Routing without tears: Bridging without danger

Radia Perlman
Sun Microsystems Laboratories
Radia.Perlman@sun.com

Before we get to RBridges

- Let’s sort out bridges, routers, switches...

What are bridges, really?

- Myth: bridges/switches simpler devices, designed before routers
- OSI Layers
  - 1: physical

Why this whole layer 2/3 thing?

- Myth: bridges/switches simpler devices, designed before routers
- OSI Layers
  - 1: physical
  - 2: data link (nbr-nbr, e.g., Ethernet)

Why this whole layer 2/3 thing?

- Myth: bridges/switches simpler devices, designed before routers
- OSI Layers
  - 1: physical
  - 2: data link (nbr-nbr, e.g., Ethernet)
  - 3: network (create entire path, e.g., IP)

Why this whole layer 2/3 thing?

- Myth: bridges/switches simpler devices, designed before routers
- OSI Layers
  - 1: physical
  - 2: data link (nbr-nbr, e.g., Ethernet)
  - 3: network (create entire path, e.g., IP)
  - 4 end-to-end (e.g., TCP, UDP)

Why this whole layer 2/3 thing?

- Myth: bridges/switches simpler devices, designed before routers
- OSI Layers
  - 1: physical
  - 2: data link (nbr-nbr, e.g., Ethernet)
  - 3: network (create entire path, e.g., IP)
  - 4 end-to-end (e.g., TCP, UDP)
  - 5 and above: boring

Definitions

- Repeater: layer 1 relay

Definitions

- Repeater: layer 1 relay
- Bridge: layer 2 relay

Definitions

- Repeater: layer 1 relay
- Bridge: layer 2 relay
- Router: layer 3 relay

Definitions

- Repeater: layer 1 relay
- Bridge: layer 2 relay
- Router: layer 3 relay
- OK: What is layer 2 vs layer 3?

Definitions

- Repeater: layer 1 relay
- Bridge: layer 2 relay
- Router: layer 3 relay
- OK: What is layer 2 vs layer 3?
  - The “right” definition: layer 2 is neighbor-neighbor. “Relays” should only be in layer 3!

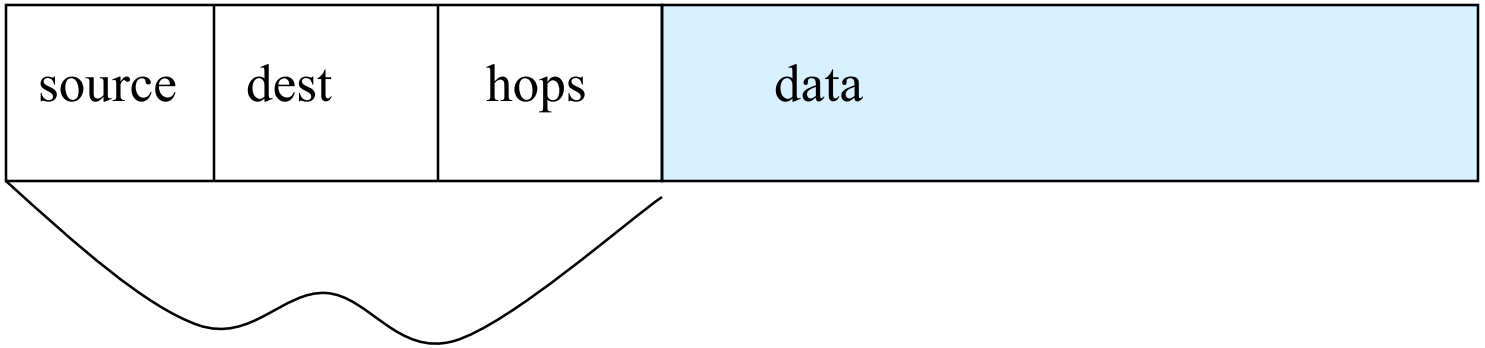
Definitions

- Repeater: layer 1 relay
- Bridge: layer 2 relay
- Router: layer 3 relay
- OK: What is layer 2 vs layer 3?
- True definition of a layer n protocol:
  Anything designed by a committee whose charter is to design a layer n protocol

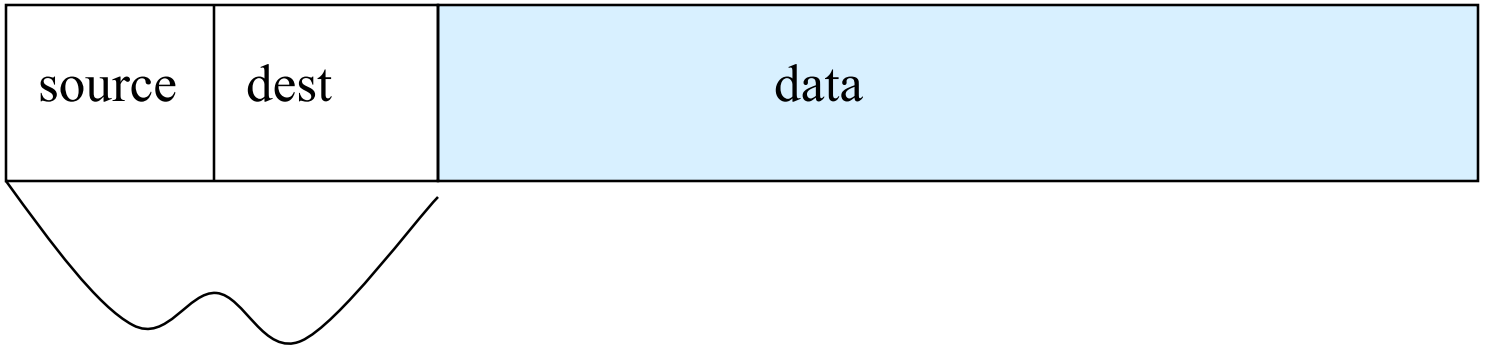
Layer 3 (e.g., IPv4, IPv6, DECnet, Appletalk, IPX, etc.)

- Put source, destination, hop count on packet
- Then along came "the EtherNET"
  - rethink routing algorithm a bit, but it's a link not a NET!
- The world got confused. Built on layer 2
- I tried to argue: "*But you might want to talk from one Ethernet to another*!"
- Thought Ethernet was a *competitor* to layer 3

