HPC Ethernet

Designing an HPC Internet

Ben Matthews
NCAR/CISL SE/3

July 19, 2021

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Abbreviations that I may forget to define…

Quiz at the end only if I don't get good questions

- EVPN-MH: Ethernet VPN Multi-home (control plane)
- VXLAN: Virtual Extensible LAN (encapsulation)
- PIM: Protocol Independent Multicast
- DAC: Direct Attached Copper (cable)
- QSFP: Quad Small Formfactor Pluggable
- BGP: Border Gateway Protocol (routing)
- IB: InfiniBand
- RoCE: RDMA over Converged Ethernet
- BUM: Broadcast, Unknown unicast, Multicast
- LAG: Link aggregation (usually referring to an aggregated link/etherchannel)
- MLAG: Multi-chassis Link Aggregation
- ECMP: Equal-Cost Multipath
- PFC: Priority Flow Control
- AS: Autonomous System
- LACP: Link Aggregation Control Protocol
- LLDP: Link Layer Discovery Protocol

- MSDP: Multicast Source Discovery Protocol
- VTEP: VXLAN Tunnel Endpoint
- ARP: Address Resolution Protocol
- ND: Neighbor Discovery
- ACL: Access Control List
- VNI: Virtual Network Identifier
- ES: Ethernet Segment

About the Presenter

- ~8 years at NCAR
- B.S. CompSci (RPI), M.S. CompSci
- HPC generalist – not a network engineer
  - So go easy on me ☺

This talk

- Why?
- Possible Designs
- An aside about open networking
- EVPN-MH and Datacenter BGP
- What I wish I knew when we started

HPC at NCAR

- ~4k node SGI ICE (now HPE Apollo)
  - EDR IB
- ~100 node heterogeneous data analysis/viz cluster ("DAV")
  - HDR IB (mostly)
  - 100 or 200gigE (hardware RDMA)
  - Some NVIDIA GPUs (V100, GP100) [Extra connectivity due to NUMA]
  - **Publicly routable on the internet**
- GPFS Storage
  - Mix of EDR IB and 40/100GigE
    - In-house failover routing daemon to make that work (based on CARP)
    - Ethernet only
    - DDN SFA (**embedded controllers**)
- ~2.8k nodes of Cray (now HPE) EX, Lustre (coming soon)
  - Pretty much Ethernet – Connected via Arista 7k series 400gigE

HPC at NCAR

- PBS Pro (now the commercial release of OpenPBS)
  - But not that OpenPBS (if you've been around a while) (ugh)
  - Each system has its own complex, plans to peer them soon
    - Ideally most jobs run wherever there are resources, not necessarily on the machine the user intended
  - Dabbling with Cloud bursting – jobs could also run on someone else's hardware
- 100gigE to Internet2 (and commercial Internet)
- 100gigE (or 2x100gigE to other NCAR sites)
  - Frgp.net

Previous High-speed Network

- Juniper QFX10008, run by networking team for Internet
  - Also carried most other traffic for facility
  - i.e. IP Phone
    - Therefore difficult change processes
- Cluster specific networks run by HPC team
- Storage servers doubled as routers for IP traffic between networks
  - Fairly complex compute note static routing tables to facilitate this with failover and load balancing

Design Goals

- HPC can go up/down independent of campus network
- No single point of failure
- Robust flow control support
  - Need to support RoCEv2 (IP)
  - Mac-pause for hardware where we can't configure to do anything better (SFA Embedded controllers)
- As close to lossless as possible
- Routing, L2 and L3 stateless filtering
- For bonus points: not add another switch OS to learn
- Roughly, 220x100gigE, 8x10gigE downstream ports initially. Another ~8x400gigE in the first year. Room for further expansion desired.
- **Cheap, Fast, Good. Available now**

How about another big Chassis Switch?

