

Rbridges: Transparent Routing

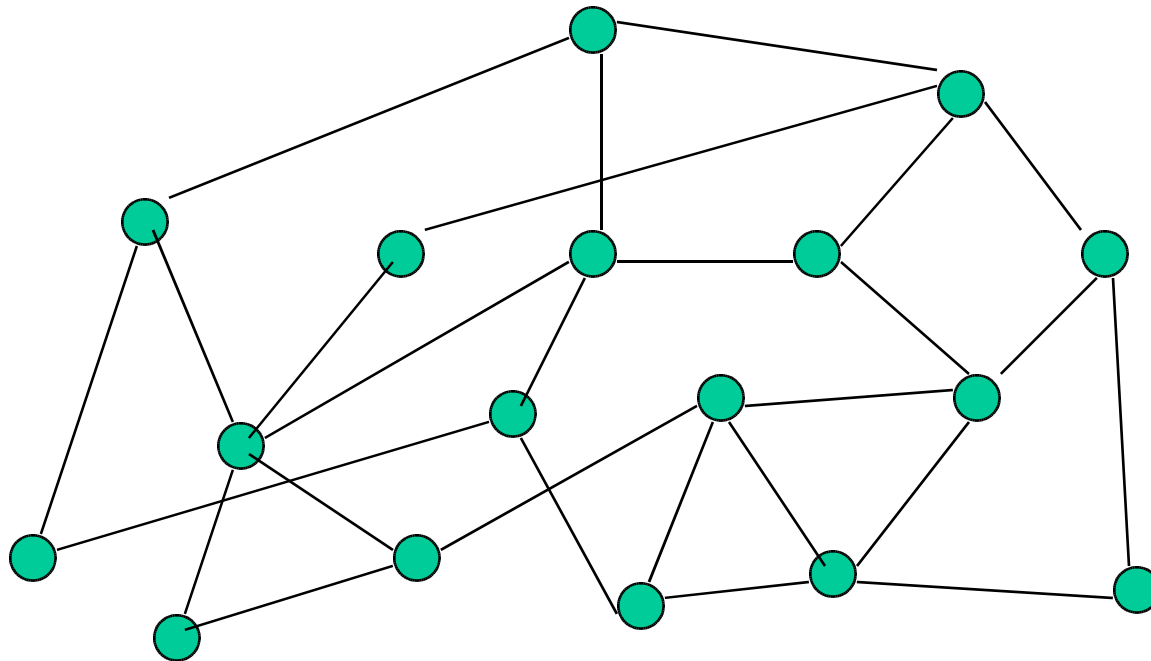
Radia Perlman

radia.perlman@sun.com

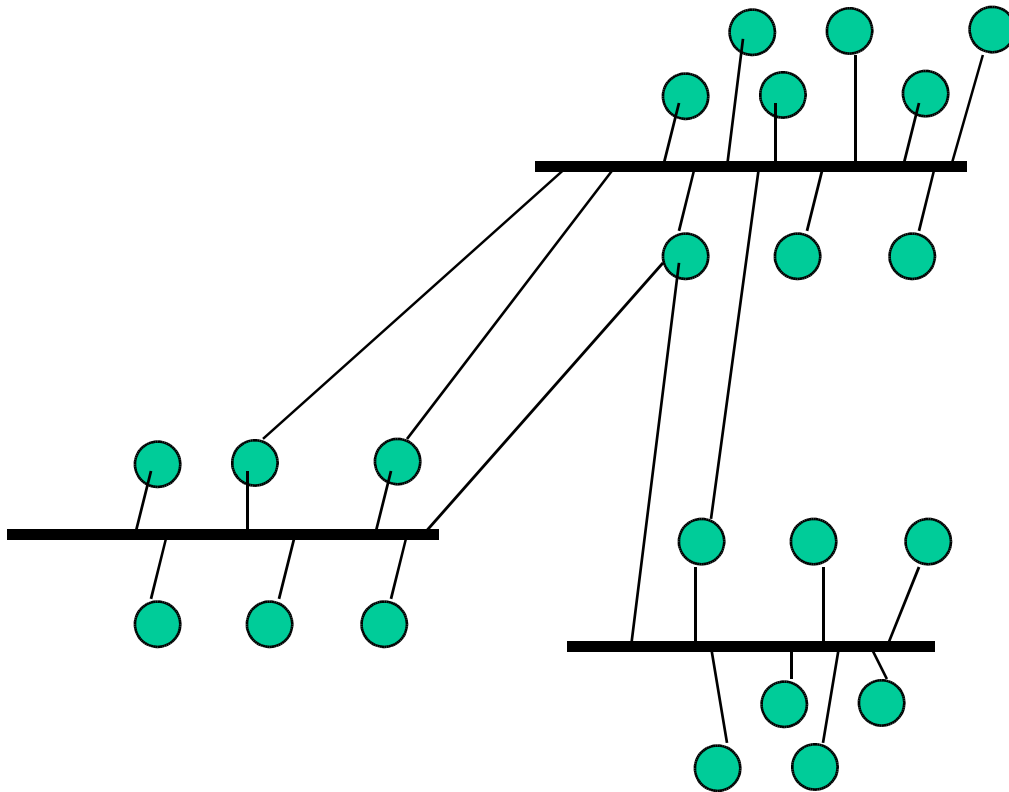


**2004
Sun Labs
Open House**

Routing Before Ethernet



With Ethernets

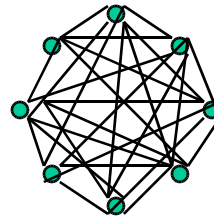


Rethink Routing Algorithm

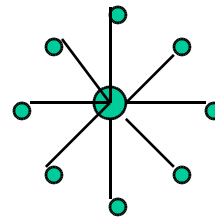
- Overhead proportional to number of links:
don't want n^2 links
- Don't want n^2 acknowledgements

Incorporating LANs

Instead of:



Add a “pseudonode”.
Make it look like this:



