RBridges: Transparent Routing

Date: 2005-03-15

Authors:

Name	Address	Company	Phone	Email
Raida Perlman	1 Network Drive, Burlington, MA 01803 USA	Sun Microsystems	+1-781-422-3252	Radia.Perlman@sun.com
Donald Eastlake 3rd	111 Locke Drive, Marlborough, MA 01752 USA	Motorola	+1-508-786-7554	Donald.Eastlake@motorola.com

Abstract

A design is proposed to retain the zero configuration advantage of bridges and have additional advantages: pair-wise shortest paths, minimal packet duplication in the presence of temporary loops, and allowing optimization for Internet Protocol traffic.

http://www.postel.org/rbridge

Rbridges: Transparent Routing

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

Donald Eastlake 3rd
Motorola Laboratories
Donald.Eastlake@motorola.com

Why 802.11 Mesh Needs The Features of RBridges

- **The wire line viewpoint is typically that connectivity changes infrequently and that, at most, the end stations are mobile.**
- **In a wireless mesh, the constantly changing radio propagation characteristics, even for fixed nodes, and the common mobility of end and intermediate nodes requires rapid adaptation to changes in topology at all levels.**

Why 802.11 Mesh Needs The Features of RBridges (cont.)

- **The typical wire line point of view is that network links are high bandwidth and most links are over provisioned.**
- **Limited and variable bandwidth of radio links mean they are commonly saturated so**
  - improved path selection such as optimal point-to-point paths is important, and
  - bridge loops are more injurious.

The Heart of RBridges

- **What Features are the Heart of RBridges?**
  - Safe encapsulation so that temporary loops are less harmful
  - Optimum point-to-point paths
  - Optimizations for IP such as proxy ARP

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!

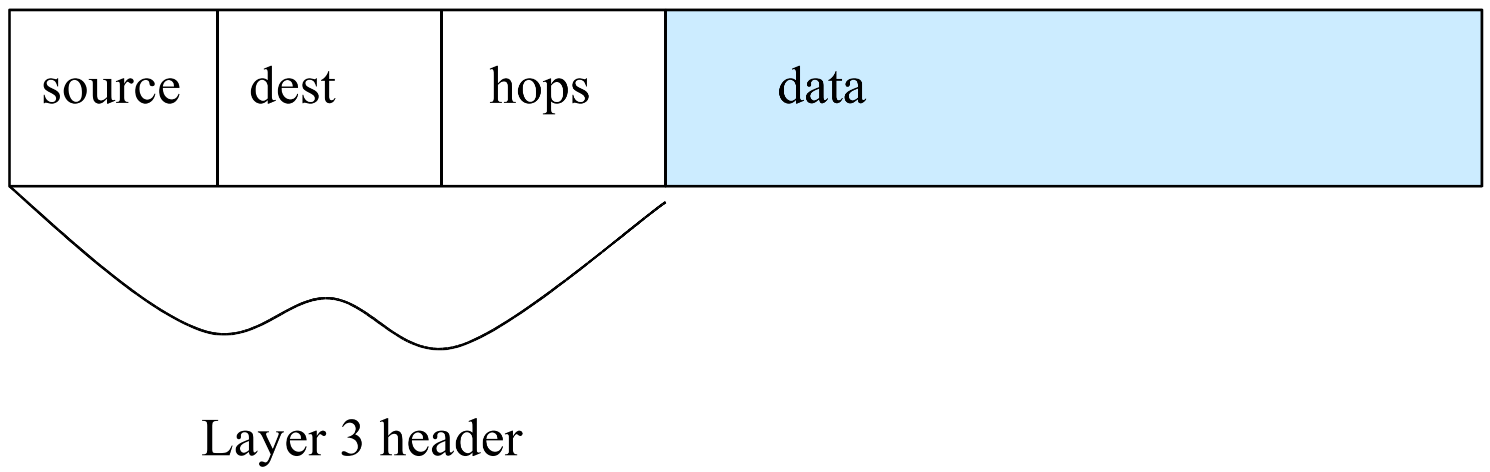
Definitions

- **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***

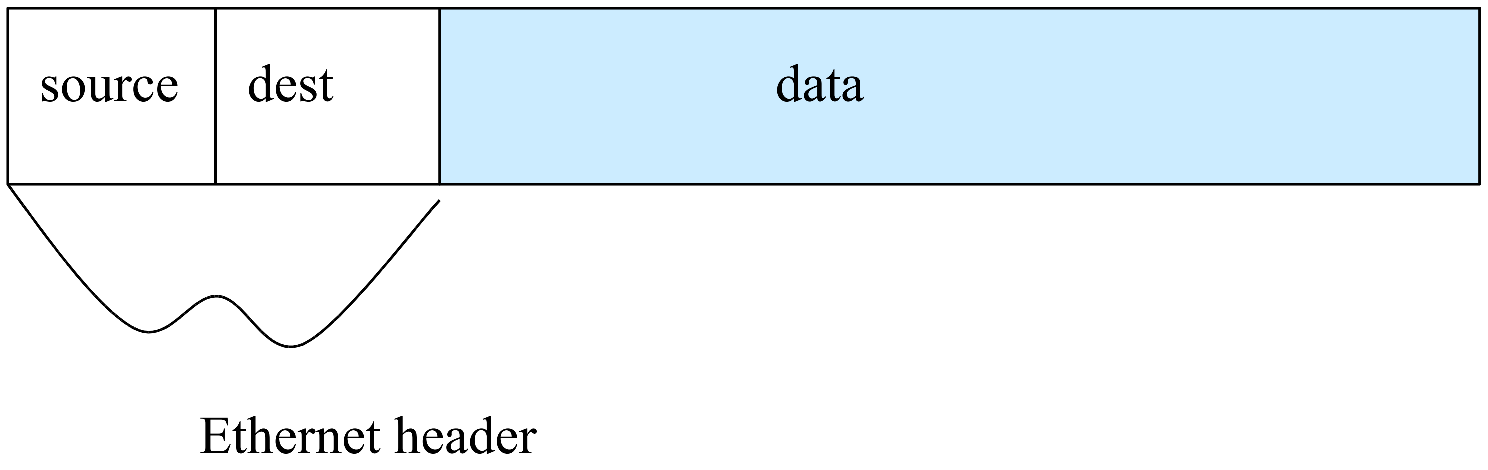
Layer 3 (e.g., IPv4, IPv6, DECnet, Appletalk, IPX, etc.)