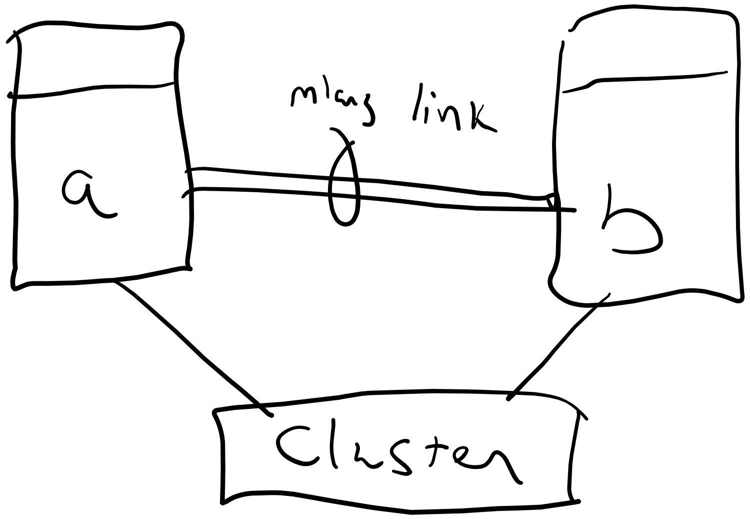
- Juniper QFX10008, Arista 7808, etc
- PSU and Switch ASIC redundancy possible, but backplane/chassis is a single point of failure
  - We've had this happen
- Simple to administer
- Expensive

Two Big Chassis Switches then?

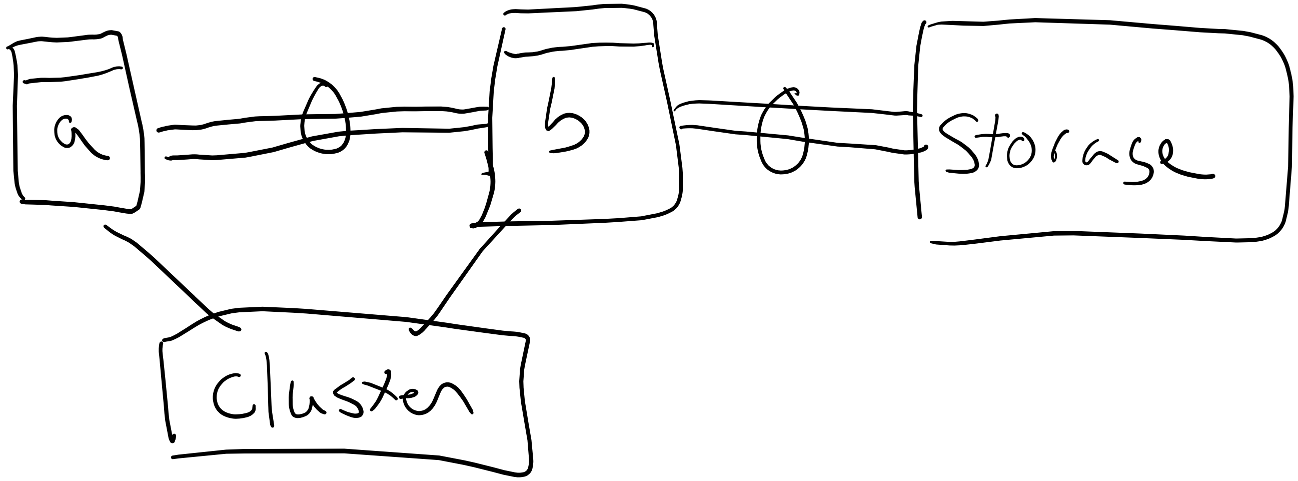
- Very expensive
- Need some way to make two switches look like one
- I'd still rather have discrete cables than backplane edge connectors
- … but let's try this design
  - MLAG?
    - Proprietary but implemented by pretty much everyone
  - EVPN-MH?
    - More on this later
  - Virtual Chassis
    - Proprietary

MLAG

- A way of dual connecting a device to two switches (generally exactly two).
- Need a path to route traffic that goes to the wrong switch
  - Actually, at least two (generally another LAG)
    - If this path goes down, bad things happen

