Understanding Design Tradeoffs in Routing

Radia Perlman
Intel Labs

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

What this is NOT about

- Looking at all the details of a particular protocol
- Telling you “the one answer”

What this IS about

- Focus separately on different aspects of protocols
- Look at alternative solutions for each orthogonal piece, together with engineering tradeoffs for each
- Be thought-provoking; spark discussion
- Suggest areas that can be researched
- Hopefully same way of thinking can work in other protocol areas

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
- And, I'm attempting to find all arguments pro and con for each aspect…please tell me anything I may have missed

Perlman's View of Network Layers

- Based on OSI layers…

Perlman's View of Network Layers

- Layer 1: Physical

Perlman's View of Network Layers

- Layer 1: Physical
- Layer 2: Data Link: Neighbor-neighbor

Perlman's View of Network Layers

- Layer 1: Physical
- Layer 2: Data Link: Neighbor-neighbor
- Layer 3: Network: create path, forward

Perlman's View of Network Layers

- Layer 1: Physical
- Layer 2: Data Link: Neighbor-neighbor
- Layer 3: Network: create path, forward
- Layer 4: "Transport": end-to-end reordering, error recovery

Perlman's View of Network Layers

- Layer 1: Physical
- Layer 2: Data Link: Neighbor-neighbor
- Layer 3: Network: create path, forward
- Layer 4: "Transport": end-to-end reordering, error recovery
- Layers 5 and above:

Perlman's View of Network Layers

- Layer 1: Physical
- Layer 2: Data Link: Neighbor-neighbor
- Layer 3: Network: create path, forward
- Layer 4: "Transport": end-to-end reordering, error recovery
- Layers 5 and above: boring!

Some mysteries

- What does it mean to forward at layer 2 vs layer 3? Layer 3 is supposed to be the thing doing forwarding.
- Why do switches have to worry about keeping packets in order? Isn't that TCP's job?

So what is routing?

- Data is put in an “envelope” with information “X” indicating where the packet should go
- A switch/router/bridge has a “forwarding table”, indicating for “X”, which port(s) to forward on

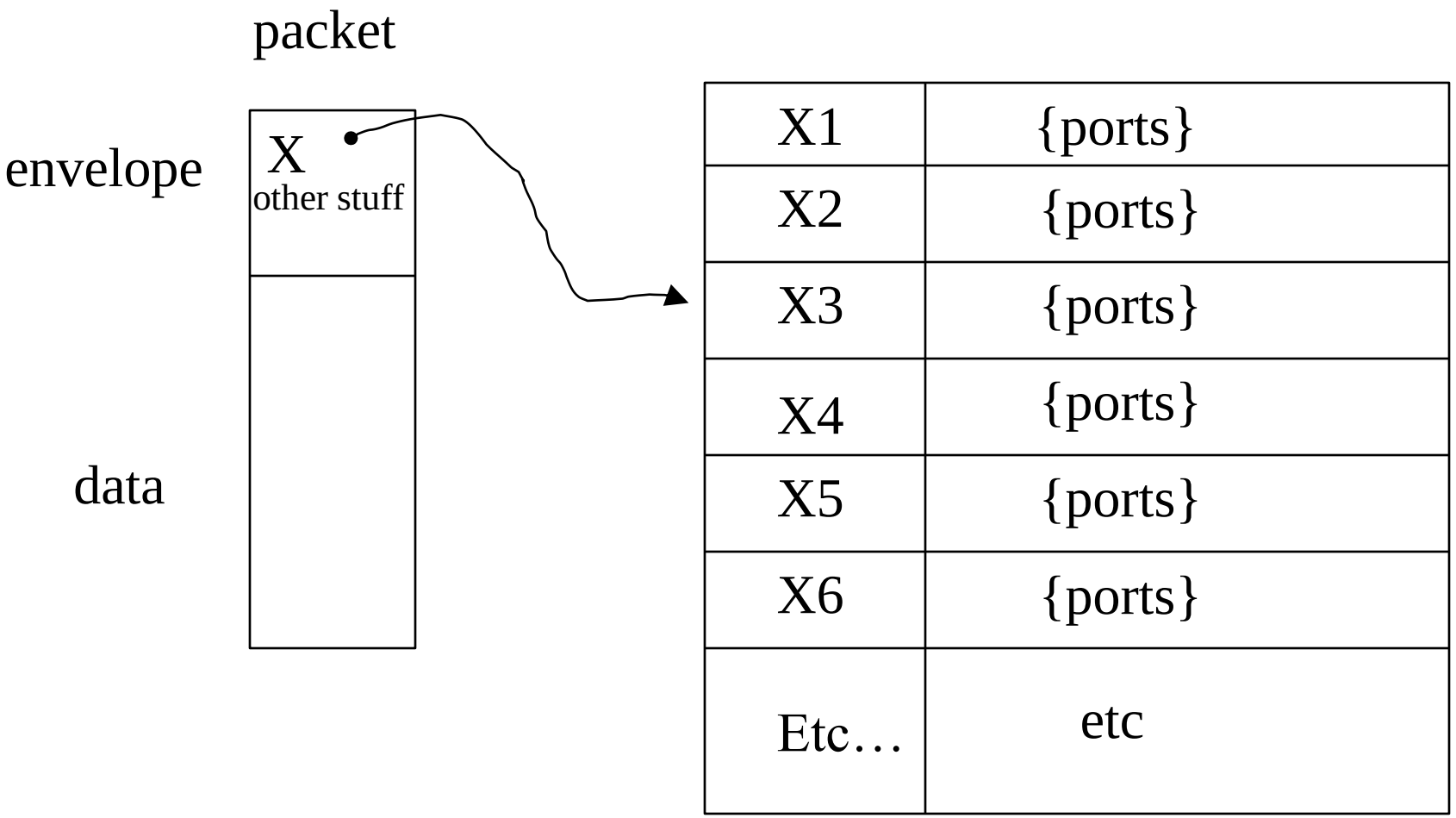
Forwarding Table

X1	{ports}
X2	{ports}
X3	{ports}
X4	{ports}
X5	{ports}
X6	{ports}
Etc…	etc

Basic components

- Information in a packet
- What the forwarding table is indexed by
- How the forwarding table is created
- Basis for spreading traffic

Forwarding Table



Network Protocols

- A lot of what we all know

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
- Try to ignore buzzwords and "which team"
- Question assumptions

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
- …

Some terminology

- “Flow” : a conversation between two applications
  - Usually networks attempt to deliver all packets for a given flow in order
- Bridge/router/switch : something that forwards packets; mostly interchangeable terms

What's in a packet; types of addresses

- Flat
  - Direct lookup
  - Hash
- Hierarchical
  - Fixed hierarchy
  - Longest prefix match

Terminology I wish people would use

- “ID” or “identifier” : doesn’t change if target or source moves
- “name” : like ID, but human-friendly string
- “Address” : changes if target moves, but source-location independent
- “Route” : changes if either target or source location changes

“Address”

- Unfortunately, people don’t use the terminology that way
- For instance, a “MAC address”

Flat Address

- “Flat” means isn’t location dependent
- (so it shouldn’t be an “address”, but oh well…)
- Flat is very convenient, with virtualization, self-configuration
- But if the entire Internet were flat addresses, forwarding table would be too large

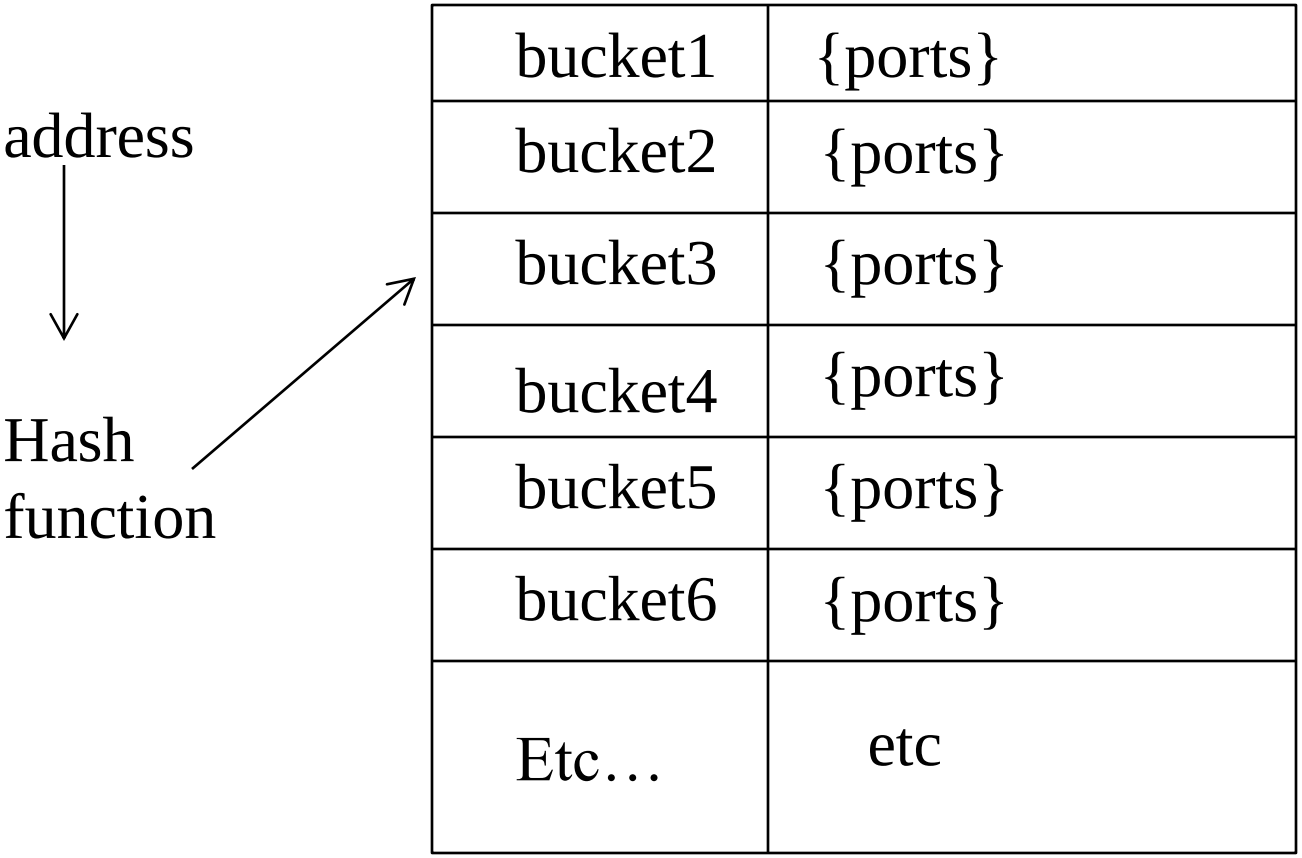
Forwarding Table

address1	{ports}
address2	{ports}
address3	{ports}
address4	{ports}
address5	{ports}
address6	{ports}
Etc…	etc

Two types of flat addresses

- Dense: address can be simple lookup: address #n is nth entry in the forwarding table
- Sparse: (like 6-byte Ethernet address) address lookup has to be hash

Forwarding Table for Sparse Flat Address



Ethernet addresses

- 6 bytes
- And for a technology originally intended to support about 1000 nodes!
- Why so huge?

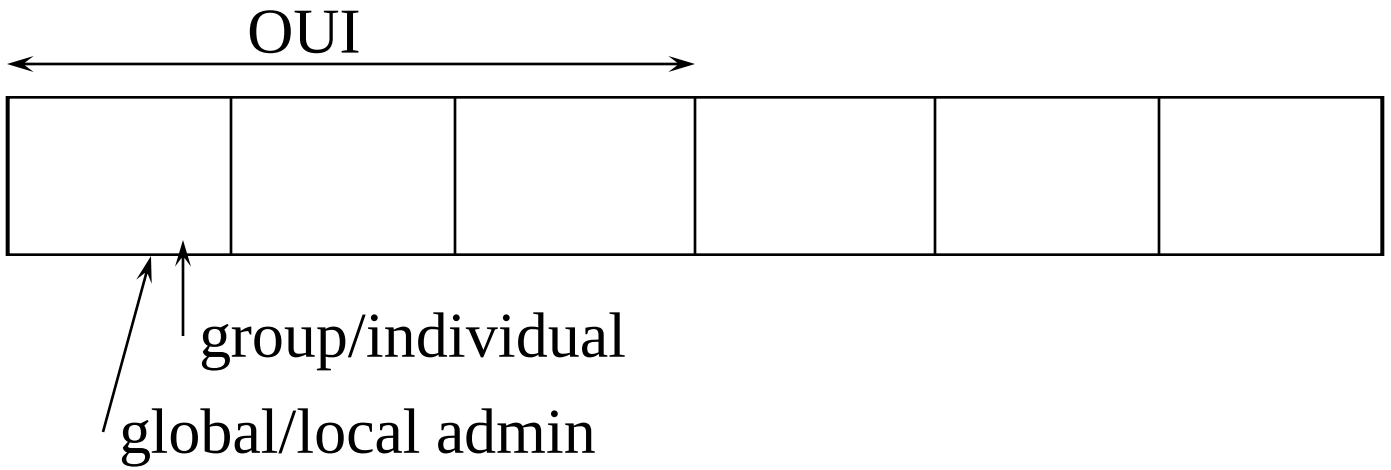
Ethernet addresses

- 6 bytes
- And for a technology originally intended to support about 1000 nodes!
- Why so huge? – actually, Ethernet's genius
  - Addresses created when device manufactured
  - No worries about configuring address when deploying a network

