THE DEFORESTATION OF L2

James McCauley, Mingjie Zhao, Ethan J. Jackson, Barath Raghavan,
Sylvia Ratnasamy, Scott Shenker
UC Berkeley, UESTC, and ICSI

The Talk

- ***What*** is AXE?
- ***Why*** look at this?
- ***How*** does it work?
- ***Really?*** This actually works?

The What

- An redesign of L2 to replace Ethernet and Spanning Tree Protocol (and its variants)
- Targets are “normal” enterprise networks, machine rooms, small private DCs
  - *Not* the Googles, Microsofts, Rackspaces
  - *Not* networks with incredibly highly utilization
  - *Not* managed by a full-time team of experts

The What: Goals

- Plug-and-play
  - If not, might as well just use L3
- Use all links for shortest paths
  - Number one shortcoming of STP
- Fast recovery from failure
  - Number two shortcoming of STP?

The What: Goals

- Plug-and-play
  - If not, might as well just use L3
- Use all links for shortest paths
  - Number one shortcoming of STP
- ~~Fast~~ Packet-timescale recovery from failure
  - Number two shortcoming of STP?

The What: Assumptions

- Failure detection can be fast
  - Not traditionally the bottleneck
    - Control plane "hellos" were sufficient
  - Need interrupt-driven LFS, BFD, etc.
- There's a market for flood-and-learn L2
  - Flooding/learn has security implications
  - No heavy unidirectional traffic
- No multi-access links
  - Everything is point-to-point

The Why: Is L2 still a problem?

- Still many largely-unmanaged, small/med L2 networks!
  - Two in our building in Berkeley!
- There have been a few interesting developments…
  - SPB, TRILL, SEATTLE, etc.
  - Provide various tradeoffs
- AXE attempts to strike a different balance
  - Focus on two key problems
  - Keeping things as simple as possible (no control plane)

The Why: Context

	Plug-and-play	Shortest Paths	Fast Recovery	No Control Plane
STP	✓	✗	✗	✗
No STP (Tree)	✓	✓	✗✗	✓
TRILL/SPB	✓	✓	✗	✗
IP (L3)	✗	✓	✗	✗
Custom	✗	✓	?	✗
AXE	✓	✓	✓	✓

The How: Extend Ethernet

- Basic flood/learn Ethernet
  - When you see a packet: learn
  - When you don't know what to do: flood

- But AXE *does not need a tree* to deal with loops
  - Means flooding works for handling failures too
    - (because alternate paths are immediately available)
  - Means that flood/learn finds short paths
    - (because you haven't removed links)

The How: Treeless flooding

- How do you get around the loop problem?

- **Duplicate-packet-detection**

- Multiple ways of doing it
- Our focus: ***hash-based deduplication filter***
  - In short: hash table where you replace upon collision
  - Straightforward
  - Amenable to hardware/P4 implementation

The How: What changes?

- Learning is more subtle
  - Source address seen on multiple ports
  - Packets may even be going *backwards!*
- Responding to failures is more subtle
  - Means we have to *unlearn* (outdated) state

The How: Extend Header

- Extend the packet header *between* switches
  - Nonce (per-switch sequence number)
    - *Used for packet deduplication*
  - Hop count
    - *Influences learning,* also *protects from loops*
  - Flooded flag: F
    - *Tracks whether a packet is being flooded*
  - Learnable flag: L
    - *Tracks whether packet can be learned from*

The How: Separate queues

- Switches have *flood queue* and *normal queue*
  - The Flooded flag in the header determines which
  - Flood queue has higher priority and is shorter
  - Normal queue sized… normally

- Intuition:
  - Delivering floods quickly *stops* flooding quickly
  - Deduplication only applies to floods, keeping fewer floods in flight makes dedup easier

