Network Myths and Mysteries

Radia Perlman
Intel Labs

radia.perlman@intel.com
radia@alum.mit.edu

All opinions expressed herein

- Are mine alone

All opinions expressed herein

- Are mine alone
- Though I’m sure…independently shared by lots of other people

Network Protocols

- A lot of what we all know…is false!

How networking tends to be taught

- Memorize these RFCs
- Nothing else ever existed
- Except possibly to make snide comments about “other teams”

Things are so confusing

- Comparing technology A vs B
  - Nobody knows both of them
  - Somebody mumbles some vague marketing thing, and everyone repeats it
  - Both A and B are moving targets

How I wish we'd compare

- Isolate conceptual pieces
- Ignore "which team"

Orthogonal ways networks can differ

- What information is in a packet
- Who computes the forwarding table
- Whether forwarding table is always complete, or created on demand when a flow starts
- Whether switch can choose among multiple next hops
- How to translate from layer 3 to layer 2 address
- …

What does a switch do?

- Forward based on:
  - Info in packet
    - Destination address
    - If need to keep things in order, other stuff in packet (e.g., TCP ports, flow ID, entropy field)
  - Forwarding table lookup
    - Destination: ➔ single port
    - Destination: ➔ {ports}
    - “Flow”: ➔ single port

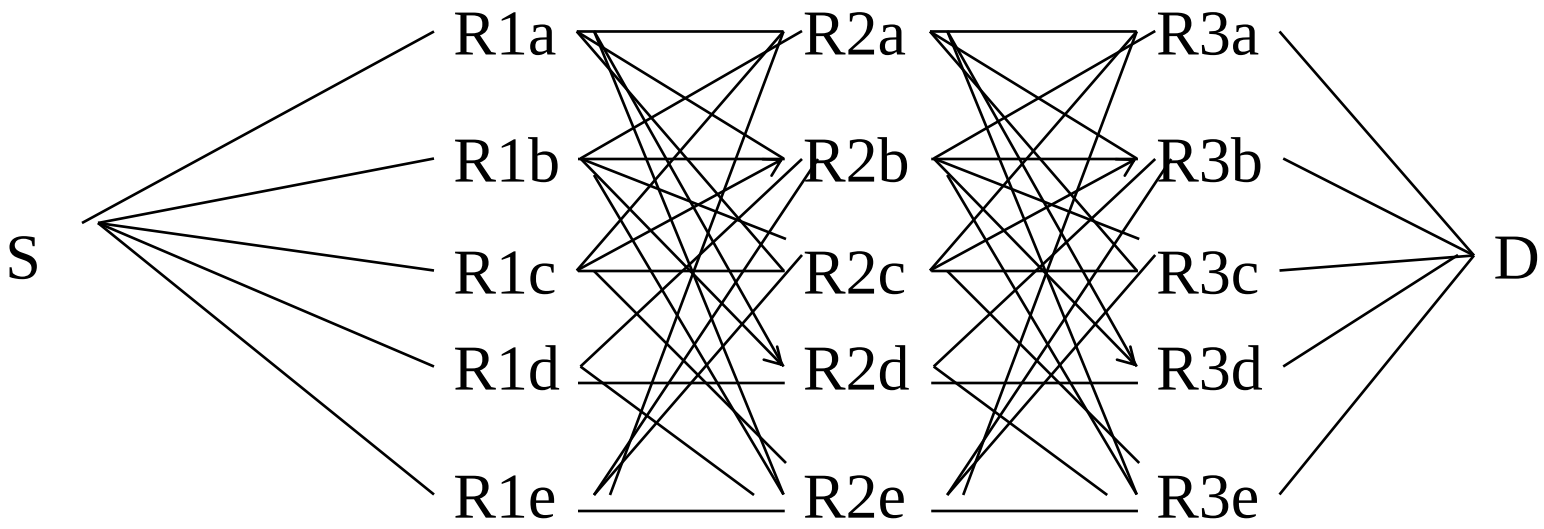
Different types of Destination Lookup

- Direct lookup: simple and fast. Only possible if small enough address that table is not too large
- Hash (like Ethernet)
- Longest Prefix match (like IP)

Thoughts

- Flow-based vs destination-based
  - Only way to make flow-based not totally explode the forwarding table is to create entry when flow starts (incur latency)
  - Switch in better position to load-split traffic than central fabric manager
  - If switch has to keep packets of a flow in order, switch can look at TCP ports, etc., or have “entropy” field in header
  - Switch can do new hash to adjust traffic

Exploiting parallel paths



Completely orthogonal concept

Where does forwarding table come from?

- Distributed algorithm
- Central fabric manager
- Neither concept new…and completely orthogonal to “data plane”
- Concept of separation of control plane from data plane not new…

Where does forwarding table come from?

- Distributed algorithm
- Central fabric manager
- Neither concept new…and completely orthogonal to “data plane”
- I think distributed algorithm is superior, because it can react to topology changes more quickly

Where does forwarding table come from?

- Distributed algorithm
- Central fabric manager
- Neither concept new…and completely orthogonal to “data plane”
- Distributed algorithm isn’t what makes switches expensive

Also orthogonal: Fancy extra features

What kinds of features are in DC switches?

- Security features (firewalls, intrusion detection/protection, IPsec, SSL session termination)
- NAT (and other stateful features)
- Multipathing
- Traffic engineering/bandwidth guarantees
- Multiprotocol ports: layer 2 and layer 3; IPv4, IPv6
- IP multicast
- Dynamic VLAN registration
- …

When does forwarding table get filled in?

- Proactively
- When a flow starts

Proactively seems better

How do you manage a network?

- From a management console, which translates “ big” commands, such as “forward based on this metric” or “traffic engineer this path” into individual commands to switches

How do you manage a network?