But People Thought Ethernet was a Competitor to DECnet (or IP)

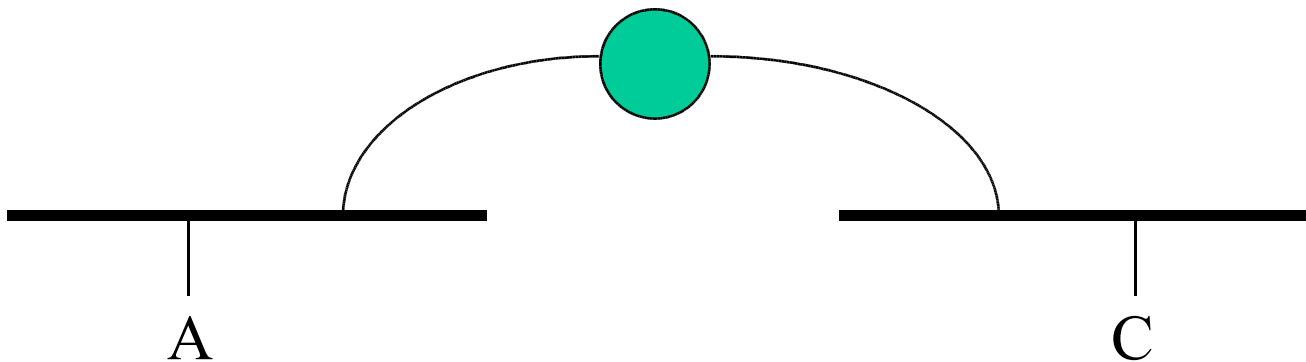
- Ethernet is a multi-access link, not a “network”
- Calling it a “network” confused the world
- People built directly on layer 2
- Routers require endnodes to cooperate by implementing layer 3

Why Isn't Ethernet Scalable?

- Addresses flat
- No hop count
- Missing associated protocols (such as fragmentation, ARP)
- Ethernet designed to be a single link
- I tried to get people to build on layer 3
- I failed