Structure of Ethernet address

- 24 bits for “OUI” (organizationally unique identifier, if you insist on expanding acronyms), assigned by IEEE
  - 2 special bits inside OUI:
    - G/I (group/individual, or multicast/unicast)
    - G/L (globally assigned vs locally assigned – if locally assigned, then you can assign 46 bits)
- For globally assigned OUI, 24 bits you can assign

802 addresses



- Assigned in blocks of 2^{24}
- Given 23-bit constant (OUI) plus g/i bit
- all 1's intended to mean "broadcast"

Trivia

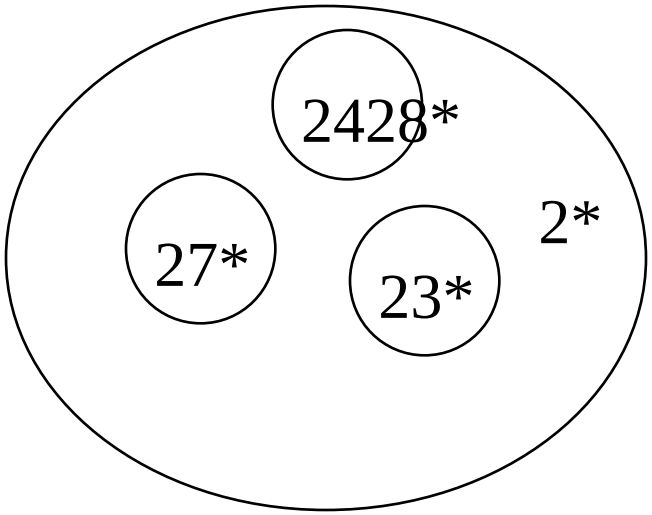
- One early proposal was to assign 48-bit MAC addresses at random…no IEEE address blocks
- 48-bits is probably big enough for this to be practical, and low enough probability of address collision

Hierarchical addresses

- Similar to country/state/city
- If outside a country, don't need to know the structure of another country…just route to that country
- Same with state
- Allows more compact forwarding table, since many destinations can be summarized in one entry (one "prefix")

Hierarchical Address

If addresses are assigned carefully, a whole blob can be summarized with a single forwarding table entry, by switches outside that blob



Fixed hierarchy

- You could set aside some # of bits for each level, e.g., fixed fields for “country”, “state”, “city”, …
- Find first field where target address differs from yours…do flat address lookup (dense or sparse) on that field
- But might not be flexible enough
- Alternative: Longest prefix match

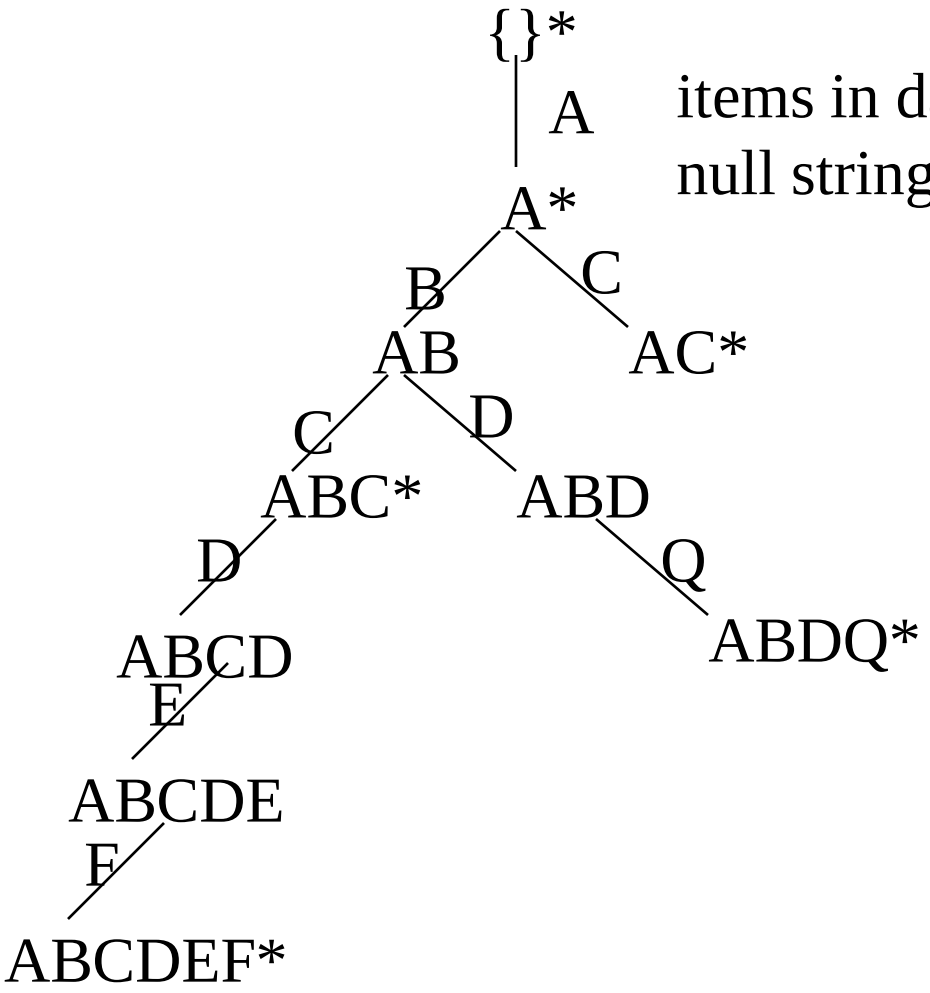
Address Prefix Routing

- Given destination address, want to find longest prefix match in forwarding table
- Two basic algorithms
  - TRIE
  - modified binary search

TRIE

- Character-by-character search
- “Character” might be single bit
- “*” means match
- remember last time “*” seen
- once nowhere to go, last “*” is longest prefix match

TRIE



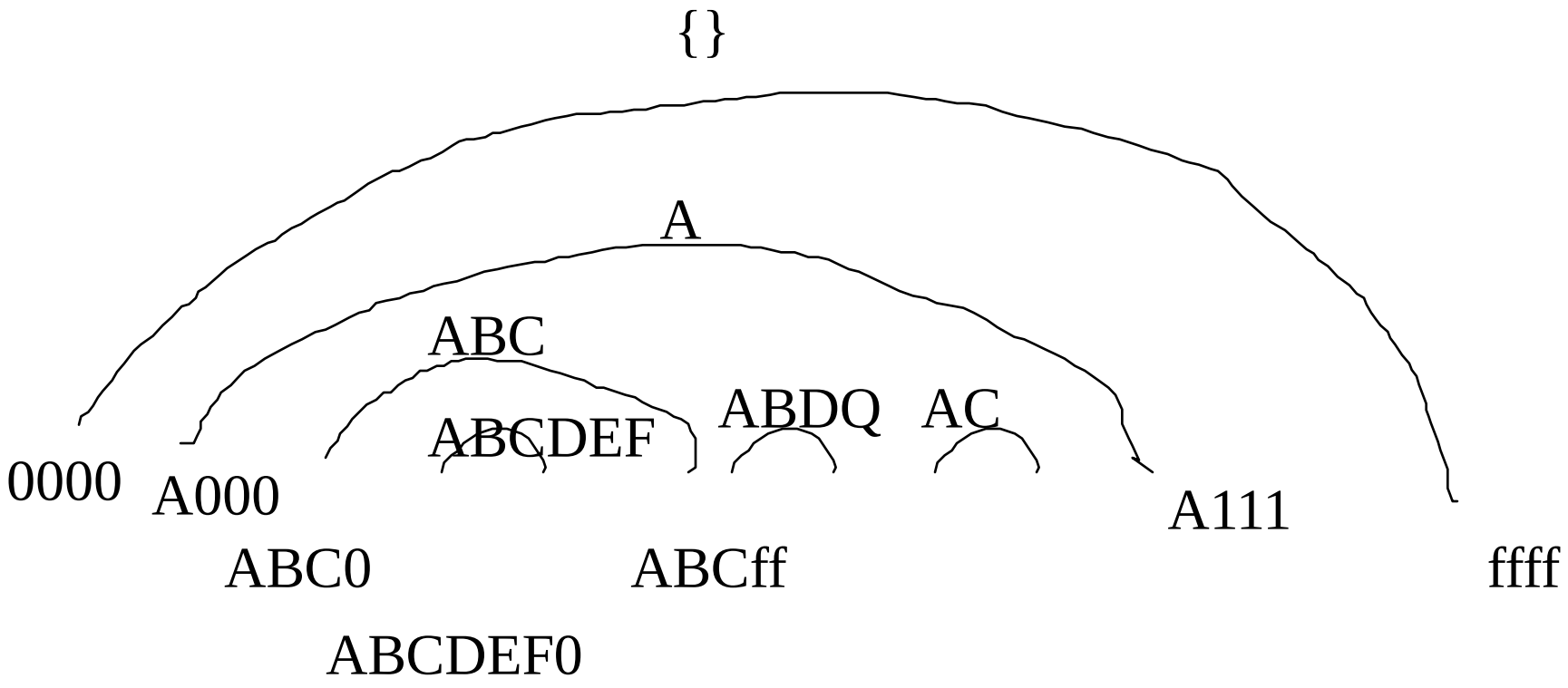
items in database:
null string, A, ABC, ABCDEF, ABDQ, A

Binary search

- Create ranges
- Take each prefix
  - pad with 0's for low order of range
  - pad with 1's for hi order of range
- Sort them
- Find where destination address fits

Binary Search

items: {}, A, ABC, ABCDEF, ABDQ, AC



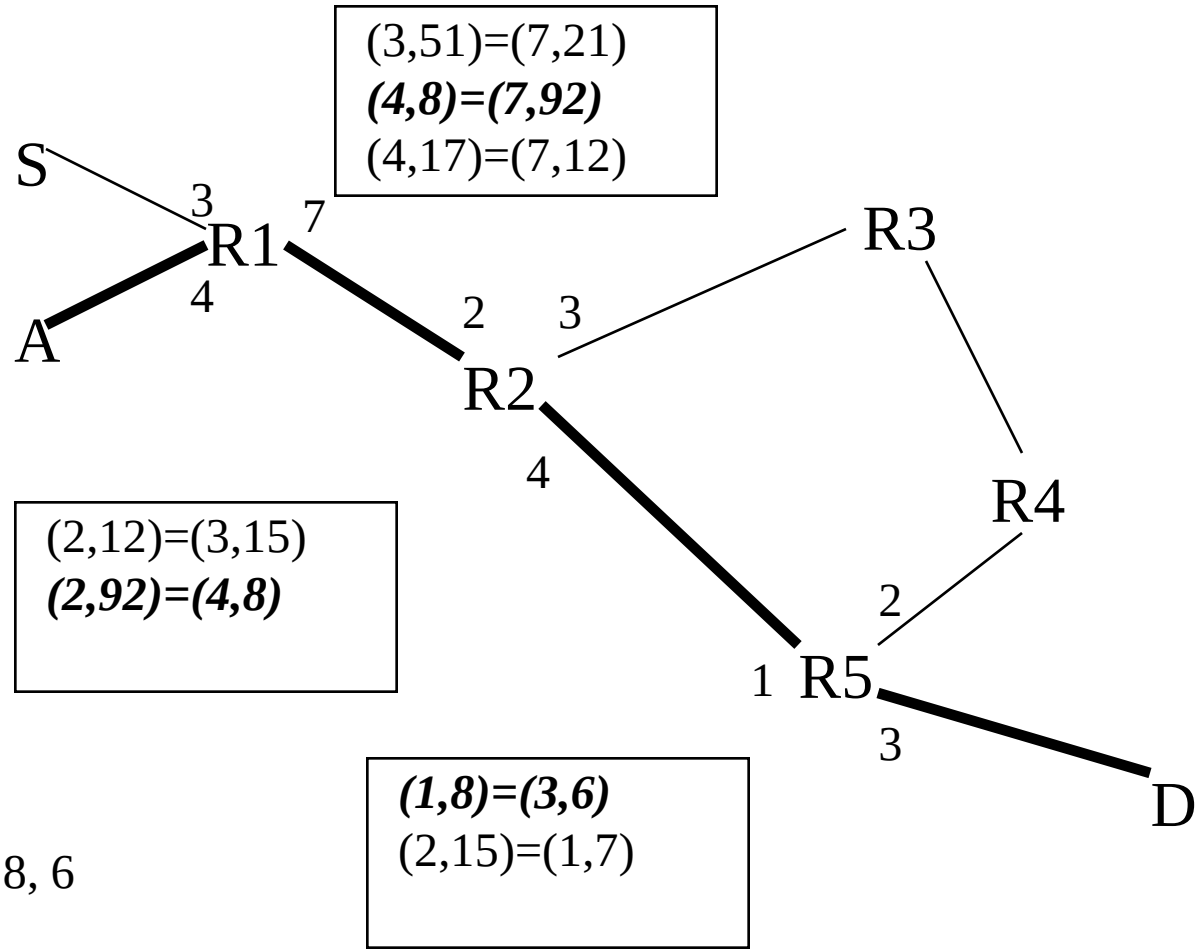
Summary of address-based alternatives

- Flat (dense or sparse) vs hierarchical
- Flat allows movement without changing address
- Hierarchical scales better by allowing summarization of all addresses in a blob
- You can move around within the blob without changing your address

Another possibility: “Label”

- Packet contains “label”, which changes hop by hop
- Forwarding table maps (input port, label) to (output port, replacement label)

Label-Based



VC=8, 92, 8, 6

Why labels?