- **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*!"**
- **"*Which will win? Ethernet or DECnet*?"**

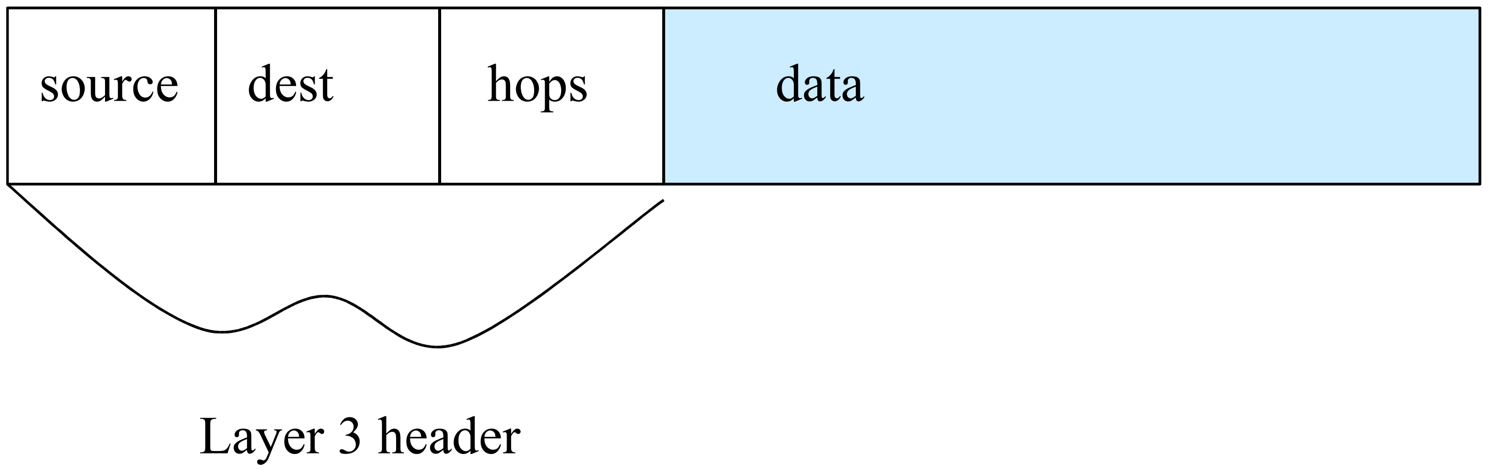
Layer 3 packet



Ethernet packet

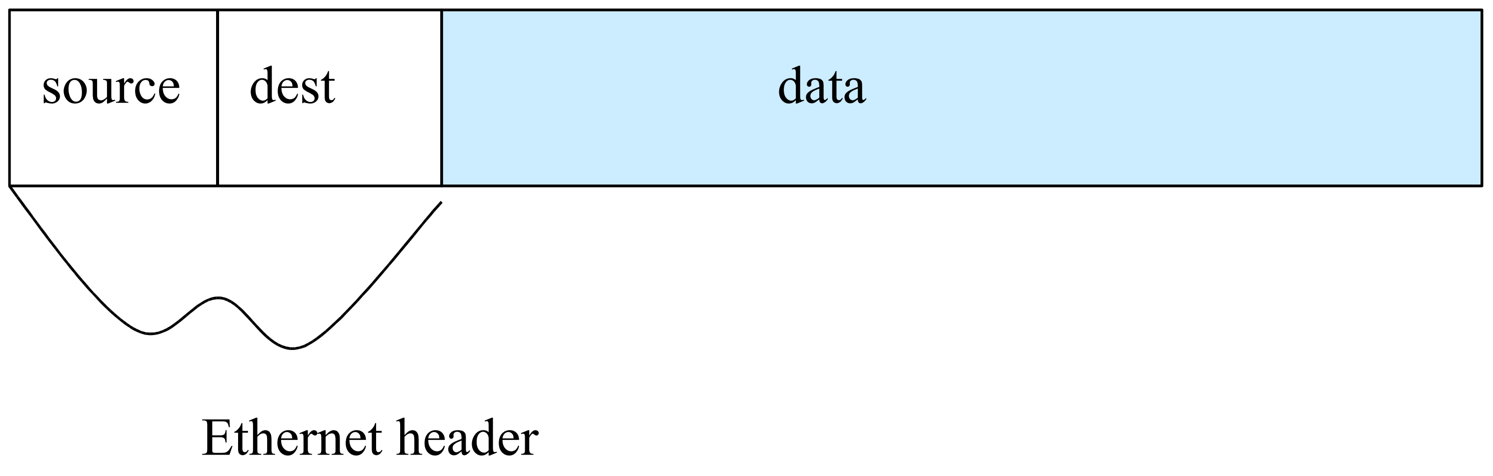


Layer 3 packet



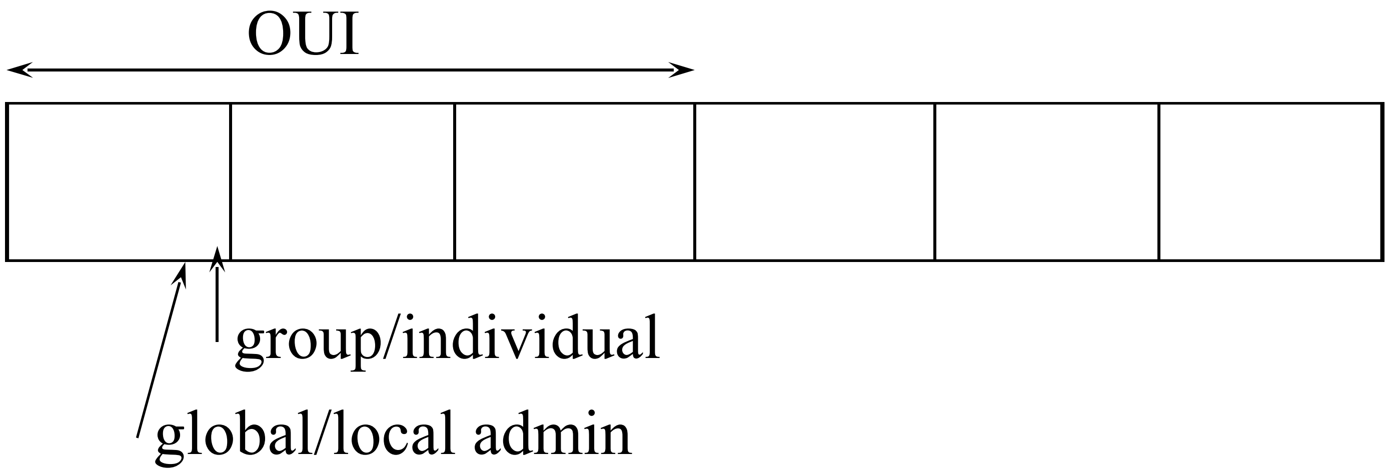
Addresses have topological significance

Ethernet packet



Addresses are "flat" (no topological significance)

Ethernet (802) addresses



- **OUI/rest split look hierarchical, but this structure is only for uniqueness, not for topological hierarchy**

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)

Horrible terminology

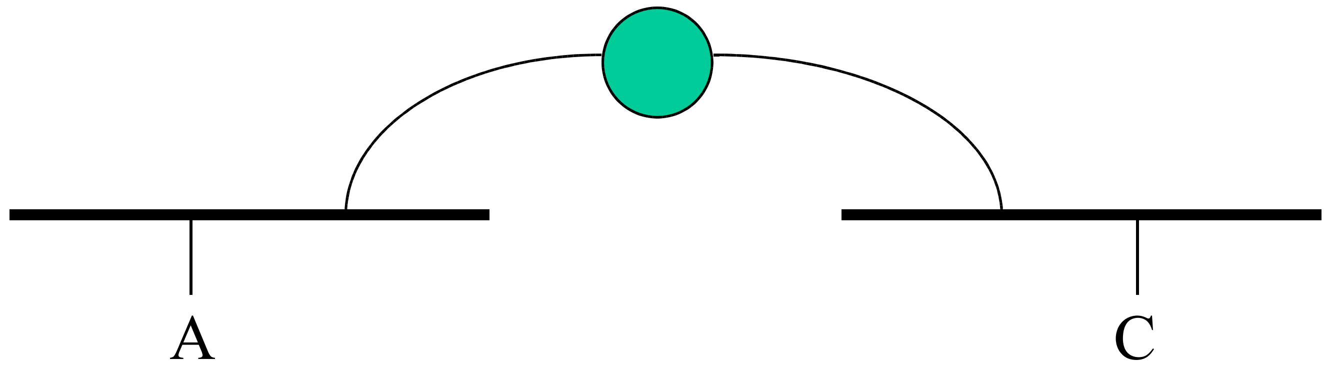
- Local area net
- Subnet
- Ethernet
- Internet

