Beyond Buzzwords: Intrinsic Network Concepts

Radia Perlman

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

This talk

- Hopefully interactive
  - Ask questions
  - (within reason) argue
- I have way too much material (I think)…so I intend to skip some of it (unless I really underestimated the time it will take)

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

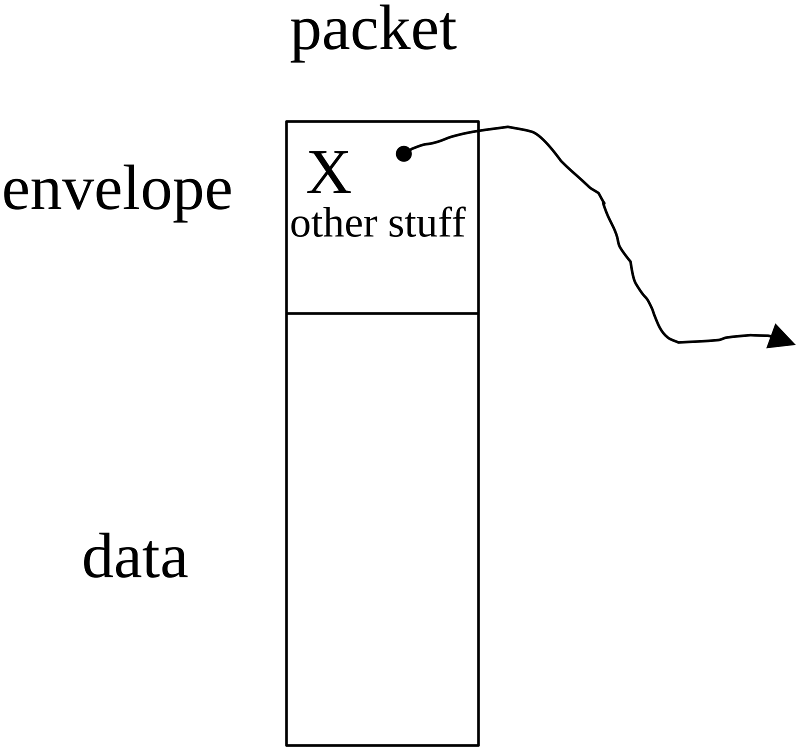
Forwarding Table

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



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

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
- …

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 (not location dependent)
  - Direct lookup (if address small)
  - 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

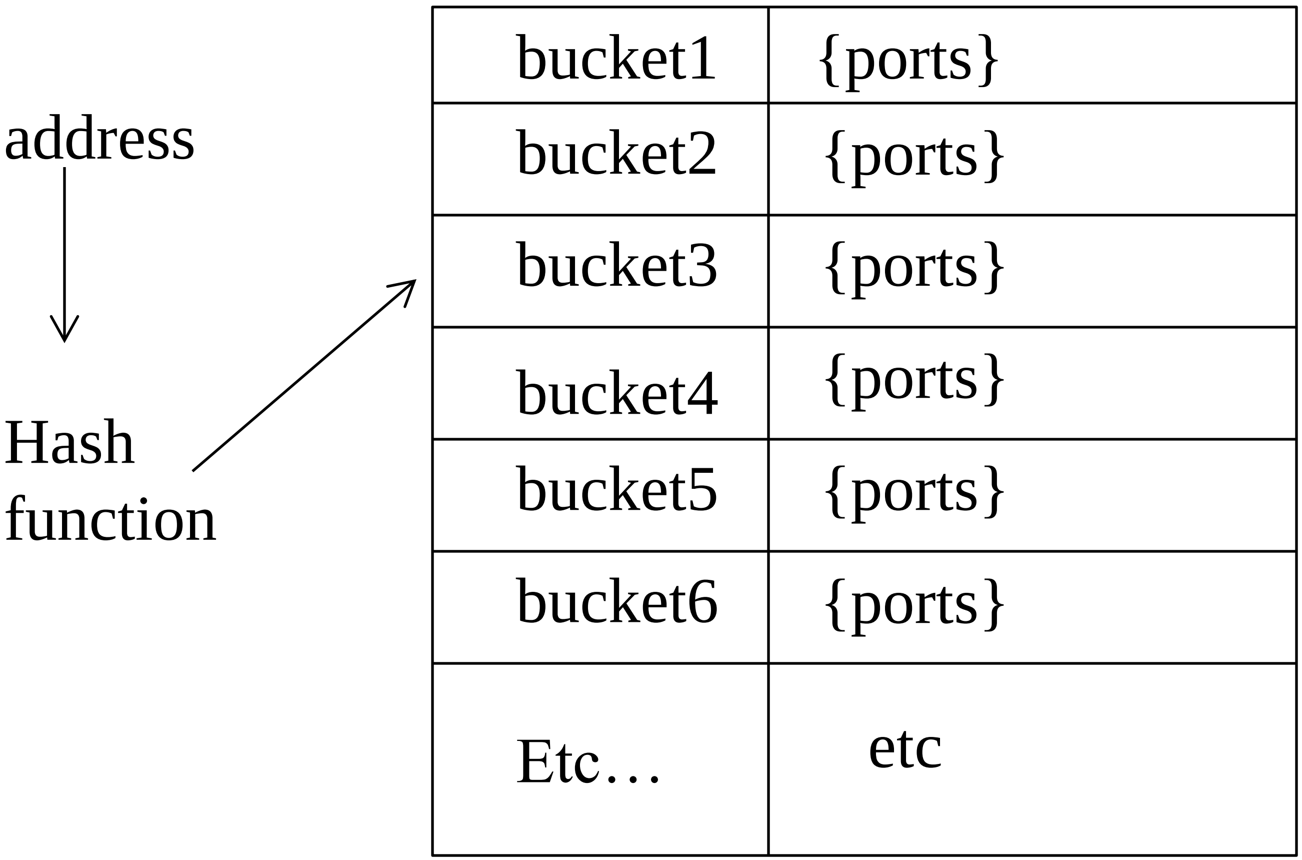
Forwarding Table

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



address

Hash function

bucket1	{ports}
bucket2	{ports}
bucket3	{ports}
bucket4	{ports}
bucket5	{ports}
bucket6	{ports}
Etc…	etc