- Originally for telephony (e.g., ATM, X.25)
  - Total nodes much greater than currently communicating pairs
    - Therefore, label can be shorter than destination address, and densely assigned on each link
  - OK to have latency while call set up
- For IP, grew out of "tag switching" (MPLS)
- Used now for "traffic engineering"
  - However, you could do traffic engineering using a destination rather than a label that changes hop by hop
  - You could have multiple destination addresses for a destination if you want multiple paths to that destination

Some thoughts

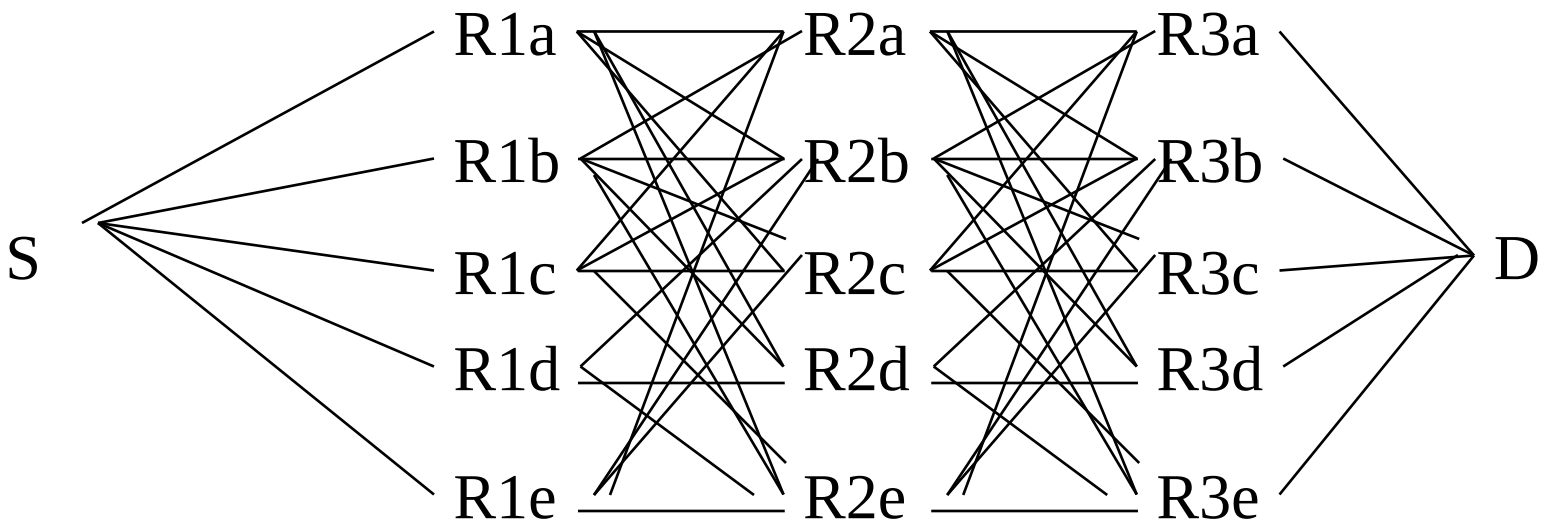
- Dest-based vs label-based
  - Destination-based is smaller (O(n)) forwarding table than label-based ($O(n^2)$)
  - Unless path set up when flow starts…but that incurs latency
  - Destination-based doesn't preclude traffic engineering
    - Infiniband "destination" is a path to the destination, set up by a central fabric manager. If you want multiple paths to a D, give D multiple addresses

Different topic: Keeping packets in order

Different topic: Keeping packets in order

- Keep packets for the same flow in order
  - Because endnode protocols or implementations will perform badly or actually fail (?)
  - Or so destination immediately knows if a packet was lost (if you get n, then n+2, and they are in order, n+1 got lost)
- We want to spread traffic among lots of paths

Exploiting parallel paths



Various strategies

- Switch R's forwarding table contains a set of ports for "X"
  - R parses more of the packet; identify "flow" by TCP ports, etc.
    - R remembers which flows it has assigned to which port (a lot of state); assigns port when new flow
    - R hashes (ports, source, etc.) to choose same port
- Switch R's forwarding table has one port for "X"
  - Source chooses

If R's forwarding table contains multiple ports for X

- Parse more of the packet, e.g., TCP ports, source address, …
  - Remember chosen port for each flow; assign port if new flow
  - Hash information to select same port

If R's forwarding table contains only one port for each X

- Have multiple entries in R's forwarding table for the destination (e.g., multiple destination addresses). Source's choice determines the entire path

“Entropy Field”

- A field added to the header for two purposes
  - To save each switch along the way from having to do deep packet inspection to find ports, etc.
  - To allow the source to spread the load for its own flow, by choosing different entropy labels

Another concept: Flow-based

- Forwarding table based on (destination, source, protocol type, TCP ports)
- Claim: better for spreading load
  - If a central entity knows what all the flows are, it can carefully place them

Seems to me…

- 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

Seems to me

- Switch in better position to path split based on local queues
- Would be much better if it didn't have to keep things in order
- Even if it has to keep things in order, it can re-hash to spread load
- "Flows" are not stable bandwidth…so central fabric manager knowing all the flows can't do as good a job as switches

Opportunities for research

- Compare traffic utilization between knowledge of all flows, and having forwarding entries with a single choice for “flow”, or letting switches choose among a set of output ports for a destination
- Is congestion n hops away useful information, or is just local queue length good enough? (there is a cost to finding out congestion at other switches)

Completely orthogonal concept

New topic: Congestion

- Drop or backpressure?
- Backpressure
  - Credit based or pause/resume
- But can only do it based on a few buffer pools (“classes”)
- So congestion spreads
- You can inform source (e.g., TCP), but what if congestion is due to zillions of sources? Is information timely enough? How does it interact with path splitting?

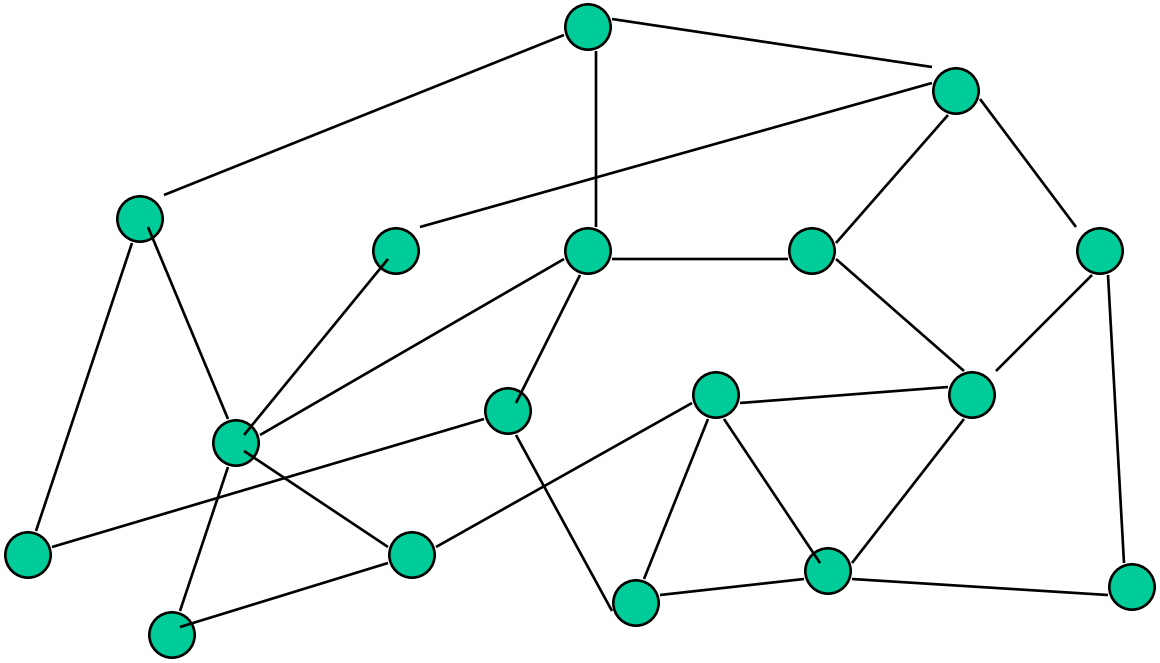
Completely orthogonal concept

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…

Distributed Routing Protocol

New topic: Distributed Routing Protocols

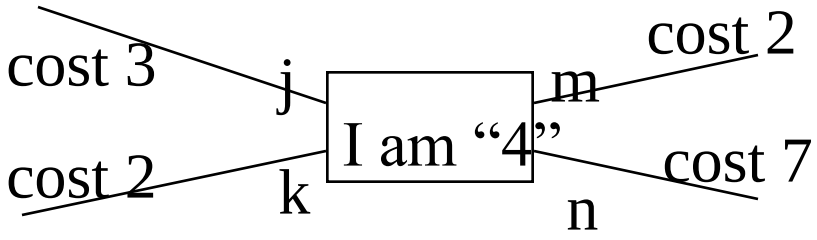


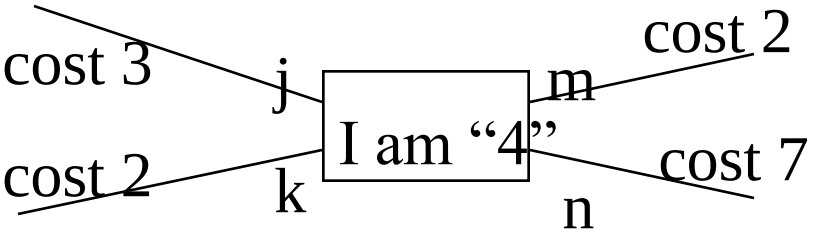
Distributed Routing Protocols

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

Distance Vector

- Know
  - your own ID
  - how many cables hanging off your box
  - cost, for each cable, of getting to nbr





distance vector rcv’d from cable j

cost 3

12	3	15	3	12	5	3	18	0	7	15

distance vector rcv’d from cable k

cost 2

5	8	3	2	10	7	4	20	5	0	15

distance vector rcv’d from cable m

cost 2

0	5	3	2	19	9	5	22	2	4	7

distance vector rcv’d from cable n

cost 7

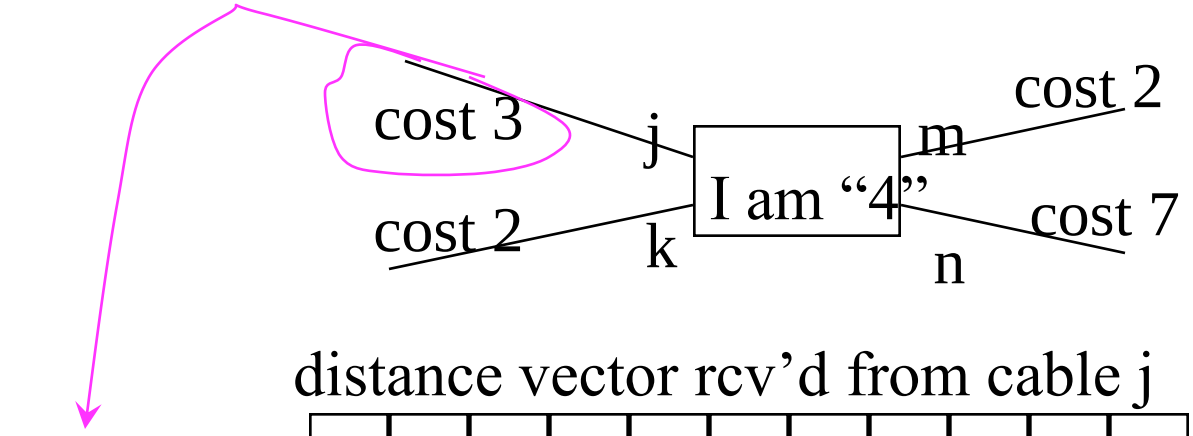
6	2	0	7	8	5	8	12	11	3	2

your own calculated distance vector

2	6	5	0	12	8	6	19	3	?	?

your own calculated forwarding table

m	j	m	0	k	j	k/j	n	j	?	?



distance vector rcv’d from cable j

cost 3

12	3	15	3	12	5	3	18	0	7	15

distance vector rcv’d from cable k

cost 2

5	8	3	2	10	7	4	20	5	0	15

distance vector rcv’d from cable m

cost 2

0	5	3	2	19	9	5	22	2	4	7

distance vector rcv’d from cable n

cost 7

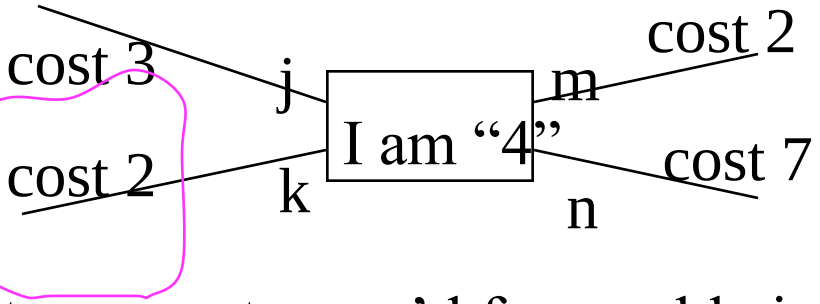
6	2	0	7	8	5	8	12	11	3	2

your own calculated distance vector

2	6	5	0	12	8	6	19	3	?	?

your own calculated forwarding table

m	j	m	0	k	j	k/j	n	j	?	?