Layer 3 packet

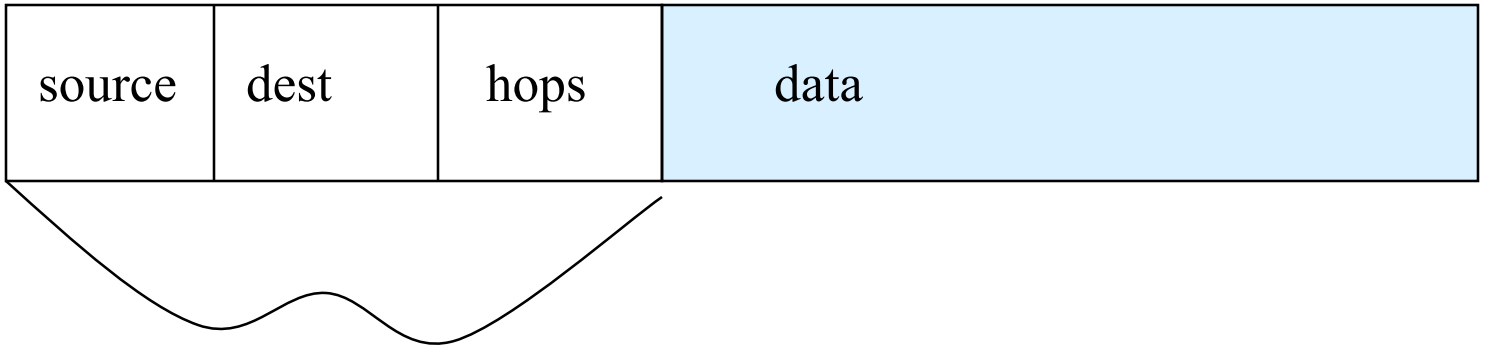


Ethernet packet



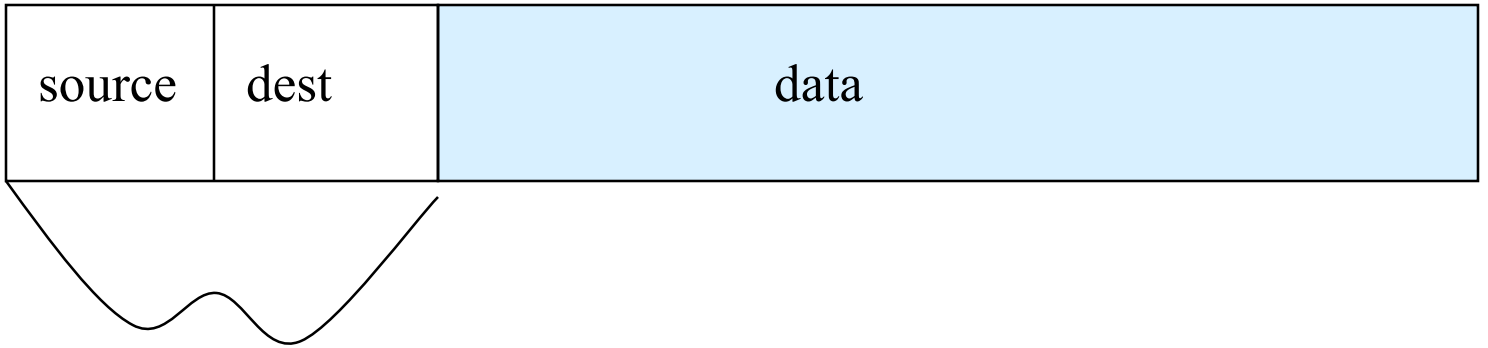
No hop count

Layer 3 packet



Addresses have topological significance

Ethernet packet



Addresses are “flat” (no topological significance)

It's easy to confuse "Ethernet" with "network"

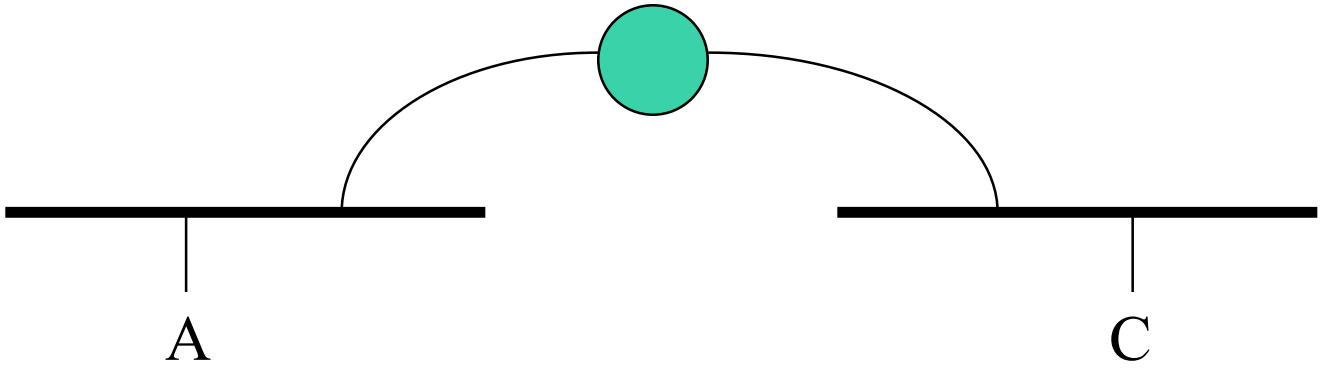
- Both are multiaccess clouds
- Why can't Ethernet replace IP?
  - Flat addresses
  - No hop count
  - Missing additional protocols (such as neighbor discovery)
  - Perhaps missing features (such as fragmentation, error messages, congestion feedback)

So, we had layer 3, and Ethernet

- People built protocol stacks leaving out layer 3
- There were lots of layer 3 protocols (IP, IPX, Appletalk, CLNP), and few multi-protocol routers

Problem Statement

Need something that will sit between two Ethernets, and let a station on one Ethernet talk to another



Basic idea

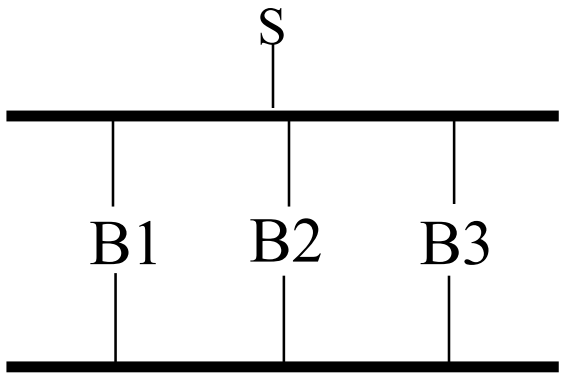
- Listen promiscuously
- Learn location of source address based on source address in packet and port from which packet received
- Forward based on learned location of destination

What's different between this and a repeater?

- no collisions
- with learning, can use more aggregate bandwidth than on any one link
- no artifacts of LAN technology (# of stations in ring, distance of CSMA/CD)