The How: Overview

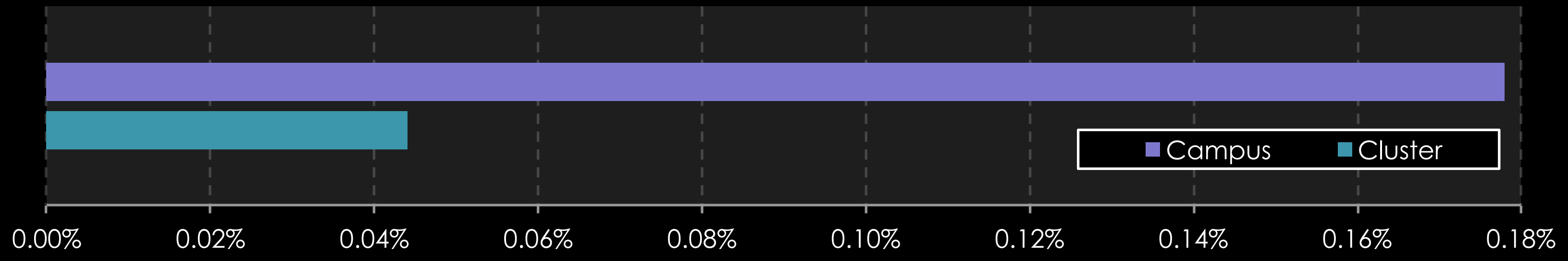
- Extend packet header
  - Nonce, Hop Count, Flooded / Learnable flags
- Learning/Unlearning Phase
  - May learn port and HC to *src*
  - May unlearn path to *dst* if trouble was observed
- Output Phase
  - If packet is a duplicate: drop
  - If unknown-*dst*/path-failed/already-flooding: flood
  - Otherwise forward according to table

The How: Example

- ***A*** sends a packet to ***B*** (L:True)
  - Destination ***B*** unknown; packet flooded from first hop (F:True)
    - All switches learn how to reach ***A***
- ***B*** sends to ***A*** (L:True)
  - Direct path following table entries to ***A*** (F:False)
    - Switches along path learn how to reach ***B***
- Link fails
- ***A*** sends another packet to ***B*** (L:True)
  - Follows along path… (F:False)
  - .. until it hits failure (L:False F:True)
  - Switch floods packet out *all* ports (even backwards)
    - Flooded packet reaches ***B*** (Successful delivery!)
    - Another duplicate of flooded packet reaches ***A***'s first hop; unlearn ***B***
- ***A*** sends *another* packet to ***B*** (L:True)
  - Destination ***B*** unknown; packet flooded from first hop (F:True)
    - All switches learn how to reach ***A*** again

Really? Preliminaries

- How much flooding do failures cause?



- How big does the deduplication filter need to be?
  – Less than 1,000 entries in our simulations
- Does it recover from overload?
  – Yes*

Really? Overview

- Thinking back to that matrix…
  - We want plug and play
  - We to support shortest paths using all links
  - We don’t want to have a control plane
  - *Packet-timescale* recovery from failures

Really? Failure benchmark

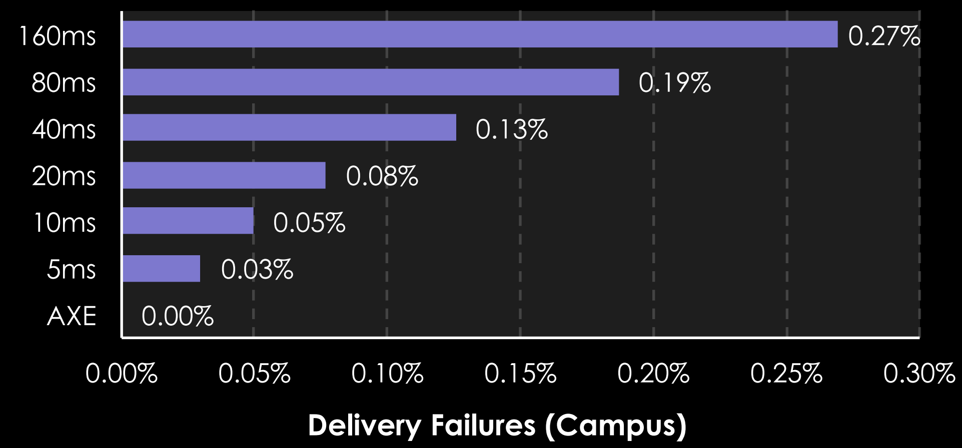
- Omniscient, randomized, shortest-path routing
- Failure → ***Adjustable delay*** → Fix routes