distance vector rcv’d from cable j

cost 3

12	3	15	3	12	5	3	18	0	7	15

distance vector rcv’d from cable k

cost 2

5	8	3	2	10	7	4	20	5	0	15

distance vector rcv’d from cable m

cost 2

0	5	3	2	19	9	5	22	2	4	7

distance vector rcv’d from cable n

cost 7

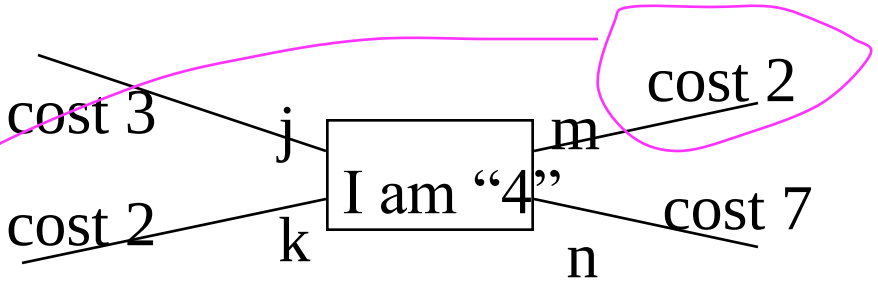
6	2	0	7	8	5	8	12	11	3	2

your own calculated distance vector

2	6	5	0	12	8	6	19	3	?	?

your own calculated forwarding table

m	j	m	0	k	j	k/j	n	j	?	?



distance vector rcv’d from cable j

cost 3

12	3	15	3	12	5	3	18	0	7	15

distance vector rcv’d from cable k

cost 2

5	8	3	2	10	7	4	20	5	0	15

distance vector rcv’d from cable m

cost 2

0	5	3	2	19	9	5	22	2	4	7

distance vector rcv’d from cable n

cost 7

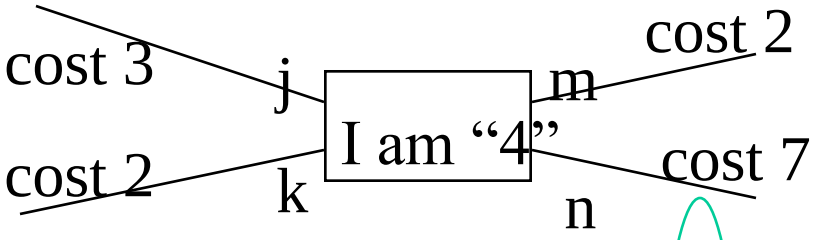
6	2	0	7	8	5	8	12	11	3	2

your own calculated distance vector

2	6	5	0	12	8	6	19	3	?	?

your own calculated forwarding table

m	j	m	0	k	j	k/j	n	j	?	?



distance vector rcv’d from cable j

cost 3

12	3	15	3	12	5	3	18	0	7	15

distance vector rcv’d from cable k

cost 2

5	8	3	2	10	7	4	20	5	0	15

distance vector rcv’d from cable m

cost 2

0	5	3	2	19	9	5	22	2	4	7

distance vector rcv’d from cable n

cost 7

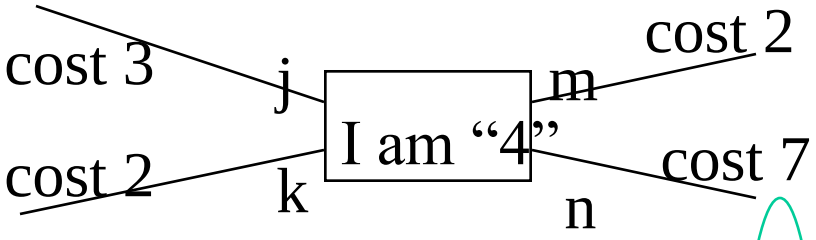
6	2	0	7	8	5	8	12	11	3	2

your own calculated distance vector

2	6	5	0	12	8	6	19	3	?	?

your own calculated forwarding table

m	j	m	0	k	j	k/j	n	j	?	?



distance vector rcv’d from cable j

cost 3

12	3	15	3	12	5	3	18	0	7	15

distance vector rcv’d from cable k

cost 2

5	8	3	2	10	7	4	20	5	0	15

distance vector rcv’d from cable m

cost 2

0	5	3	2	19	9	5	22	2	4	7

distance vector rcv’d from cable n

cost 7

6	2	0	7	8	5	8	12	11	3	2

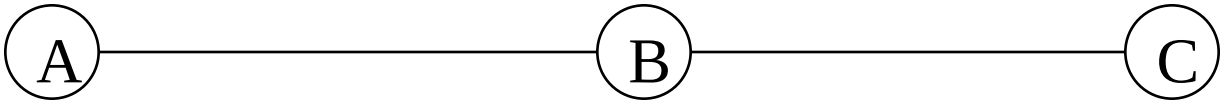
your own calculated distance vector

2	6	5	0	12	8	6	19	3	?	?

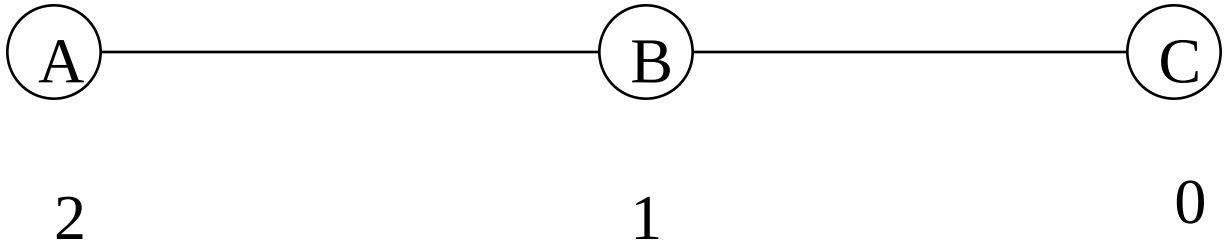
your own calculated forwarding table

m	j	m	0	k	j	k/j	n	j	?	?

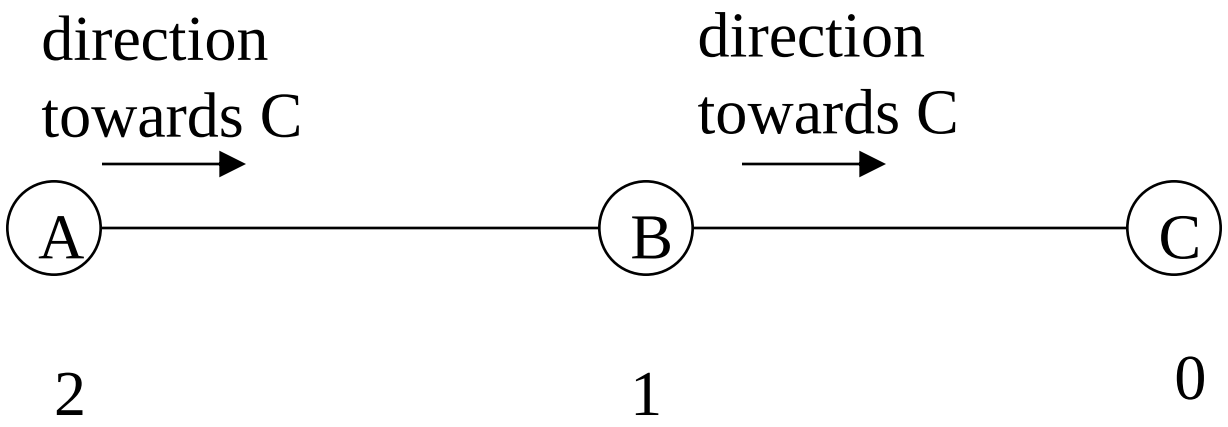
Looping Problem



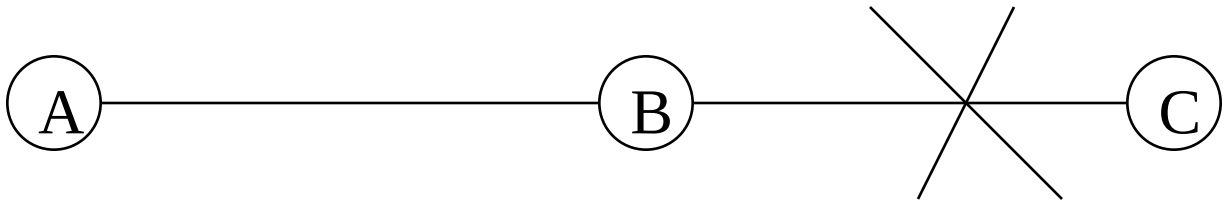
Looping Problem



Looping Problem

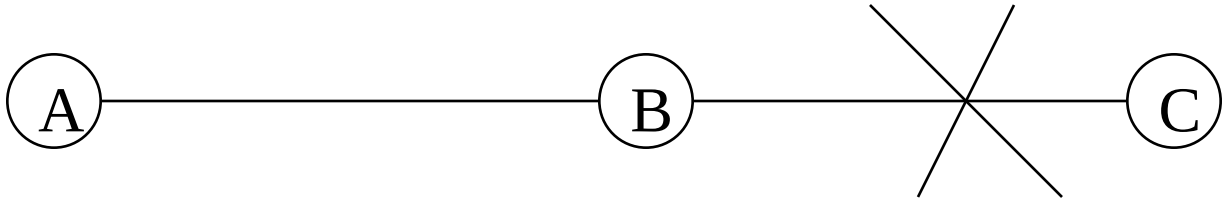


Looping Problem



What is B's cost to C now?

