Cloud Fabric: Myths, Missteps, and Mysteries

Radia Perlman
Intel Labs

radia.perlman@intel.com

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

What about “facts”?

- What if you measure A vs B?

What about “facts”?

- What if you measure A vs B?
- What are you actually measuring?...one implementation of A vs one implementation of B

How I wish we’d compare

- Isolate conceptual pieces
- Try to ignore buzzwords or “which team”

Some really confusing stuff

- We talk about “layer 2 solutions” vs “layer 3 solutions”….what’s that about?

Basic network protocols

- Simple…an envelope in which you put your data
- Envelope contains, e.g., source, destination
- Switch has forwarding table that indicates (based on info in packet) output port or set of ports

“Switch”

- Something that forwards (e.g., bridge, router, switch)

What does a switch do?

- Forward based on:
  - Info in packet
    - Destination address or “label” (like MPLS, changes at each hop and represents an S-D path)
    - If need to keep things in order, other stuff in packet (e.g., TCP ports, flow ID, entropy field)
  - Forwarding table

When does forwarding table get filled in?

- Proactively
- When a flow starts

Seems to me…

- Proactively is better…otherwise latency while setting up a path for a new flow

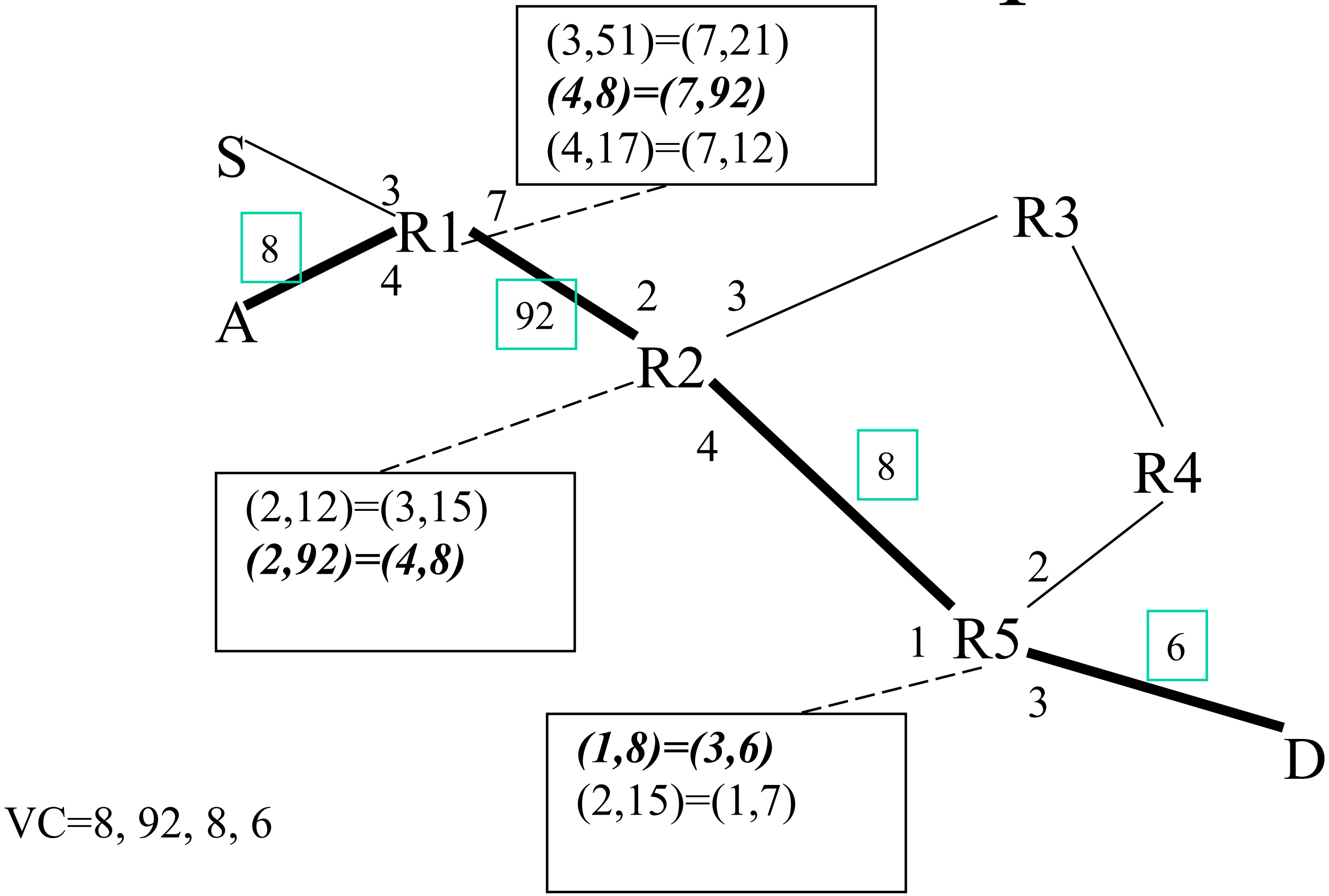
Info in packet

- Forwarding table indexed by
  - destination vs label vs flow
- Forwarding table gives single port or set of ports (allowing switch to choose)
- Preview: I think destination-based is best, with set of ports

Destination alternatives

- Flat or hierarchial
  - Flat
    - Convenient for moving without changing address
    - Dense vs sparse: dense can be direct lookup, sparse (as in 6-byte Ethernet address) requires hash
  - Hierarchical
    - Makes forwarding table smaller
    - Either reserve certain bits for each level, or be flexible and have to do longest prefix match to find proper forwarding entry

“Label”: is a path



(3,51)=(7,21)
(4,8)=(7,92)
(4,17)=(7,12)

(2,12)=(3,15)
(2,92)=(4,8)

(1,8)=(3,6)
(2,15)=(1,7)

VC=8, 92, 8, 6

Flow-based

- Each forwarding table entry is for a single conversation…more specific than (S-D)
  - E.g., source, destination, TCP ports