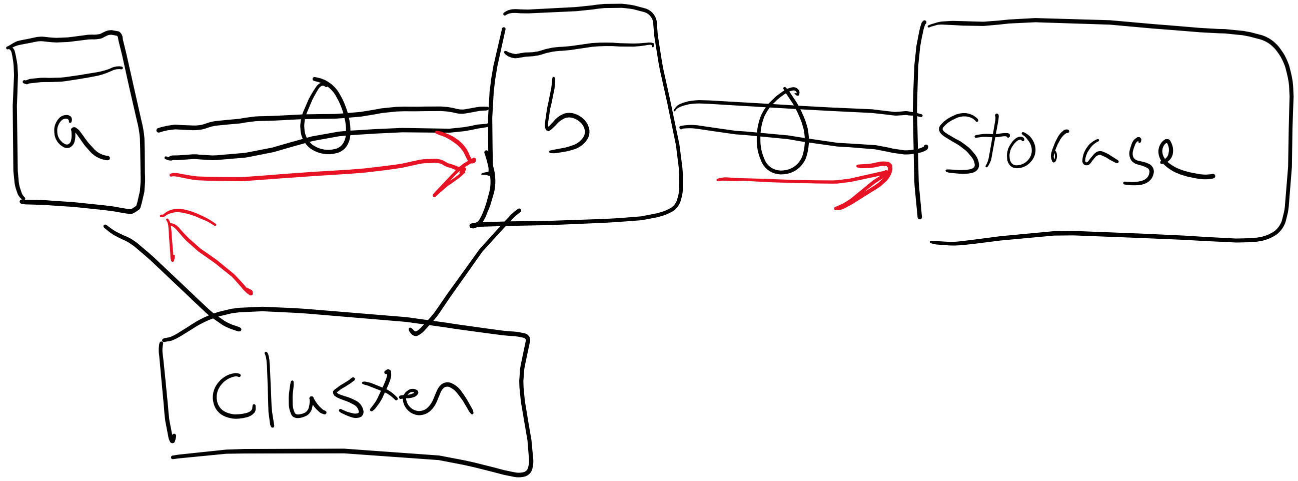


MLAG



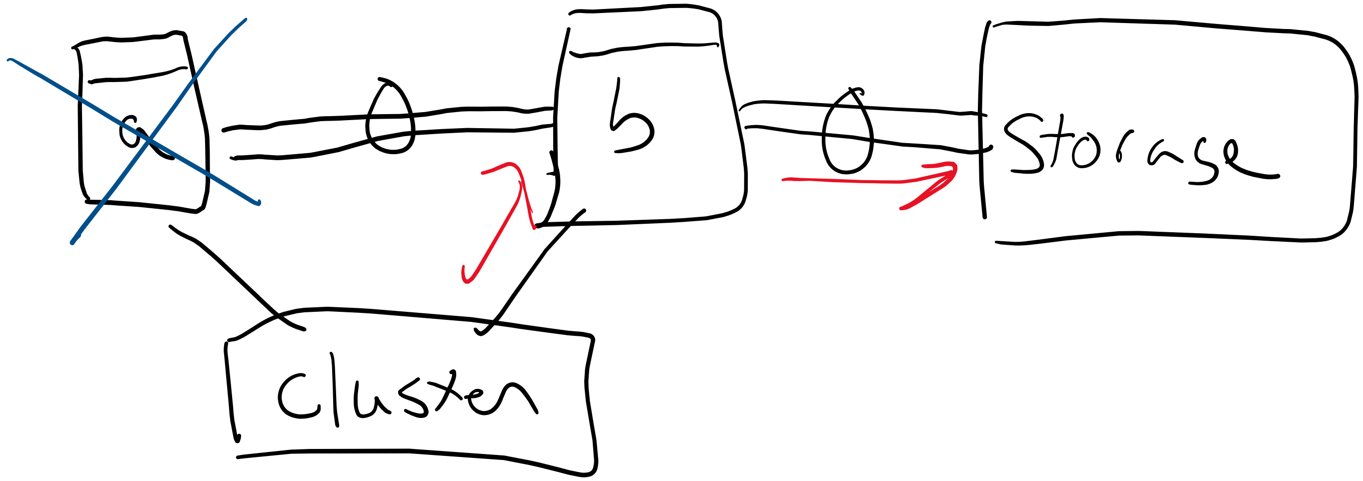
MLAG

- Packet flow (non-ideal hashing)