Looping Problem



2

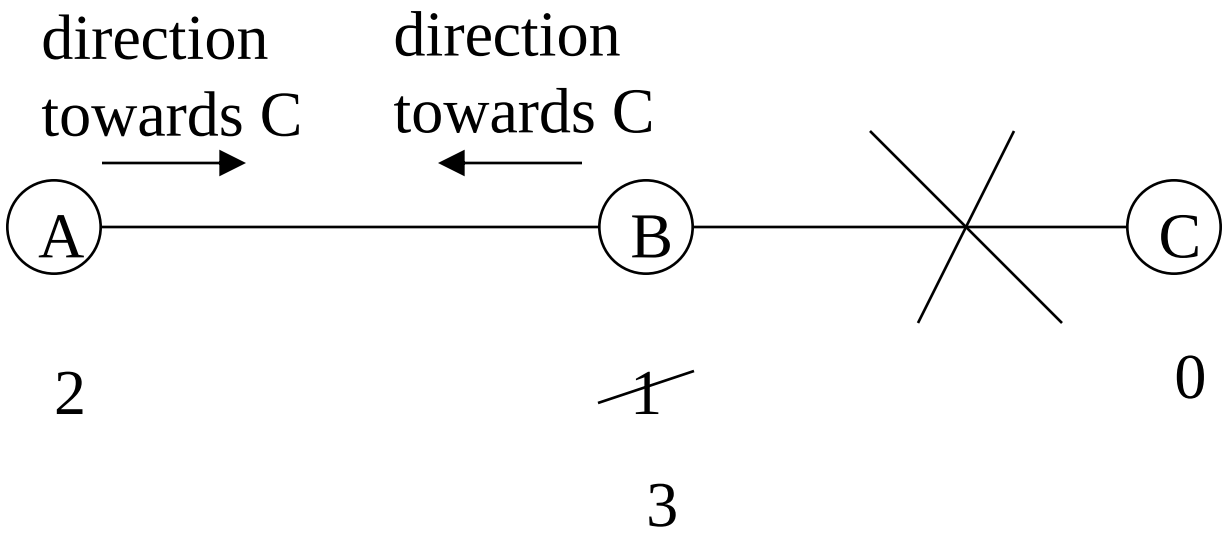
~~1~~

3

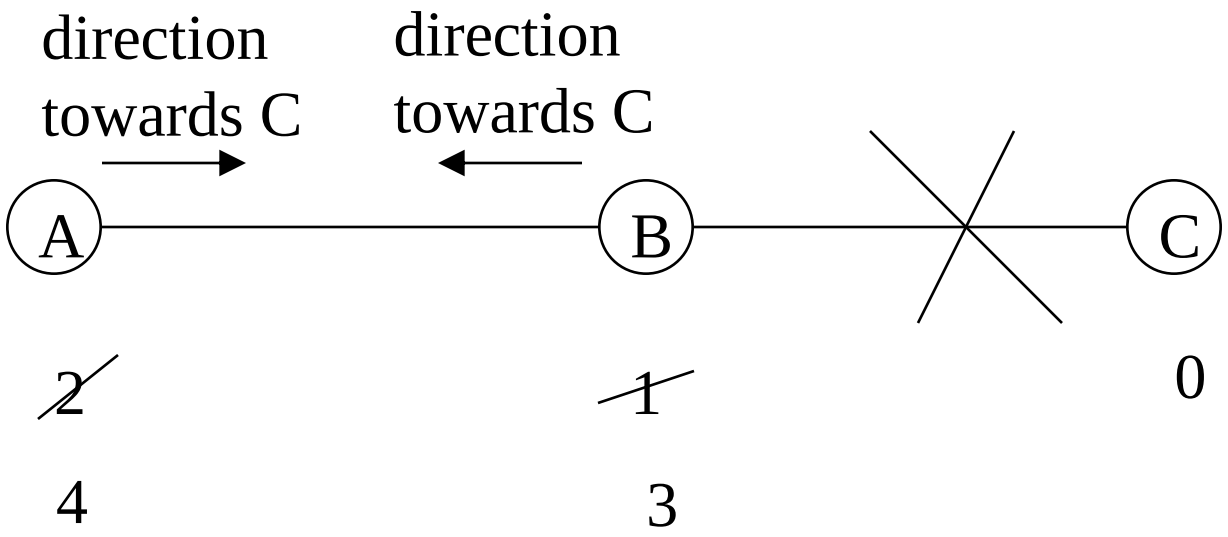
0

Cost to C

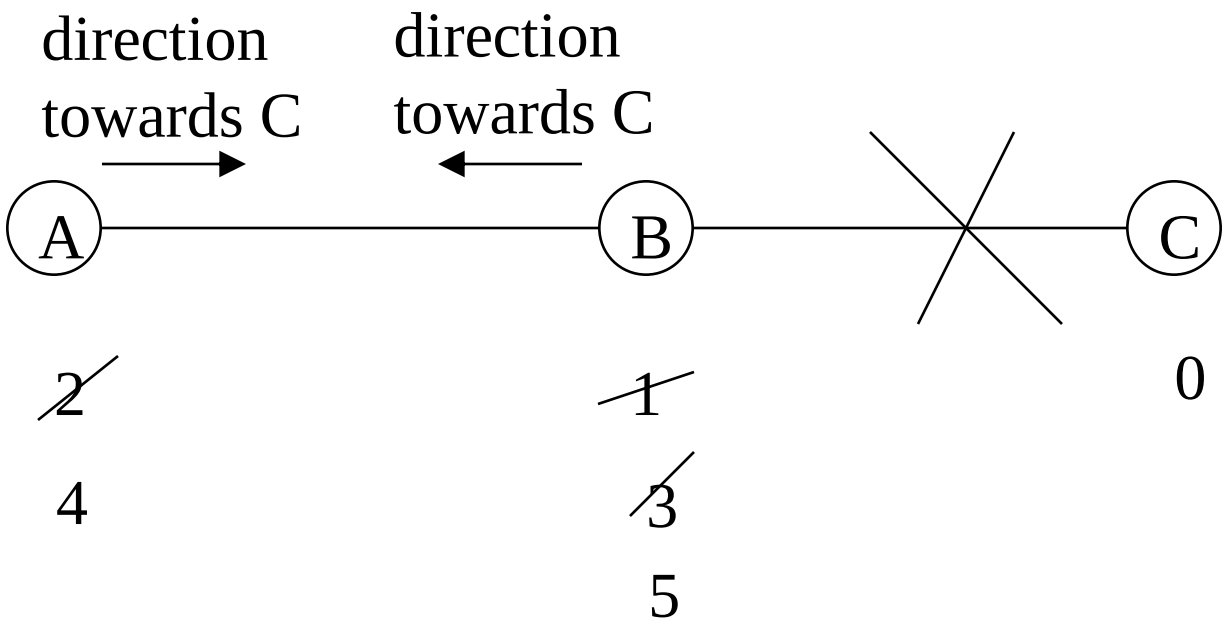
Looping Problem



Looping Problem



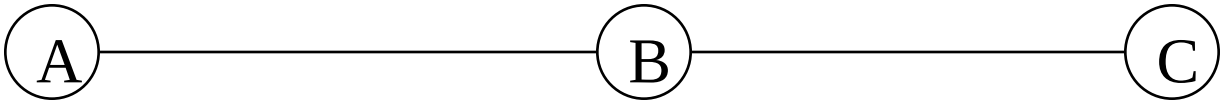
Looping Problem



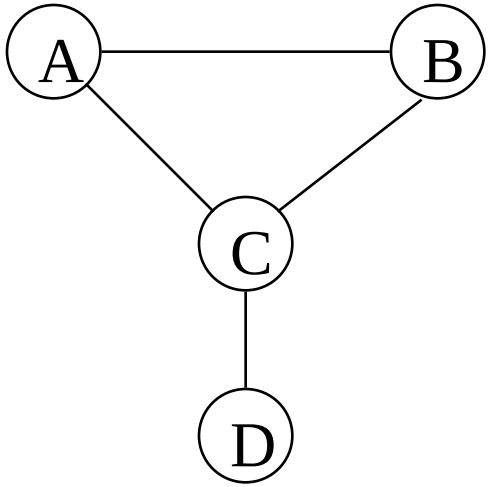
Looping Problem
worse with high connectivity

Split Horizon: one of several optimizations

Don’t tell neighbor N you can reach D if you’d forward to D throug

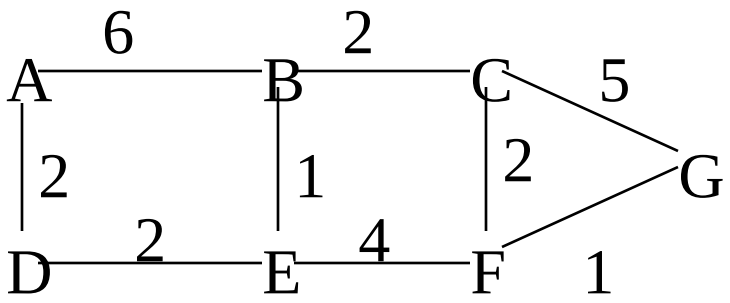


Split Horizon: …but it won’t work with loops of more than 2 nodes



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)



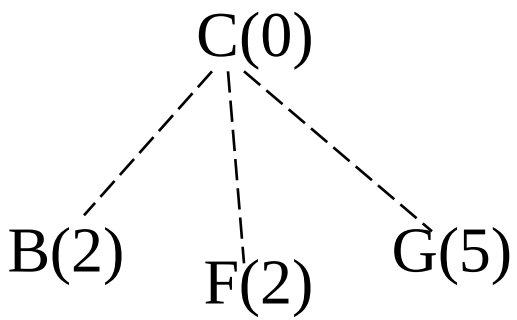
A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	