Ethernet addresses

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

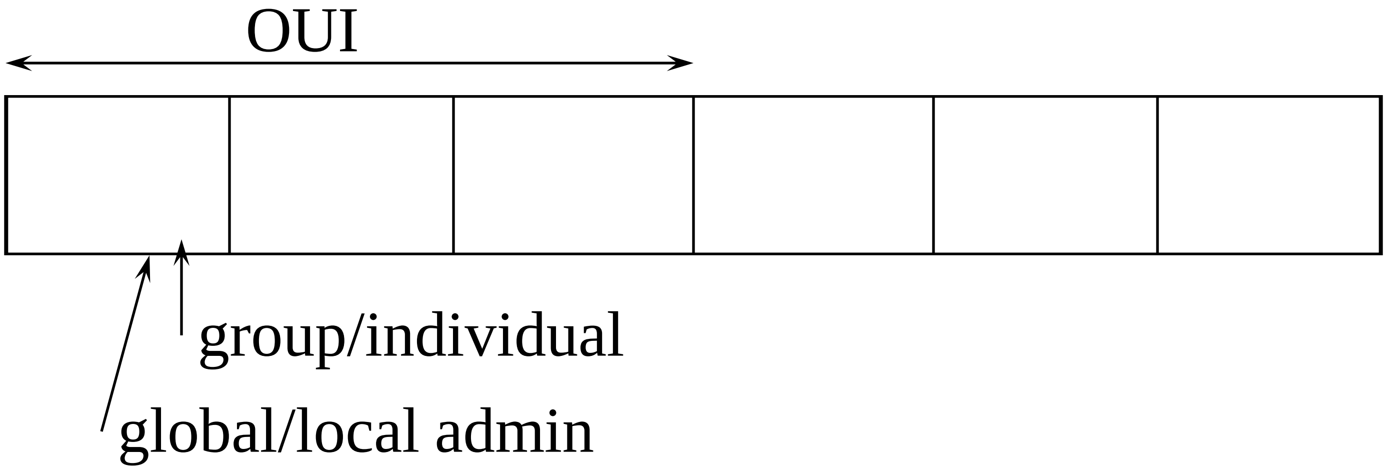
Ethernet addresses

- 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

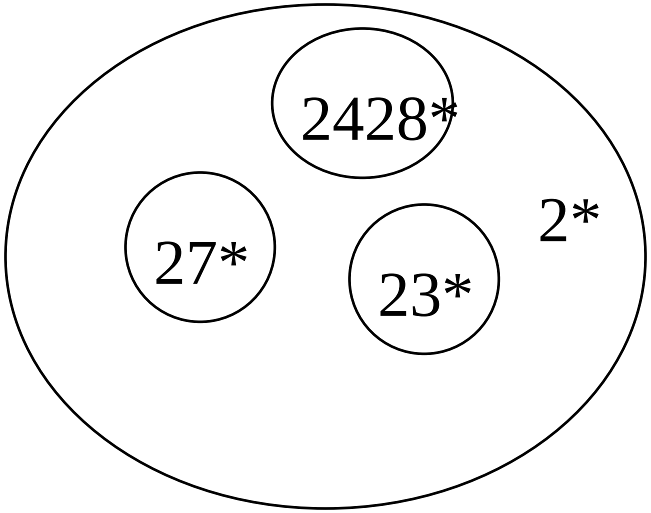
- One early proposal was to assign 48-bit MAC addresses at random…no IEEE address blocks
- 48-bits is probably big enough for this to be practical, and low enough probability of address collision

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

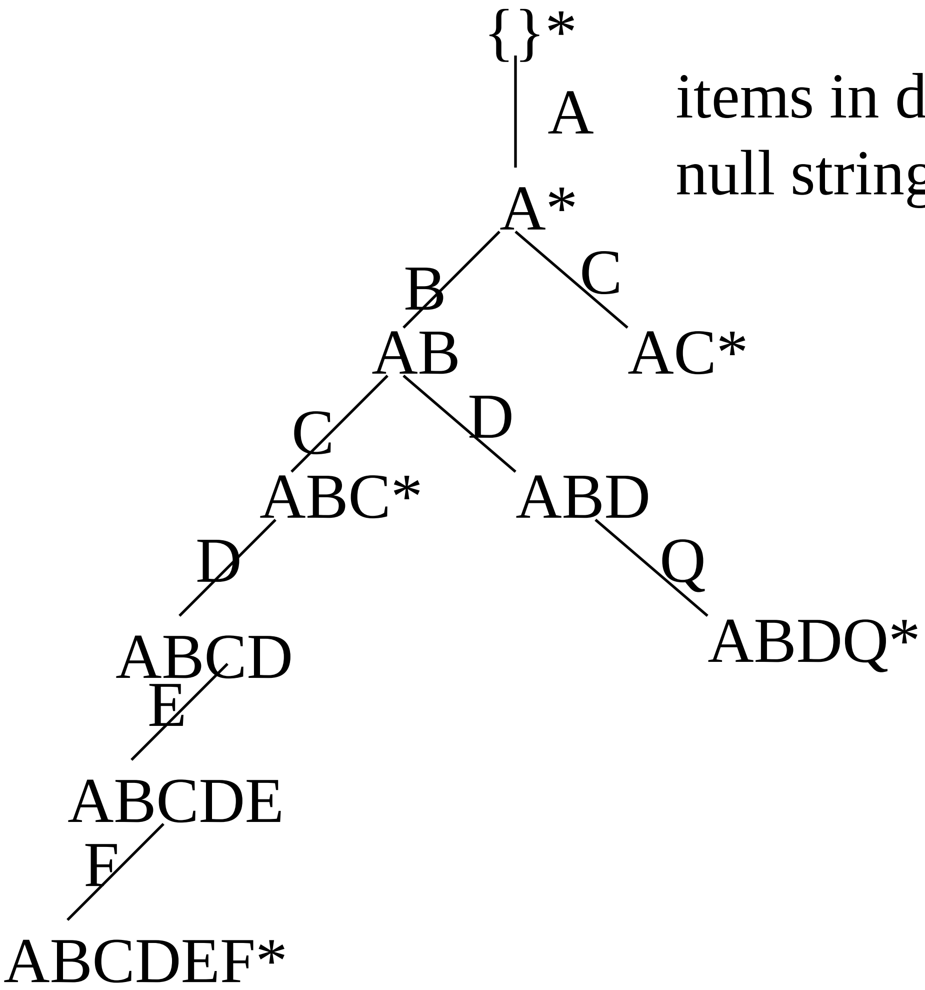
Address Prefix Routing

- 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:

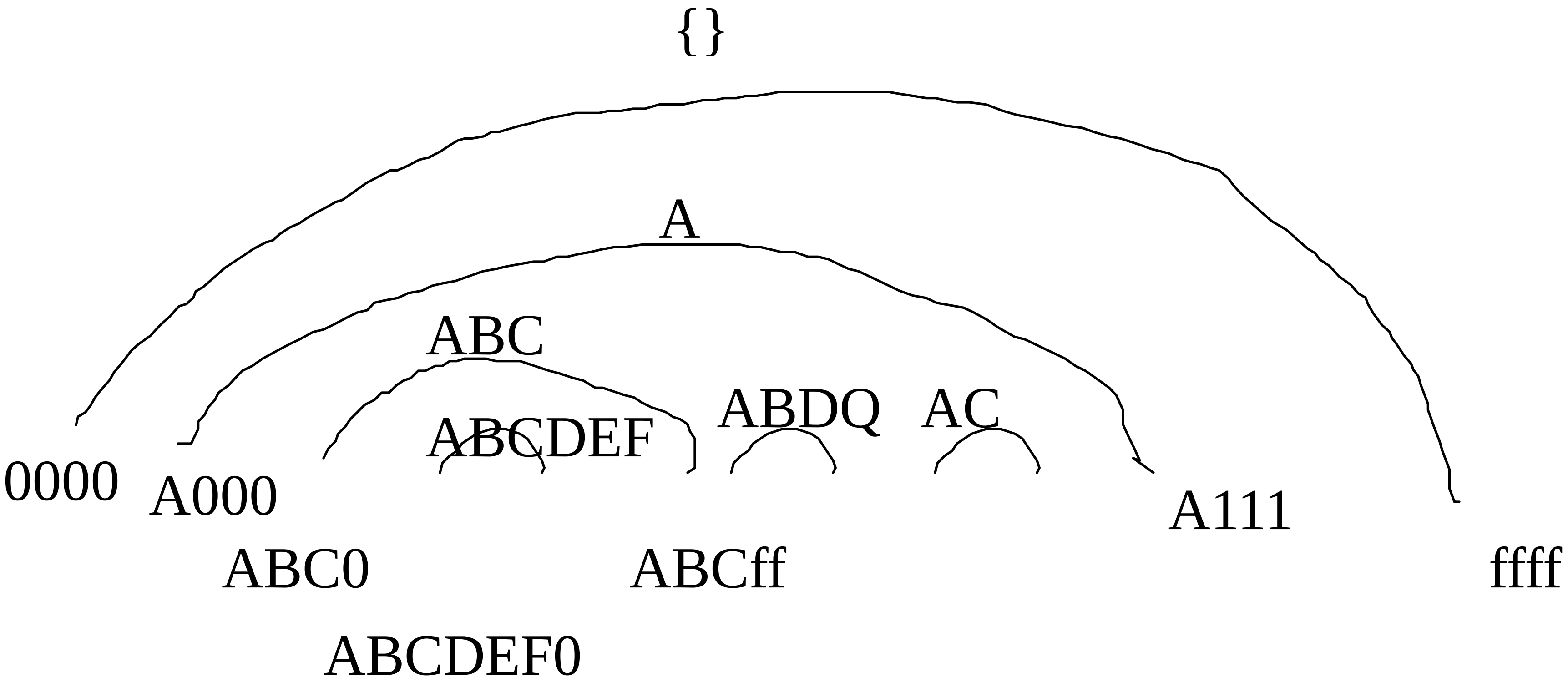
null string, A, ABC, ABCDEF, ABDQ,

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



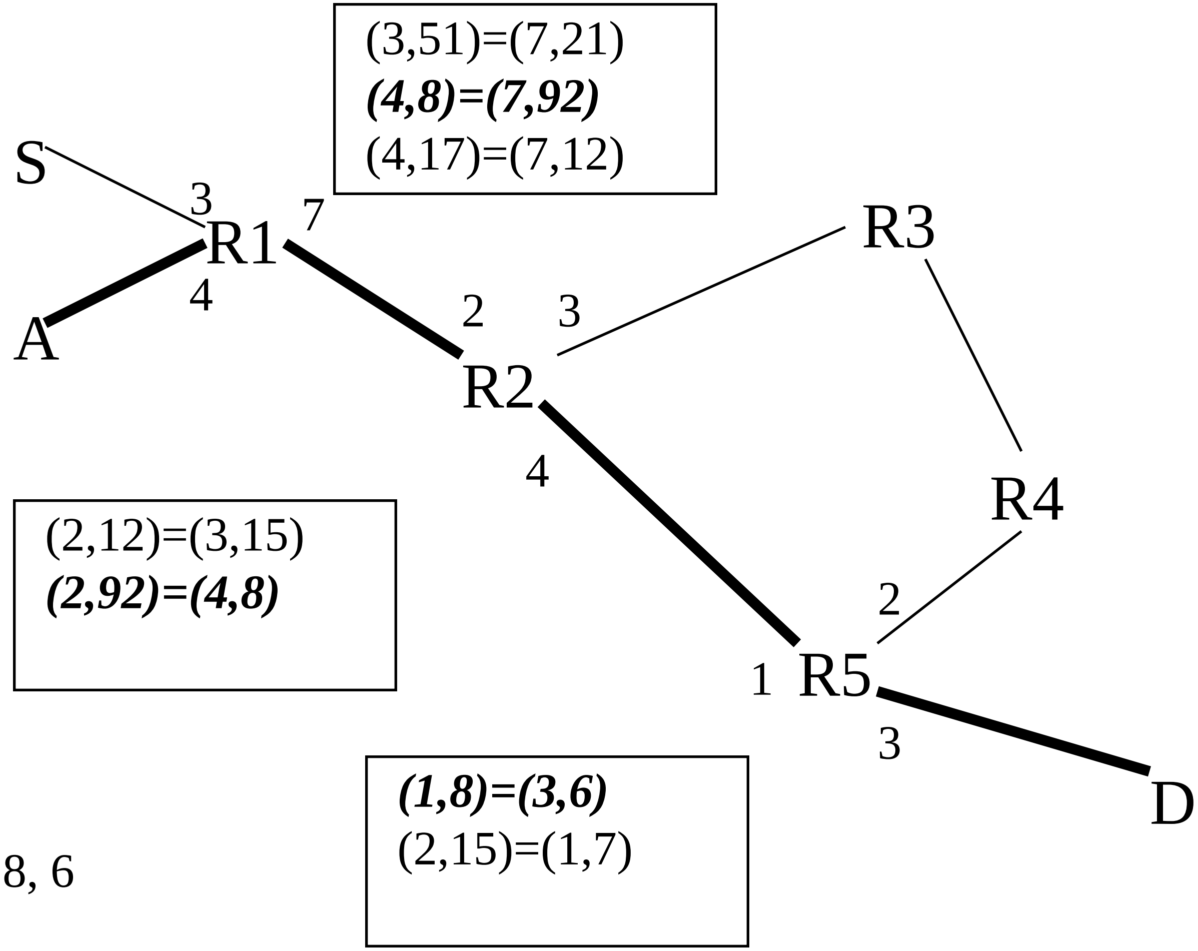
Summary of address-based alternatives

- 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



(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

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

Controversial thought

- I think anything MPLS is doing (and what is that?) could be done better with a simple header of “source/destination”