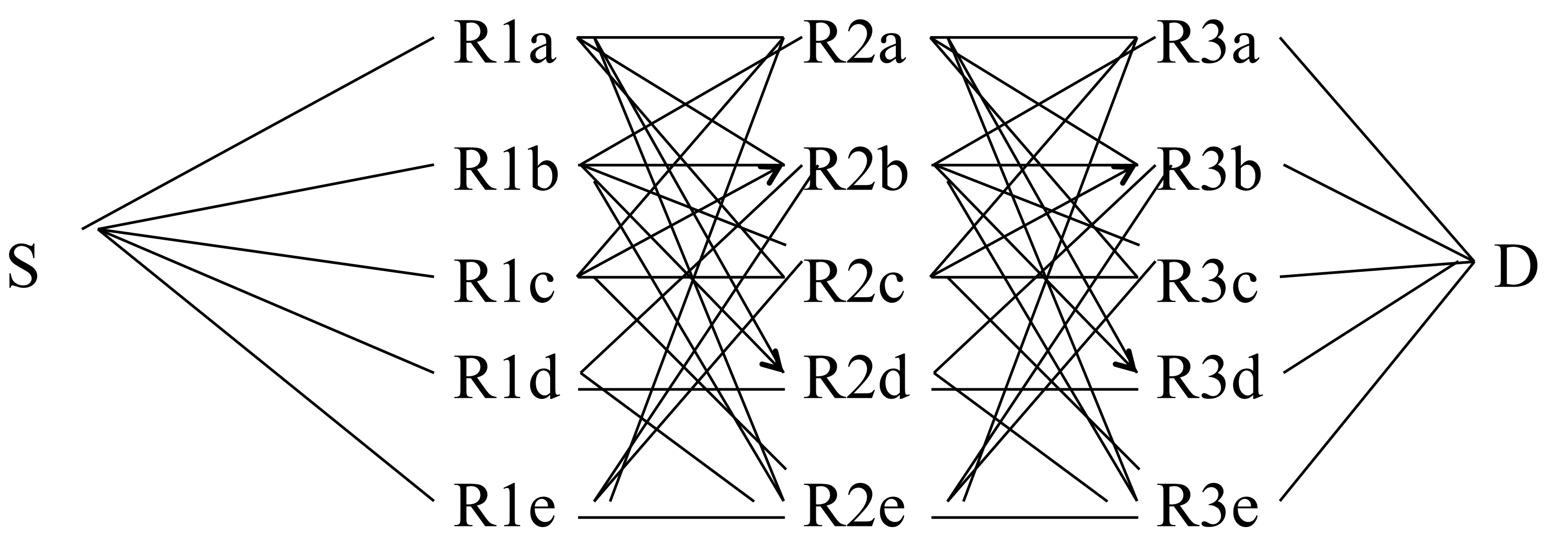
Some thoughts

- Dest-based vs label-based
  - Destination-based is smaller (O(n)) forwarding table than label-based ($O(n^2)$)
  - People think label-based is for traffic engineering, but can do traffic engineering with destination-based using some special destination addresses
  - ATM did label-based because
    - # of currently communicating pairs much smaller than total number of destination
    - OK to have latency to set up a conversation
  - MPLS did it because it grew out of “tag-switching”

More thoughts

- Flow-based vs destination-based
  - Only way to make flow-based not totally explode the forwarding table is to create entry when flow starts (incur latency)
  - Switch in better position to load-split traffic than central fabric manager

Exploiting parallel paths



Load splitting and keeping packets in order

- Source chooses the path
  - With a label or with choice of destination addresses for a destination (each one having a different path)
- Forwarding table based on flow
- Switch looks at other info to choose port
  - Deep packet inspection (e.g., TCP ports)
  - “entropy field”
  - Either way, deterministically choose same path for same flow

Research Suggestion

- Suppose a central place knows about all the flows
- What spreads traffic better?
  - Switches based on local output queues?
    - What about knowing about congestion k hops away?
  - Central place carefully placing all the paths for all the flows?

Seems to me…

- Better to give switches choices per destination, and have them load split
- If have to keep order, can occasionally re-hash to move flows around
- I believe flows are inherently bursty

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…
- I don’t believe the distributed algorithm makes switches expensive

Seems to me…

- Distributed algorithm is superior, because it can react to topology changes more quickly
- But if there are very few topology changes, then perhaps less overhead with central?

How do you manage a network?

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

How do you manage a network?

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

To my astonishment

- That original vision degraded

To my astonishment

- That original vision degraded
- If we reinvent that vision with a new language for managing the switches, will the same vision degrade for the same reason?

New topic

What is Ethernet?

Why this whole layer 2/3 thing?

- Perlman’s View of ISO Layers
  - 1: physical

Why this whole layer 2/3 thing?

- Perlman's View of ISO Layers
  - 1: physical
  - 2: data link (nbr-nbr, e.g., Ethernet)

Why this whole layer 2/3 thing?

- Perlman’s View of ISO 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?

- Perlman’s View of ISO 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?

- Perlman’s View of ISO 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:

Why this whole layer 2/3 thing?

- Perlman’s View of ISO 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

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…

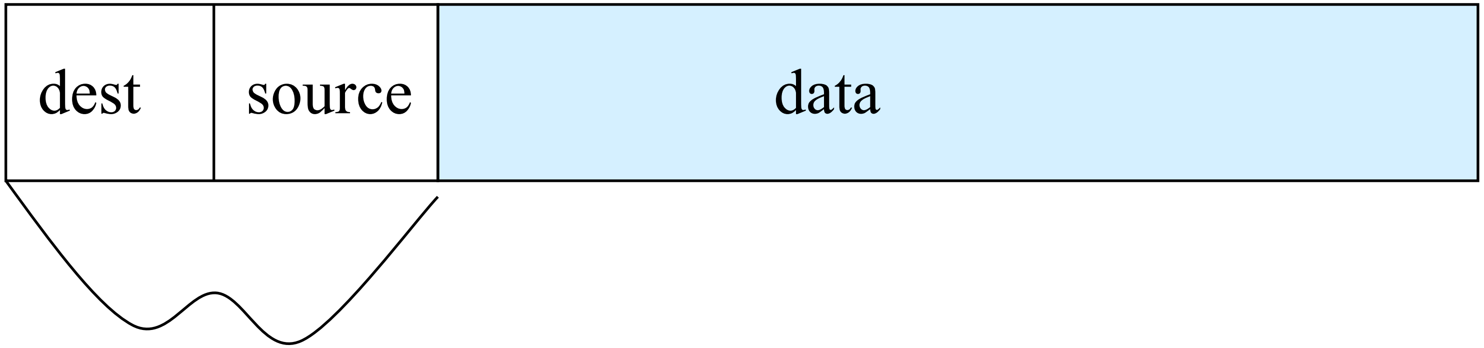
- I was layer 3 architect for DECnet
- Layer 3 calculate paths, and forwarded packets
- Layer 2 just marked beginning and end of packet, and checksum
- Then along came Ethernet