- From a management console, which translates “ big” commands, such as “forward based on this metric” or “traffic engineer this path” into individual commands to switches
- Protocols define parameters that are settable, readable, events that trigger alerts

Management commands to switches

- SNMP
- Netconf
- YANG
- OpenFlow

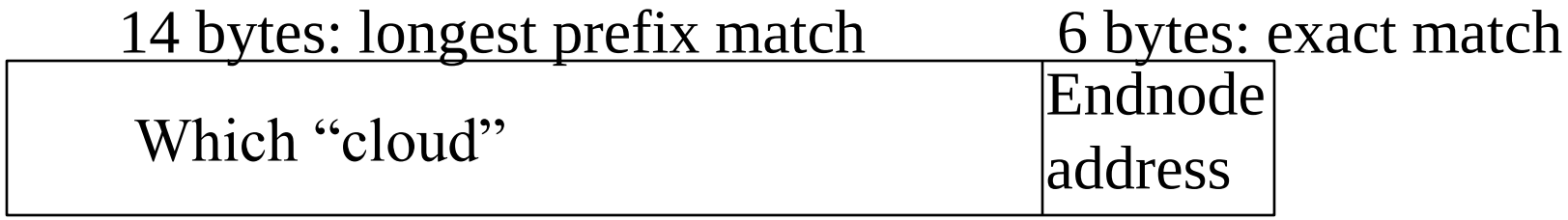
Now, sorting out a few existing things

- IP, Ethernet, TRILL, etc.

A bit of layer 3 history

- ISO's layer 3 (CLNP: ConnectionLess Network Protocol)
- 20 byte address
- Top 14 bytes an entire "cloud"
- Inside cloud
  - No configuration of switches
  - Flat address space (bottom 6 bytes)
  - Endnodes help by announcing to routers where they are
  - Endnodes can move around and keep address

CLNP addresses

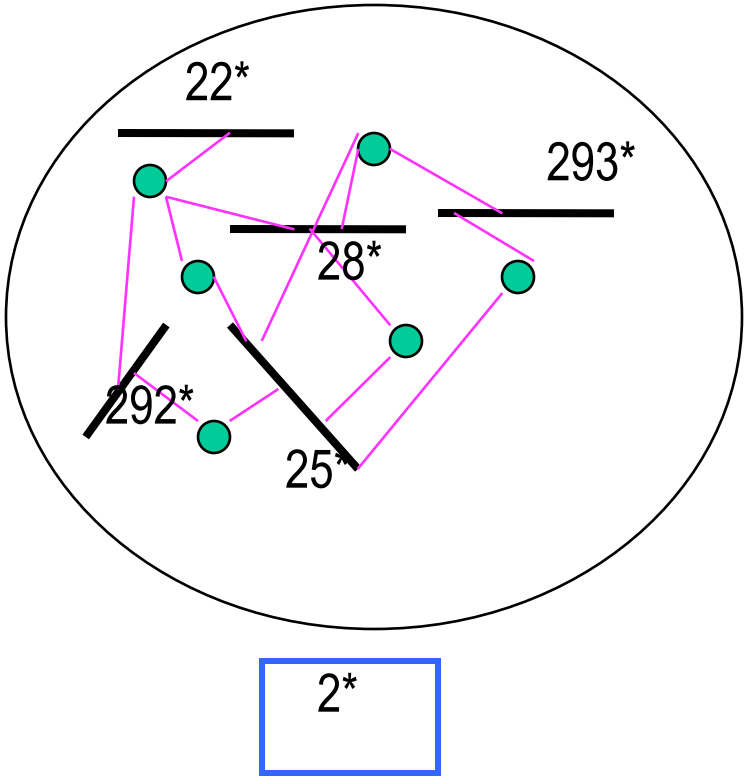


In contrast: IP

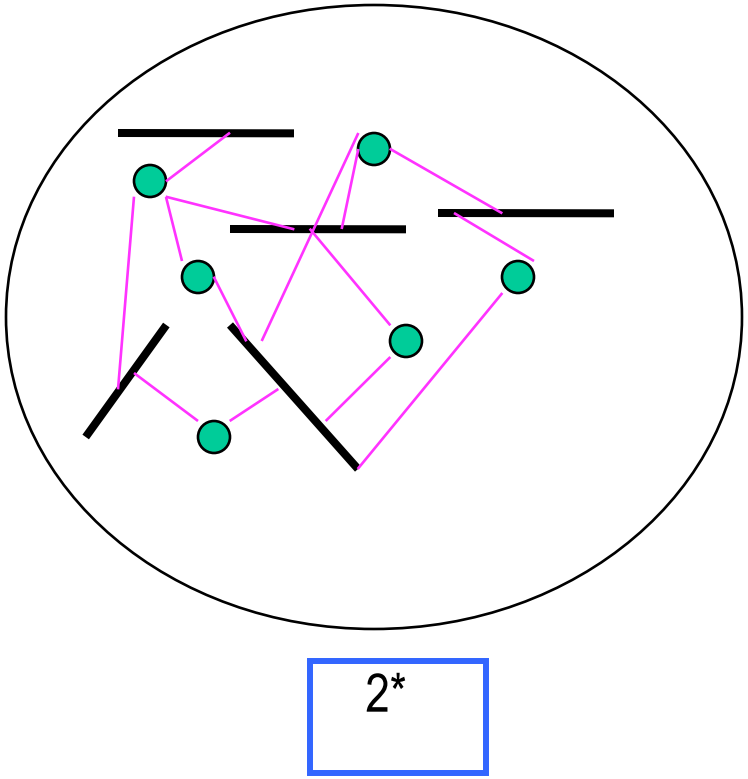
- Every link has its own prefix
- Routers need to be configured
- If node moves, address has to change

Hierarchy

One prefix per link (like IP)



One prefix per campus (like CLNP)



Advantages of CLNP over IP

- No configuration of routers inside a “cloud”
- Flat address within cloud so endnodes can move around within the cloud
- No need for ARP (since cloud address is carried within the layer 3 address)
- Other things…not enough time to go into

Worst decision ever

- 1992…Internet could have adopted CLNP
- Easier to move to a new layer 3 back then
  - Internet smaller
  - Not so mission critical
  - IP hadn't yet (out of necessity) invented DHCP, NAT, so CLNP gave understandable advantages
- CLNP still has advantages over IPv6 (e.g., large multilink routed bottom layer)