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

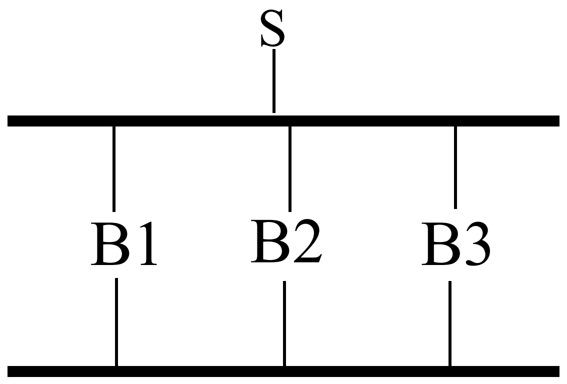
- **Listen promiscuously**
- **Learn location of source address based on source address in packet and port from which packet received**
- **Forward based on learned location of destination**

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)

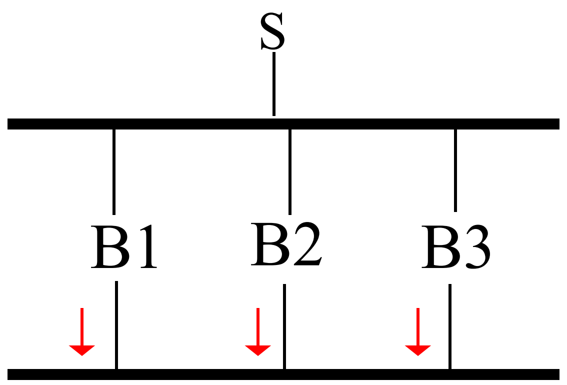
But loops are a disaster

- No hop count
- Exponential proliferation



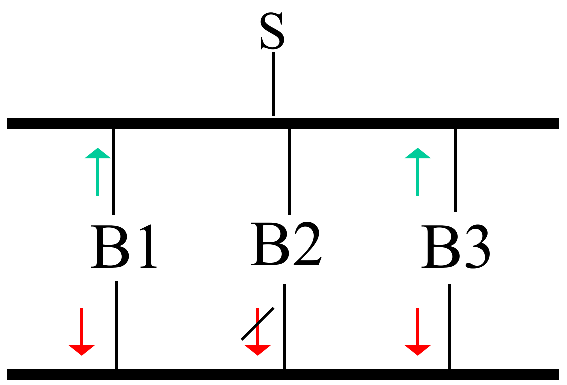
But loops are a disaster

- No hop count
- Exponential proliferation



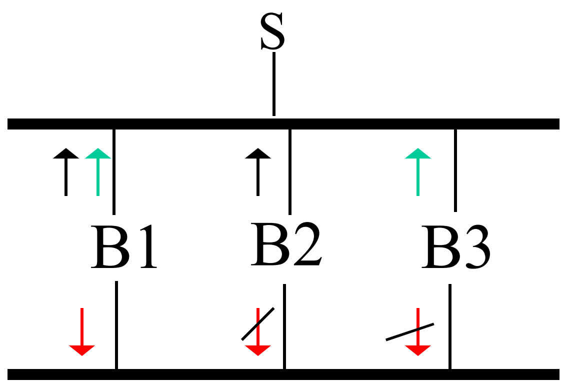
But loops are a disaster

- No hop count
- Exponential proliferation



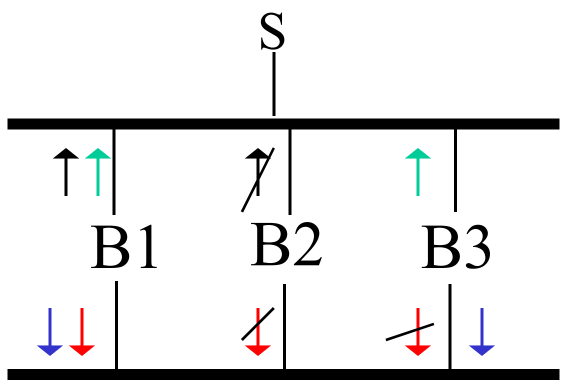
But loops are a disaster

- 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.

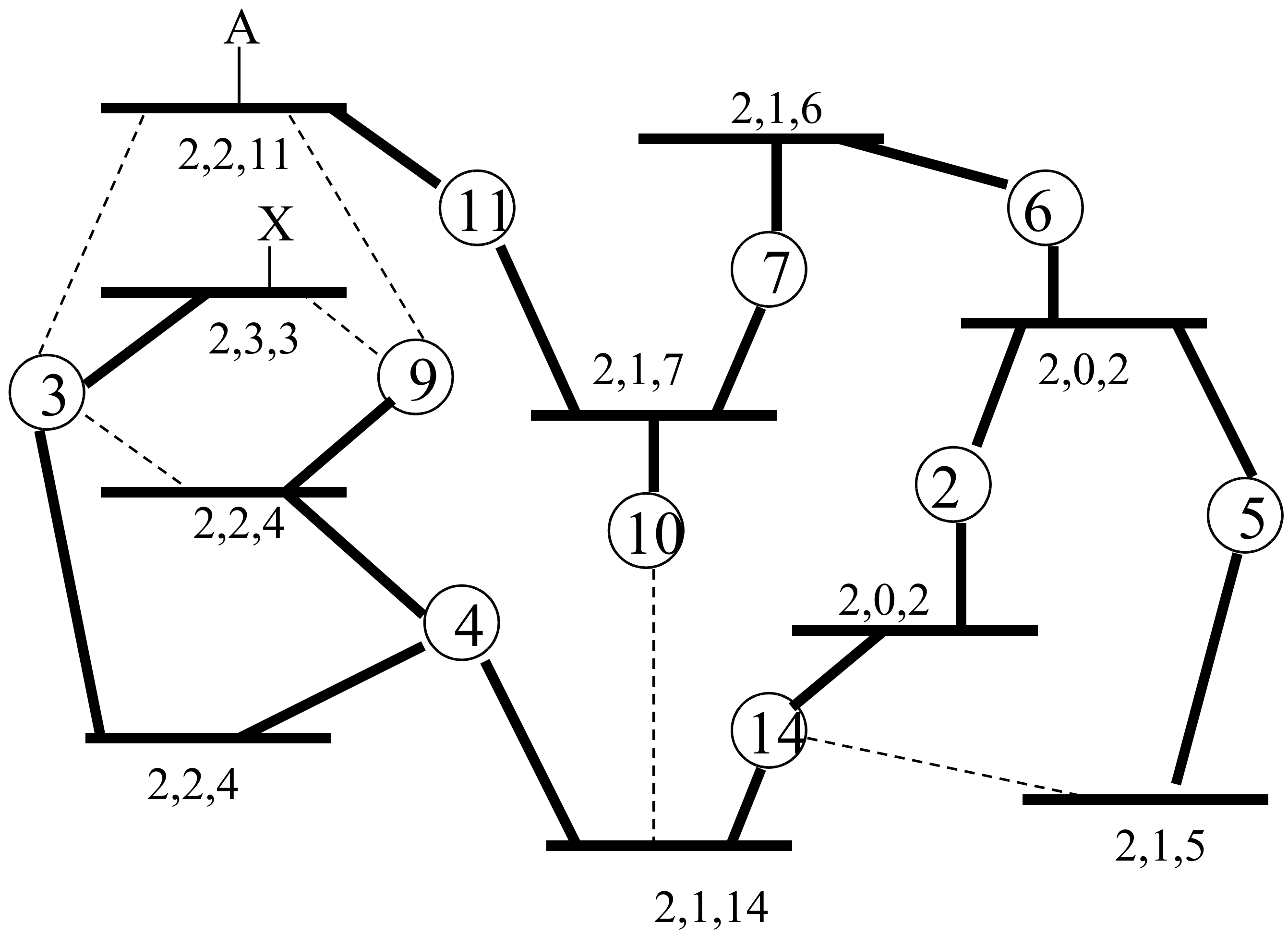
A tree which must be sure to span
So packets can reach every LAN.