MLAG

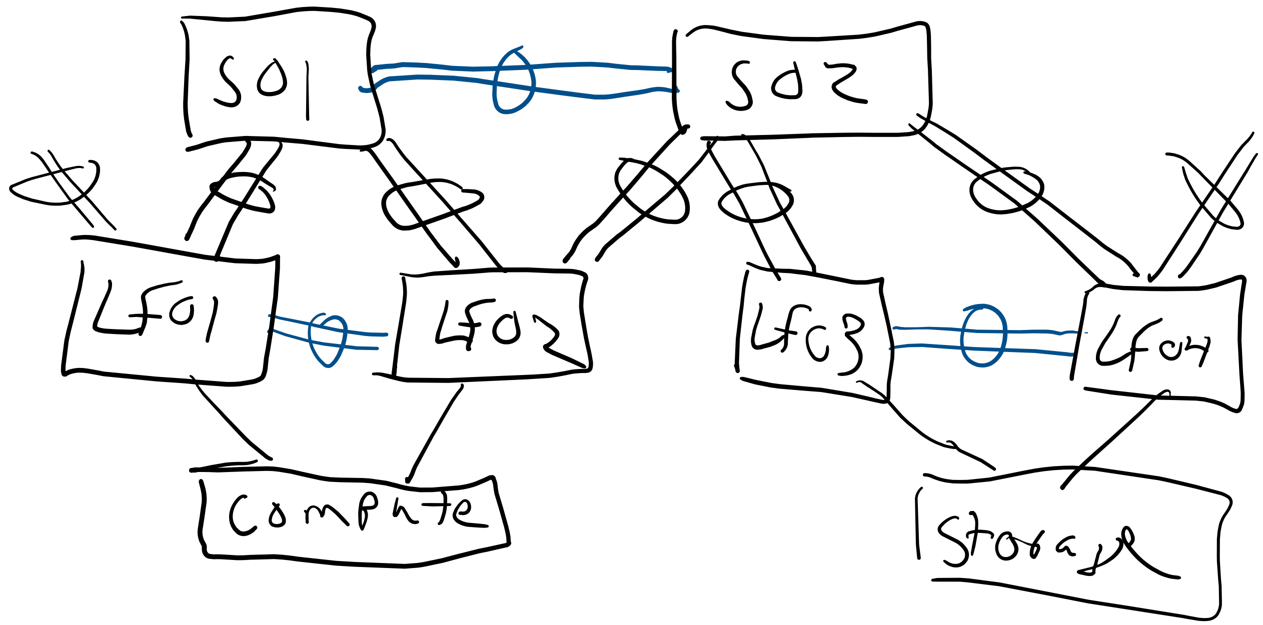
- Failure case



Scaling Up

- Q: Can this scale farther?
  - Yes, but with significant constraints
  - Tree of trees
- Q: Can we use smaller/cheaper/more robust/commodity 1U switches?
  - Sure, but generally MLAG is limited to a relatively small number of links and pairs of switches
- Consider this topology…

Bigger MLAG
(fat-tree, give or take)



Notice:

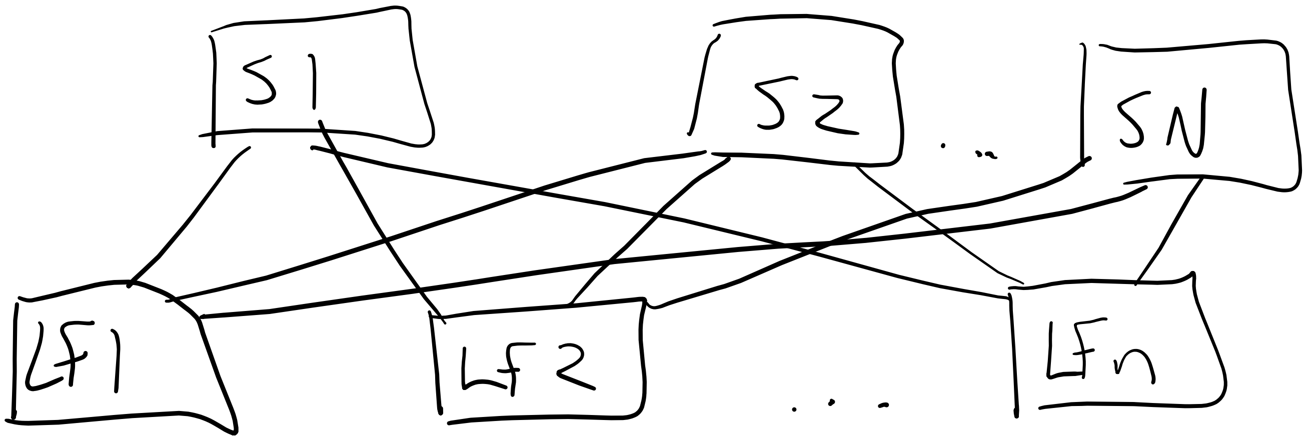
- At least 6 links wasted to MLAG (blue)
- Can only LAG to pairs of switches
- Sub-optimal routing (don't want the MLAG links to get busy)

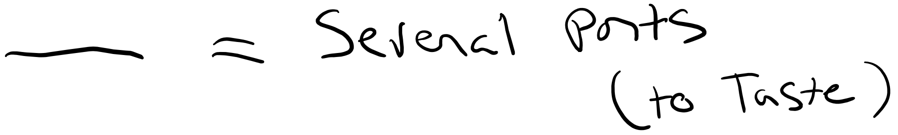
S=spine LF=leaf

Fat-Tree

- Can't we just do a fat-tree like we'd do on IB?
- Just plug everything together and it'll be fine… right?
  - Uh, no. Where did this misconception come from?
- Still, this might be possible
  - What if we run our Layer2 network on top of a routed Layer3 network of point to point links?
  - We'd be able to build up something like a traditional HPC interconnect
  - Enter BGP, EVPN and VXLAN
- Physically we want to get to this: (…)