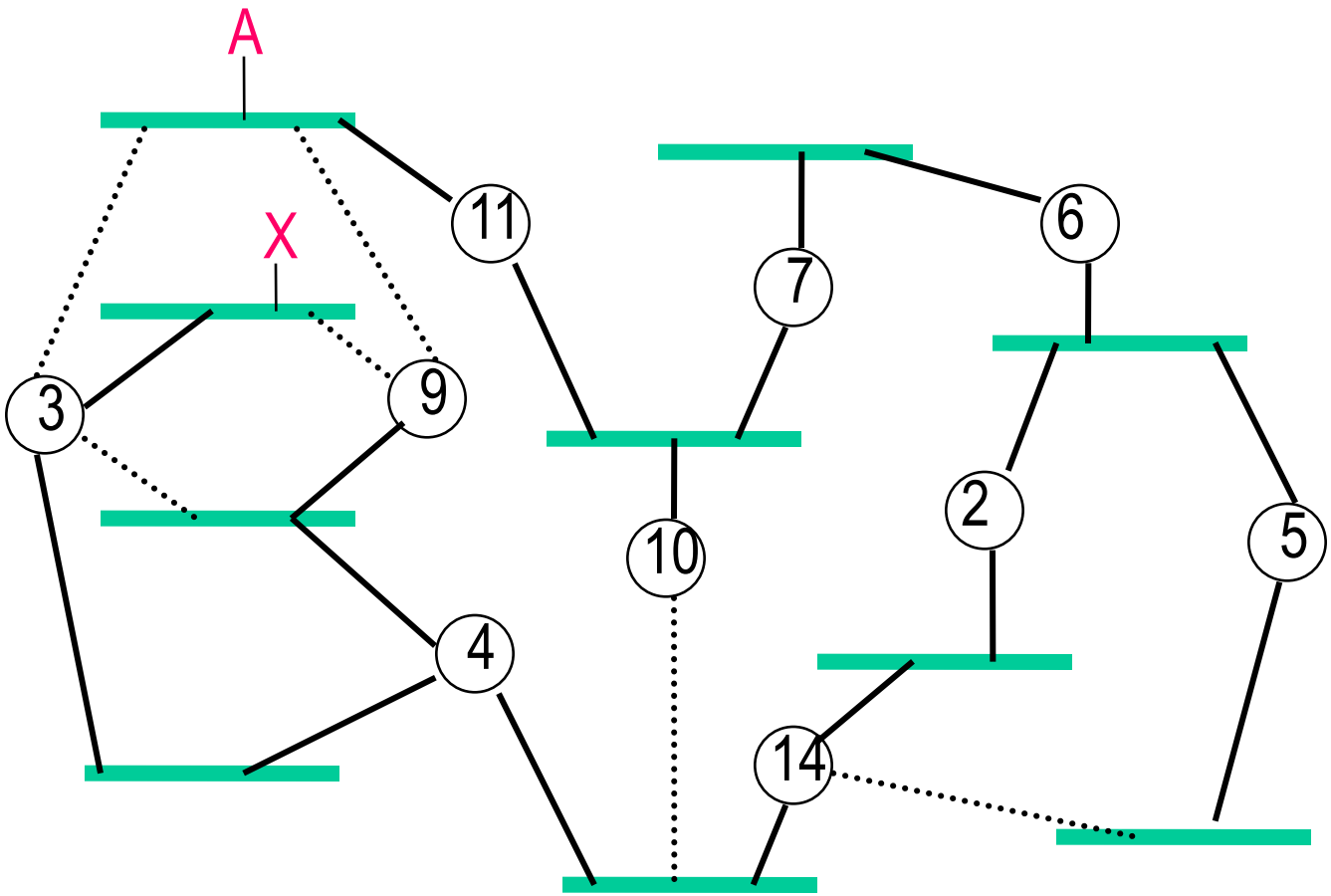
So that's why we are stuck with both Ethernet and IP

- Need "Ethernet" to create a flat cloud at bottom layer for IP
- Original Ethernet CSMA/CD
- But now based on spanning tree

Constraint in 1983 for designing spanning tree (“transparent”) bridge

- No change to Ethernet frame (including adding to length)
- No change to endnode behavior

Spanning Tree doesn't give optimal paths



Then…length restriction ended…

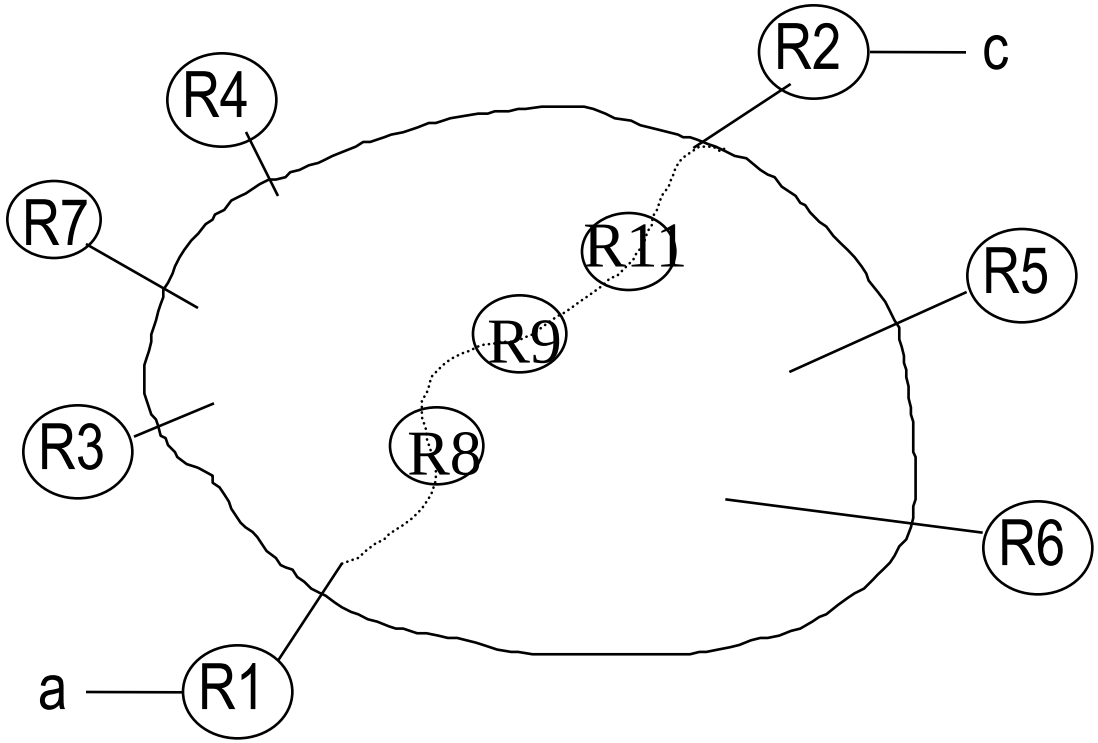
So…TRILL (TRansparent Interconnection of Lots of Links