Some thoughts

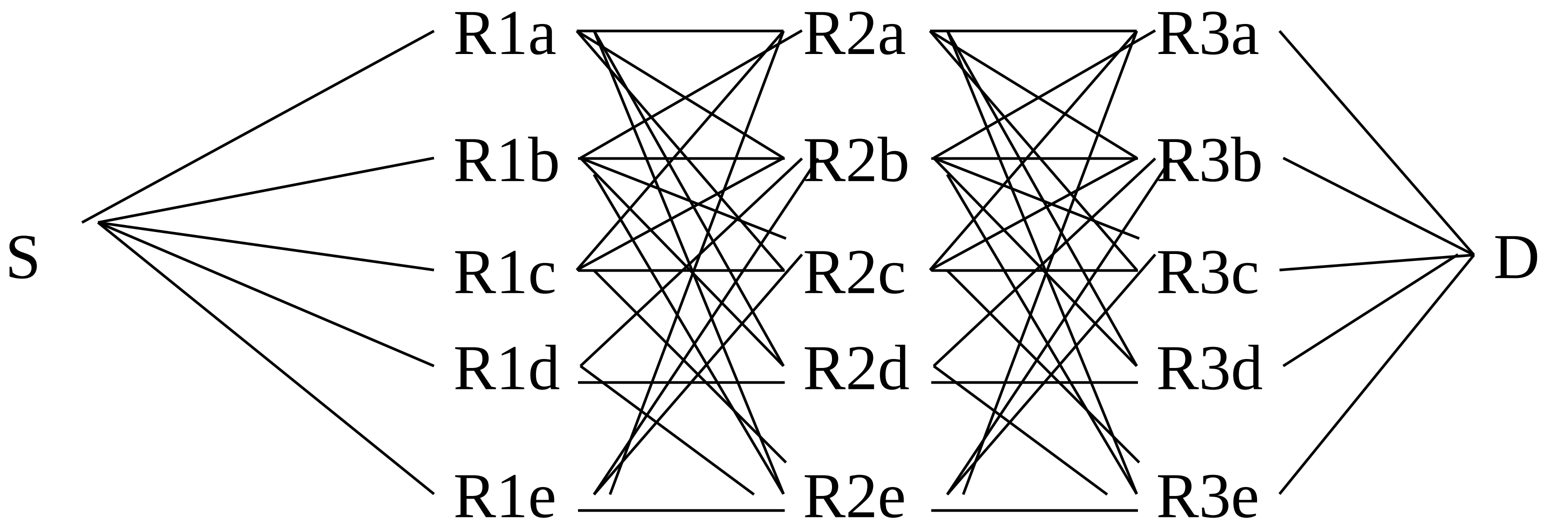
- Dest-based vs label-based
  - Destination-based is smaller (O(n)) forwarding table than label-based (O(n^2))
  - Unless path set up when flow starts…but that incurs latency
  - Destination-based doesn't preclude traffic engineering
    - For instance, Infiniband "destination" is a path to the destination, 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

- (And you have to keep packets for a flow in order)
- 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 of which address to use for the destination 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)