Fat-Tree





VXLAN

- An encapsulation format for Ethernet and/or IP packets on top of IP
- We could build up a matrix of VXLAN tunnels to route our "real" traffic over, bridge them all and make a bunch of IP networks look like one
  - But that'd be a lot of work
  - And break every time the topology changed
  - Let's punt on this and figure out our underlay network first
- "Underlay network" -> The underlying IP network over which the VXLAN tunnels run
- "Overlay network" -> The network that is tunneled – it looks mostly like the chassis switch design if you squint really hard

Underlay Network

- Assign each switch a unique IP on its loopback interface for routing purposes
- Assign a unique address to each uplink port (switch to switch link)
- Run BGP over each uplink, advertising a route to the loopback interface
  - Assign each leaf switch a unique AS number
  - Spines share an AS number
  - Enable ECMP

I got this wrong, so NCAR's spines have unique AS numbers as well. It still works, but could be better

Quick BGP Review

- Each BGP entity is assigned an autonomous system (AS) number
- BGP neighbors establish a TCP session over which they exchange routes
  - Be careful when you write your ACLs/filters – don't filter important BGP sessions
- A single AS might advertise several routes
- We assume that routes with the same cost between the same pair of AS are interchangeable
- Can also do unequal cost multipath to use more costly routes, but this is HPC, we're sticking to a nice clean fat tree.
  - At least we really want to
- This is how the internet works, but we can also use it in the datacenter

ECMP

SM == Subnet Manager
== routing 🧙

LID == assigned address

- Equal Cost Multipath
- Kind of like a layer 3 LAG
- Packets are hashed and allocated among links with the same path cost
- For InfiniBand, the SM allocates paths to physical links based on the source and destination addresses (LIDs) in a predetermined forwarding table
- ECMP is kind of similar, except we don’t do it based on a static table of possible hashes and we might hash more than just the source/destination (configurable)

Underlay Network

BUM:
- Broadcast
- Unknown-Unicast
- Multicast

- So, what do we have so far?
  - A bunch of point to point links
  - Routing from any switch to any other switch via those point to point links
  - Load balancing among the point to point links via ECMP
  - Our own mini-internet
- What are we missing?
  - A single layer 2 domain
    - You might not care about this if you're very sure that you're only doing IP
  - A single subnet that we can cleanly connect servers to
  - BUM forwarding
  - Multi-leaf LAGing
    - Unless you want to run BGP or complex static routes on your compute nodes (which you could do)

Overlay Network

- Associate a VTEP (VXLAN Tunnel Endpoint) with each VLAN that you want to pass (I like to use the VLAN number for the VNI but it doesn't matter)
  - You probably want both layer2 and layer3 VTEPs
- Configure EVPN
- Your switch will listen for packets on each port and announce EVPN routes to IPs and MAC addresses that it hears locally
- When we want to communicate with a device on a different switch, we look it up in the routing table. If we have a route, VXLAN encapsulate that packet and send it to the correct switch via the underlay network
  - No route? Generate an ARP or ND packet
- A daemon periodically makes sure that everything in the routing table is still alive

Overlay Network
What about Broadcast/Multicast

- We still need a way to forward BUM traffic
  - We could just flood it to all the switches
  - We could multicast it on the underlay network
  - Different vendors handle this differently
    - Cumulus uses Protocol Independent Multicast (PIM)

BUM Forwarding: Pitfall

- This is a trap – your network will tend to work without BUM forwarding .. Mostly. As long as a device occasionally sends unsolicited traffic it will be reachable
  - (and you don't need multicast)
  - So the symptom is that a device is unreachable right after boot and then magically starts talking
  - No broadcast = no ARP = no IP. However, the switch will respond to ARP for things that it knows about so as long as you're in the BGP table, you're fine (or connected to the same bridge as the device making the request)
  - This is a nice performance win ("ARP suppression") since in most cases ARP doesn't actually need to get broadcast
  - But if the switch doesn't know about you, and BUM forwarding isn't working and the thing trying to reach you is on a different switch, you're SOL.