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.

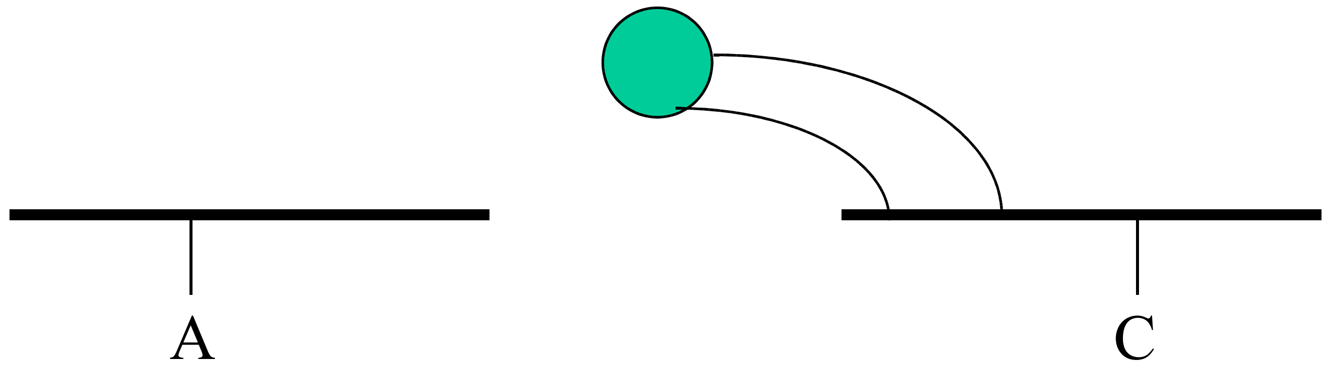
Radia Perlman


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

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 <u>on</u> link**
- **So if too much traffic causes spanning tree messages to get lost…**
  - loops
  - exponential proliferation of looping packets

Slight digression

- **What are switches?**

So what is Ethernet?

- CSMA/CD, right? Not any more, really...
- source, destination (and no hop count)
- limited distance, scalability (not any more, really)

Switches

- Ethernet used to be bus
- Easier to wire, more robust if star (one huge multiport repeater with pt-to-pt links
- If store and forward rather than repeater, and with learning, more aggregate bandwidth
- Can cascade devices…do spanning tree
- We’ve reinvented the bridge!

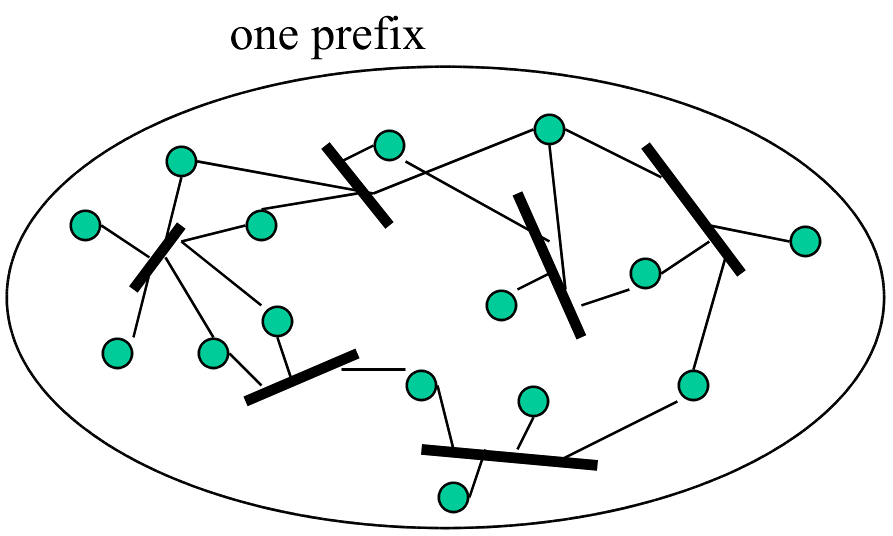
Why are there still bridges?

- **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

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”
- **ES-IS has endnodes proactively advertise**

CLNP areas



CLNP level 1 routing

- **Depended on protocol "ES-IS"**
  - endnodes periodically multicast presence to rtrs
  - (also, rtrs periodically multicast to endnodes)
- **Rtrs tell each other, within area, location of all endnodes in area**
- **IS-IS originally designed for CLNP. "Level 2" was to longest prefix. "Level 1" was to exact match of bottom 6 bytes.**

“Level 1 routing” with IP

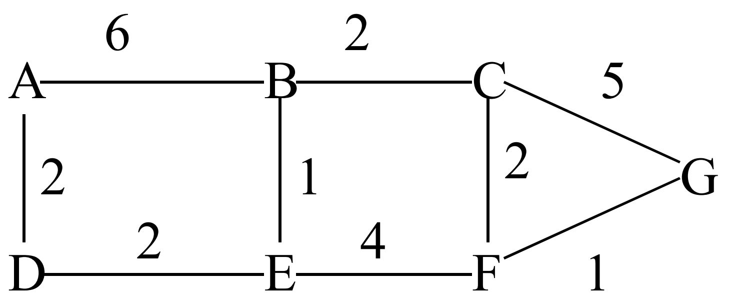
- IP has never had true level 1 routing
- Each link has a prefix
- Multilink node has two addresses
- Move to new link requires new address
- Bridging is used to create a campus in which all nodes share the same prefix
- But bridging isn’t as good as routing

Distributed Routing Protocols

- **Rtrs exchange control info**
- **Use it to calculate forwarding table**
- **Two basic types**
  - distance vector
  - link state