- Allows clouds with flat address space, optimal routing, and no configuration of switches
- Evolutionary: Can replace any subset of spanning tree bridges with TRILL switches
- First published about 10 years ago
- Been an IETF working group for about 8 years, initial version standardized about 3 years ago

Basic TRILL concept

- TRILL switches find each other (perhaps with bridges in between)
- Calculate paths to other TRILL switches
- First TRILL switch tunnels to last TRILL switch

Basic TRILL concept

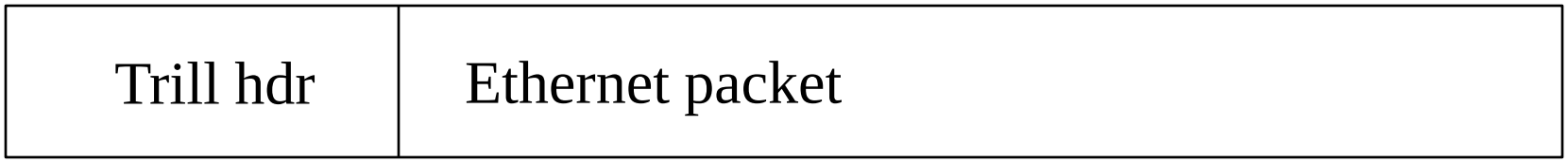


What's in TRILL header?

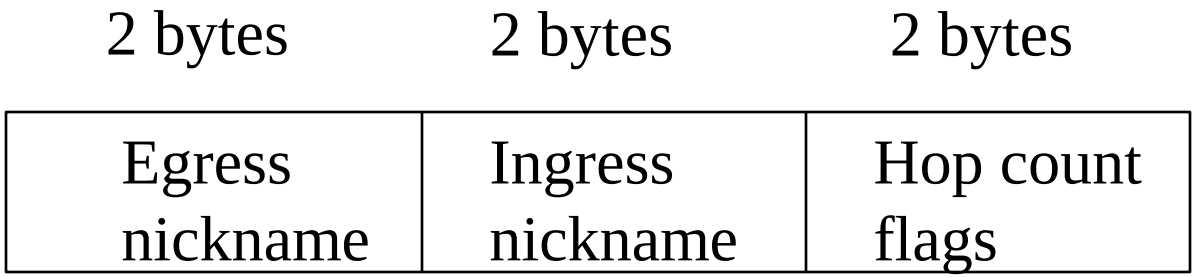
- TRILL switches autoconfigure a 2-byte "nickname"
- So header contains
  - Ingress TRILL switch (2 bytes)
  - Egress TRILL switch (2 bytes)
  - Hop count
  - Flags (e.g., unicast vs multicast)

TRILL-encapsulated Ethernet pkt

6 bytes



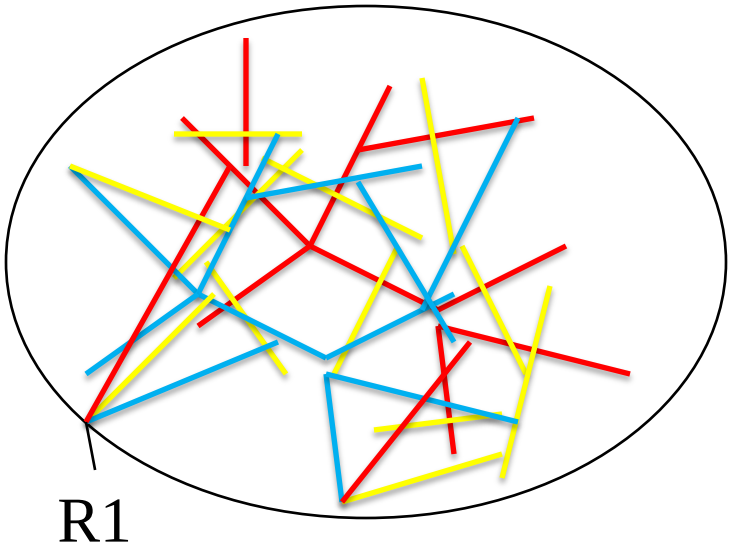
TRILL Header



Unicast vs multicast use of “egress” field in TRILL header

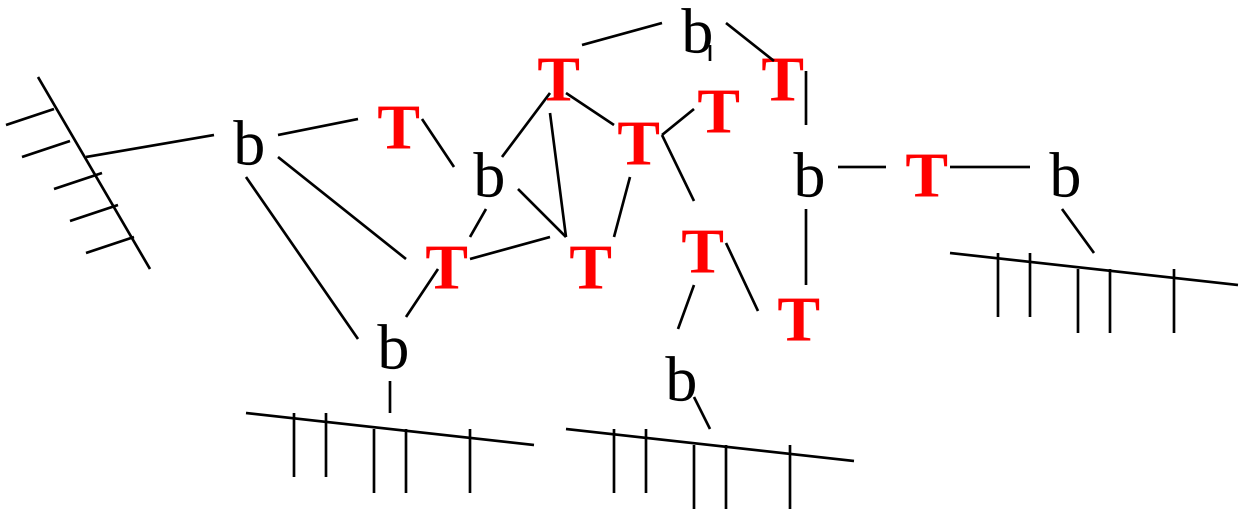
- Unicast: “egress” is last switch
- Multicast: “egress” is “which tree”