PIM

- Ok, so for BUM forwarding to work, we need multicast routing
- To configure PIM we designate a rendezvous point (RP) (potentially one per multicast-group, but that's usually overkill)
- When a switch gets a multicast join, it sends a message toward the RP, forming a distribution tree with the RP at the root. This tree is then used to forward multicast messages
  - This is *really* simplified and more representative of PIM sparse than PIM dense.
- Q: You promised no single point of failure. What happens when the RP dies?
  - Configure MSDP to get failover for PIM RPs
  - Also, even if BUM forwarding/PIM breaks for a brief period, your network will still mostly work

Summary of Protocols

- EVPN-MH configuration can be a bit of a choose-your-own-adventure type situation. Here's what you need to configure:
- BGP
- EVPN
- VXLAN VTEPs
- PIM or BUM flooding
- MSDP
- ECMP
- (probably want to turn on ARP suppression too)

- Yum.. Alphabet soup ☺

EVPN-MH

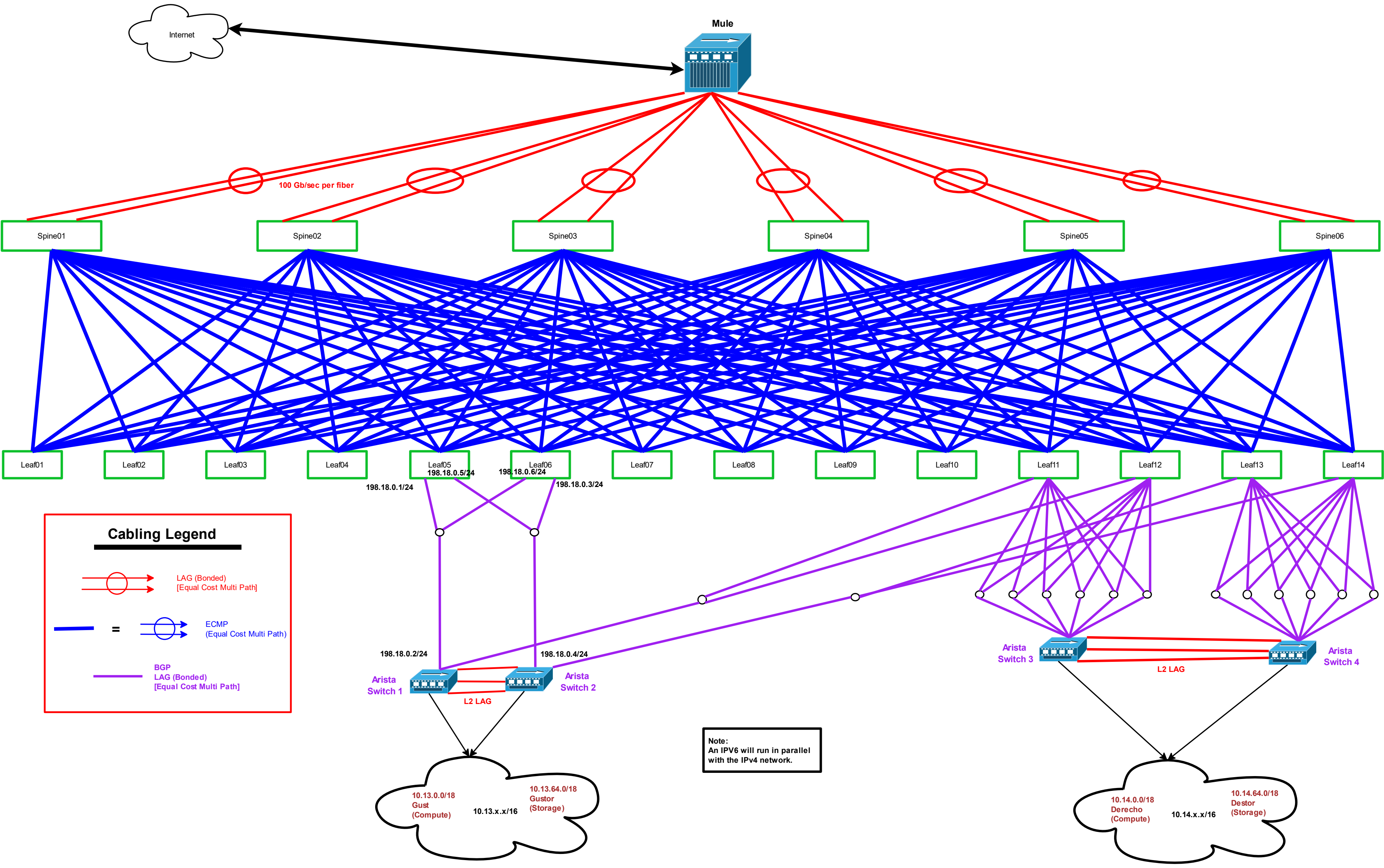
- So, how do I do a cross leaf lag?
- On the client, configure LACP as you normally would
- Define an Ethernet Segment ID for the LAG (this should be globally unique)
- Configure an ethernet segment sys-mac. This should be in one of the site assigned ranges and unique across the site
- Tell all the switches involved in the LAG about the ES-id and ES-sys-mac
- Note: You probably want to use a unique sys-mac for each LAG. This isn't mandatory, but if you don't LACP won't be able to help you detect miss-cables
  - 1/N of your packets go to the wrong place in this case. Super fun to troubleshoot.