The story of Ethernet

The story of Ethernet

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

Ethernet packet

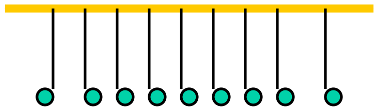


Ethernet header: 6 byte addresses – strangely large…because it allows autoconfiguration

Plus stuff like protocol type and VLAN

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)

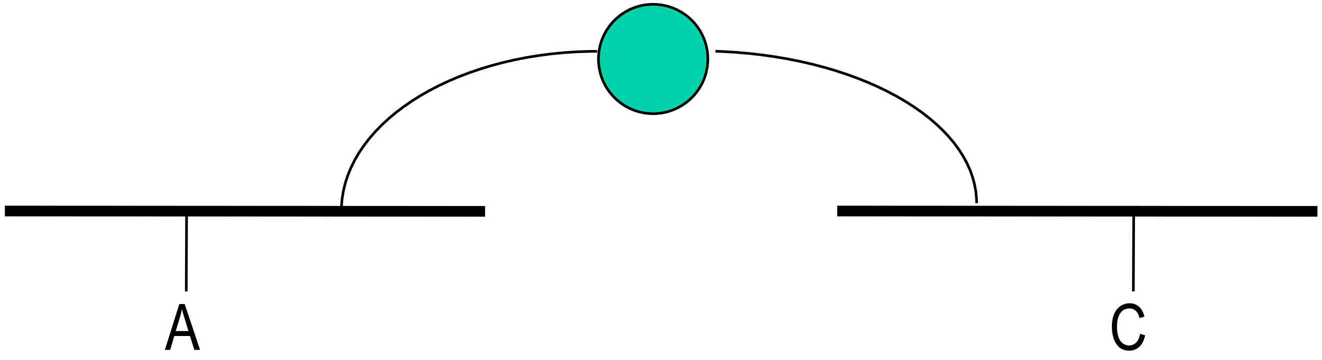
But Ethernet hasn’t been CSMA/CD for decades

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)

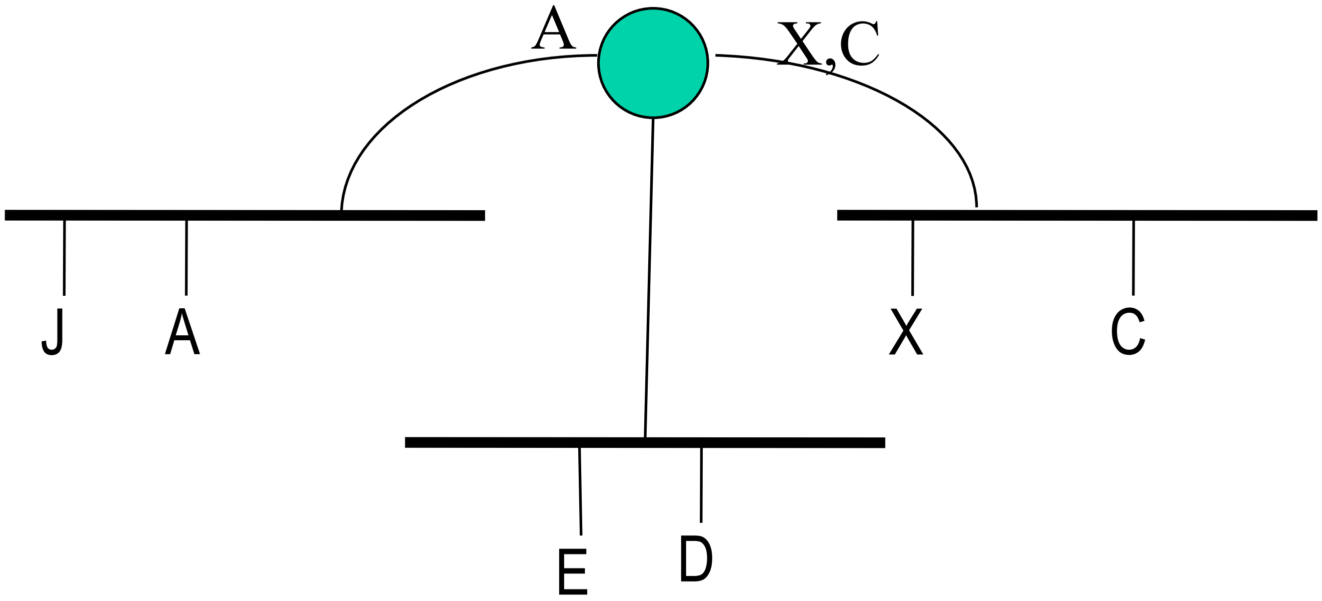
Need something that will sit between two Ethernets, and let a station on one Ethernet talk to another