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)**



A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	

Computing Routes

- **Edsgar Dijkstra's algorithm:**
  - calculate tree of shortest paths from self to each
  - also calculate cost from self to each
  - Algorithm:
    - step 0: put (SELF, 0) on tree
    - step 1: look at LSP of node (N,c) just put on tree. If for any nbr K, this is best path so far to K, put (K, c+dist(N,K)) on tree, child of N, with dotted line
    - step 2: make dotted line with smallest cost solid, go to step 1

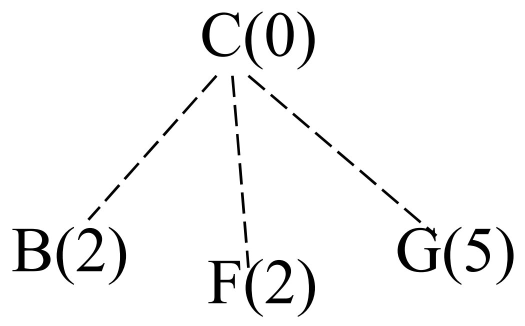
Start with Self(0)

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	

C(0)

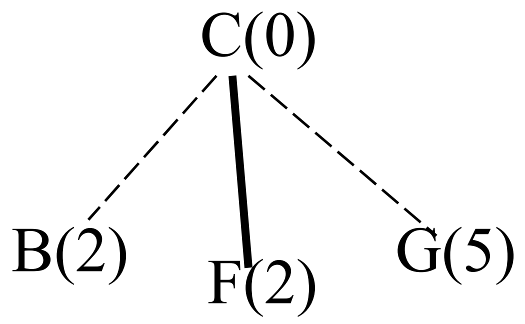
Look at LSP of new tree node

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



Make shortest TENT solid

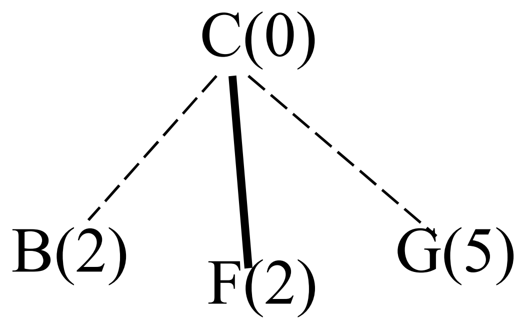
A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



Look at LSP of newest tree node

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	

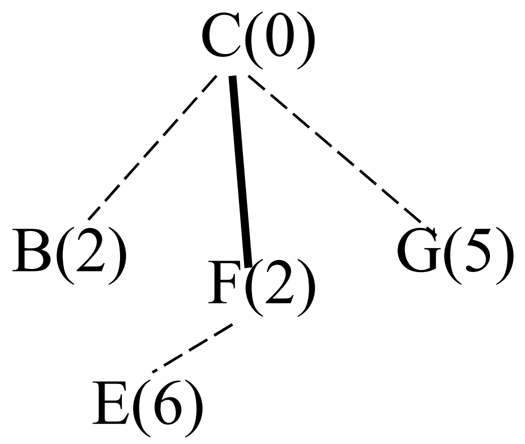
C: 2+2 worse than 0



Look at LSP of newest tree node

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	

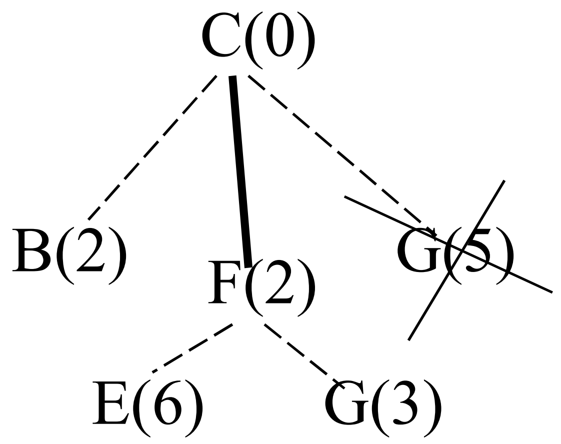
E:2+4 best so far



Look at LSP of newest tree node

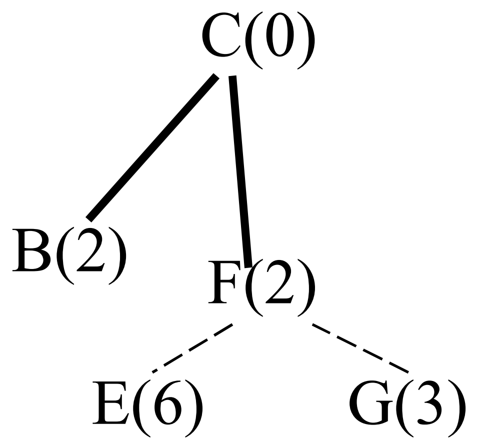
A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	