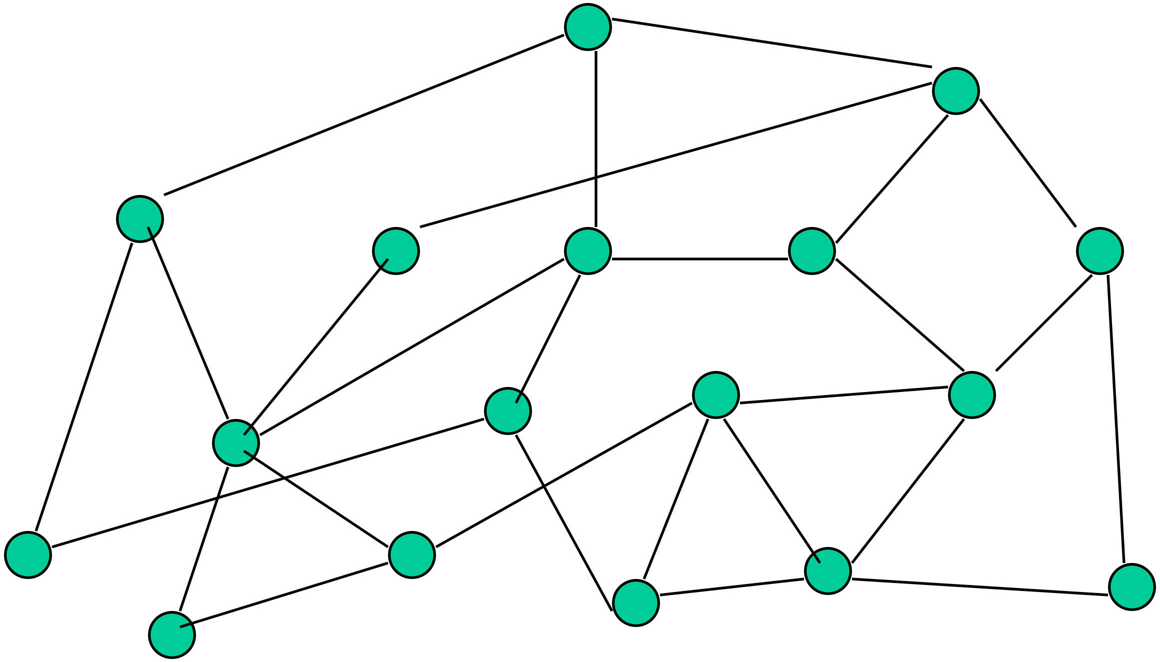
Completely orthogonal concept

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

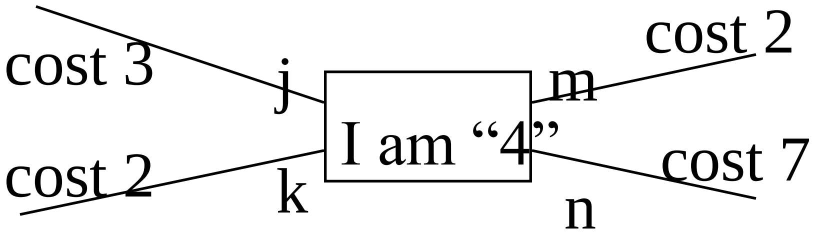


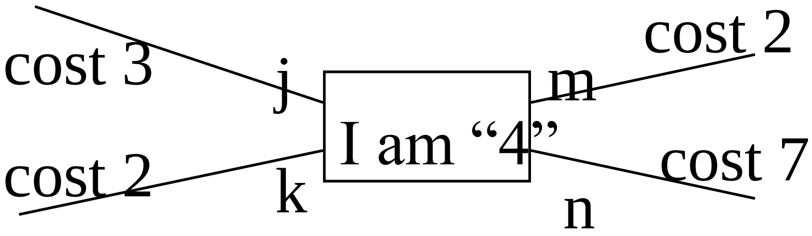
Distributed Routing Protocols

- 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

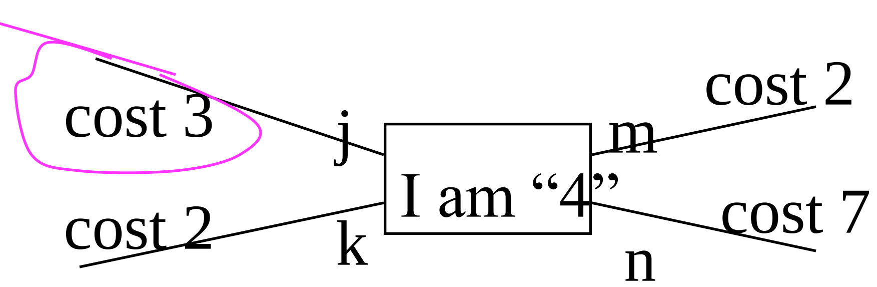
cost 7	6	2	0	7	8	5	8	12	11	3	2

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

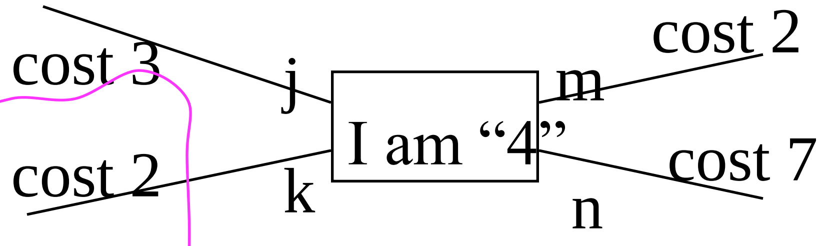
cost 7	6	2	0	7	8	5	8	12	11	3	2

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

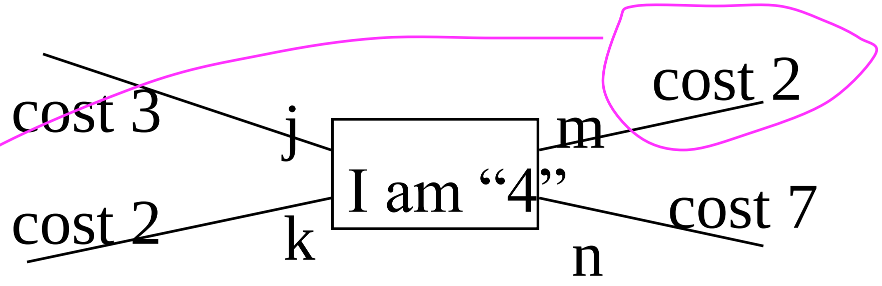
cost 7	6	2	0	7	8	5	8	12	11	3	2

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

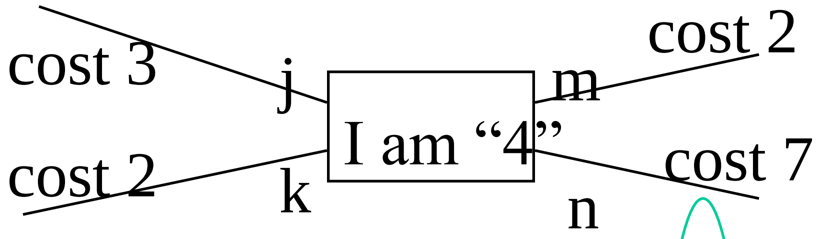
cost 7	6	2	0	7	8	5	8	12	11	3	2

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

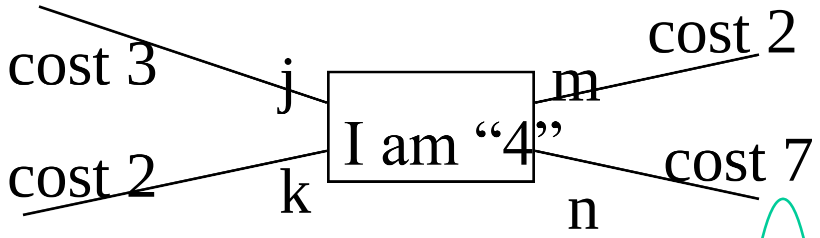
6	2	0	7	8	5	8	12	11	3	2

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

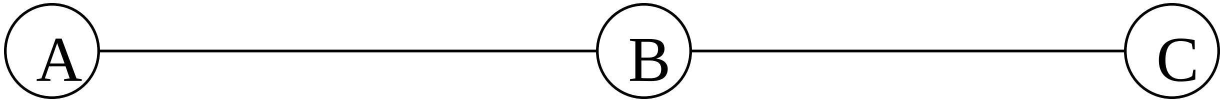
your own calculated distance vector

2	6	5	0	12	8	6	19	3	?	?