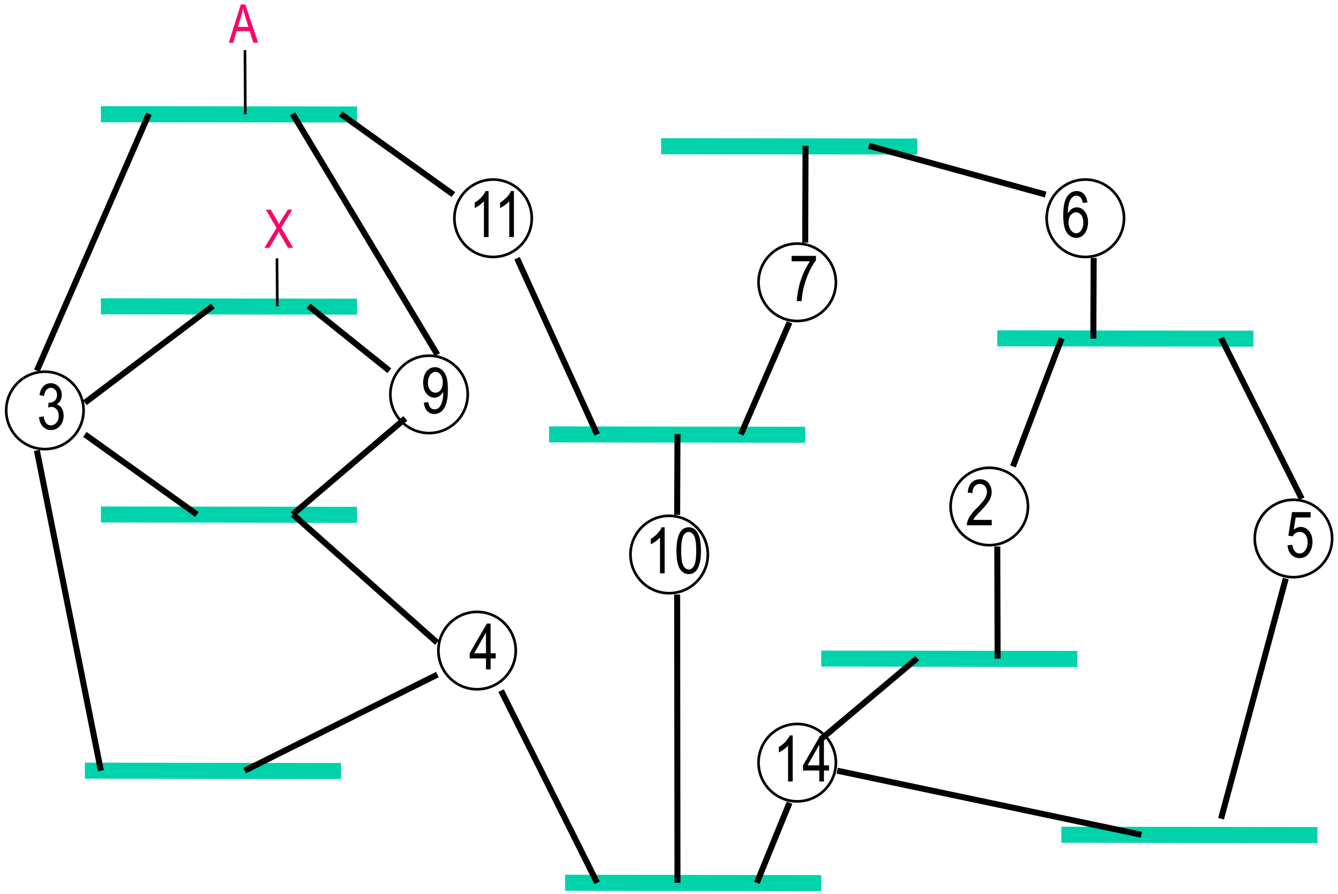
Without modifying the endnode, or Ethernet packet, in any way

The basic concept

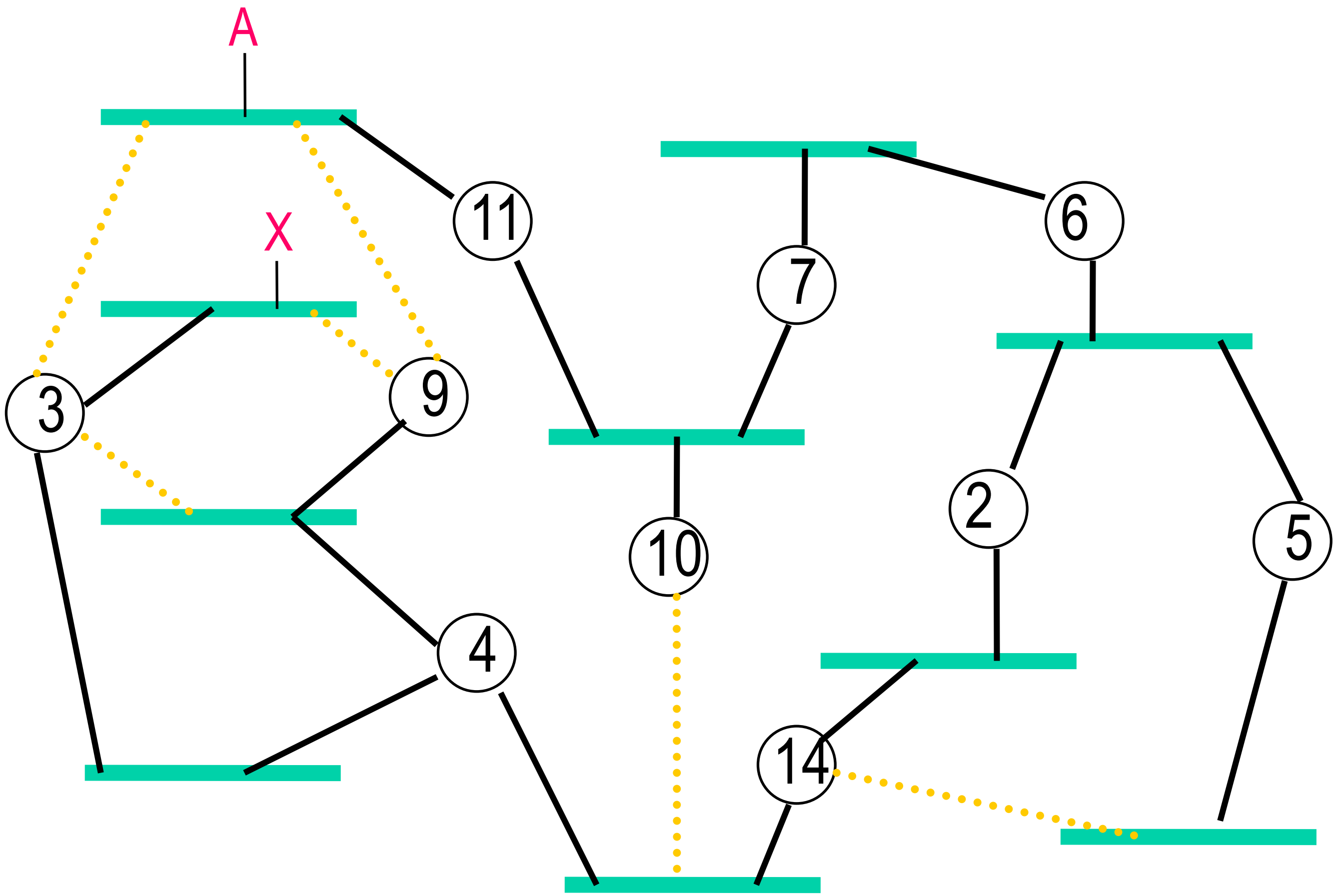
- Bridge just listens promiscuously, and forwards to each other port when the ether is free
- Learn (Source=S, input port). Once learned, if see a packet with destination=S, know where to forward it (rather than “all the ports”)
- This requires a tree (no loops) topology