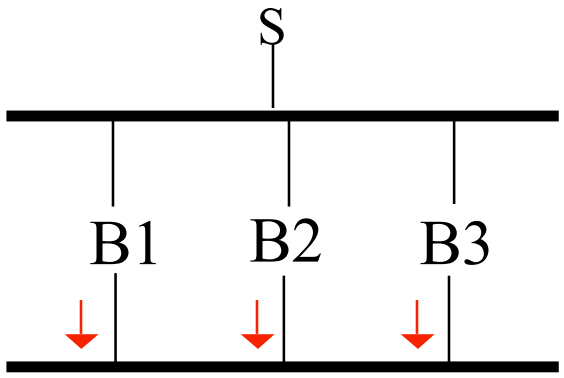
But loops are a disaster

- No hop count
- Exponential proliferation



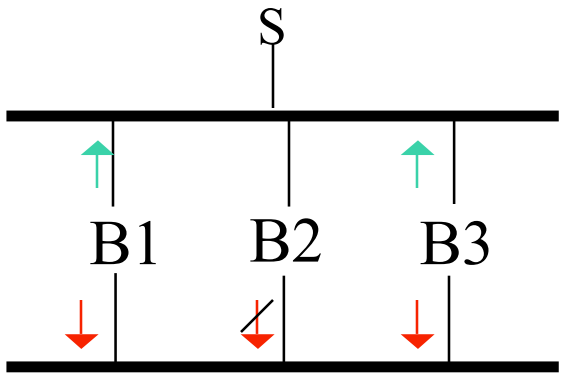
But loops are a disaster

- No hop count
- Exponential proliferation



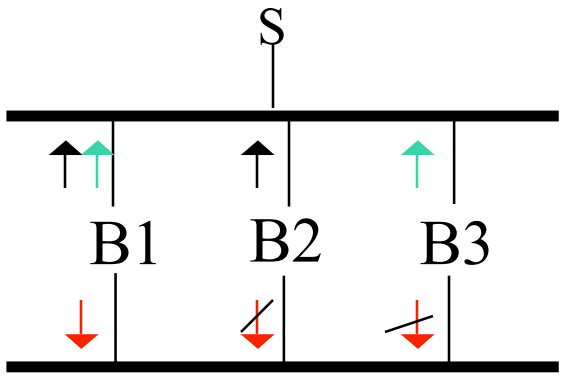
But loops are a disaster

- No hop count
- Exponential proliferation



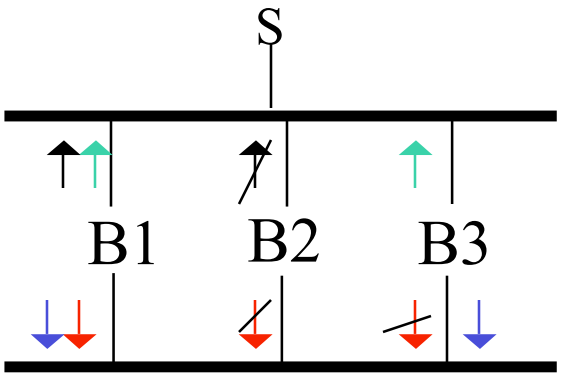
But loops are a disaster

- No hop count
- Exponential proliferation



But loops are a disaster

- No hop count
- Exponential proliferation



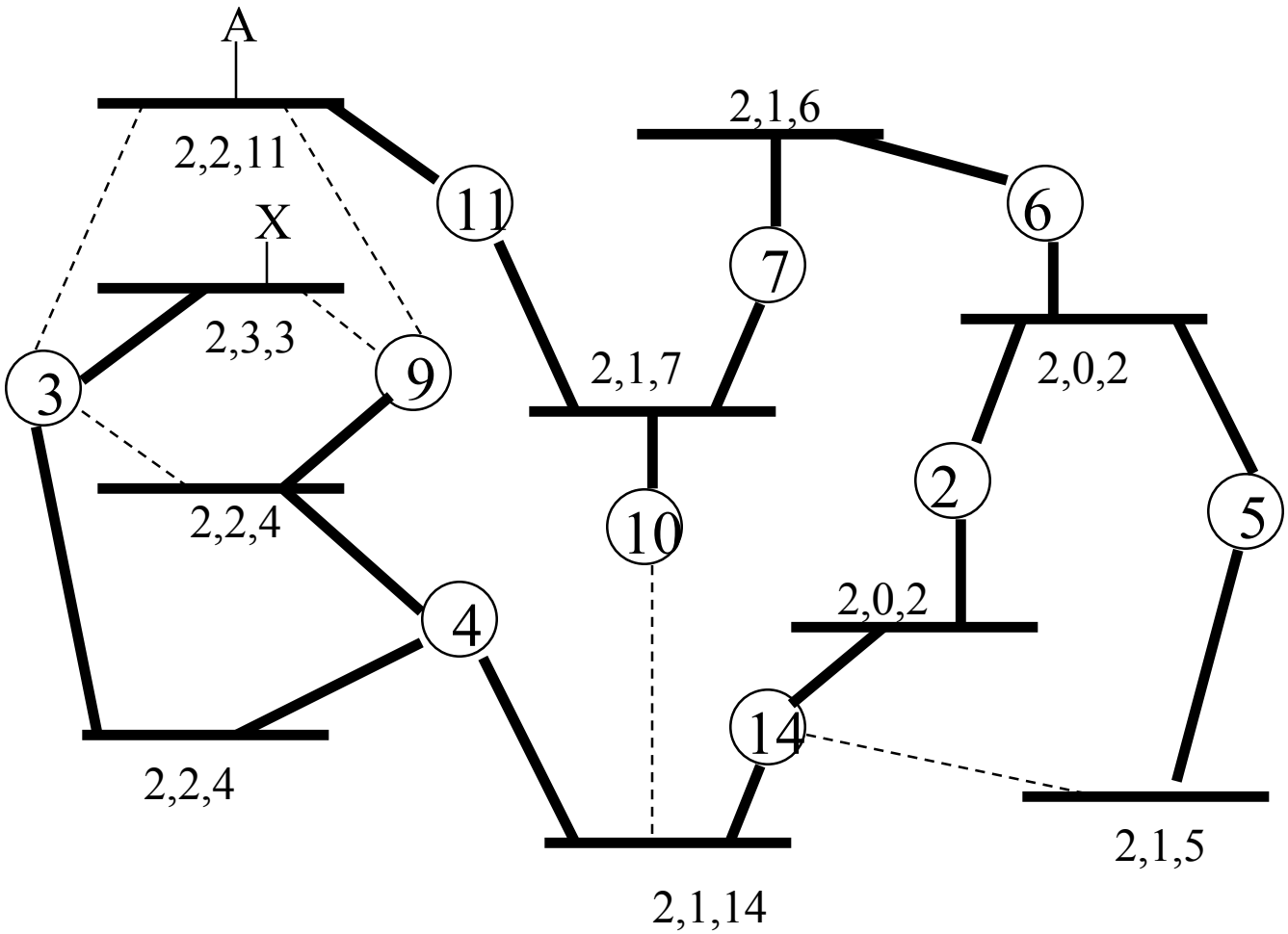
What to do about loops?

- Just say “don’t do that”
- Or, spanning tree algorithm
  - Bridges gossip amongst themselves
  - Compute loop-free subset
  - Forward data on the spanning tree
  - Other links are backups

Algorhyme

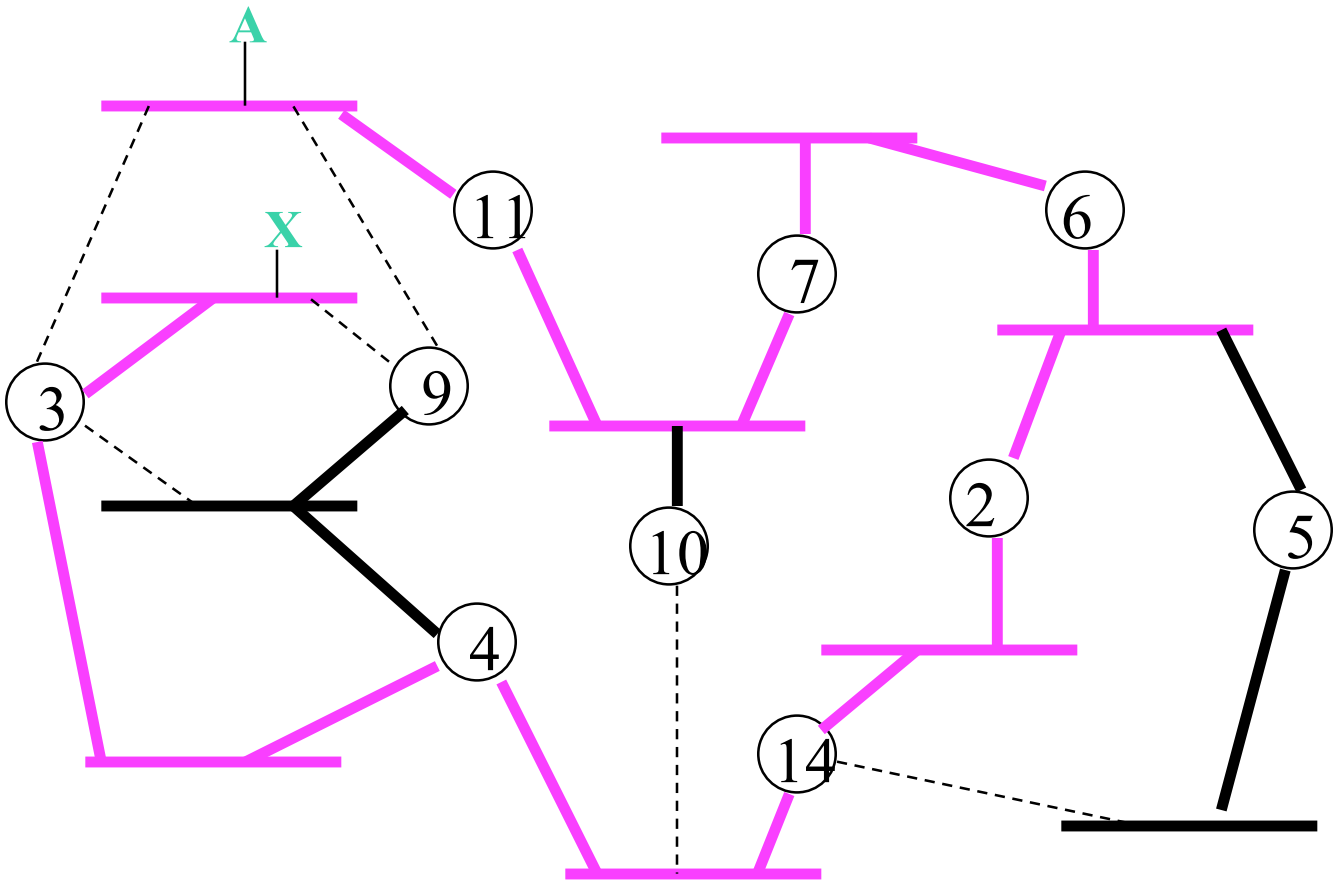
I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree which must be sure to span
So packets can reach every LAN.
First the Root must be selected
By ID it is elected.
Least cost paths from Root are traced
In the tree these paths are placed.
A mesh is made by folks like me.
Then bridges find a spanning tree.

