bridge fdb show

mac1 dev swp1

/* On swp1 link failure event, control plane
updates each fdb entry to point to peerlink */

$bridge fdb show

mac1 dev peerlink
```

After:

Bridge port swp1 has peerlink as backup port:

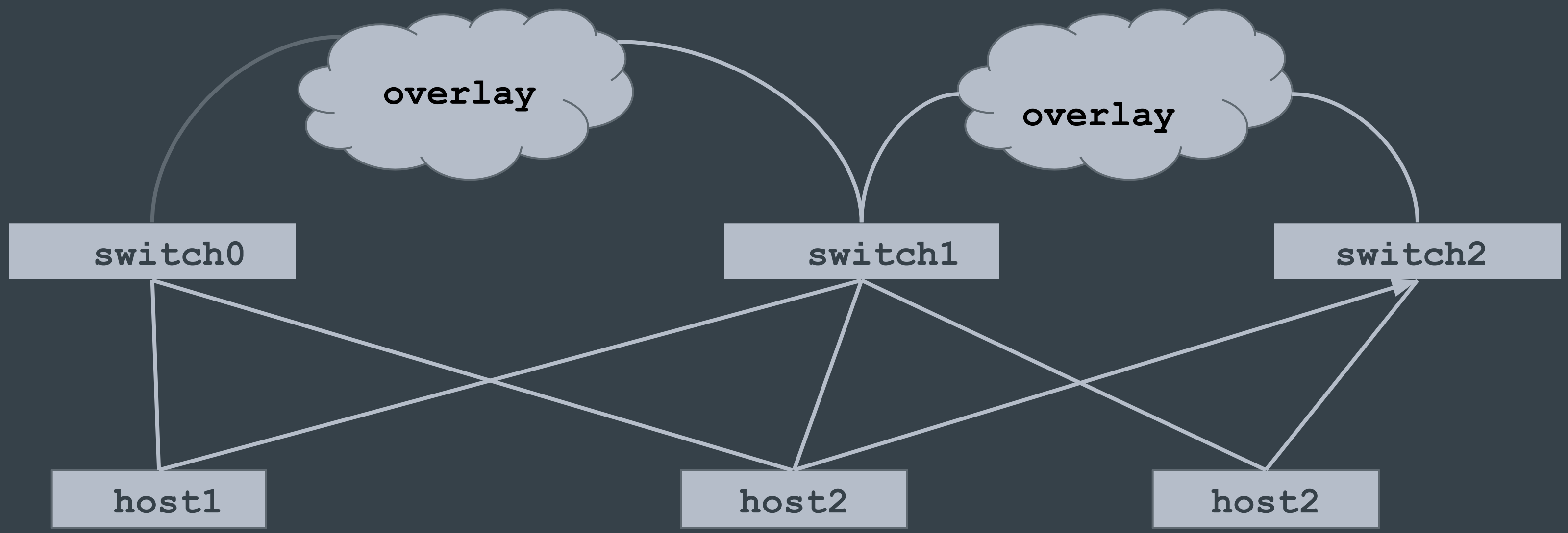
```
$ip link set dev swp1 type bridge_slave
backup_port peerlink

$bridge fdb show

mac1 dev swp1

/* On swp1 link failure event, kernel
implicitly forwards traffic to backup port
peerlink. No change to fdb entry */