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.

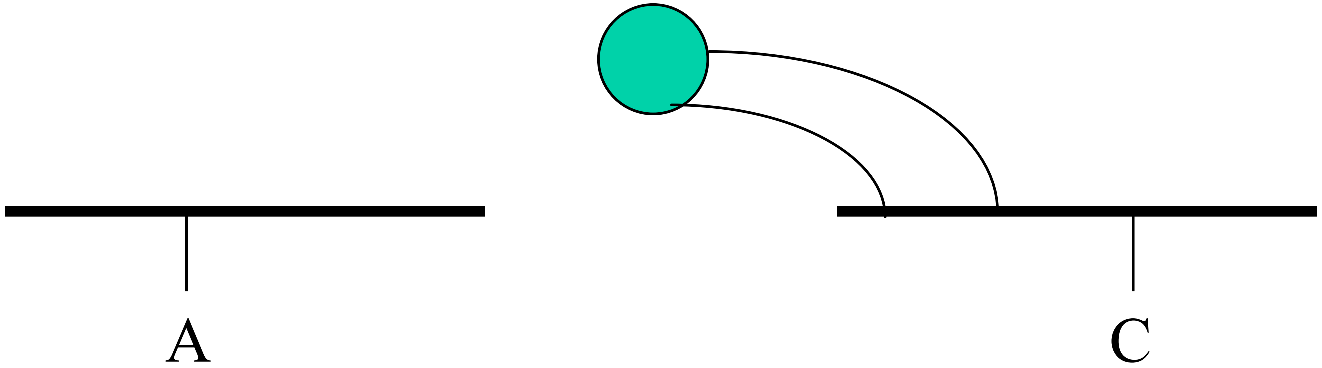
A mesh is made by folks like me.
Then bridges find a spanning tree.

Radia Perlman

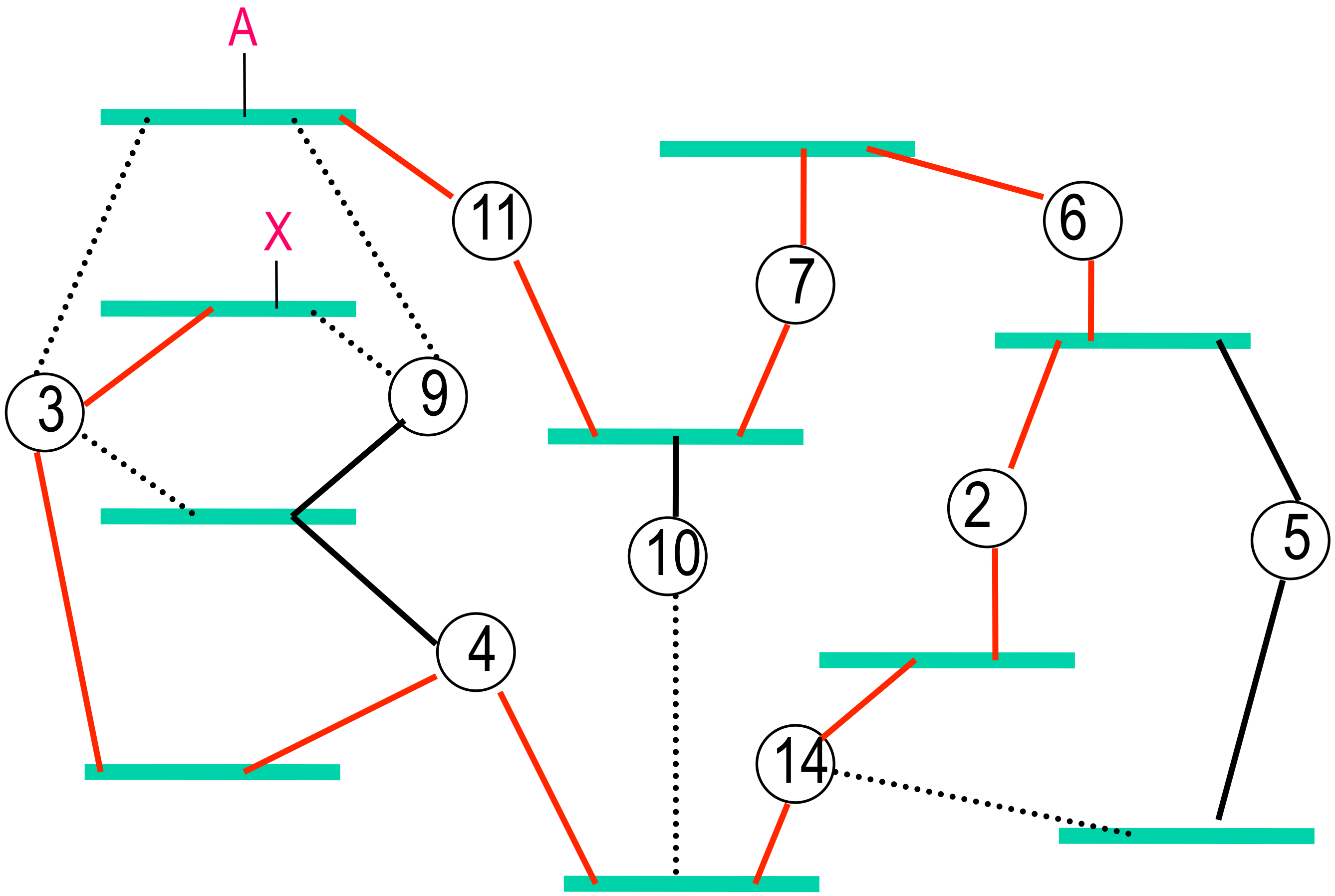
Bother with spanning tree?