Radia Perlman


A
2,2,11
11
X
2,3,3
3
9
2,2,4
4
2,2,4
2,1,6
7
6
2,1,7
2,0,2
2
5
10
2,0,2
14
2,1,5
2,1,14

Notice you don’t get optimal pairwise paths

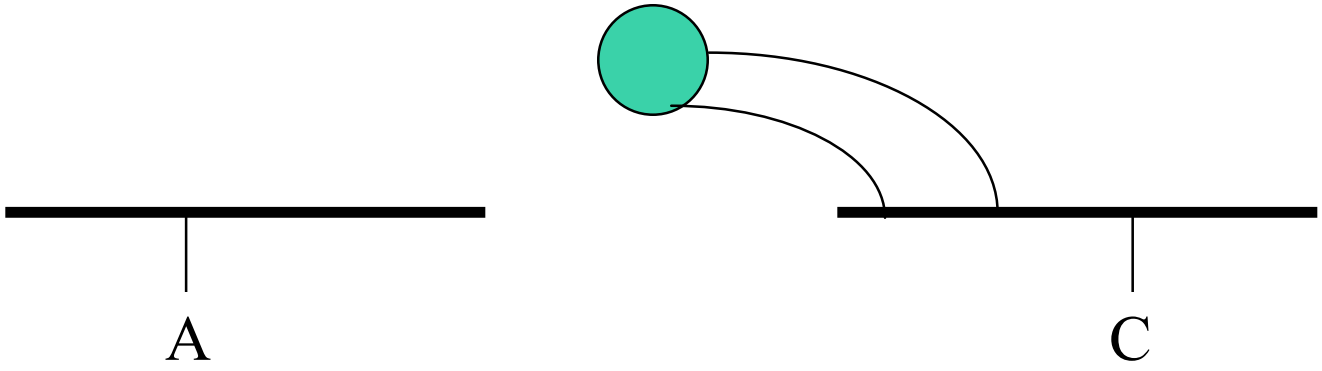
A talks to X



Bother with spanning tree?

- Maybe just tell customers “don’t do loops”
- First bridge sold...

First Bridge Sold



Bridges are cool, but…

- Routes are not optimal (spanning tree)
  - STA cuts off redundant paths
  - If A and B are on opposite side of path, they have to take long detour path
- Temporary loops really dangerous
  - no hop count in header
  - proliferation of copies during loops
- Traffic concentration on selected links

Bridge meltdowns

- They do occur (a Boston hospital)
- Lack of receipt of spanning tree msgs tells bridge to turn on link
  - So if bridge can't keep up with wire speed…
- In contrast with routers: lost messages will cause link to be brought down
  - Note: original Digital bridge spec said bridges had to be wire speed
- Also, some additions to bridging involve configuration, which if wrong…meltdowns

Why are there still bridges?

- Why not just use routers?
  - Bridges plug-and-play
  - Endnode addresses can be per-campus
- IP routes to links, not endnodes
  - So IP addresses are per-link
  - Need to configure routers
  - Need to change endnode address if change links

What if you routed to endnodes, not to links?

- Suppose you could have a whole campus, all with one prefix
- That's what DECnet/CLNP called "level 1 routing"
- Used a special protocol called "ES-IS"
  - Endnodes periodically announce to routers
  - Routers periodically announce to endnodes

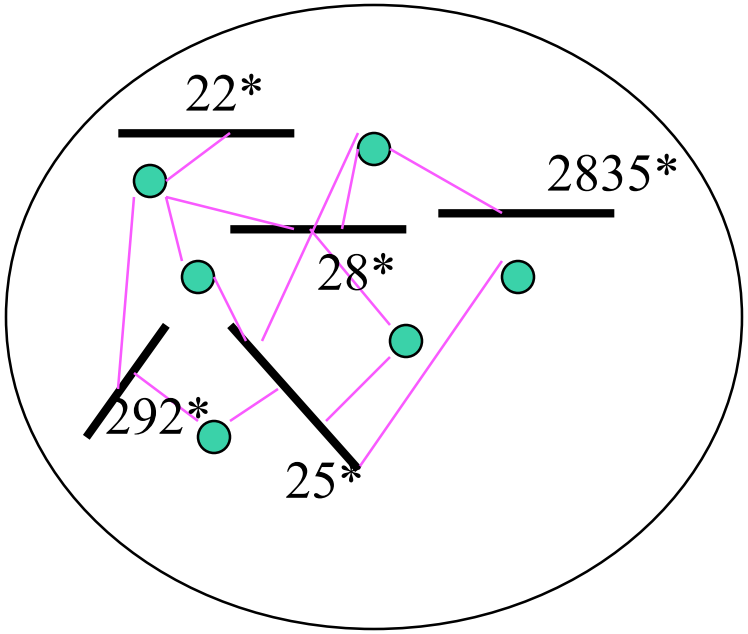
True “level 1” routing

- CLNP addresses had two parts
  - “area” (14 bytes…)
  - node (6 bytes)
- An area was a whole multi-link campus
- Two levels of routing
  - level 1: routes to exact node ID within area
  - level 2: longest matching prefix of “area”

Hierarchy

IP-style

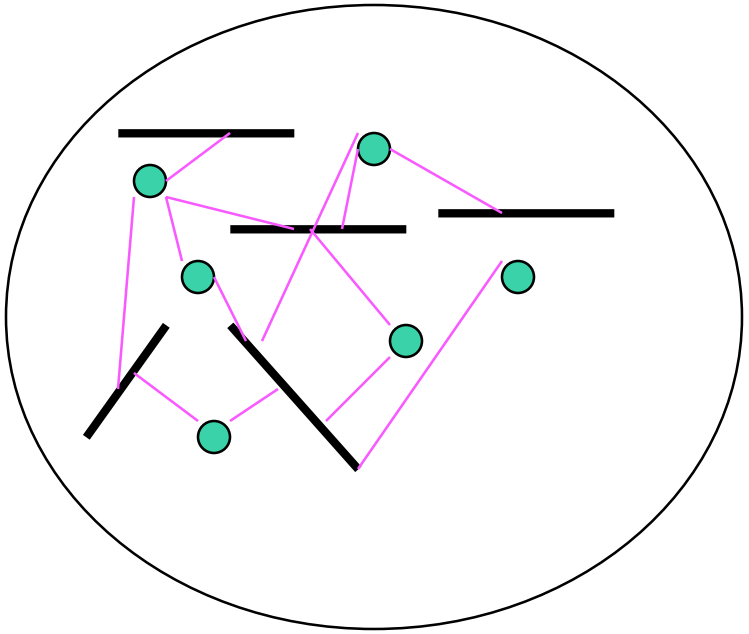
One prefix per link



2*

CLNP-style

One prefix per campus



2*

CLNP level 1 routing

- Autoconfiguration
  - Rtrs discover “area” prefix
  - Tell endnodes, which plug in their MAC to form their layer 3 address
- Rtrs tell each other (using link state routing protocol), within area, location of all endnodes in area