NCAR's Implementation

- 4 spines, 10 leaves. Soon to be expanded to 6 spines and 14 leaves
- Mellanox SN3700, mix of DAC and SR4 optics
- Cumulus Linux
  - Docker container runs dhcpd/nginx on a management node
  - Pushes a configuration script and boot image to ONIE
  - Zero-touch provision new switches
  - If anything happens, just erase and start over. It's basically a stateless HPC cluster
    - I told you I'm not a network engineer
  - I'm using shell scripts, but you could use Ansible or Puppet. Cumulus has some examples for this.
    - It's pretty much just a Debian box. Config goes into frr.conf and /etc/network/interfaces

NCAR’s implementation

- 1:1.6 oversubscription on paper
  - SN3700 is 200G/4 lane
  - Some of our hosts have Connectx-4 cards, which can only do 25G/lane, so we burn a full port for those
  - The rest have Connectx-6 cards, which can do 100G-SR2 (2x100 per port – more density)
  - If you balance these well, you can get the oversubscription ratio to almost 1:1 and still fill the switches
- 400G devices connected via splitter. 400G-SR8 QSFP-DD on one side, 2x 200G-SR4 QSFP56 on the other (custom MPO16->2xMPO8 cable from fiberstore)
- 10G devices via 4>1 splitters



Thanks to Lee Myers (NCAR/CASG) for this diagram

Aside: Cumulus Linux and Open Networking

- It looks like Linux
- Various kernel data structures are sync'd with hardware
  - This is *fantastic* when it works
  - When it doesn't, good luck
- For example, iptables rules are compiled into hardware ACLs
  - You can do whatever you want with iptables
  - But it won't work if the hardware doesn't support it
  - Sometimes spectacularly
- Same with routes
  - ip r will show you the routing table
  - Unless something happened when programming the ASIC, then you have to drop into proprietary debugging magic
- "switchd" is the closed-source piece that makes the magic happen

Aside: Cumulus Linux and Open Networking

- Provisioning is fantastic. It’s an x86 box that runs a piece of software called ONIE
- ONIE will go download an install image for you
- Like PXE, but there’s a little Linux in the firmware that you can use to debug
- Likewise with monitoring. Install telegraf or ganglia or whatever and it’s just a Linux box with a lot of network interfaces
- ethtool –s and ethtool –m for debugging optics
- Also some networky options if you prefer
  - net show interface
  - net show interface plugables

Aside: Cumulus Linux and Open Networking

- Plan was for a cross platform NOS
- We have some Broadcom based 10gig switches running Cumulus as well
- Then Mellanox bought Cumulus
- And NVIDIA bought Mellanox
- Now Broadcom isn’t playing ball and it’s again mostly a single vendor platform
- ;-(
- SoNIC looks like a promising replacement but they don’t really have EVPN support yet.
- At least ONIE remains open enough, so it’s easy to load something else when something else is available.

Aside: Cumulus Linux and Open Networking

- Accepts most optics
  - Arista or Juniper would want branded optics
  - We're using mostly ~$100 FS.com optics
    - Huge savings
  - Still test first

Talking to the outside world

- More BGP
- Collapsed boarder-spine
- Each spine has a link to the internet and a BGP session with the router
- ECMP load-balancing
- Eventually, redundant paths to the internet