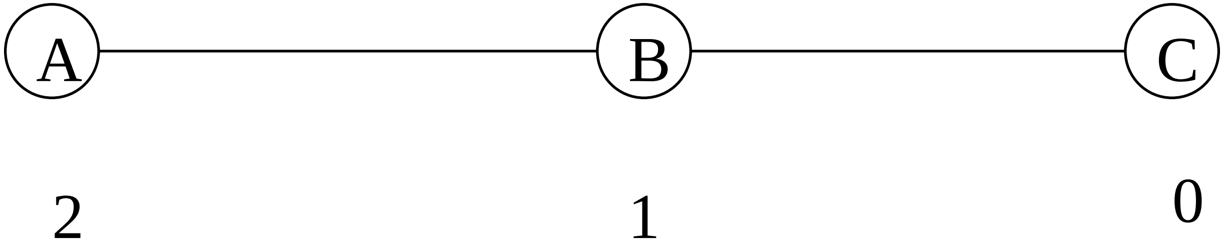
your own calculated forwarding table

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

Looping Problem

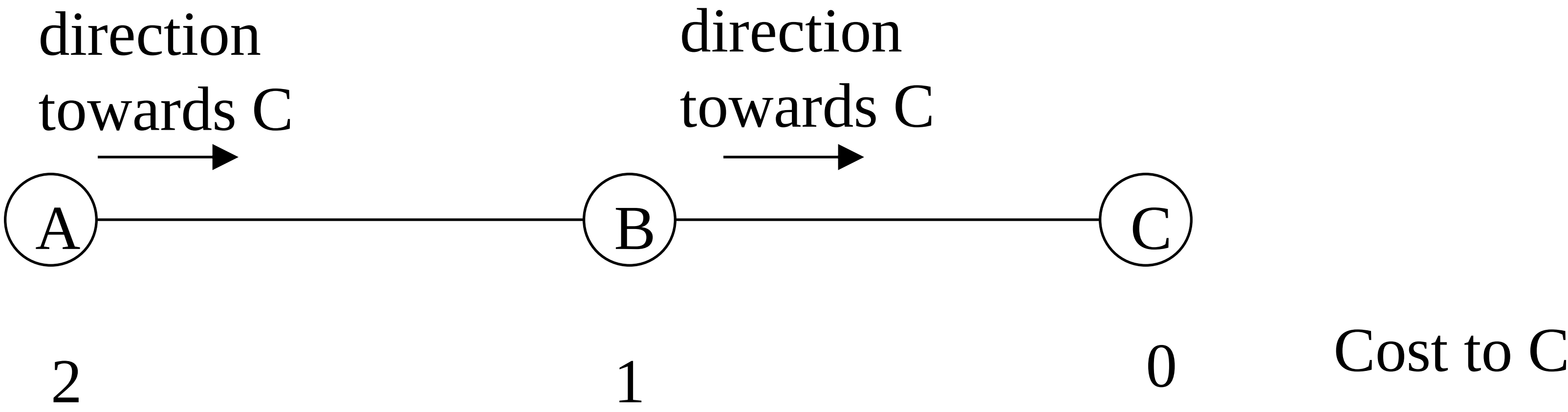


Looping Problem

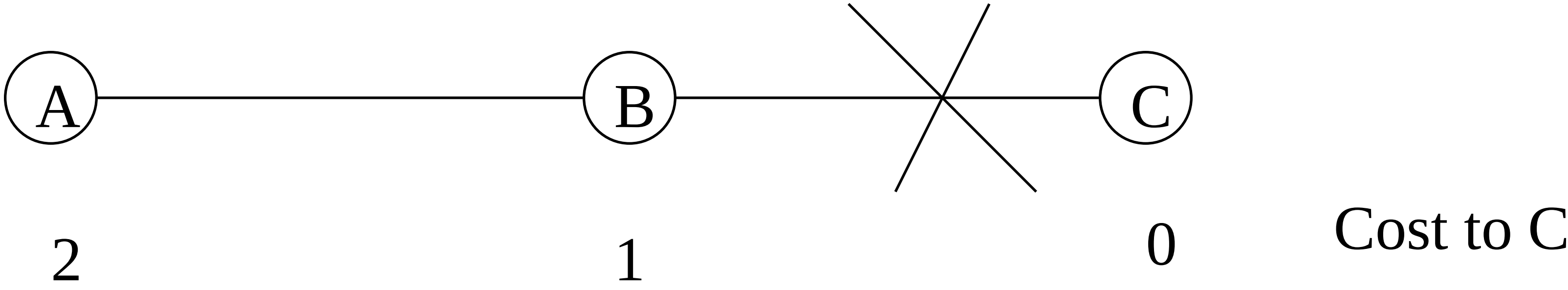


Cost to C

Looping Problem

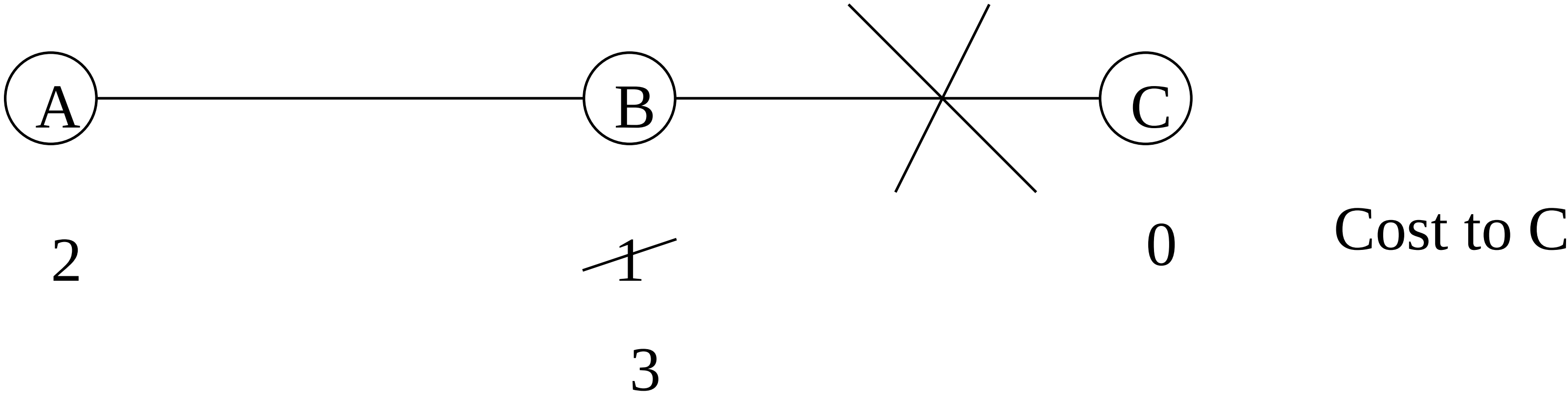


Looping Problem

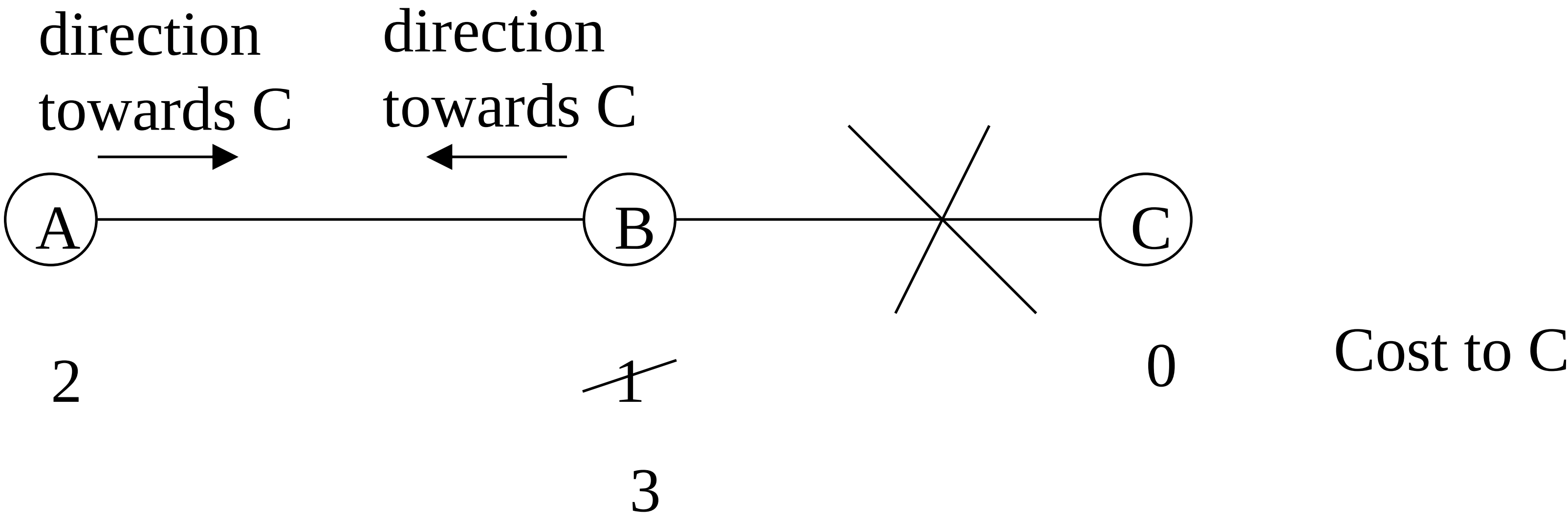


What is B’s cost to C now?