G:2+1: best so far



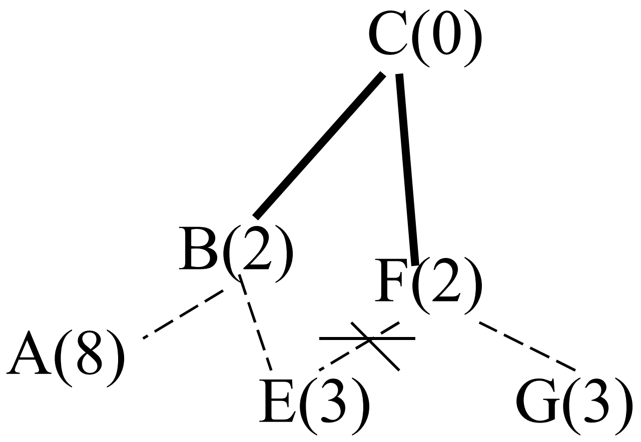
Make shortest TENT solid

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



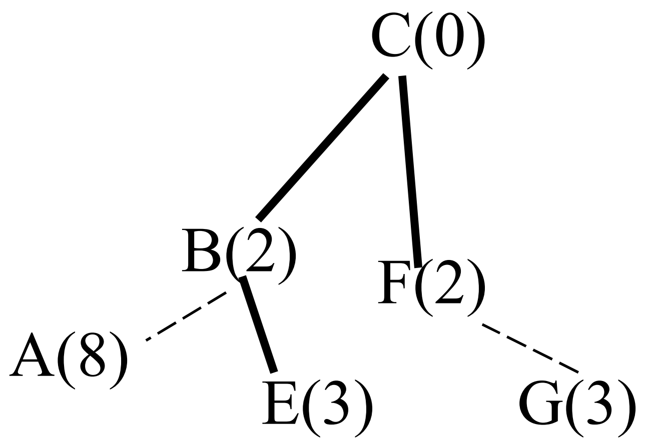
Look at LSP of newest tree node

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



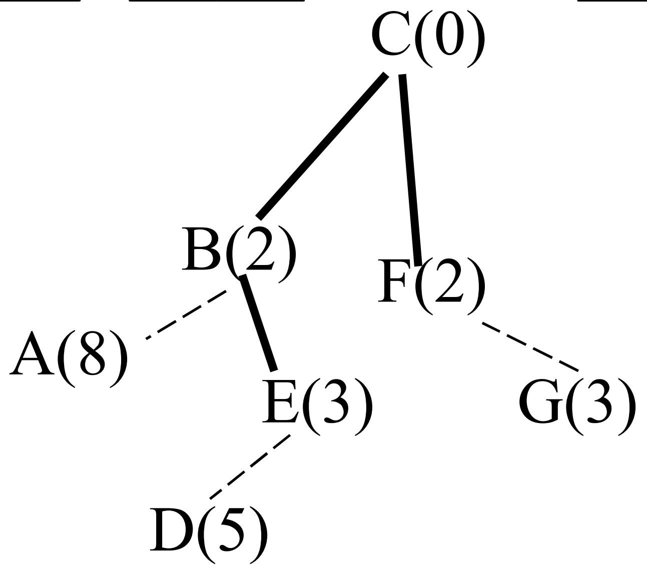
Make shortest TENT solid

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



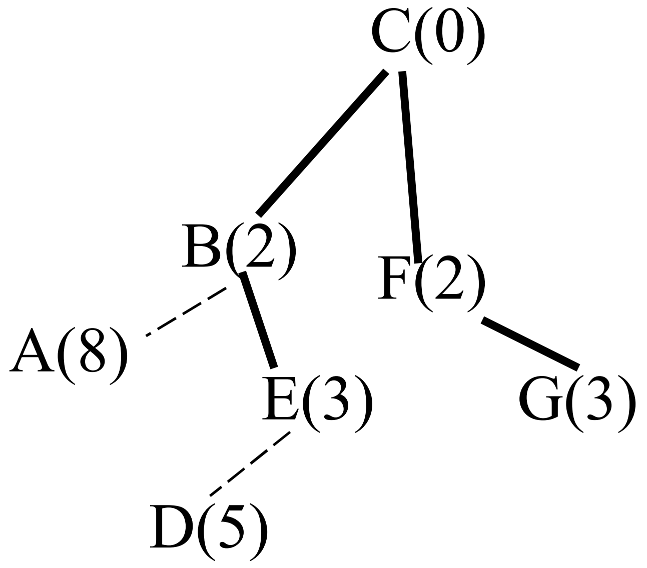
Look at LSP of newest tree node

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



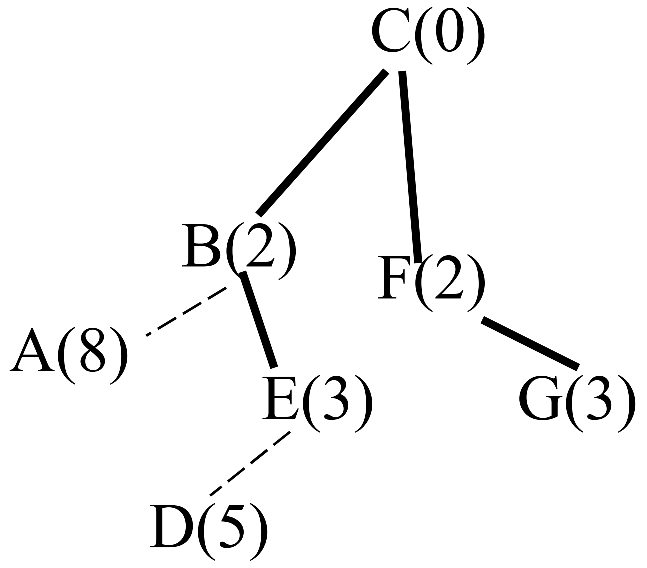
Make shortest TENT solid

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



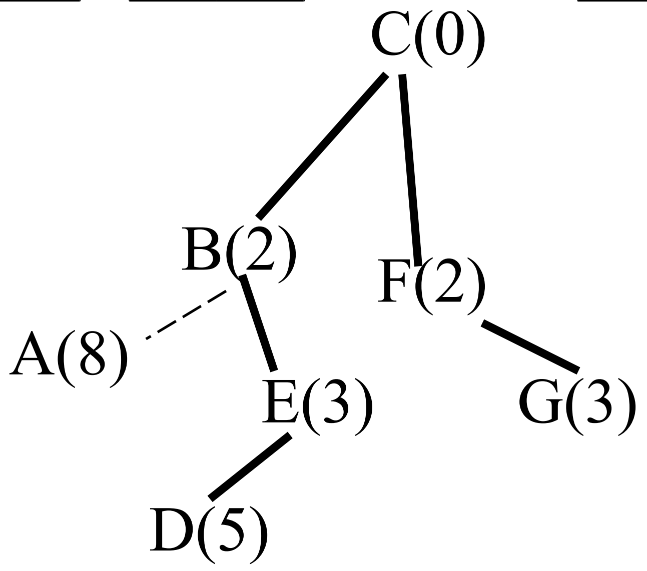
Look at newest tree node’s LSP

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



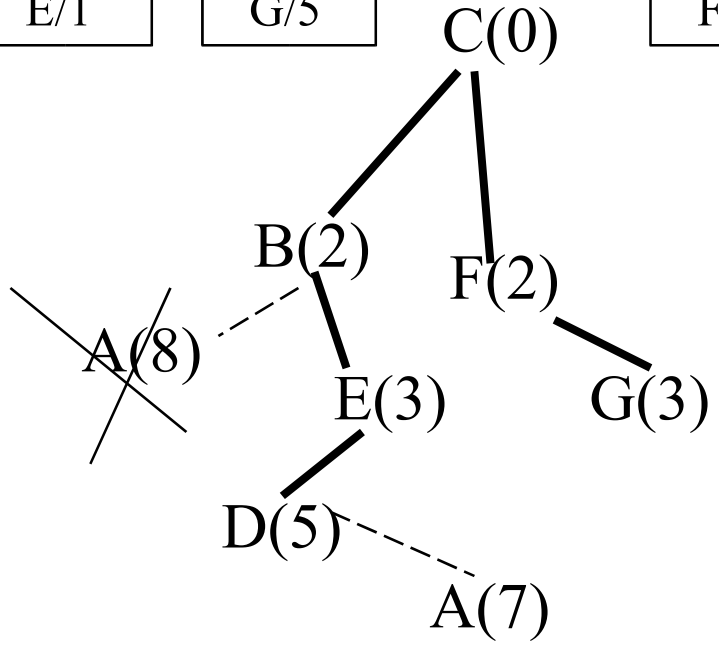
Make shortest TENT solid

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



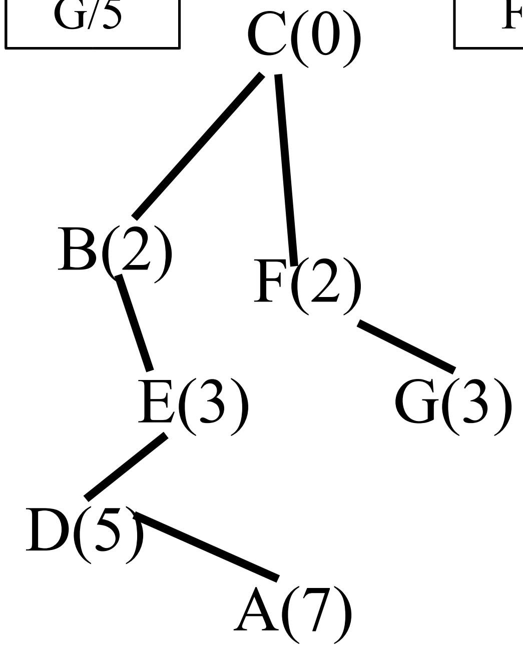
Look at newest node’s LSP

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



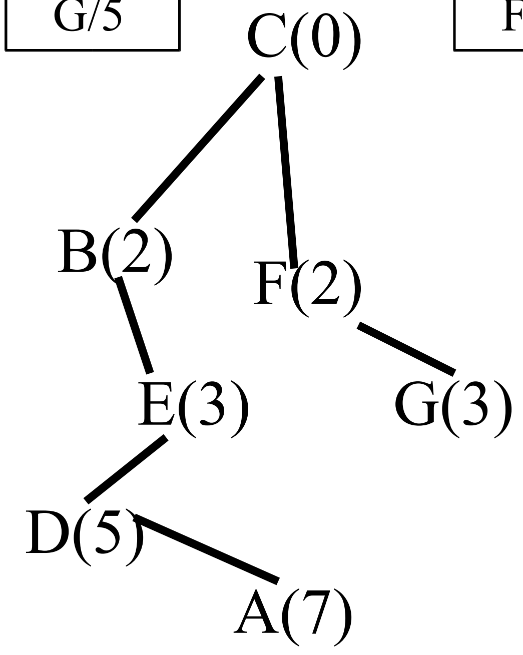
Make shortest TENT solid

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



We're done!

A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	



Not quite: need forwarding table

- So everything in subtree gets forwarded through that port

"A miracle occurs"

- First link state protocol: ARPANET
- I wanted to do something similar for DECnet
- My manager said "Only if you can prove it's stable"
- Given a choice between a proof and a counterexample…