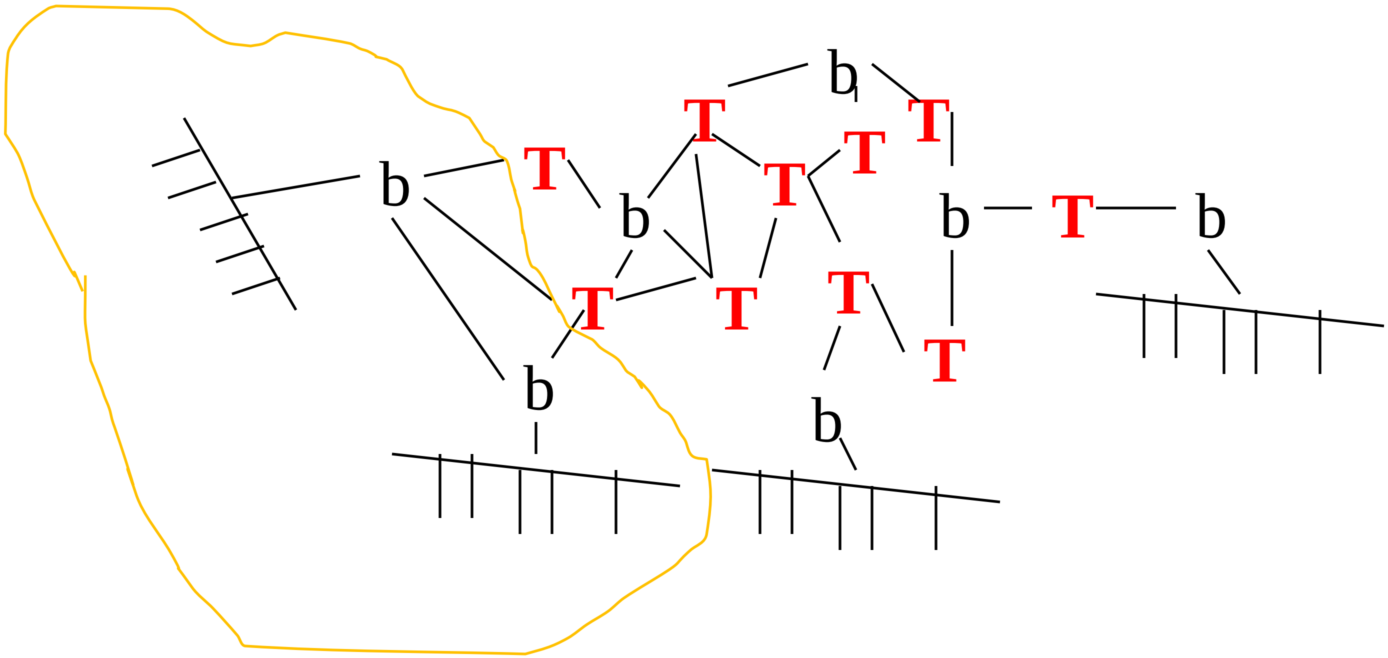
Multiple trees for multicast

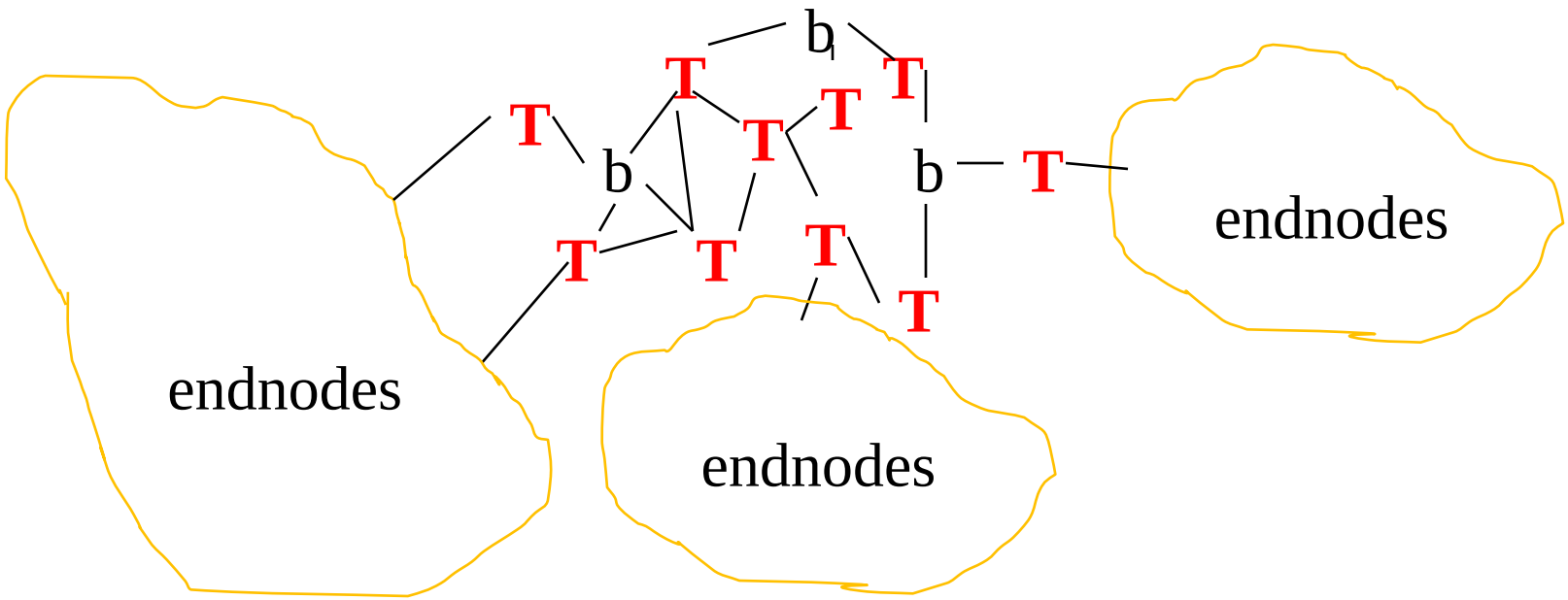


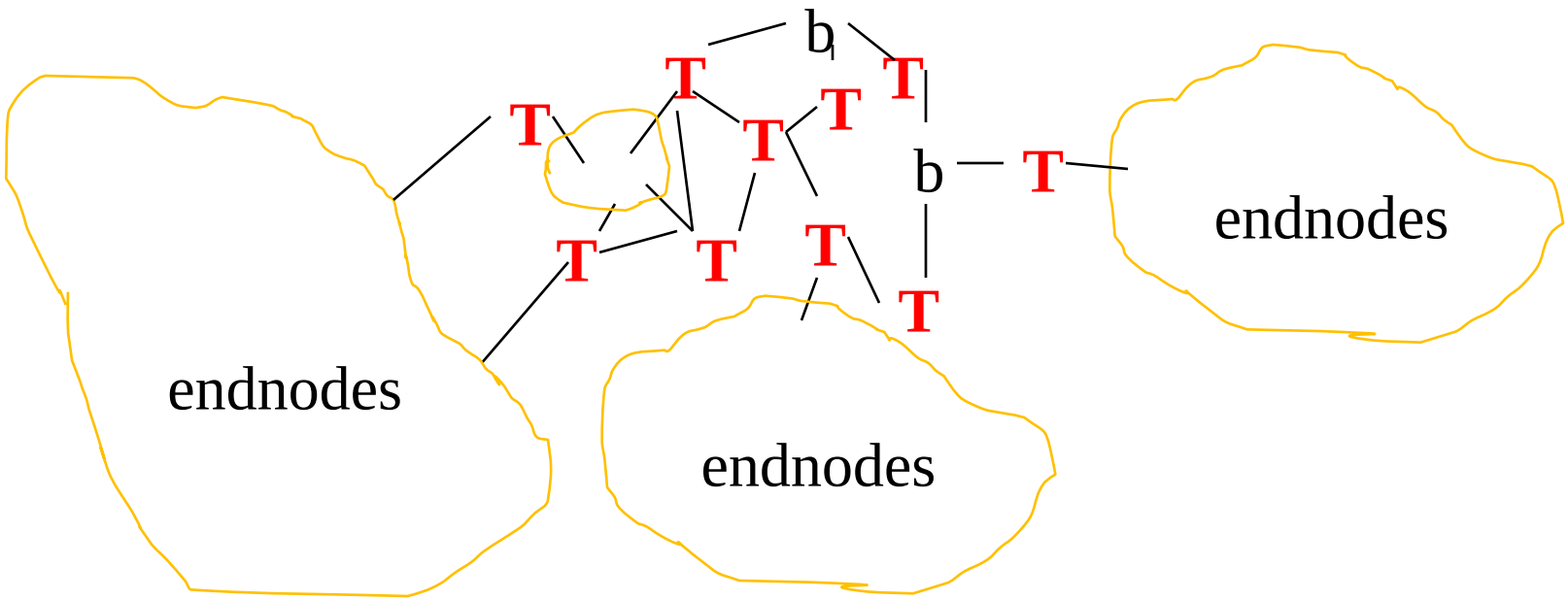