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

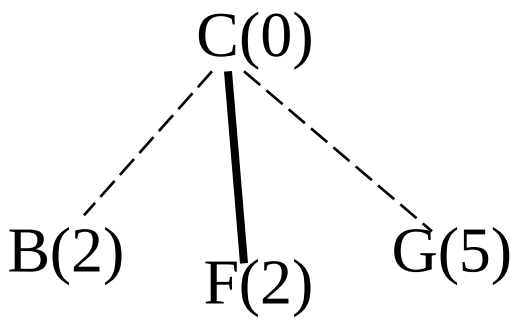
Look at LSP of new tree node

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	



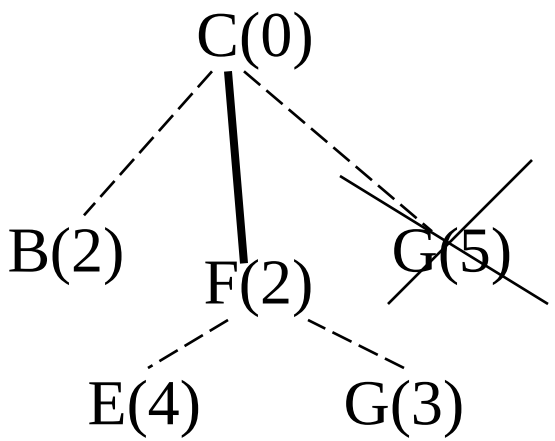
Make shortest TENT solid

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	



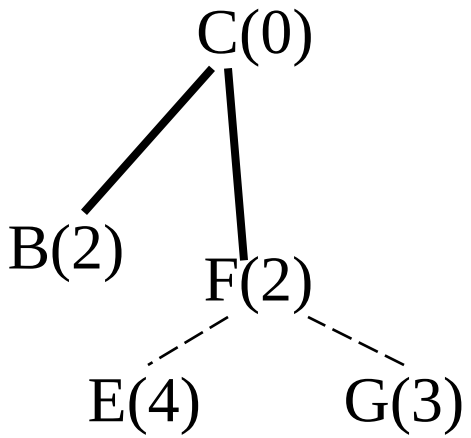
Look at LSP of newest tree node

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	



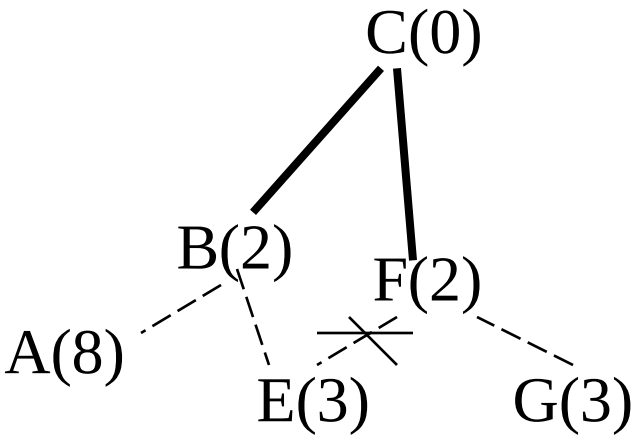
Make shortest TENT solid

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	



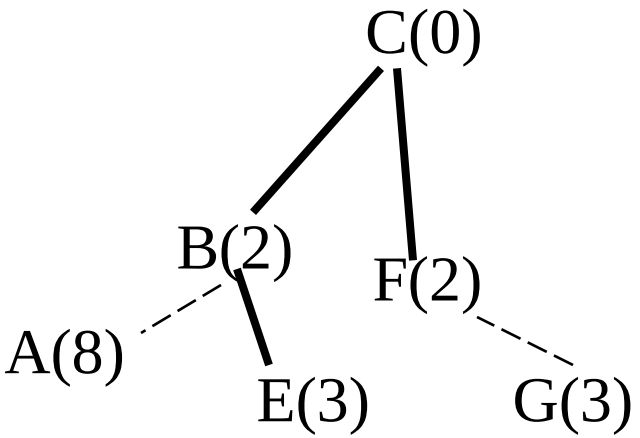
Look at LSP of newest tree node

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	



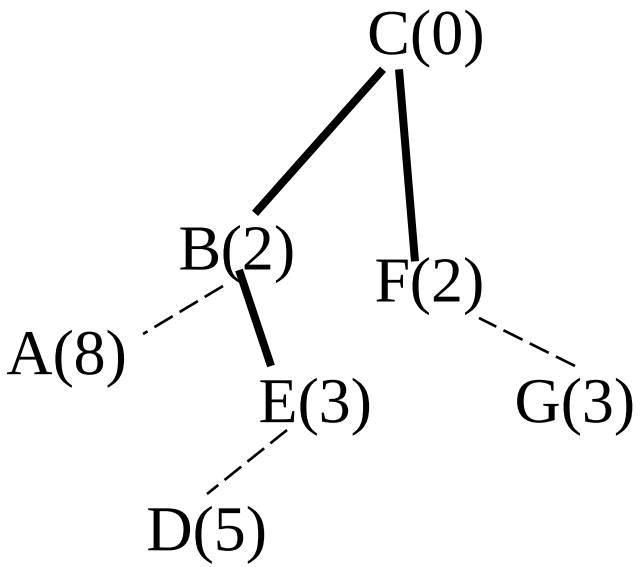
Make shortest TENT solid

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	



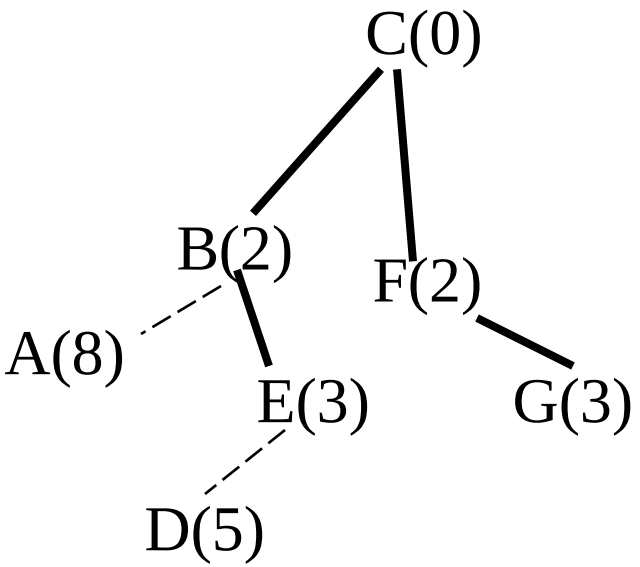
Look at LSP of newest tree node

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	



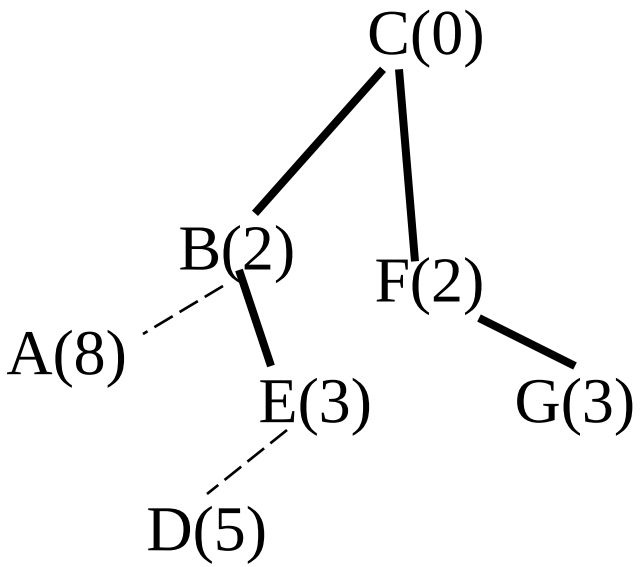
Make shortest TENT solid

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	



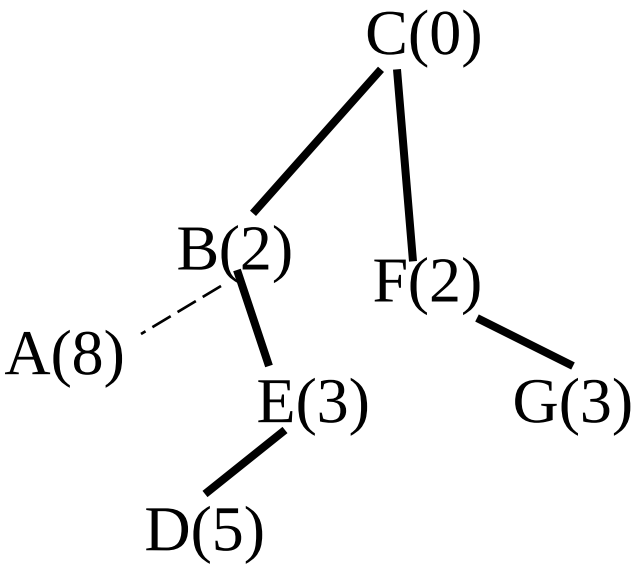
Look at newest tree node's LSP

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	



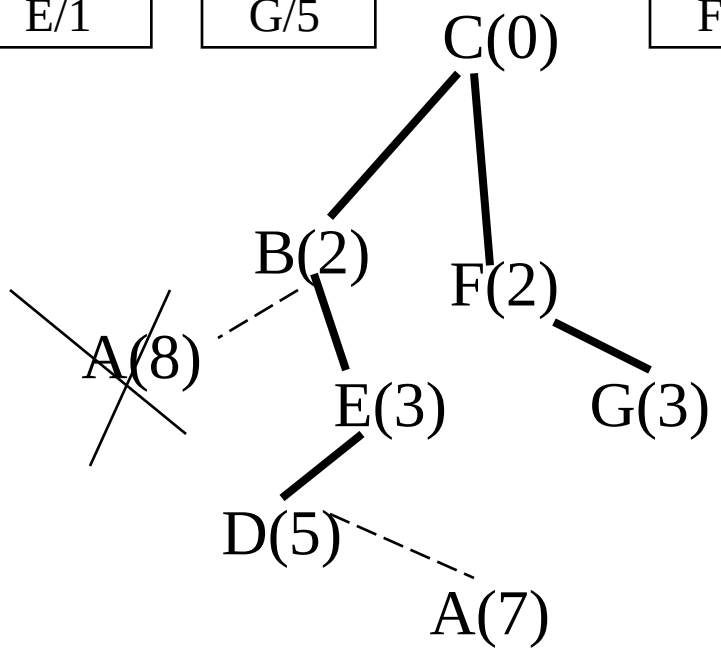
Make shortest TENT solid

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	



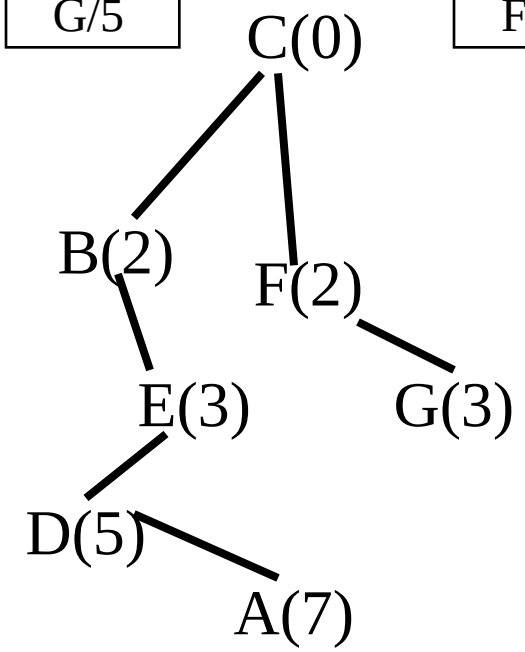
Look at newest node's LSP

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	



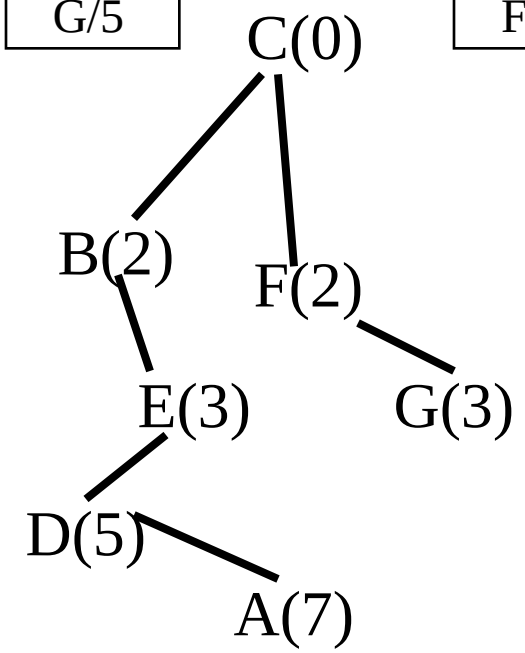
Make shortest TENT solid

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	

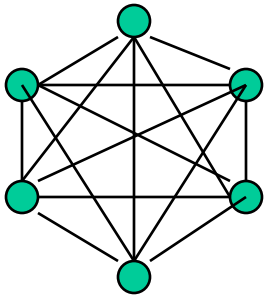


Another interesting detail of link state

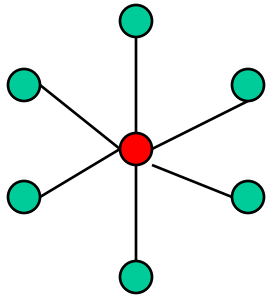
- Pseudonodes
- Since routing algorithm is proportional to the number of links
- If an Ethernet with 100s of nodes were considered fully connected, the link state database would be too large

Pseudonodes

Instead of:



Use pseudonode



Designated Routers

- Elect a router to be the master of the link
- It names the pseudonode
  - In IS-IS, a node's ID is 7 bytes: 6 bytes of system ID (usually the MAC address of one of its ports), plus an extra byte. E.G., R1 is DR, names link R1.25
- All routers (including R1) claim a link to R1.25
- R1 (pretending to be the pseudonode), claims connectivity to each of the routers on the link

Central Fabric Manager

- This has been deployed (e.g., ATM, Infiniband)
- One place (“fabric manager”) collects topology information, calculates paths, distributes forwarding tables to all the switches

Seems to me…

- Link state routing protocol will be more responsive to topology changes than central fabric manager…especially if link failure affects path to fabric manager
- Route computation criteria can be changed by modifying link costs

When does forwarding table get filled in?

- Proactively
- When a flow starts

Proactively seems better

- Rather than paying latency to create entry

How do you manage a network?

Original vision:

- A management console translates “big” commands, e.g.,“forward using this metric” or “traffic engineer this path” into individual commands to switches
- Protocols define a MIB (management information base) that says which parameters are readable, writeable, what events alert management station
- Define a standard language to read/set parameters remotely (e.g., SNMP)

Mystery (to me)

- Apparently that original vision degraded into often remotely logging into each switch and doing CLI (command line interface) commands
- Why? (proprietary features not defined in MIB…vendor's fault? Standards body fault? Need new features (like atomic transaction?) in SNMP?
- If we define a new thing like SNMP, will things degrade over time the same way? Is a new thing easier than reviving SNMP/Netconf?

New Topic: What is layer 2 vs layer 3?

Forwarding at layer 2 vs layer 3

- Extremely confusing, without knowing the history of IP, Ethernet, TRILL, etc.

So…why are we forwarding Ethernet packets?

- Ethernet was intended to be layer 2
- Just between neighbors – not forwarded

So…why are we forwarding Ethernet packets?

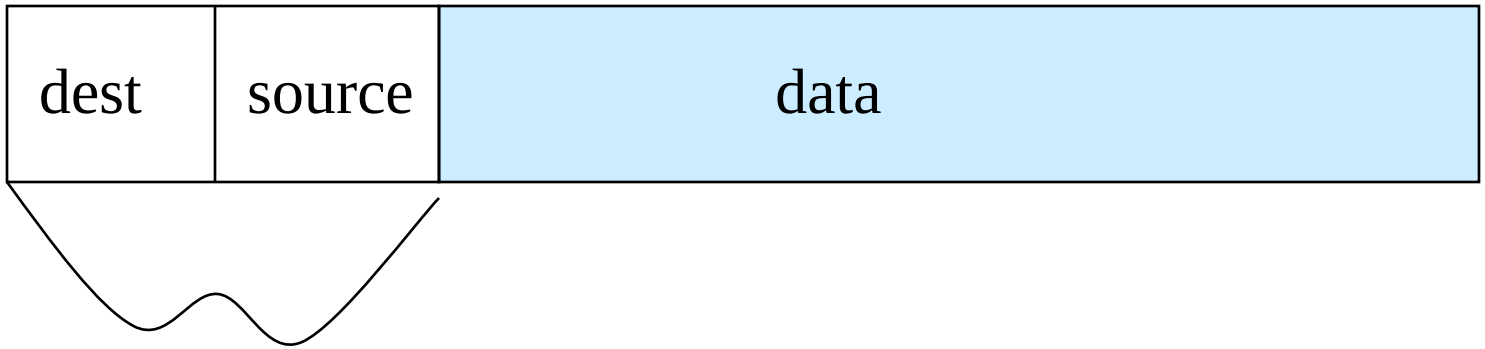
- Ethernet was intended to be layer 2
- Just between neighbors – not forwarded
- What exactly is Ethernet?

Back then…

- I was layer 3 architect for DECnet
- Layer 3 calculate paths, and forwarded packets
- Layer 2 just marked beginning and end of packet, and checksum
- Then along came Ethernet

The story of Ethernet

Ethernet packet



Ethernet header: 6 byte addresses – strangely large…because it allows autoconfiguration

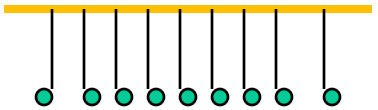
Plus stuff like protocol type and VLAN

The story of Ethernet

- CSMA/CD
- Spanning Tree
- TRILL
- Futures?

CSMA/CD Ethernet

- CSMA/CD…shared bus, peers, no master
  - CS: carrier sense (don't interrupt)
  - MA: multiple access (you're sharing the air!)
  - CD: listen while talking, for collision



- Lots of papers about goodput under load only about 60% or so because of collisions
- Limited in # of nodes (maybe 1000), distance (kilometer or so)

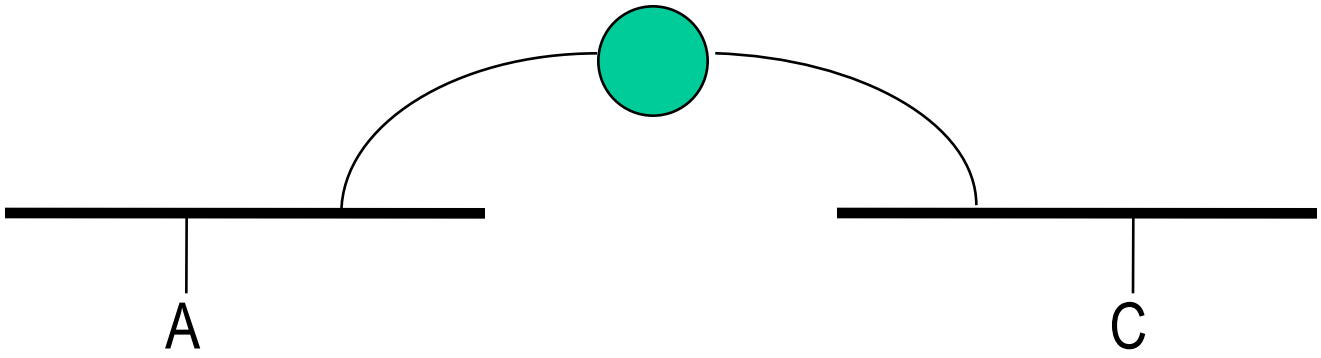
But Ethernet hasn't been CSMA/CD for decades

How it evolved to spanning tree

- People got confused, and thought Ethernet was a network instead of a link
  - Link (layer 2) = nbr-nbr
  - Network (layer 3) = forward along a path
- Built apps on Ethernet, with no layer 3
- Router can’t forward without the right envelope

Problem Statement (from about 1983)

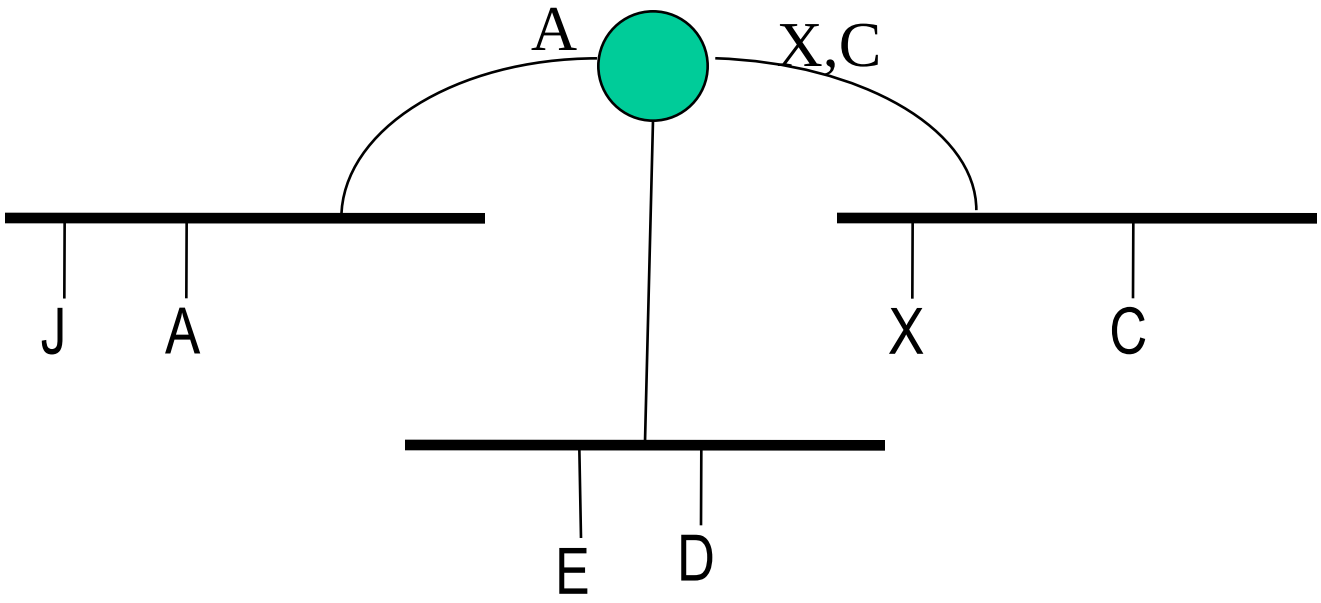
Need something that will sit between two Ethernets, and let a station on one Ethernet talk to another