Broadcasting LSP

- **Can’t depend on routing info being correct**
- **Basic idea is flooding**
  - send to every nbr except from which LSP rcv’d
- **Flooding is exponential, but we can do better than that since we store LSPs, and only flood them the first time we see them**
- **How do you tell if an LSP is newer than the stored one?**

Comparing LSPs

- **Different from what's in database? Which is newer?**
  - most recently received?
  - globally synchronized clocks
  - local battery-backup clock
  - sequence numbers
    - wrap-around with partitions, restarts
  - sequence number plus age field

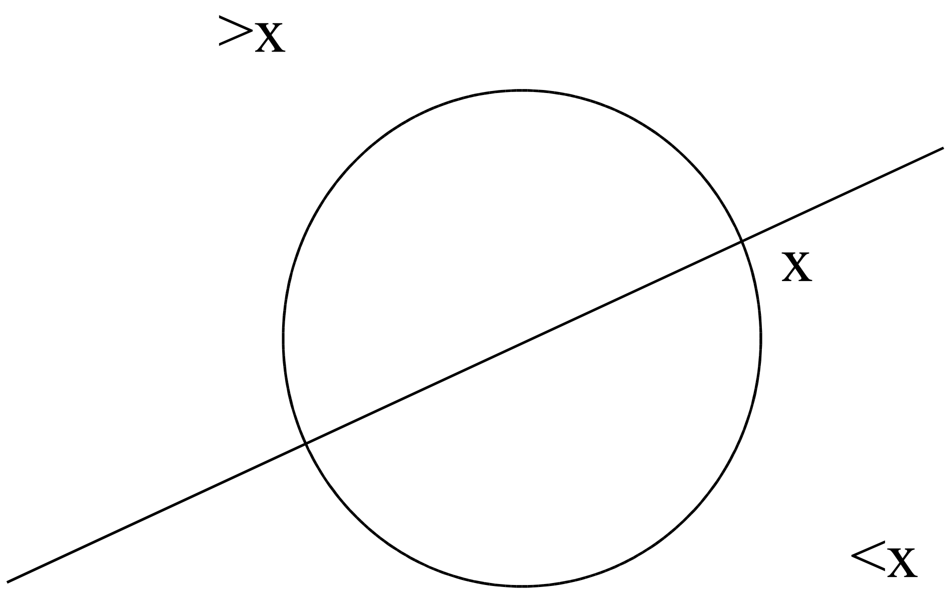
Sequence Number Handling

- **If rcv LSP from Fred through neighbor N:**
  - compare sequence number with what’s in database for Fred.
  - If rcv’d one bigger
    - overwrite database
    - flood to all nbrs except Fred.
  - Else ignore

Age Field

- source sets age to MAX-AGE (64 seconds, 3 bit field, units of 8 seconds)
- decrement age after hold LSP for 8 seconds)
- if age=0, “too old”, don’t propagate
- Generate new LSP within MAX-INT (60 seconds)
- When starting, wait RESTART-TIME (90 seconds)

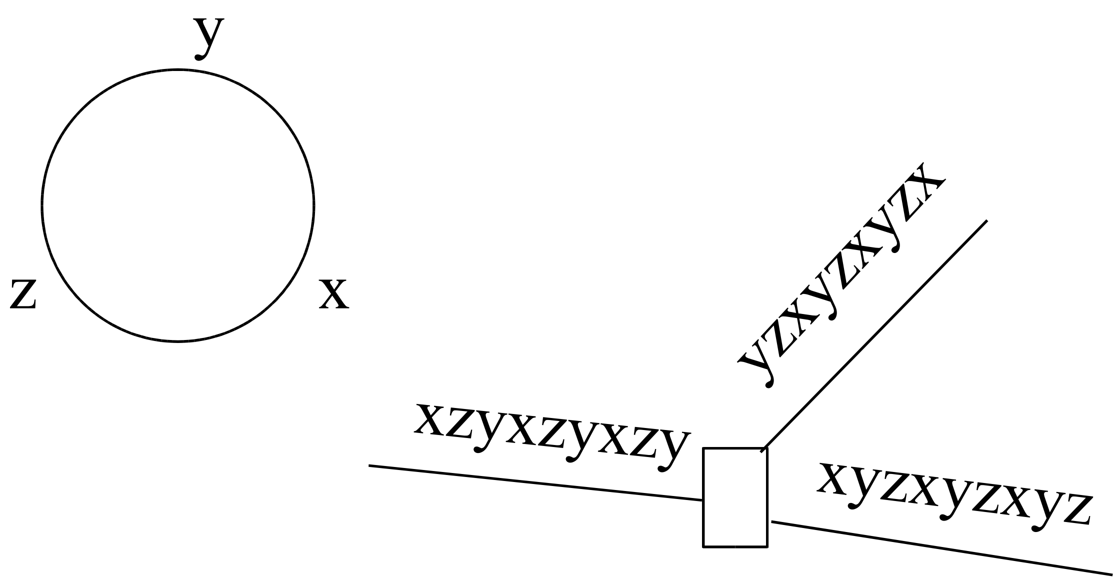
Arithmetic in circular space



ARPANET disaster

- symptom: net didn’t work
- how do you diagnose and manage a network?
- Note: these guys were really really lucky!
- What had happened: Fred, a sick router, generate bad LSPs before dying, with sequence numbers x, y, z

ARPANET disaster



What now?