Problem Statement

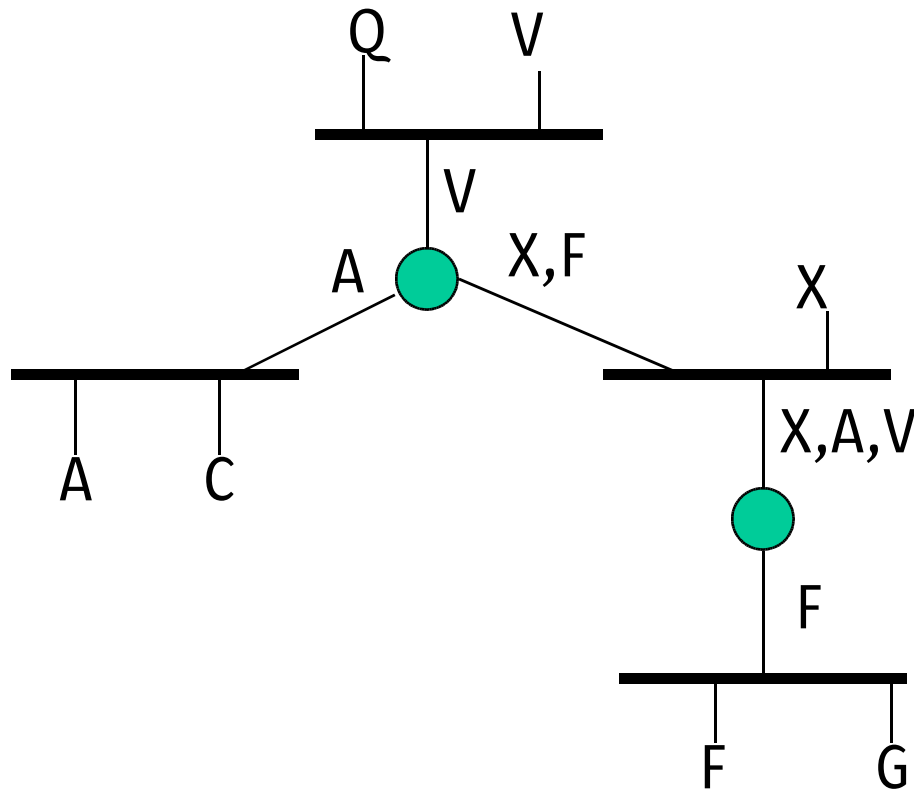
Need something that will sit between two Ethernets, and let a station on one Ethernet talk to another



Basic Idea of Transparent Bridge

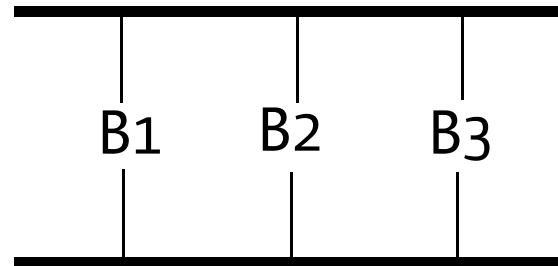
- Listen promiscuously
- Learn location of source address based on source address in packet and port from which packet received
- Forward based on learned location of destination
- When in doubt, forward

Station Learning



But Loops Are a Disaster

- No hop count
- Exponential proliferation



What to Do About Loops?

- Give up on bridges: “I guess that idea didn’t work”
- Document the restriction
- Spanning tree algorithm--automatically and continuously prune the topology to use a loop-free subset