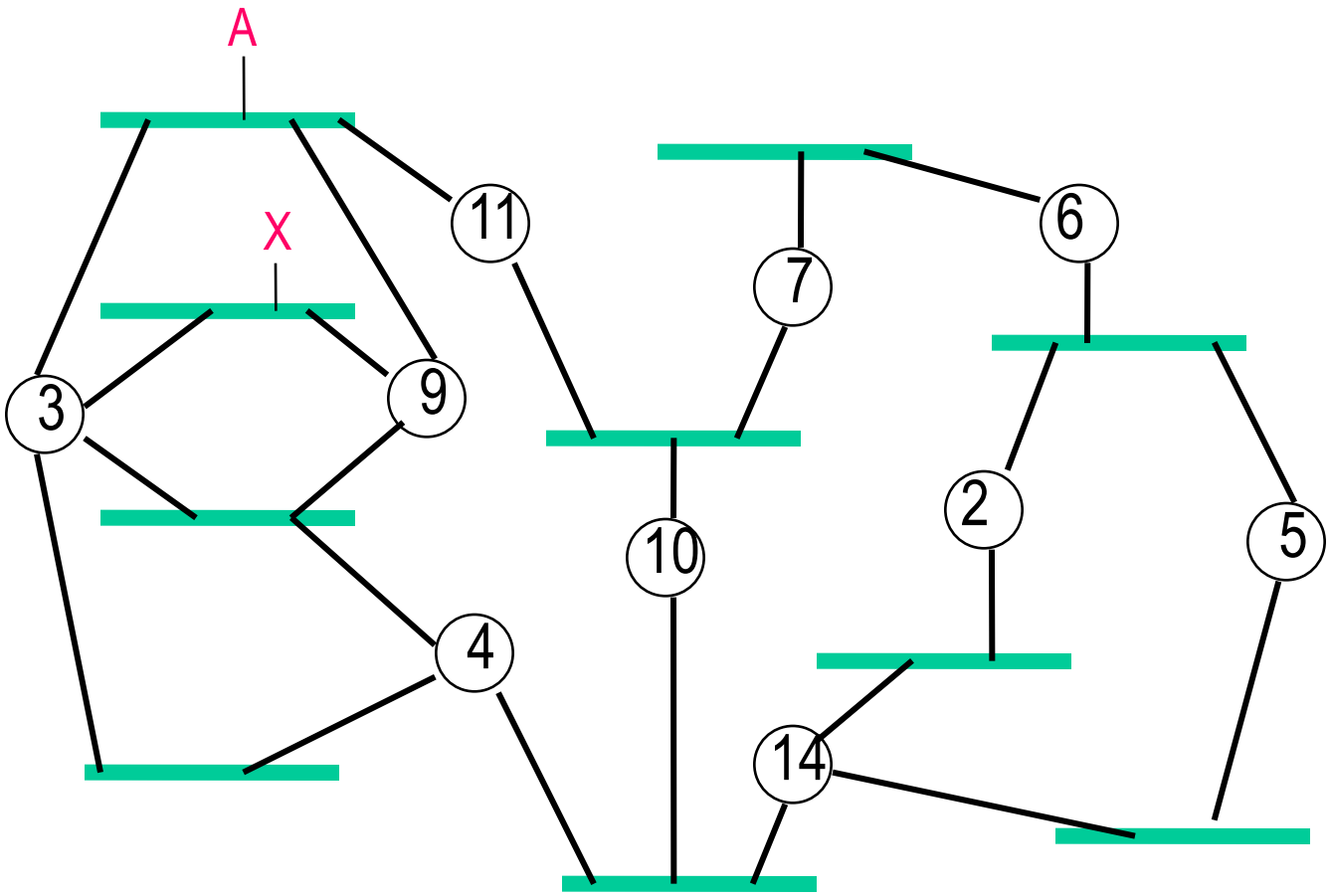
Without modifying the endnode, or Ethernet packet, in any way

The basic concept

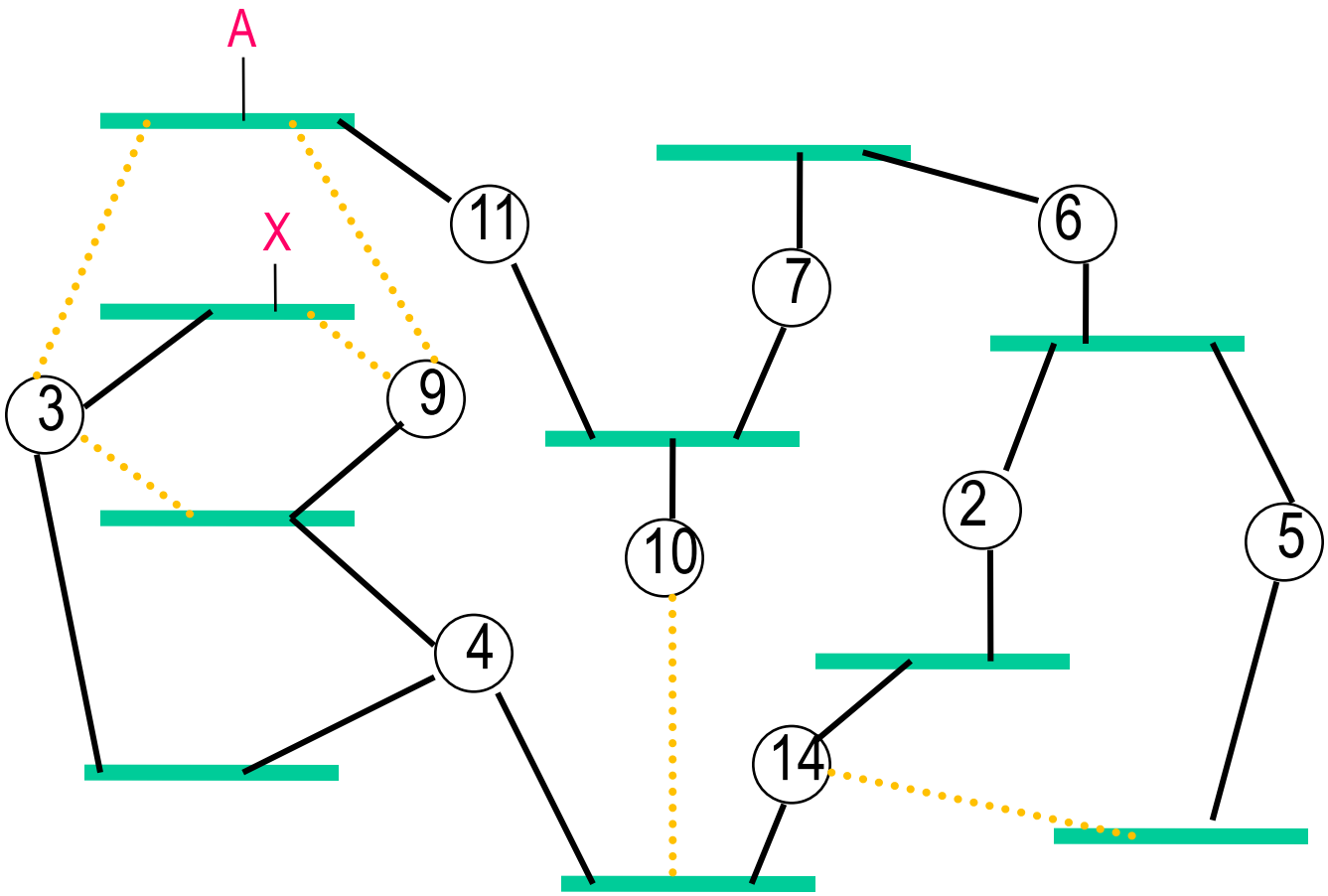
- Bridge just listens promiscuously, and forwards to each other port when the ether is free
- Learn (Source=S, input port). Once learned, if see a packet with destination=S, know where to forward it (rather than “all the ports”)
- This requires a tree (no loops) topology


A
X,C
J
A
X
C
E
D

Physical Topology



Pruned to Tree



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.

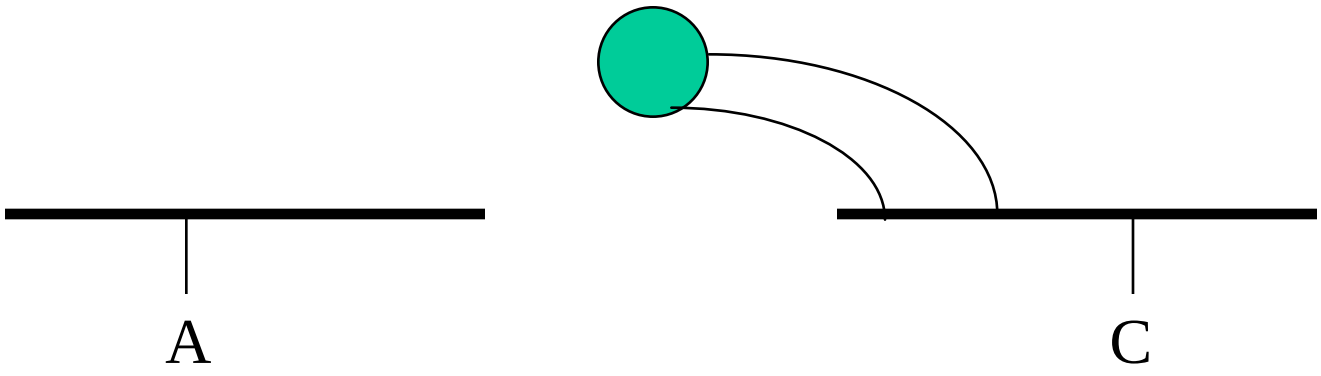
A mesh is made by folks like me.
Then bridges find a spanning tree.

Radia Perlman

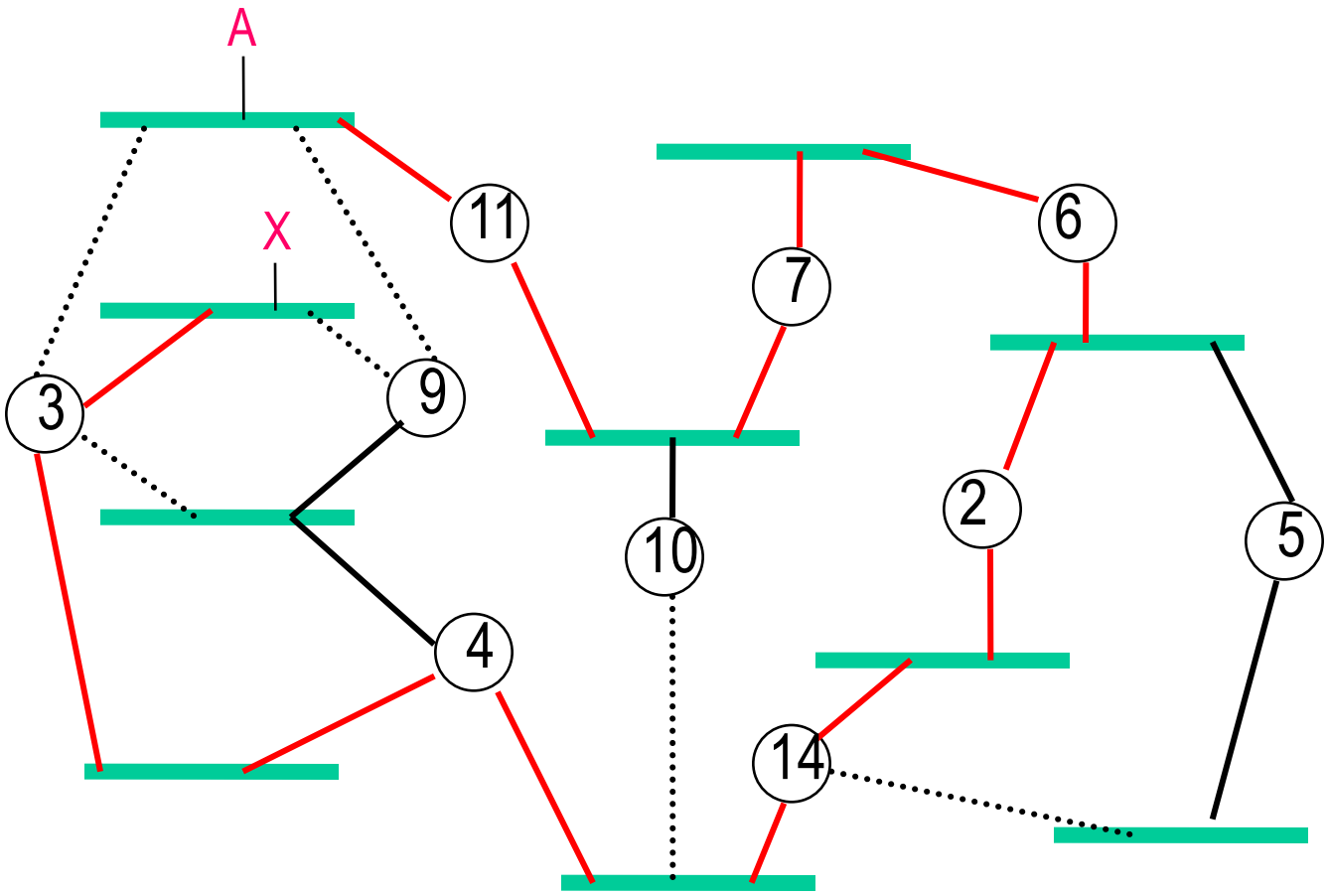
Bother with spanning tree?