R1 specifies which tree (yellow, red, or blue)

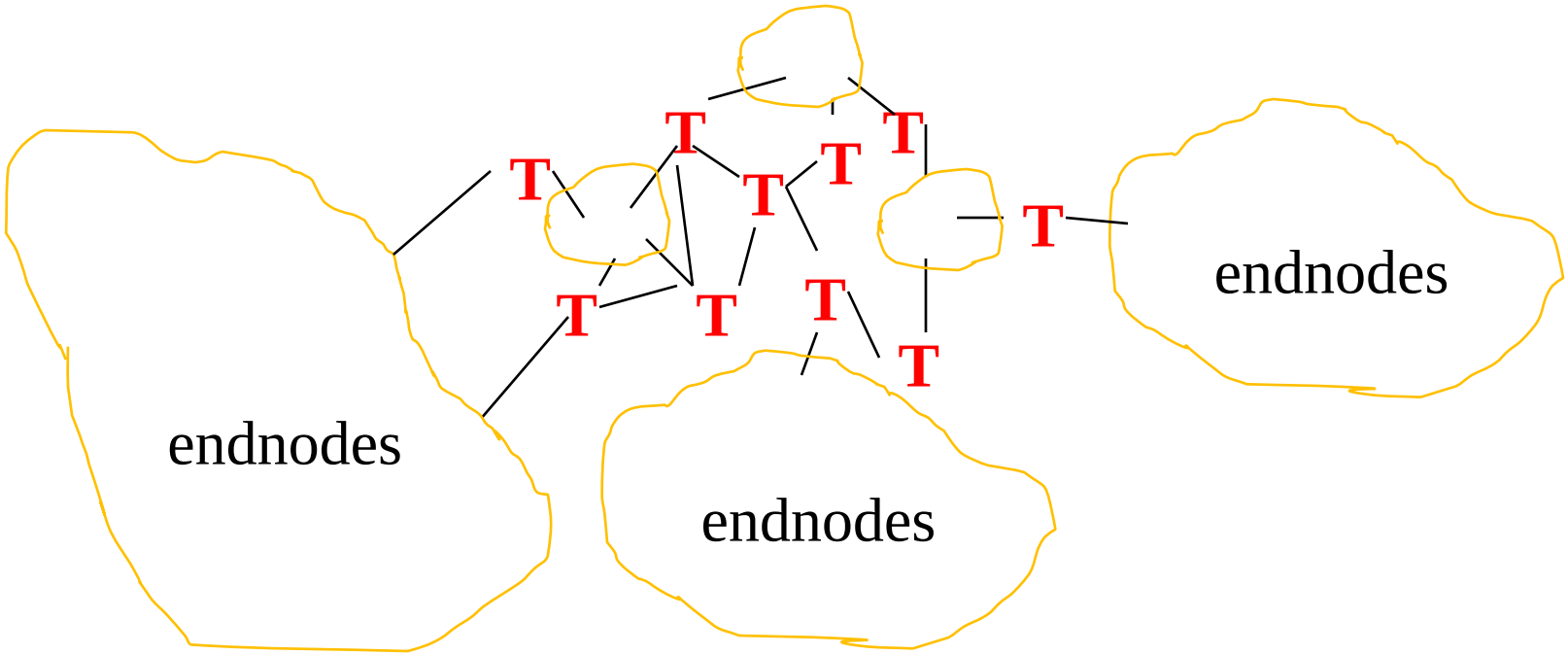
TRILL switches find each other, creating network of just TRILL switches