Looping Problem



Looping Problem



direction
towards C

direction
towards C

A

B

C

2

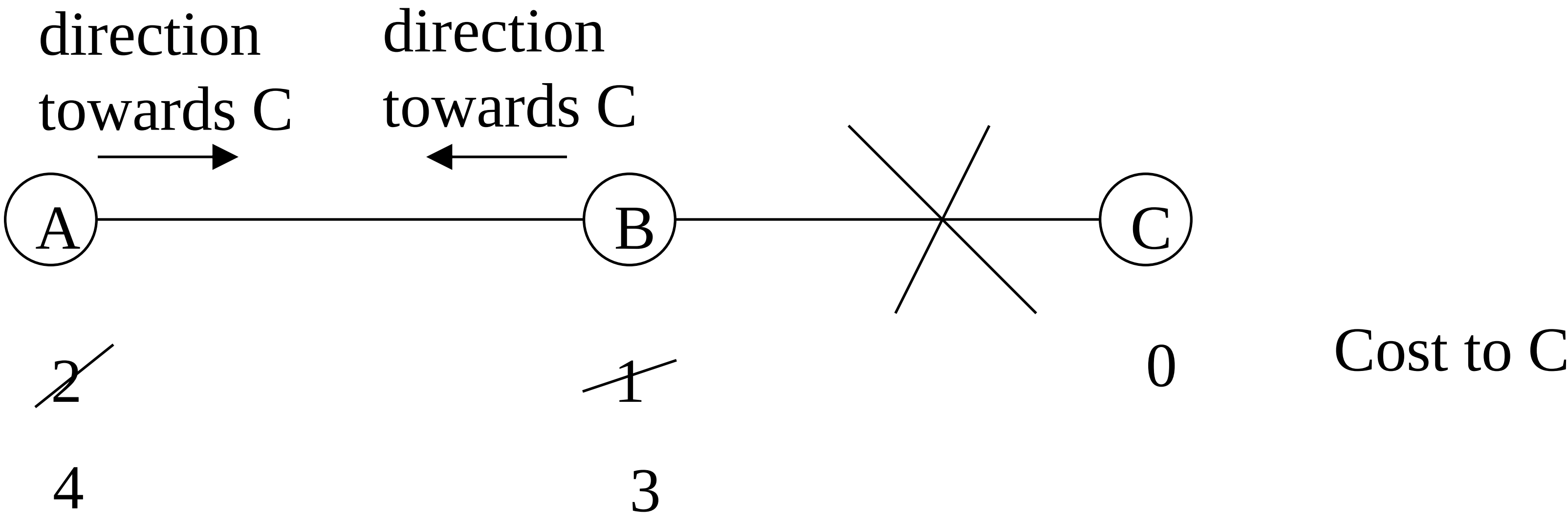
~~1~~

3

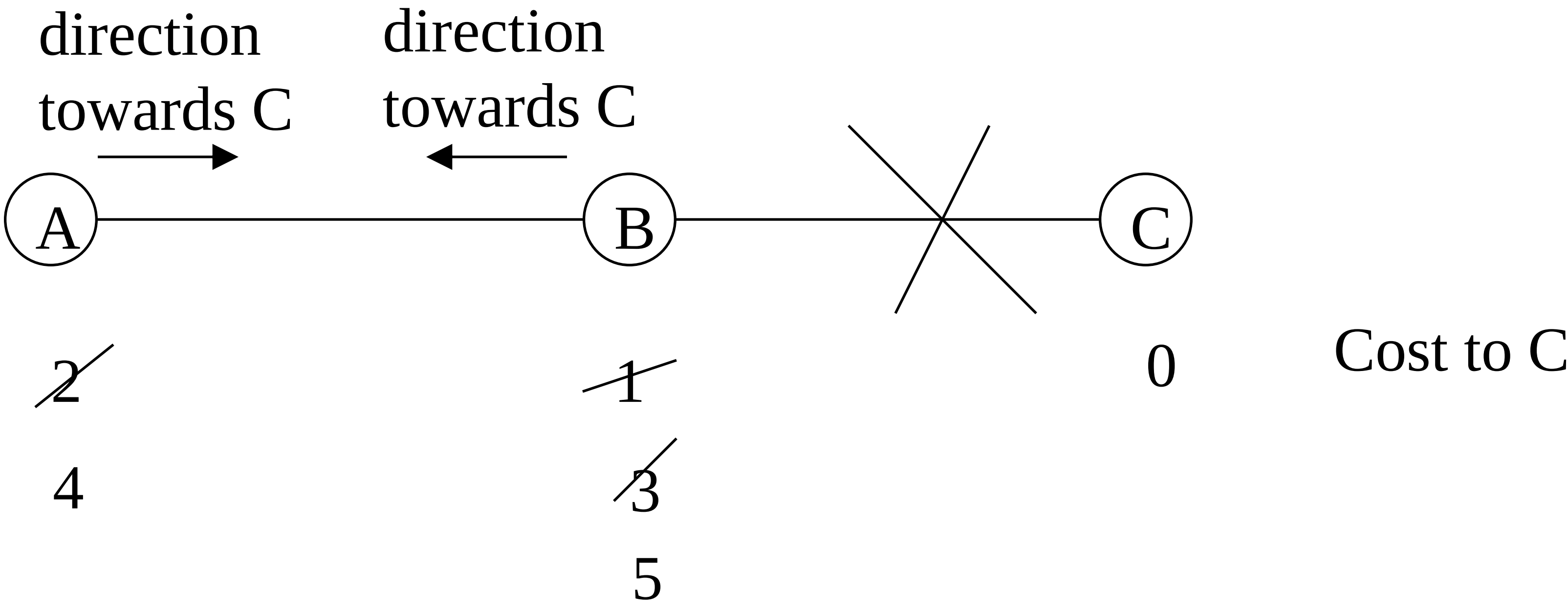
0

Cost to C

Looping Problem



Looping Problem



direction towards C

direction towards C

~~2~~

4

~~1~~

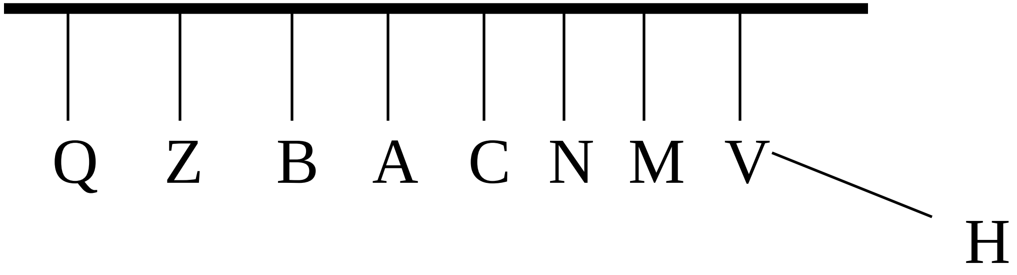
~~3~~

5

0

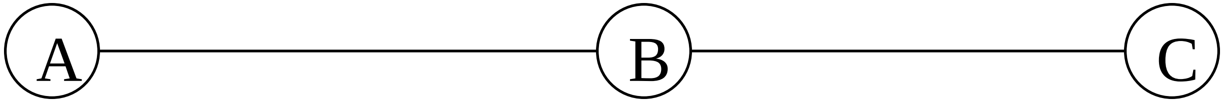
Cost to C

Looping Problem worse with high connectivity

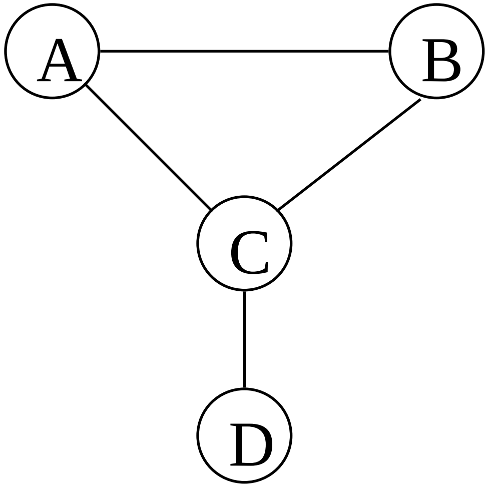


Split Horizon: one of several optimizations

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

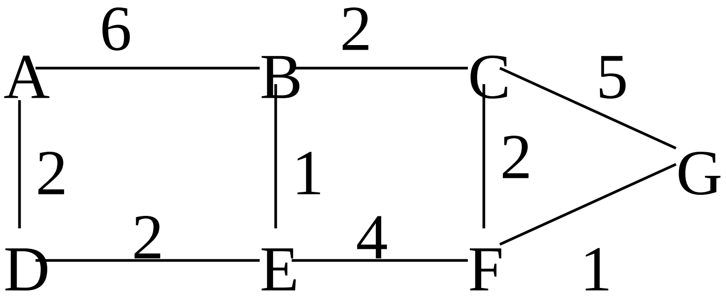


Split Horizon: …but it won’t work with loops of more than 2 nodes



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)