- 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

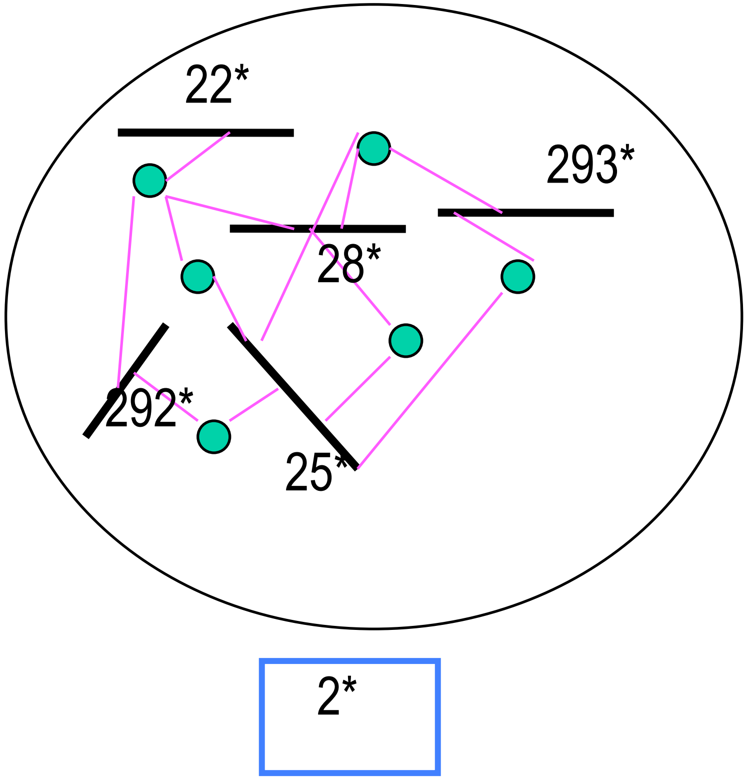
Layer 3 doesn’t have to work that way!

- 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

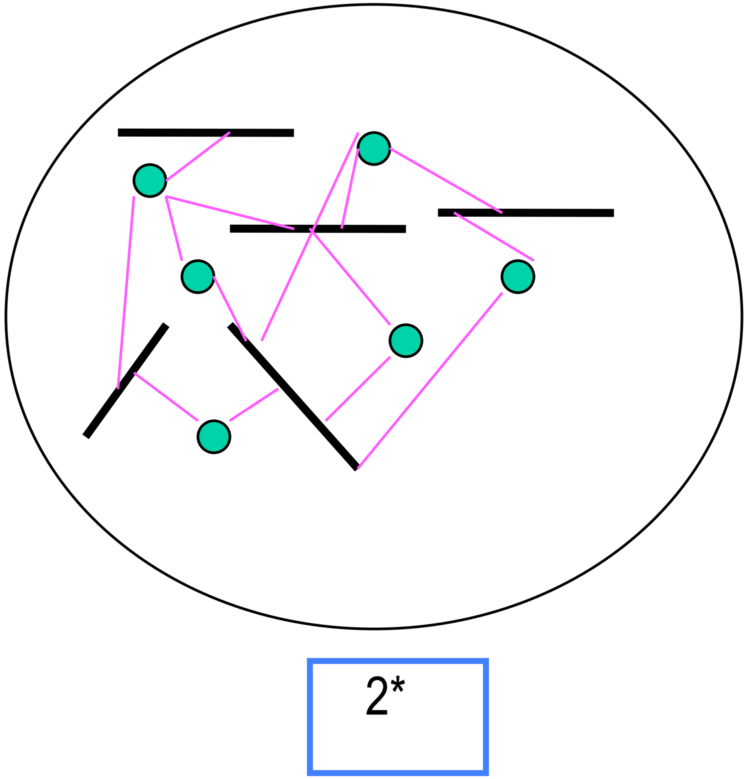
14 bytes	6 bytes
Prefix shared by all nodes in large cloud	Endnode ID