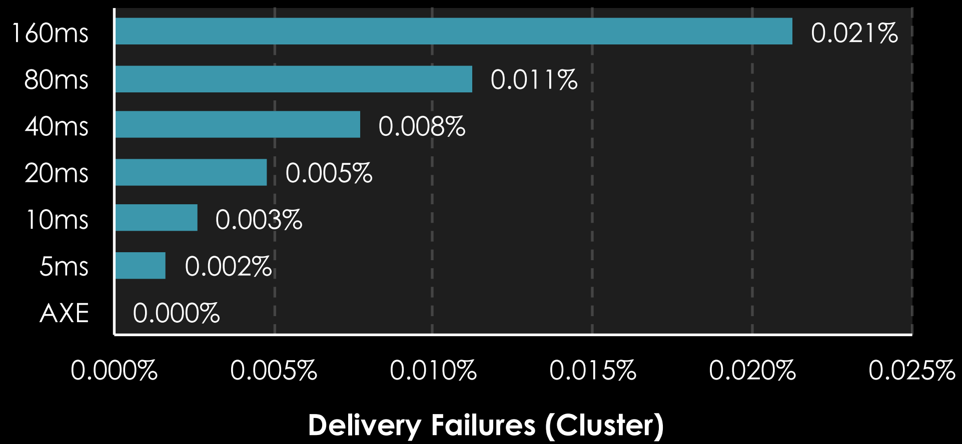
- *Delay of zero* is *optimal routing* / an upper bound
- *Nonzero delay* meant to roughly simulate…
  - OSPF, IS-IS, TRILL, SPB, etc.
  - .. without needing to model each one in detail

- Random shortest-cost tree rooted at each destination
- Note: we don't compare ourselves to STP at all

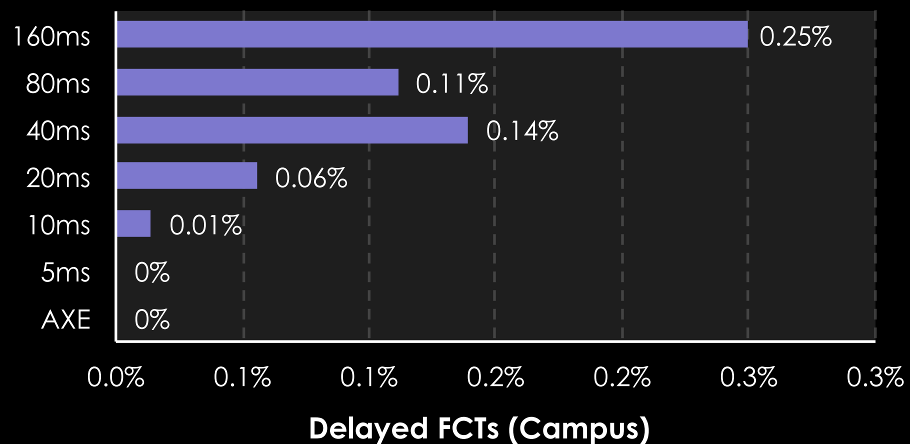
Really? Failure recovery - UDP

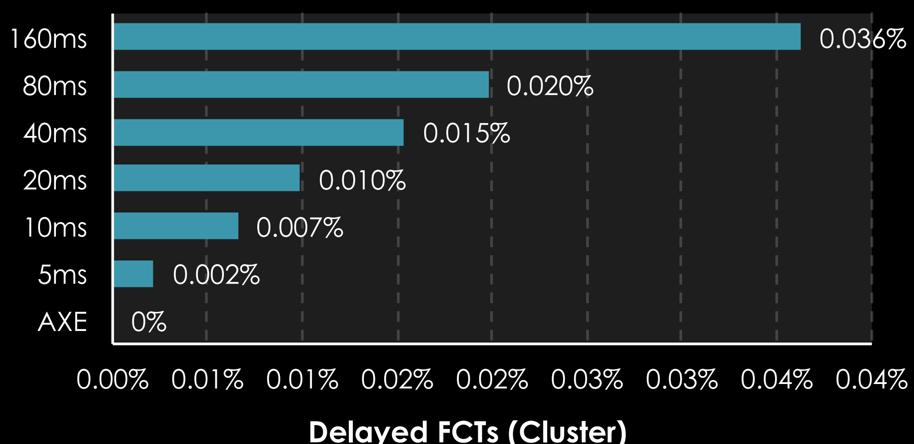
- Send traffic on network with high failure rate
- Metric is *unnecessary delivery failures* – packets that weren't delivered even though optimal routing could have delivered them
- AXE has *no* unnecessary delivery failures





Really? Failure recovery - TCP





- Similar setup, but with TCP
- Metric is number of flows with significantly worse FCT than optimal routing
- AXE has *no* significantly worse FCTs

The End: Not Mentioned Here

- Multicast AXE
  - On any change (failure; join), flood+dedup and prune
  - Flooded packets have all data needed to build tree
- AXE with Hedera
  - Use AXE for mice & recovery
  - Centralized SDN routing for elephant flows
- P4 implementation
  - AXE is expressible in P4
  - Performance on real hardware is open question

THE END