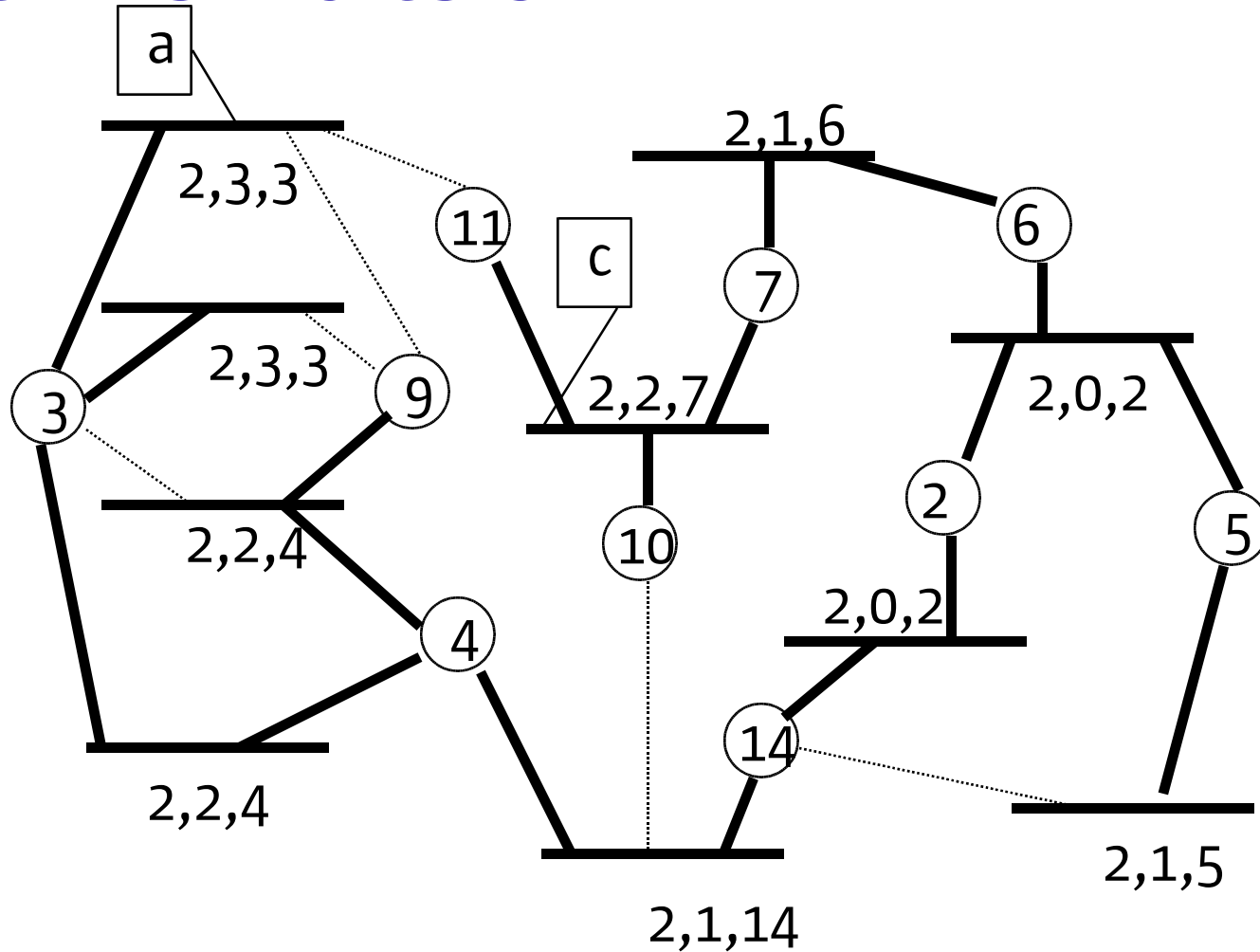
Thus the Spanning Tree Algorithm

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.

Problems with Bridges

- 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
 - So, should be conservative in transition

Path from a to c



Why Bridge Temporary Loops Are a Disaster

- No hop count – looping packets don't go away
- Exponential proliferation – routers only forward in one direction. Routers also specify next hop.

Why Bridges Are Slow to Start Forwarding

- Temporary loops might cause meltdown
- Can't (except in certain special cases, like a port to an endnode) know if turning on a link might cause temporary loop
- Simple solution: wait before turning on link, so other bridges can turn off links first
- People want instant failover (but they don't want meltdowns)

Bridge Meltdowns

- They do occur (a Boston hospital)
- Lack of receipt of spanning tree msgs tells bridge to turn on link
- So if too much traffic causes spanning tree messages to get lost...
 - loops
 - exponential proliferation of looping packets

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 IP address if change links

Using Bridges with IP

- With IP, each link has a prefix
- Multilink node has two addresses
- Move to new link requires new address
- Bridging is used to create a campus in which all nodes share the same prefix
- But bridging isn't as good as routing

Review: Why Bridging Not As Good As Routing

- Suboptimal paths
- Traffic concentration (onto links chosen by the spanning tree)
- Temporary loops a disaster
 - Therefore need to be conservative about turning on links (slow failover or meltdowns)

What We'd Like: Best of Both Worlds

- Keep transparency to endnodes and zero configuration
- Optimal paths
- Make temporary loops safe
 - Fast failover
 - No meltdowns