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

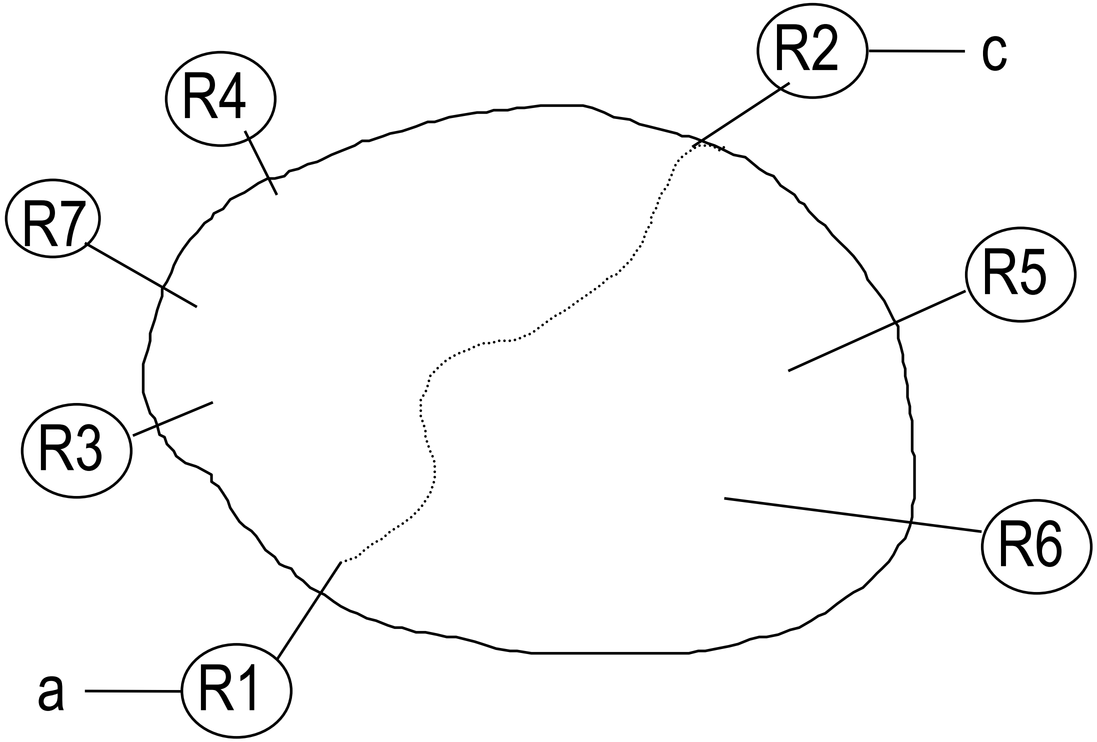
- So Ethernet provides a large cloud in which switches can autoconfigure, and nodes (e.g., VMs) can move around transparently
- But don’t want limitations of spanning tree

Next step in evolution: TRILL

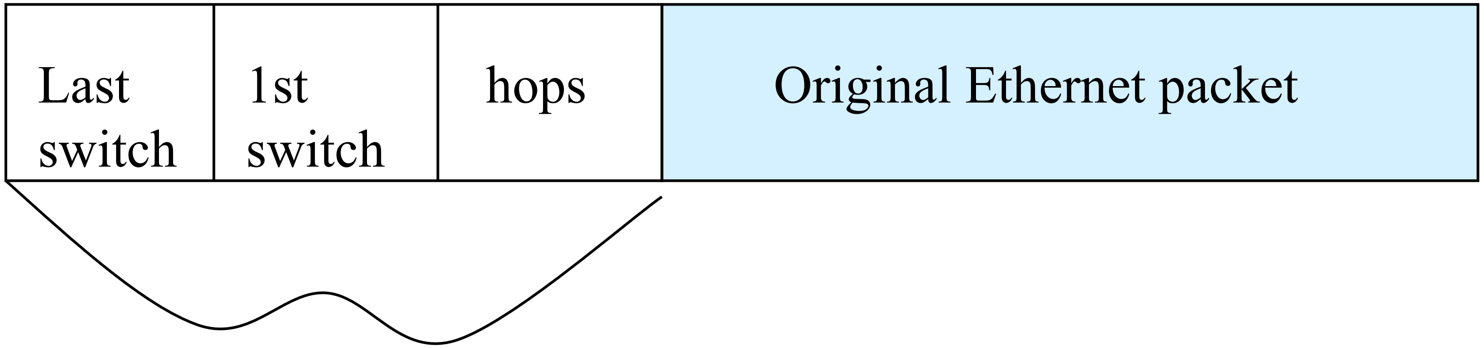
TRILL

- 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



Last switch

1st switch

hops

Original Ethernet 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

What does encapsulation header address?

- Last switch?
  - Smaller forwarding tables
  - Last switch has to look at inner header to know where to forward
- Output port of last switch?
  - Can avoid making forwarding tables bigger if there is a fixed hierarchy:
    - Last switch | Port on last switch

Interesting (to me, anyway) note

- CLNP vs IP+TRILL
  - With CLNP, no need for ARP to get address on final link…it's part of the header
  - With these encapsulation things, forwarding table inside final cloud can be smaller…with CLNP, routers have to keep track of all endnodes inside the cloud

Some heresy

- Fabrics should be allowed to reorder packets…make smarter endnodes, including work of middle boxes
- Congestion by telling source too slow
- Cost of making fabric “lossless” is too high
  - Congestion spreads if
    - You never drop packets
    - You backpressure, based on a few classes

Protocol Folklore

- Obvious stuff everyone gets wrong

What’s a Version Number?

- Version number
  - what is “new version” vs “new protocol”?
    - same lower layer multiplex info
  - therefore, must always be in same place!
  - drop if version # bigger

Version

- Nobody seems to do this right
- IP, IKEv1, SSL unspecified what to do if version # different. Most implementations ignore version number field
- SSL v3 moved version field!

Parameters

- Minimize these:
  - someone has to document it
  - customer has to read documentation and understand it
- How to avoid
  - architectural constants if possible
  - automatically configure if possible

Settable Parameters

- Make sure they can’t be set incompatibly across nodes, across layers, etc. (e.g., hello time and dead timer)
- Make sure they can be set at nodes one at a time and the net can stay running

Example: Hello Timer

- IS-IS
  - pairwise parameters reported in “hellos”
  - So you know what to expect from that neighbor
- OSPF
  - Kind of copied IS-IS, but decided…