Flow Control

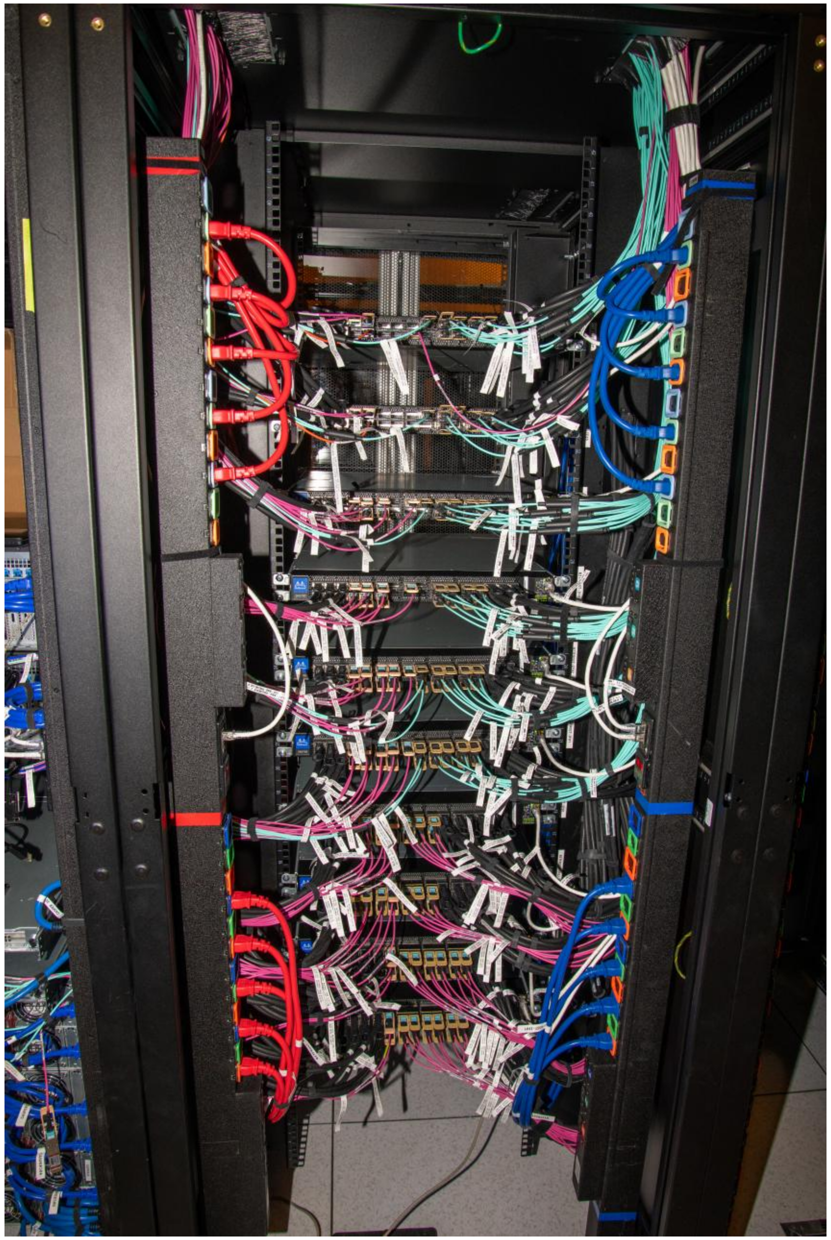
- This can be as simple or complex as you want
- Need to allocate buffer for packets in-flight on the wire (fiber?)
  - Propagation delay is a function of fiber type
- Our DDN gear doesn't like to do PFC, so that leaf needs to do pause frames
- ECN is nice for IP flows since it covers the entire round trip (maybe even some internet hosts)
- PFC is generally recommended by the vendors
  - In our situation, we generally want to lose no traffic
  - Inside the datacenter, can't drop RoCE traffic
  - Longer distance, retries are too costly
  - So, just do mac pause and ECN
  - So far, so good

Deployment Thoughts

- Try to layout each switch the same as much as you can. It's hard to get all the port maps correct
- Run lldpd on everything you can
  - Still not as nice as ibdiagnet, but it's a start
- Make sure optics are *clean*. Get a scope and learn to use it
  - ethtool –m
    - Results can vary by optic, but generally you should expect all the lanes to be about the same. If one is much lower, clean it again
  - ethtool –S
    - Look for symbol errors or ports with many more corrected FEC events than the others
- Verify multicast and BUM forwarding (iperf can do this… who knew?)

Deployment Thoughts

- DAC cables are nice between switches – they're cheap, but be mindful of bend radius
- Prefer deep door racks to allow better bend radius for fiber
- Prefer wider racks for better cable management
- Be mindful of cooling for both NICs and switches. 200gigE Optics may dissipate as much as 5W!
- Keep things clean →



References and Reading Material

- https://resource.nvidia.com/en-us-bgp-datacenter/bgp-datacenter-ebook
- https://resource.nvidia.com/en-us-ethernet-switching/evpn-datacenter?lx=GUdR1P

Questions?

- matthews@ucar.edu
- hsg@ucar.edu