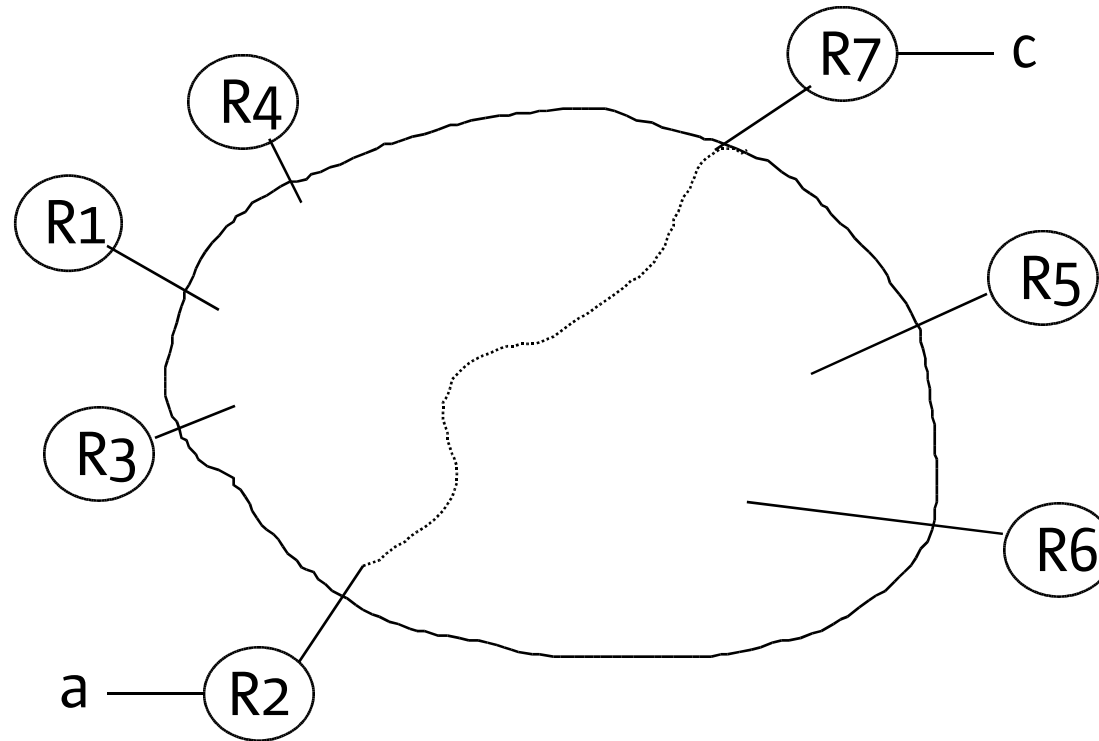
Rbridges

- Compatible with today's bridges and routers
- Like routers, terminate bridged LAN
- 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 (or ARP replies)
- But since traffic on optimal paths, need to distinguish originating traffic from transit
- So encapsulate packet

Rbridging



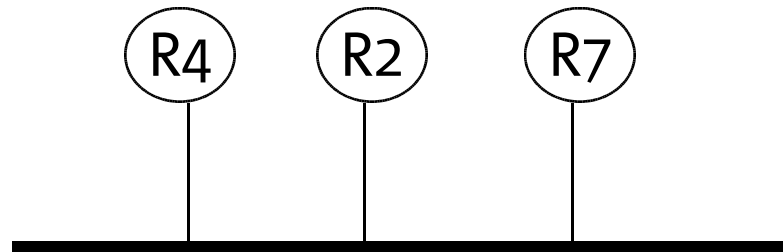
Encapsulation Header

| | | |
|---|-----------|---------------------------------|
| S=Xmitting Rbridge D=Rcvng Rbridge pt="transit" | hop count | original pkt (including L2 hdr) |
|---|-----------|---------------------------------|

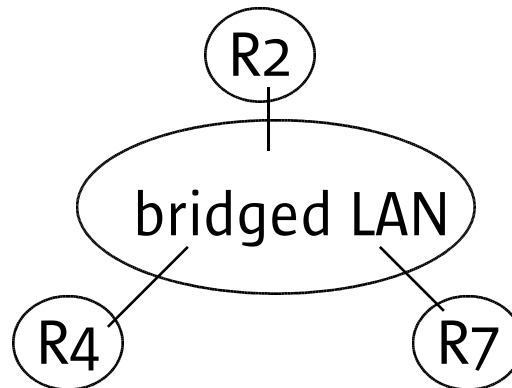
- Outer L2 hdr must not confuse bridges
- So it's just like it would be if the Rbridges were routers
- Need special layer 2 destination address
 - for unknown or multicast layer 2 destinations
 - can be L2 multicast, or any L2 address provided it never gets used as a source address

Rbridges and Bridges

Seems like:



Actually can be:



Endnode Learning

- On shared link, only one bridge (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

Naked Pkt Forwarding

- If layer 2 Dest D known: encapsulate and forward towards D
- Else, send to “destination=flood”, meaning send on spanning tree
 - calculated from LS info, not sep protocol
 - each DR decapsulates

Encapsulated Packet Forwarding

- If packet is encapsulated (new Ethertype):
 - Decrement encapsulation header hop count
 - Forward towards inner header layer 2 destination
 - If forwarding table indicates destination is directly attached, on port P, decapsulate and forward onto P

Rbridging IP

- Rbridging at layer 2 will do it
- Optimization: locally answer ARPs
 - learn (layer 3, layer 2)
 - pass that in link state info
- Another optimization for IP: shorter endnode cache timer (since can ping)

Conclusions

- Looks to routers like a bridge
 - invisible, plug-and-play
- Looks to bridges like routers
 - terminates spanning tree, broadcast domain

Conclusions, cont'd

- Much better replacement for bridging
 - optimal paths
 - still plug and play and transparent
 - fast convergence
 - no meltdowns
- For IP
 - allows plug-and-play single-prefix campus

radia.perlman@sun.com



**2004
Sun Labs
Open House**