“Level 1 routing” with IP

- IP has never had true level 1 routing
  - Each link has a prefix
  - Multilink node has two addresses
  - Move to new link requires new address
- Bridging is used with IP to sort of do “level 1 routing”
  - But not as good: spanning tree paths rather than optimal pt-to-pt paths, meltdowns

One prefix per campus vs per link

- Advantages
  - Zero configuration of routers within campus
  - Move nodes within campus without changing address
  - Multiple points of attachment: same address
  - Don’t partition address space
- Disadvantages
  - Bigger routing tables of level 1 routers

Bridging vs CLNP-style level 1 Routing

- Better routes: optimal pt-to-pt routes, can use all links, can path split, do traffic engineering
- Stable protocol (lost messages bring link *down*, not up)
- Forwarding with safe hdr (hop count, and specify next hop)
- But CLNP depended on ES-IS: We'll have to do our best without it

Link State Routing

- meet nbrs
- Construct Link State Packet (LSP)
  - who you are
  - list of (nbr, cost) pairs
- Broadcast LSPs to all rtrs (“a miracle occurs”)
- Store latest LSP from each rtr
- Compute Routes (breadth first, i.e., “shortest path” first—well known and efficient algorithm)

IS-IS

- A specific link state protocol
- Similar to OSPF, but more suitable for RBridges because
  - No need to configure IP addresses (which OSPF depends on)
  - Easy to add new fields with TLV encoding

What we'd like

- Best features of bridges
  - Transparency to endnodes
  - Plug-and-play switches
- Best features of routers
  - Optimal paths
  - Safe header for forwarding
- Interoperate with existing routers, endnodes, and bridges

RBridges

- Compatible with today's bridges and routers
- Like routers, terminate bridges' spanning tree
- Like bridges, glue LANs together to create one IP subnet (or for other protocols, a broadcast domain)
- Like routers, optimal paths, fast convergence, no meltdowns
- Like bridges, plug-and-play

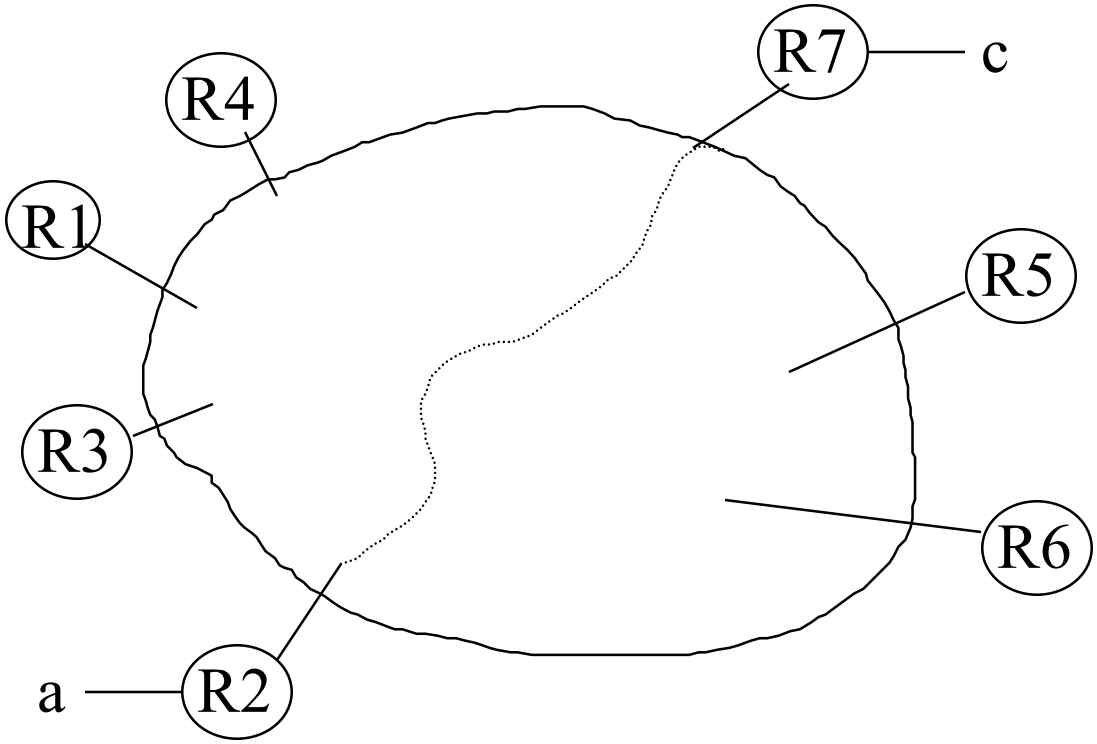
RBridging layer 2

- Link state protocol among Rbridges (so know how to route to other Rbridges)
- Like bridges, learn location of endnodes from receiving data traffic
- But since traffic on optimal paths, need to distinguish originating traffic from transit
- So encapsulate packet

“Layer 2” routing

- Think of it just like routers
  - At each hop, add a layer 2 header to get to the next hop RBridge
  - The “destination” is the egress RBridge
- First RBridge
  - Looks up destination endnode, finds egress RB
  - Adds shim header, forwards to egress RB
- Egree RBridge decapsulates

Rbridging



Encapsulation Header

S=Xmitting Rbridge D=Rcving Rbridge pt="transit"	hop count In/out RBridge	original pkt (including L2 hdr)

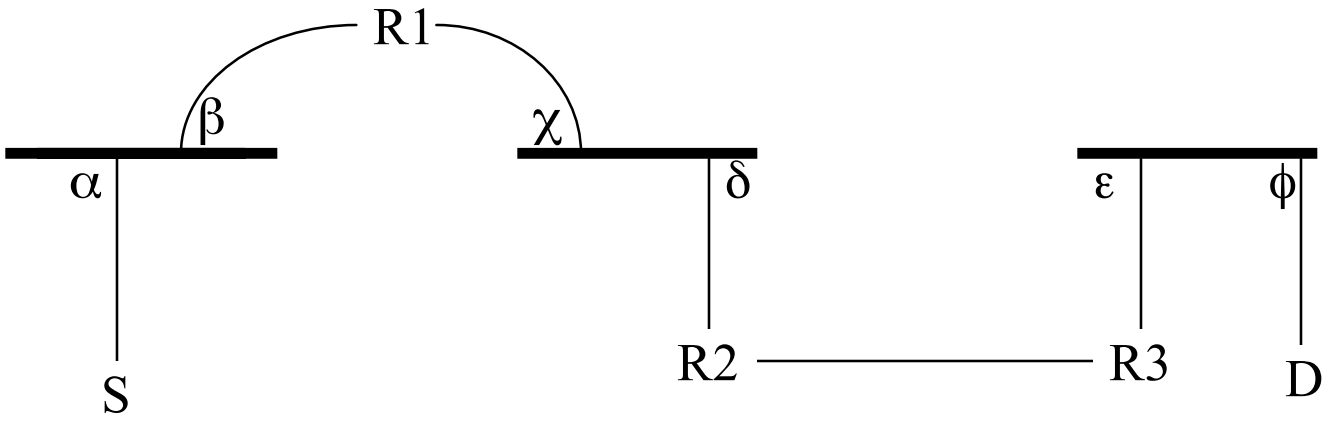
Outer header: new p-t: otherwise, ordinary Ethernet hdr