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

First Bridge Sold



Problems with spanning tree: suboptimal paths, Unused links



Why not just use IP routers?

- World has converged to IP as layer 3, and it’s in the network stacks

Why not just use IP routers?

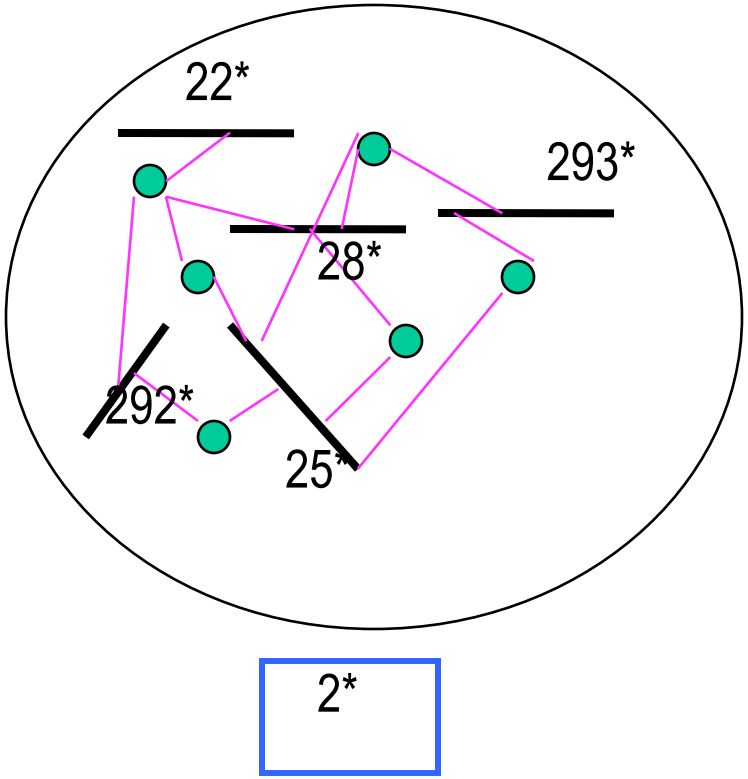
- IP is configuration intensive, moving VMs disruptive
  - IP protocol requires every link to have a unique block of addresses
  - Routers need to be configured with which addresses are on which ports
  - If something moves, its address changes

Layer 3 doesn't have to work that way!

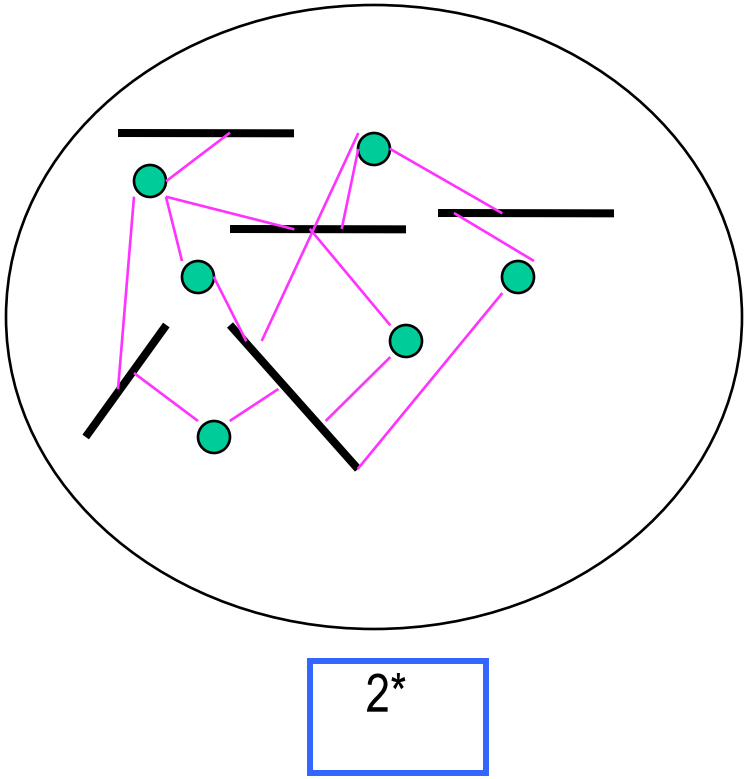
- CLNP / DECnet...20 byte address
  - Bottom level of routing is a whole cloud with the same 14-byte prefix
  - Routing is to 6 byte ID inside the cloud
  - Enabled by "ES-IS" protocol, where endnodes periodically announce themselves to the routers

Hierarchy

One prefix per link (like IP)



One prefix per campus



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 level 1 clouds)

Ethernet looks like a single IP link

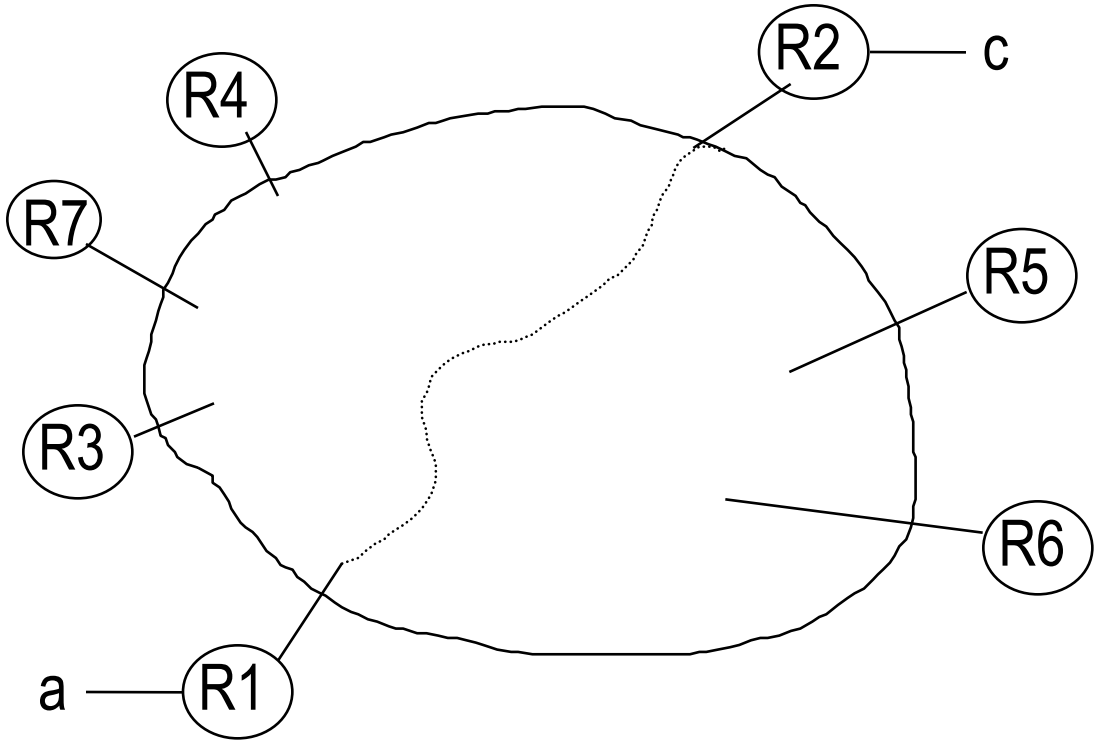
- So Ethernet provides a large cloud in which switches can autoconfigure, and nodes (e.g., VMs) can move around transparently
- But don’t want limitations of spanning tree

Next step in evolution: TRILL

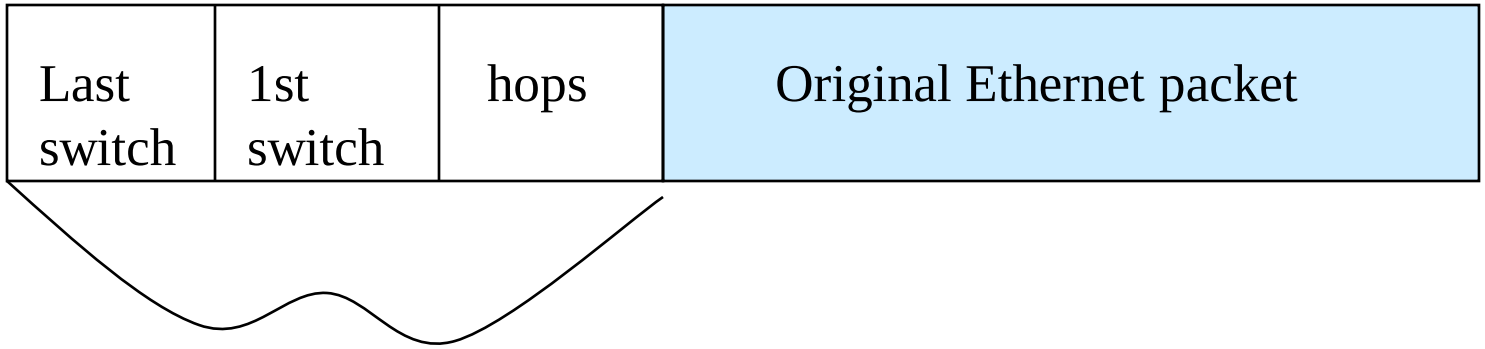
TRILL

- TRansparent Interconnection of Lots of Links
- Basic idea: Put Ethernet in another envelope that acts more like a layer 3 envelope, and can be routed

TRILL



TRILL packet



TRILL header
Switch addresses are 16 bits

16-bit TRILL switch "nicknames"

- Allows 64,000 switches…many more endnodes
- TRILL autoconfigures nicknames
- Allows simple forwarding table lookup
  - Direct table lookup
  - Don't need associative memory, or hash, or longest prefix match

Advantage of extra header

- Switches inside cloud don't need to know about all the endnodes…
  - Forwarding table size of # of switches
- The outer header is like a layer 3 header, and can use all the layer 3 techniques, e.g.,
  - Shortest paths
  - Multiple paths (exploit parallelism)
  - Traffic engineering

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

Note: TRILL is evolutionary

- Endnodes just think it's Ethernet…no changes
- Even interworks with existing spanning tree switches
- The more switches you upgrade to TRILL, the better the bandwidth utilization
- This could have been implemented by a single vendor, without standardizing

Orthogonal concept

Who encapsulates/decapsulates?

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

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 transparently.
Without a common spanning tree.

Ray Perlner

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

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

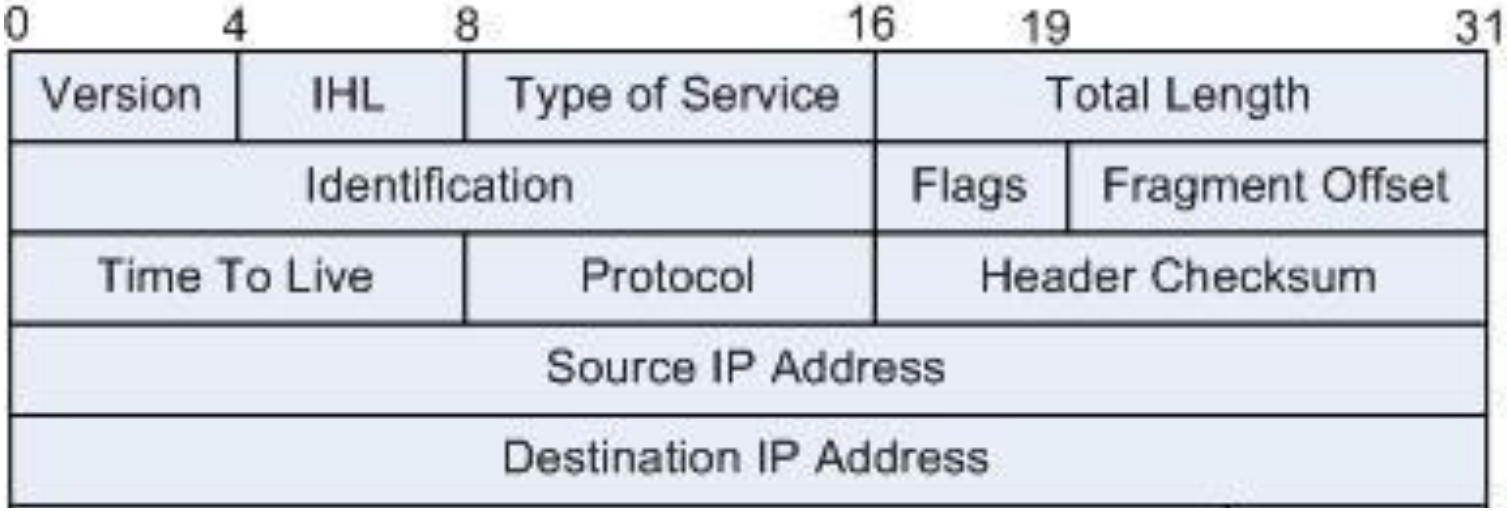
- Easier forwarding lookup inside bottom layer (16 bit nicknames)
- Smaller forwarding table in inner switches (table size # of switches rather than endnodes)
- Ability to multipath multicast
- VLANs

Last Orthogonal Thought: Latency

Latency

- Store-and-forward vs cut-through
- Cut through can start after the forwarding decision is made
- What field do you need to see for forwarding decision?

IPv4 header


0
4
8
16
19
31
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time To Live
Protocol
Header Checksum
Source IP Address
Destination IP Address

IPv6 header


32 Bits
8
8
8
8
Version = 6
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address

Another latency mistake

- TCP has checksum in the header
- So can’t start transmitting until you see the whole packet

Parting thoughts

- Don’t believe anything about “technology X” unless there is a plausible inherent reason for it
- Don’t get carried away by buzzwords
- Know what problem you’re solving before you start on the solution

Questions?