$bridge fdb show

mac1 dev swp1
```

Future enhancements

Debuggability:

- FDB dumps to carry indication that backup port is active

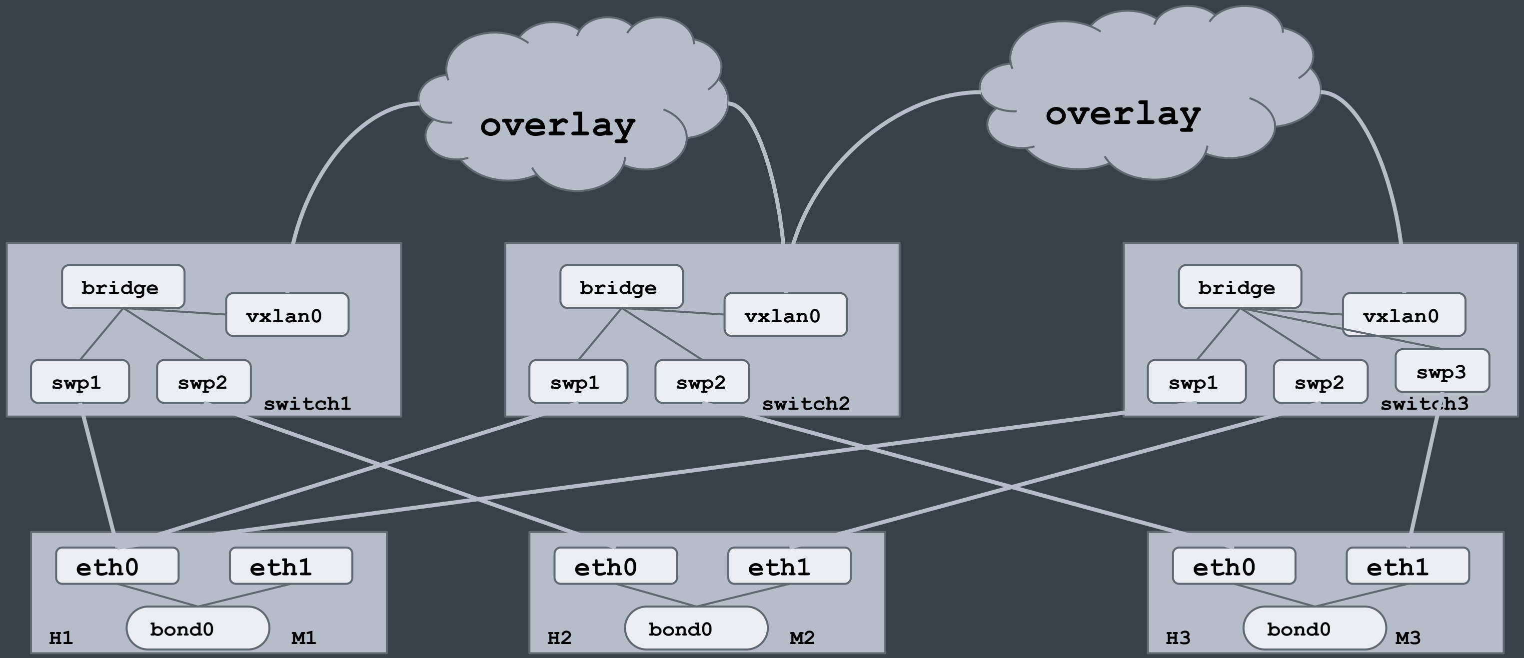
Multihoming: network overlay



Multihoming with network virtualization

- E-VPN RFC [2]: BGP based multihoming control plane
- No dedicated link between the clustered switches in a multihomed environment
- Dedicated switch peer-link is now replaced by the overlay
    - Eg a vxlan tunnel port in a vxlan environment
- More than 2 switches in a cluster
- In the active-active case, more than one remote dst in the underlay:
    - **mac <remote-end-point-underlay-ip-list>**
    - Requires mac ECMP (FDB entry mac pointing to ecmp group containing remote dsts)

Multihoming: network overlay



Control plane strategies for faster convergence

- Designated forwarder: avoid duplicating pkts [2,3]
- Split horizon checks [4]
- Aliasing: Instead of distributing all macs and withdrawing during failures infer from membership advertisements [5]

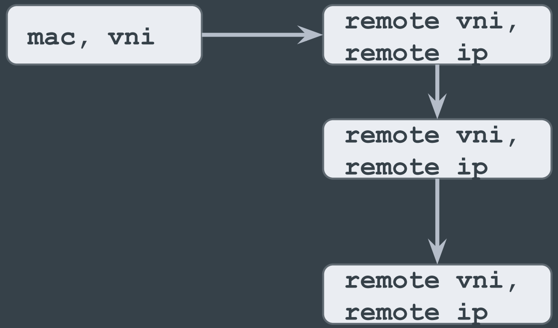
Forwarding database changes for faster convergence

- Backup port: to redirect traffic to network overlay on failure [1]
- Mac dst groups (for faster updates to FDB entries):
  - FDB entry points to dst group (dst is an overlay end-point)
  - Dst group is a list of vteps with paths to the MAC
  - Think FDB entries as routes:
    - Ability to update dst groups separately is a huge win
      - Similar to recent updates to the routing API [6]

New way to look at overlay FDB entry: dst groups

Current vxlan fwding database

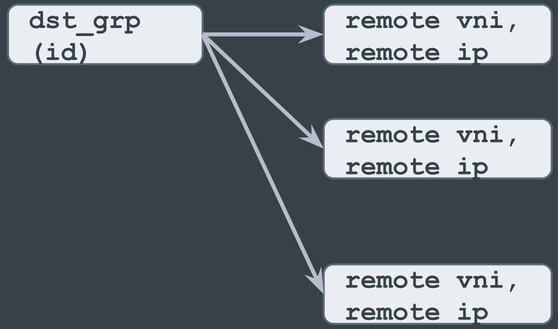
Eg: Vxlan fdb entry:



New proposed vxlan fwding database

Eg: Vxlan fdb entry:

Dst group db:



Fdb database API update

New fdb netlink attribute to link an fdb entry to a dst group:

- NDA_DST_GRP

New dst group API

To create/delete/update a dst group: RTM_NEW_DSTGRP/RTM_DEL_DSTGRP /RTM_GET_DSTGRP

```
enum {
        NDA_DST_GROUP_UNSPEC,
        NDA_DST_GROUP_ID,
        NDA_DST_GROUP_FLAGS,
        NDA_DST_GROUP_ENTRY,
        __NDA_DST_GROUP_MAX,
};
#define NDA_DST_GROUP_MAX (__NDA_DST_GROUP_MAX - 1)
```

```
enum {
        NDA_DST_UNSPEC,
        NDA_DST_IP,
        NDA_DST_IFINDEX,
        NDA_DST_VNI,
        NDA_DST_PORT,
        __NDA_DST_MAX,
}
#define NDA_DST_MAX (__NDA_DST_MAX - 1)
#define NTF_DST_GROUP_REPLICATION 0x01
#define NTF_DST_GROUP_ECMP      0x02
```

Other considerations for the dst group api

- Investigating possible re-use of route nexthop API [6]

Acknowledgements

We would like to thank Wilson Kok, Anuradha Karuppiah, Vivek Venkataraman and Balki Ramakrishnan for discussion, knowledge and requirements for building better Multihoming solutions on Linux.

References

[1] net: bridge: add support for backup port: https://patchwork.ozlabs.org/cover/947461/

[2] E-VPN Multihoming: https://tools.ietf.org/html/rfc7432#section-8

[3] E-VPN Multihoming: Fast convergence: https://tools.ietf.org/html/rfc7432#section-8.2

[4] E-VPN multihoming split horizon: https://tools.ietf.org/html/rfc7432#section-8.3

[5] E-VPN Aliasing and Backup Path: https://tools.ietf.org/html/rfc7432#section-8.4

[6] Nexthop groups: https://lwn.net/Articles/763950/

Thank you