Shim: hop count, and ingress or egress RBridge

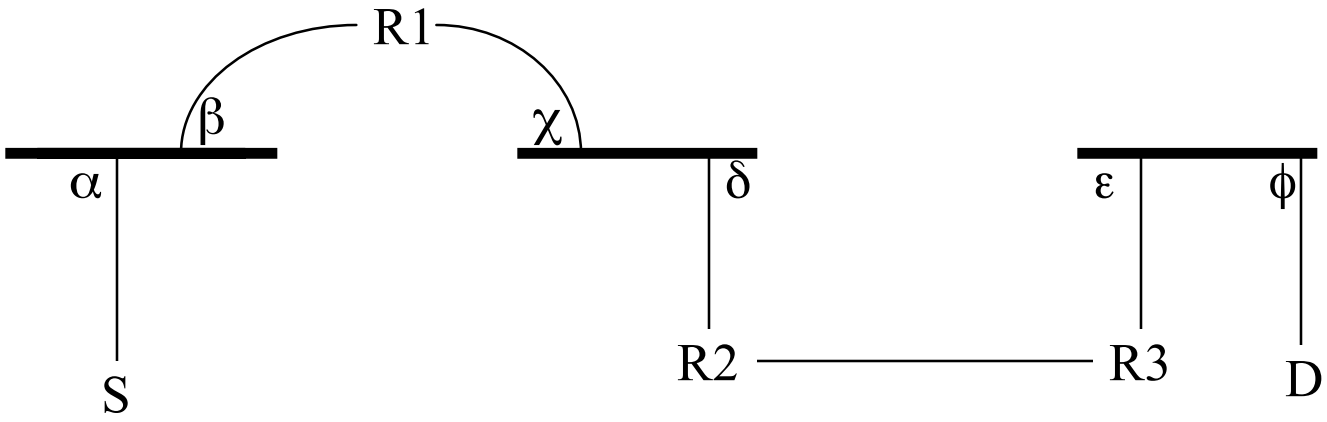
If unicast, it is egress RBridge

If multicast or unknown destination, it is ingress RBridge

Hdrs inside hdrs



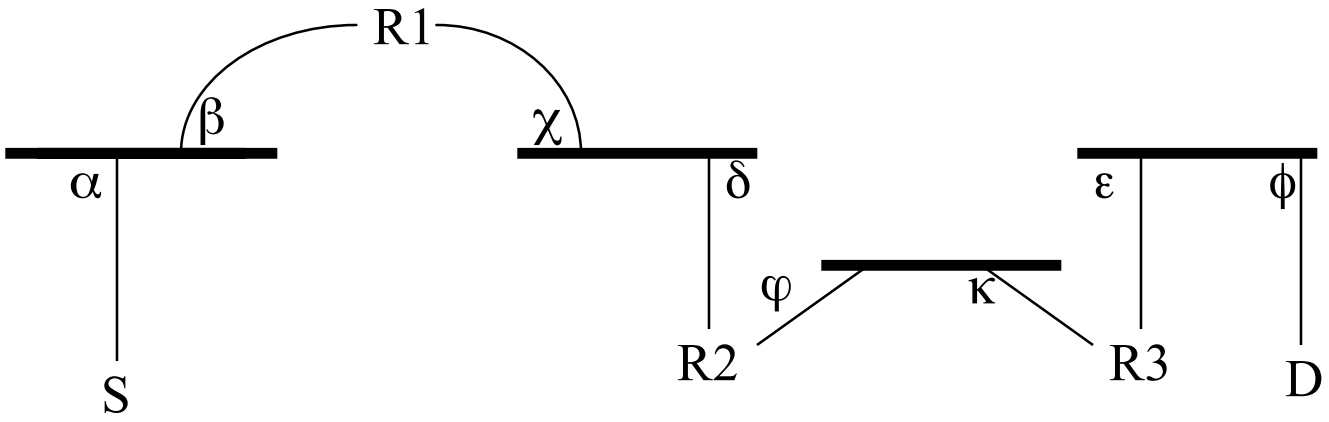
Hdrs inside hdrs

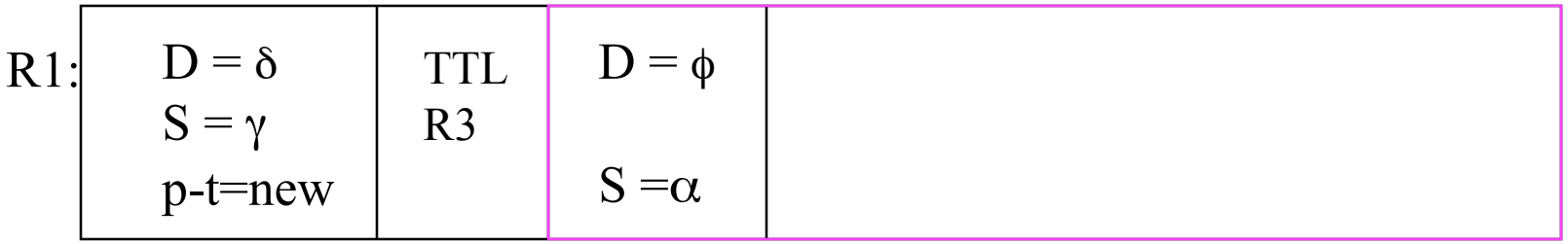


S:

D = ϕ S =α	

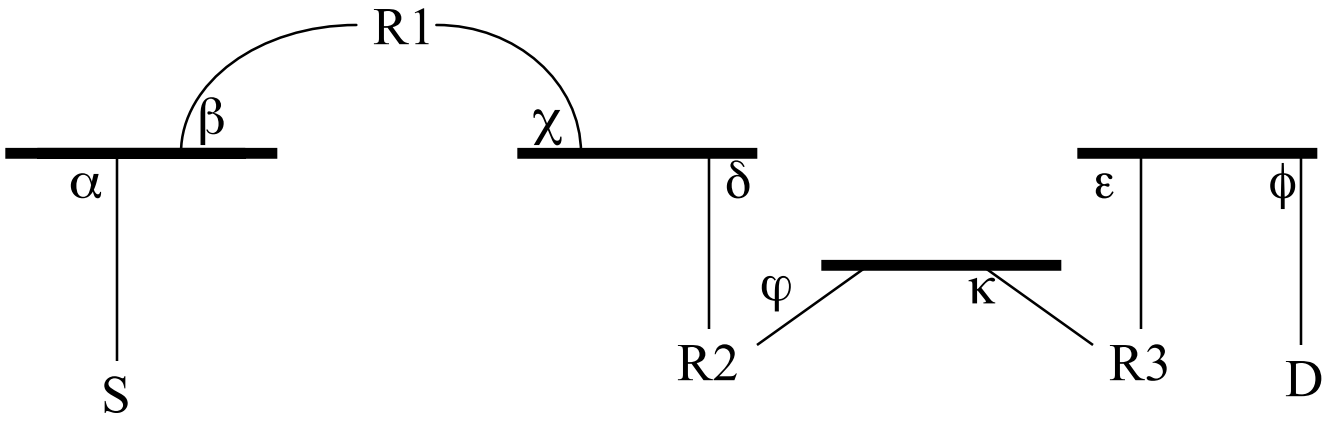
Hdrs inside hdrs

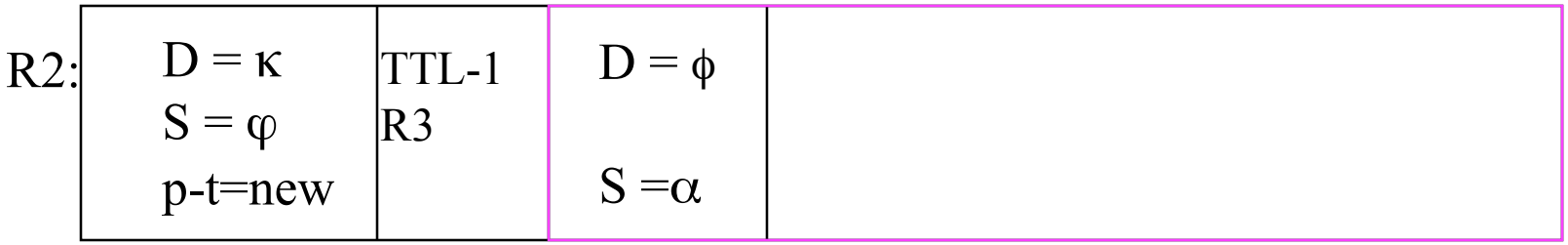




shim

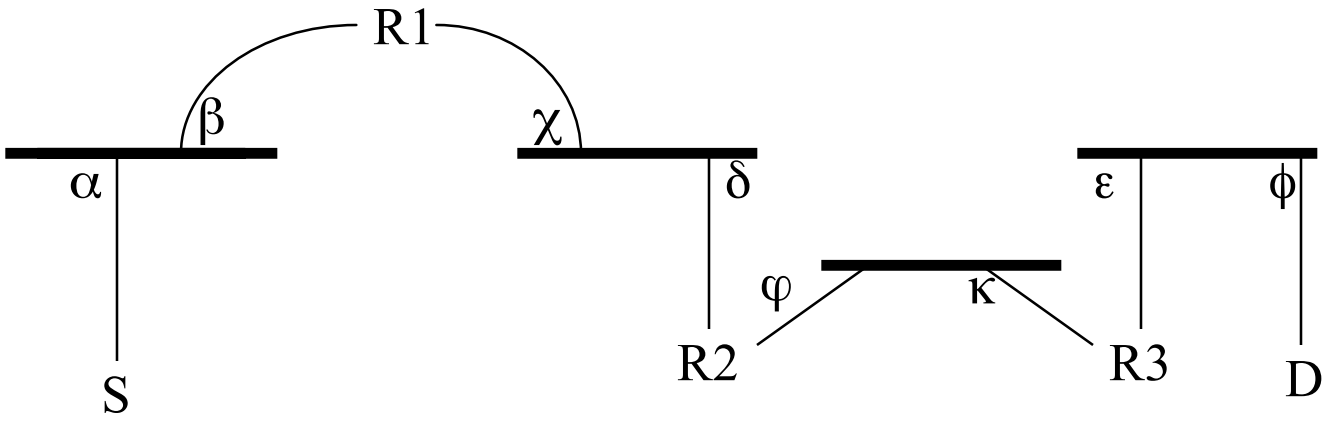
Hdrs inside hdrs





shim

Hdrs inside hdrs



R3:

D = ϕ S =α	

Flooded traffic

- Some traffic needs to be sent to all links
  - Unknown destinations
  - Multicast traffic
- Could use a single spanning tree
  - Spanning tree computed from link state database (not separate spanning tree protocol)
- But we decided on per-ingress trees

Why per-ingress trees?

- Yeah, it's more computation (but not more protocol messages…link state database gives all the info you need)
- Advantages
  - IP multicasts, and VLAN delivery not to ***all*** links, so optimized delivery
  - Packets won't get out of order when switching from unknown to known (take same path)

VLANs

- VLANs allow partitioning of a bridged campus
- A packet marked with a VLAN tag must only be delivered to links in that VLAN
- With bridges, VLAN membership of links is configured into the bridges
- Usually bridges add and remove the VLAN tag

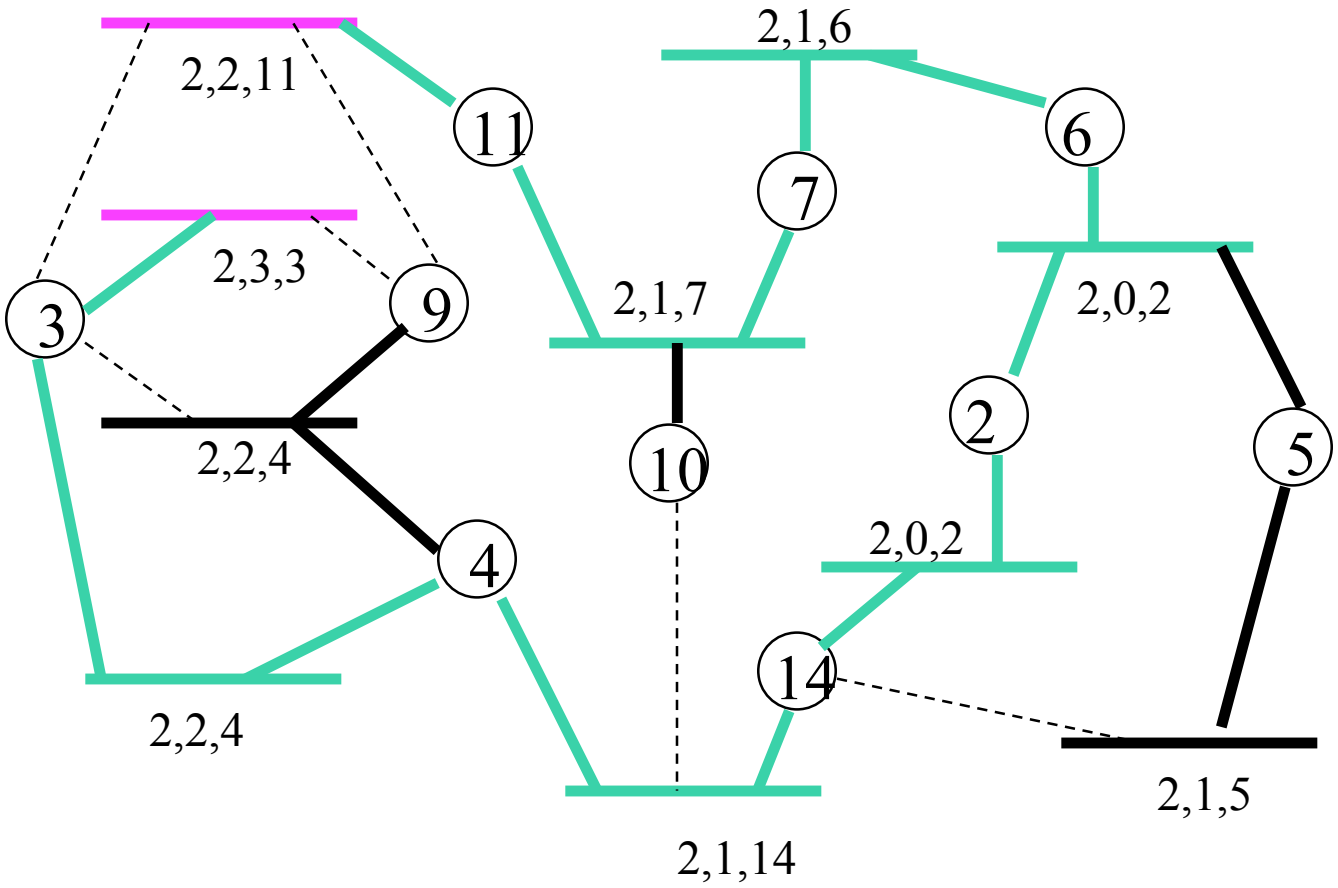
Calculating a spanning tree

- No need for separate spanning tree protocol
- Link state protocol gives enough info
- For per-RBridge: calculate a spanning tree rooted at that RBridge