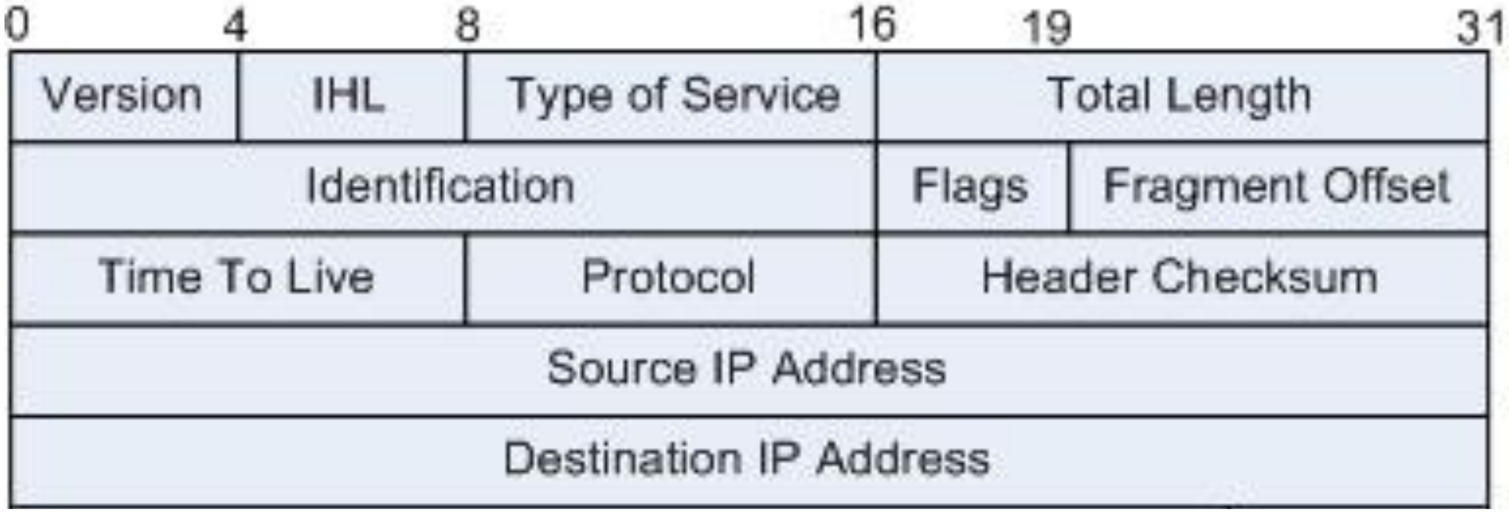
Example: Hello Timer

- IS-IS
  - pairwise parameters reported in “hellos”
  - So you know what to expect from that neighbor
- OSPF
  - Kind of copied IS-IS, but decided…
  - Refuse to talk if timers not identical with neighbor’s!

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?

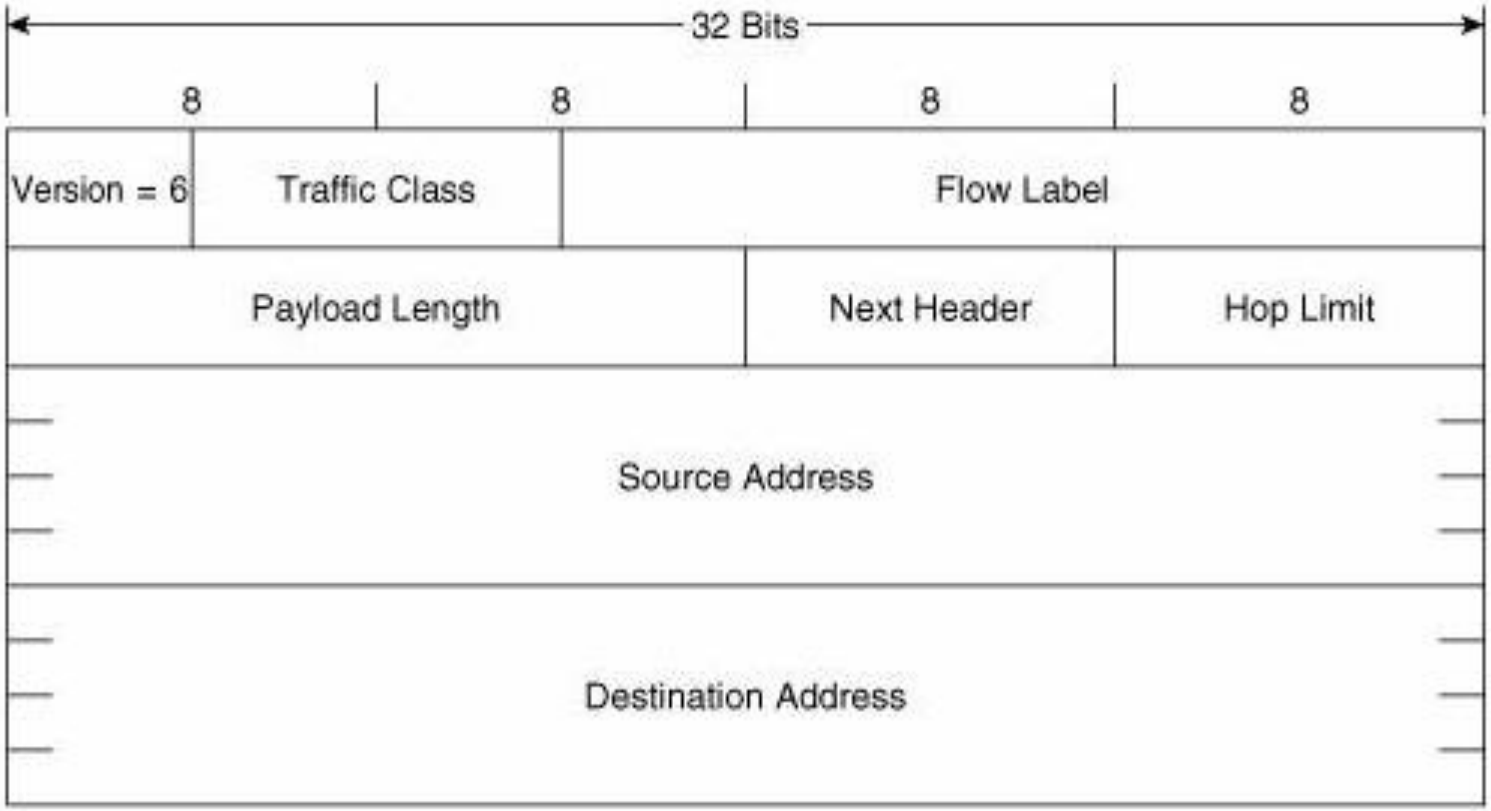
IPv4 header



0 4 8 16 19 31

<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td colspan="2">Time To Live</td><td>Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
</table>

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