b
T
T
b
T
T
b
T
T
b
T
b
T
T
T
b
T
b


b
T
T
T
T
T
b
b
T
T
T
T
T
endnodes
endnodes
endnodes

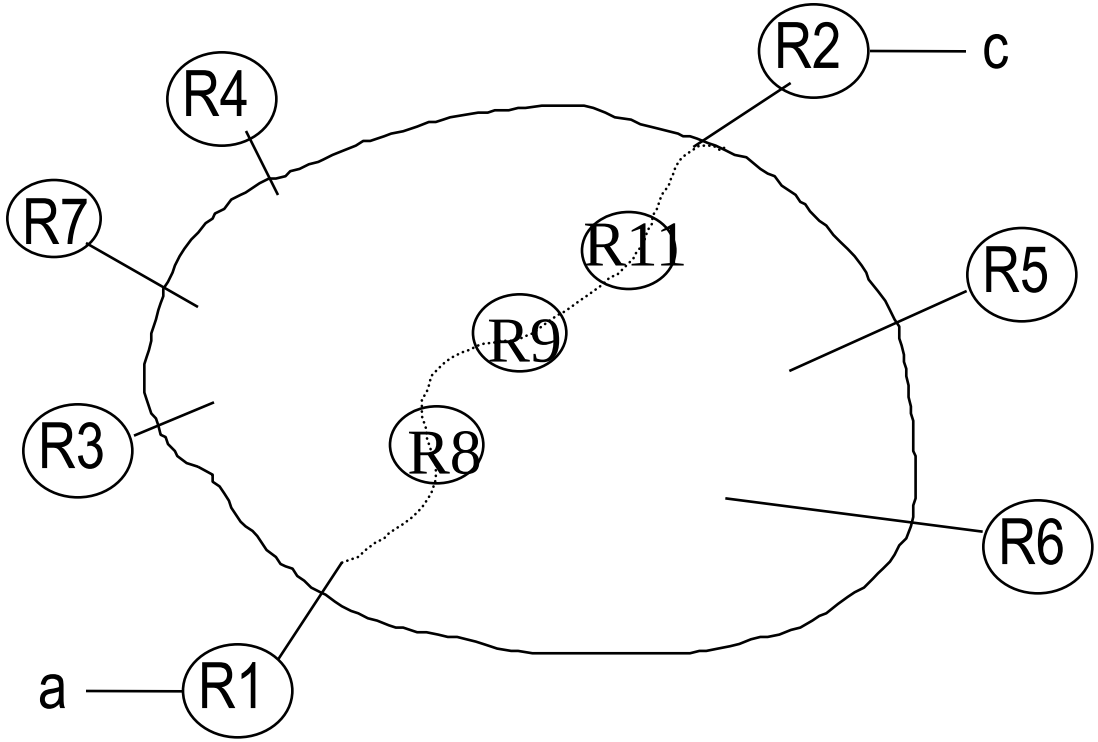

b
T
T
T
T
T
b
T
T
T
T
T
endnodes
endnodes
endnodes


T
T
T
T
T
T
T
T
T
T
endnodes
endnodes
endnodes

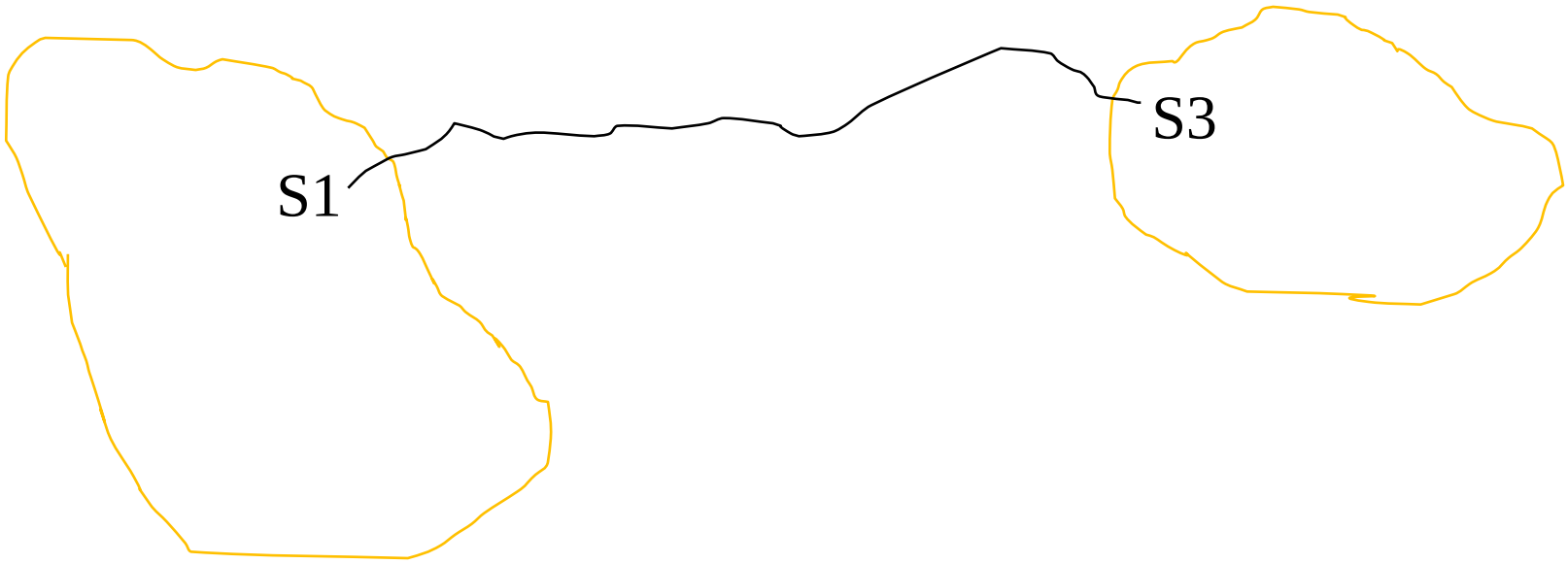
Reason for extra header

- Forwarding table in TRILL switches just size of # of TRILL switches (not # of endnodes)
- Layer 3-like header (hop count)
- Small, easy to look up, addresses (16 bits can be direct lookup rather than prefix match or hash)

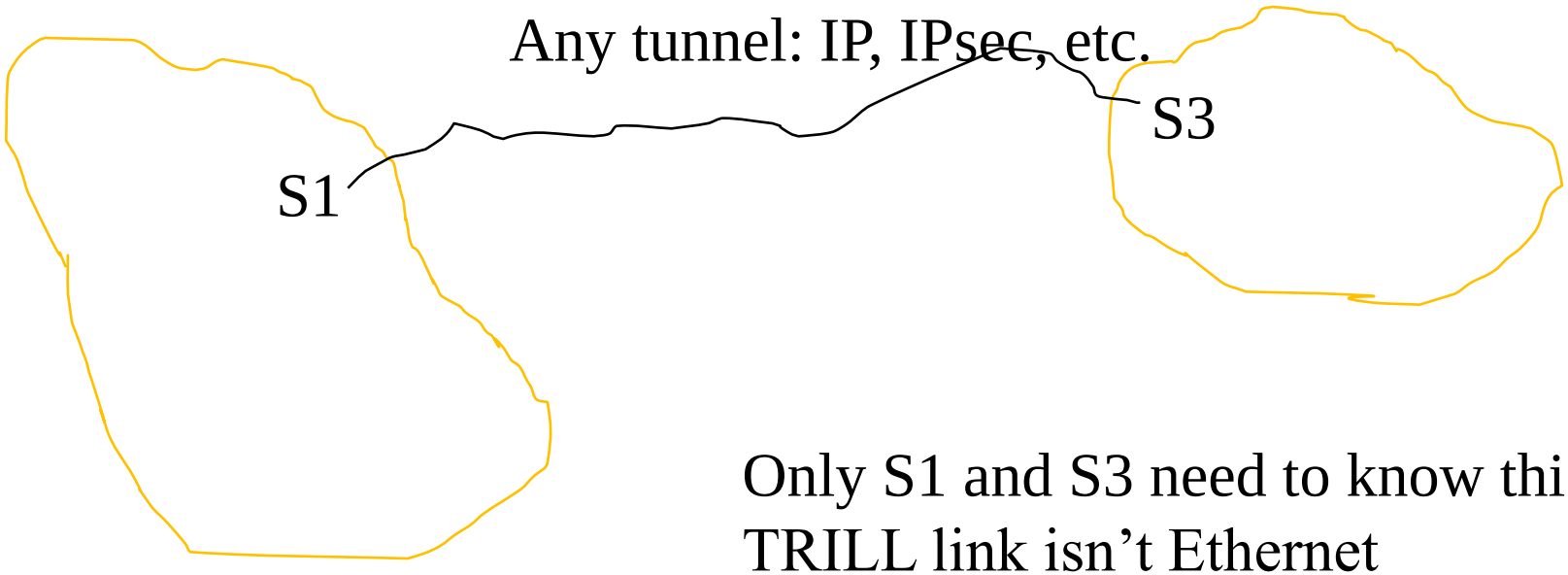
Switches inside cloud don't need to know about endnodes



Gluing two clouds together



Gluing two clouds together



Only S1 and S3 need to know this TRILL link isn't Ethernet

Advantage of CLNP vs IP+TRILL

- No need for ARP: no “layer 2 address”. It’s all part of layer 3

Some advantages of IP+TRILL vs CLNP

- Easier forwarding lookup inside cloud (16 bit nicknames allows direct lookup)
- Smaller forwarding table in switches inside cloud (table size # of switches rather than endnodes)
- Ability to multipath multicast
- VLANs

How does R1 know R2 is “last switch”?

- Orthogonal concept to rest of TRILL
- R1 needs table of (destination MAC, egress switch)
- Various possibilities
  - Edge switch learns when decapsulating data, floods if destination unknown
  - Configuration of edge switches
  - Directory that R1 queries
  - Central fabric manager pushes table

Orthogonal concept

Who encapsulates/decapsulates?

- Could be
  - first switch
  - Or hypervisor
  - Or VM
  - Or application
- Having endnode do it saves work for switch, easier to eliminate stale entries

Recently, a bunch of similar things invented

- NVGRE, VXLAN, …

How to compare

- "Inner" packet based on flat address space
  - IP or Ethernet…
    - IP header bigger, addresses smaller, well-known how to get unique Ethernet addresses without configuring
- "Outer" header location dependent
  - TRILL header small, nickname; simple forwarding lookup

So review some of what I said

- Proactive forwarding table vs incur latency when “flow” starts
- Distributed algorithm vs central fabric manager
- Load split based on fabric manager placing “flows” vs switch choosing among a set of next hops
- TRILL evolutionary, efficient outer header, Ethernet for flat cloud, can glue distant clouds together by having TRILL link which is a tunnel over IP

Questions?