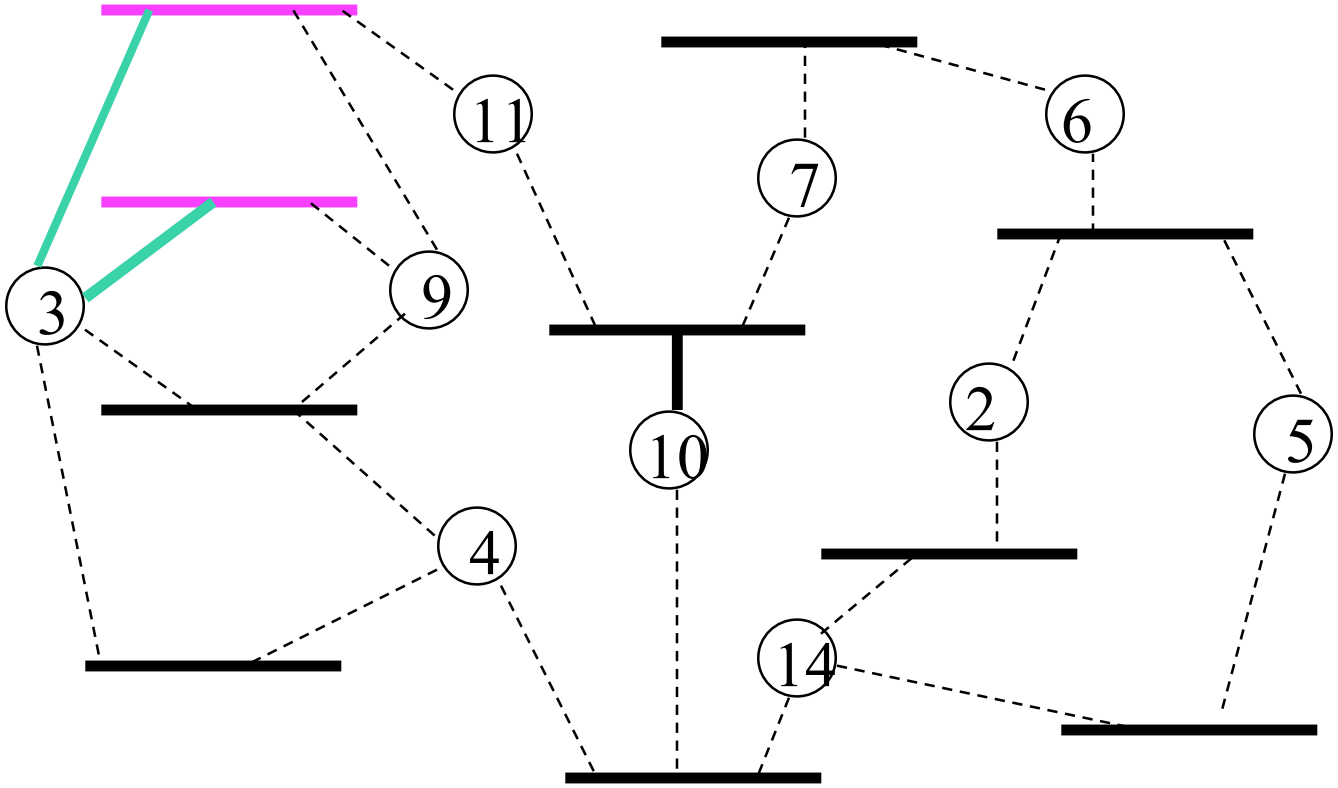
Need a broadcast domain per VLAN

- That's the definition of a VLAN
- A packet for VLAN A must only be delivered to links in VLAN A
  - Could be filtered at egress RBridges
    - Makes forwarding info for intermediate RBridges trivial, but packet delivery more expensive
  - Or could be filtered if no downstream receivers for VLAN A

Delivery path for VLAN A if one spanning tree, and pinks are VLAN A



Delivery path for VLAN A if per VLAN spanning tree, and pinks are VLAN A



Likewise for IP-derived multicasts

- If receivers are on just a few links, then it would be worth calculating per-RBridge spanning tree
- IP multicast has IGMP to let RBridges know where receivers are

Filtering info

- With spanning tree, have a set of ports
- For each port, mark what VLANs, and IP multicasts are downstream on that branch
- Select the spanning tree (based on ingress RBridge from shim header, or VLAN tag), then apply filtering rules

Shim Header

- Information needed
  - Hop count
  - Ingress RBridge
  - Egress RBridge
- But never need BOTH ingress and egress RBridge, so can overload the field

MPLS-like header

- Some people argued that some hardware already adds 4-byte shim header (for MPLS)
- MPLS contains TTL, and “label”
- But “label” is 20 bits
- How to fit 6-byte address into 20 bits?

Nicknames

- Piggyback “nickname acquisition protocol” onto link state protocol
- Stick “I want this nickname” into LSP
- If someone else claims your nickname, whoever has lower system ID keeps the nickname; the other one chooses a different one

Endnode Learning

- On shared link, only one Rbridge (DR) can learn and decapsulate onto link
  - otherwise, a "naked" packet will look like the source is on that link
  - have election to choose which Rbridge
- When DR sees naked pkt from S, announces S in its link state info to other Rbridges

Pkt Forwarding: Ingress RBridge

- If D known: look up egress RBridge R2, encapsulate, and forward towards R2
- Else, send to “destination=flood”, meaning send on spanning tree
  - Which tree: “ingress RBridge”
  - each DR decapsulates

IP optimization: proxy ARP

- For IP, learn (layer 3, layer 2) from ARP (ND) replies
- Pass around (layer 3, layer 2) pairs in LSP info
- Local RBridge can proxy ARP (i.e., answer ARP reply) if target (layer 3, layer 2) known

Possible IP optimization: tighter aliveness check

- Can check aliveness of attached IP endnodes by sending ARP query
- Can assume endnode alive, until you forward traffic to it, or until someone else claims that endnode

VLANs

- VLAN A endnodes only need to be learned by RBridges attached to VLAN A
- All RBridges must be able to forward to any other RBridge
- Egress RBridge in the encapsulation header

Endnode Information

- Only need to know location of VLAN A endnodes if you are attached to VLAN A
- So, different instances of IS-IS
  - One for basic RBridge-RBridge connectivity
  - One per VLAN, to share endnode membership for that VLAN

Possible variant

- Don’t explicitly pass around endnode information
- Instead, egress RBridge learns (ingress RBridge, source) from data packet

Decided not to do that because

- Sometimes layer 2 is definitive about enrollment (so better than caches and timeouts)
- Won’t need to flood as often
- Don’t want to starve between two bales of hay

Conclusions

- Routing without tears
  - Zero configuration, optimal paths
- Bridging without danger
  - No meltdowns
- Can gradually replace bridges with RBridges

Algorhyme v2

I hope that we shall one day see
A graph more lovely than a tree.
A graph to boost efficiency
While still configuration-free.
A network where RBridges can
Route packets to their target LAN.
The paths they find, to our elation,
Are least cost paths to destination.
With packet hop counts we now see,
The network need not be loop-free.
RBridges work effectively.
Without a common spanning tree.

Ray Perlner