- Networks don’t have on/off switches
- First crash and reload BBN router
- Still broken---examine core dump
- Realize the problem
- Create patched code to ignore LSPs from Fred
- One by one, crash and load rtrs with patch
- One by one, load rtrs with real code again
- Hope it never happens again by accident (or on purpose!)

So how do you fix a broken net?

- Patched version of code that ignore LSPs from Fred
- One by one crashed systems (not easy!) and reloaded with patched code
- Only after all routers reloaded, can they be reloaded with correct version again

Routing Robustness

- I showed how to make link state distribution “self-stabilizing”… but only after the sick or evil node was disconnected
- I said “but you can’t expect a network to work *while* the evil router is connected”
- When I went to grad school 10 years later someone challenged me to “prove that statement, or design Byzantine robustness”
- Given a choice of impossibility proof vs doing it...

Plug for my thesis

- **It’s actually not that hard to design a network such that**
  - if any nonfaulty path between A and B exists, A and B can talk, even if
  - all other routers evil
    - lie about who they are connected to
    - corrupt packets they see
    - babble garbage
    - do the routing algorithm properly, but misroute data
- **Thesis is online at lcs.mit.edu, tech reports (#429)**

What we'd like, part 1: replace bridging with Rbridging

- **keep transparency to endnodes**
- **keep plug-and-play**
- **have best paths**
- **eliminate problems with temporary loops**
  - have a hop count
  - don't exponentially proliferate packets
- **then can converge optimistically (like rtrs)**

What we'd like, part 2: true "level 1 routing" for IP

- **allow plug-and-play campus sharing a prefix**
- **allow optimal routing**
- **don't require any endnode changes (e.g., implement ES-IS)**
- **Interwork with existing routers and bridges**

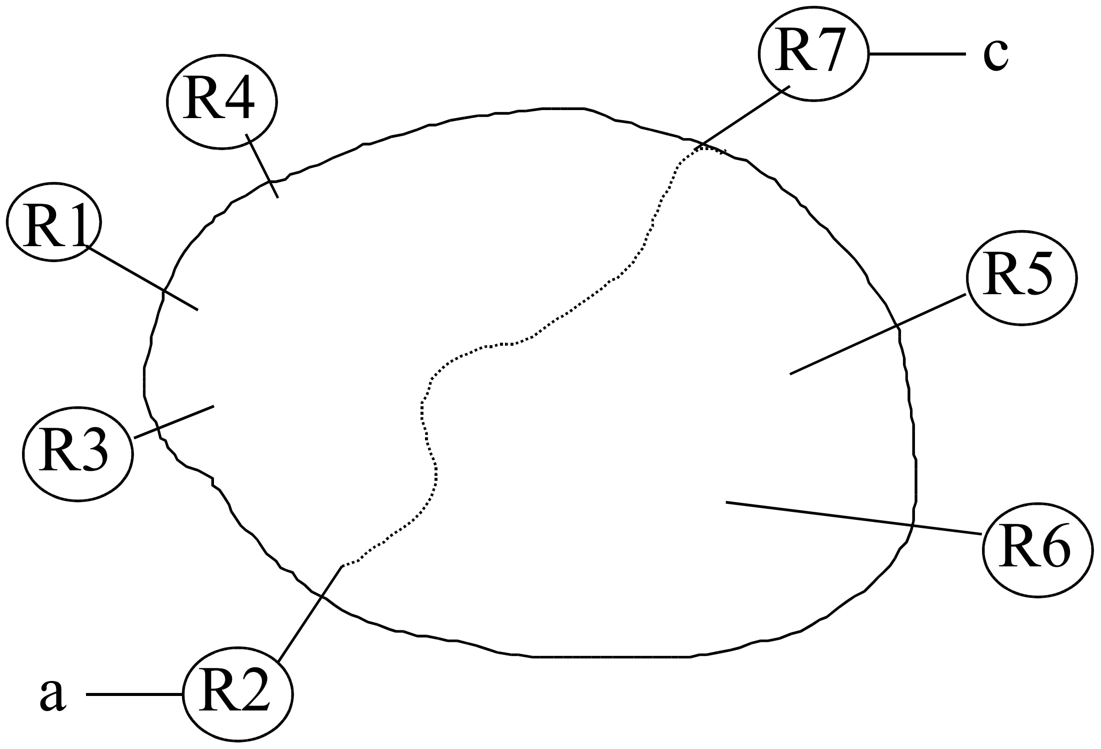
Rbridges

- 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

Rbridging



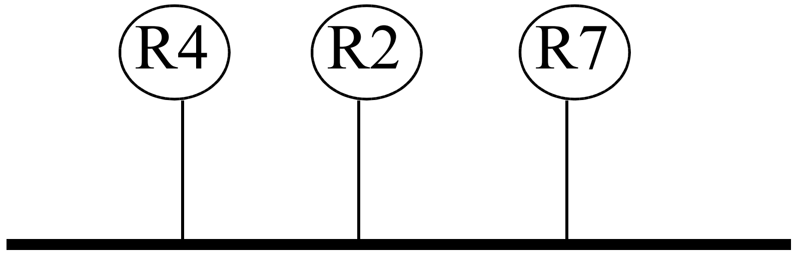
Encapsulation Header

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

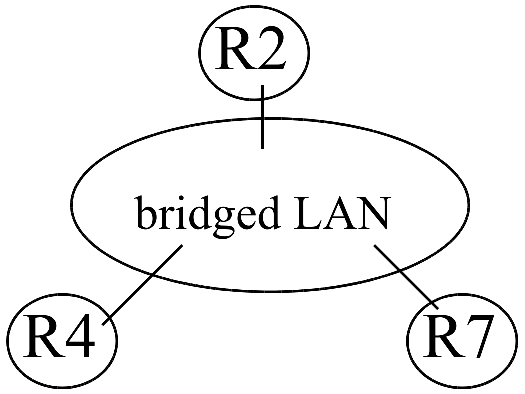
- Outer L2 hdr must not confuse bridges
- So it's just like it would be if the Rbridges were routers
- Need special layer 2 destination address
    - for unknown or multicast layer 2 destinations
    - can be L2 multicast, or any L2 address provided it never gets used as a source address

Rbridges and Bridges

Seems like:



Actually can be:



Endnode Learning

- **On shared link, only one 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**
  - calculated from LS info, not sep protocol
  - each DR decapsulates

Calculating spanning tree

- **No reason to have additional protocol to calculate a spanning tree**
- **The link state database gives enough information for RBridges to deterministically compute a spanning tree**
  - Do it from viewpoint of lowest ID RBridge
  - When tie in Dijkstra calculation, choose parent with lowest ID

Possible 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 is a broadcast domain**
- **So a VLAN A packet must only be forwarded to VLAN A links**
- **RBridges must announce which VLANs they connect to**
- **RBridges must be able to flood a VLAN A pkt to just VLAN A links**
  - could do it with one spanning tree, and just not send on non-A links
  - or one spanning tree, and filter if no A-links downstream
  - or per-VLAN spanning tree

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

Conclusions

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

Conclusions, cont’d

- **Advantages**
  - optimal pairwise paths
  - still plug and play and transparent
  - fast convergence (no artificial delays like in original version of STP)
  - safe behavior when temporary loops
  - trivial modification of IS-IS to carry RBridge info
- **For IP**
  - allows plug-and-play single-prefix campus