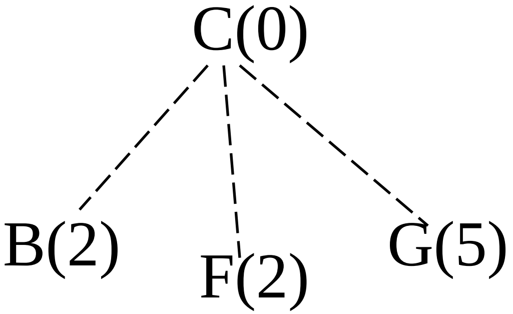
A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	

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

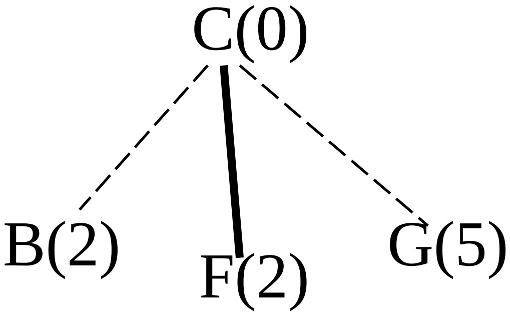
Look at LSP of new tree node

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	



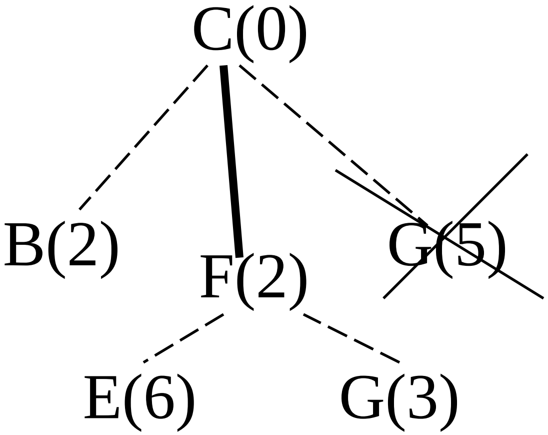
Make shortest TENT solid

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	



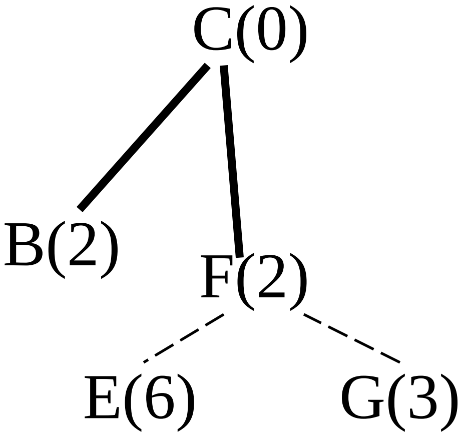
Look at LSP of newest tree node

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	



C(0)

B(2)

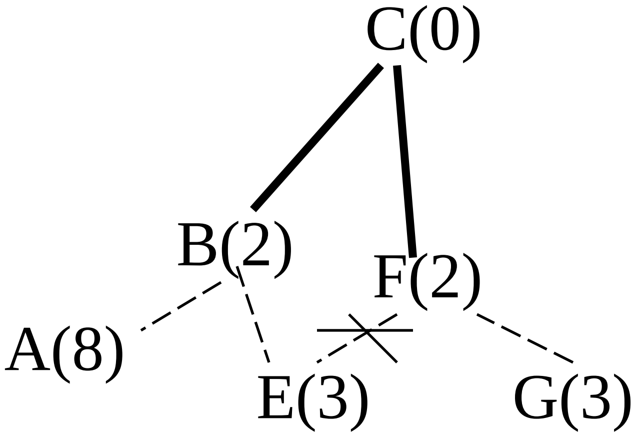
F(2)

E(6)

G(3)

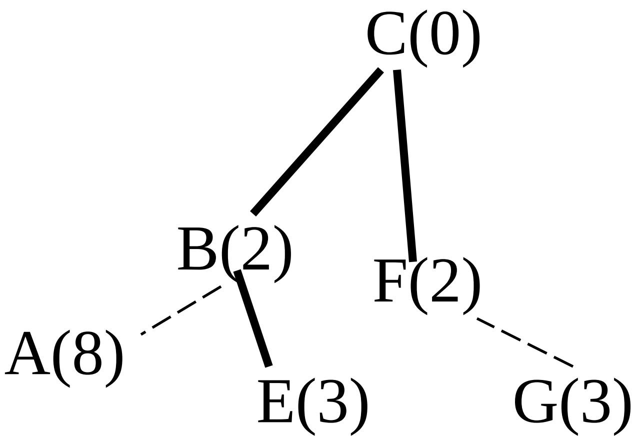
Look at LSP of newest tree node

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	



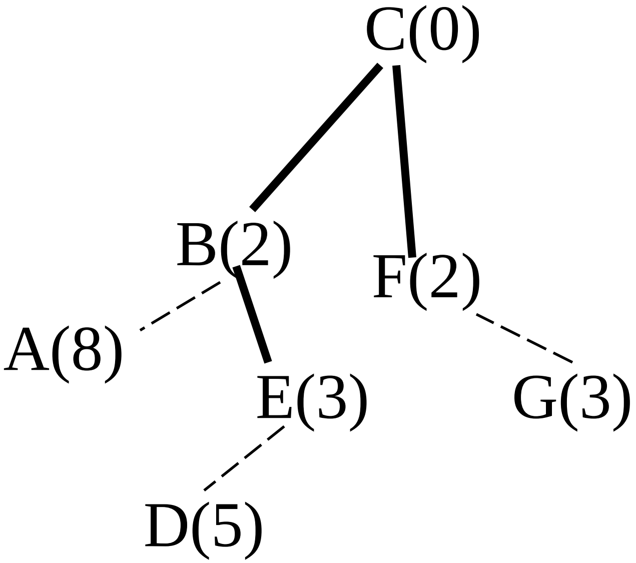
Make shortest TENT solid

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	



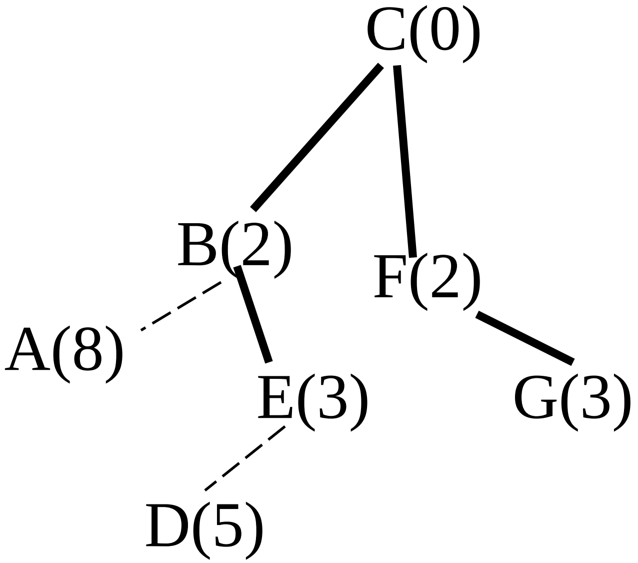
Look at LSP of newest tree node

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	



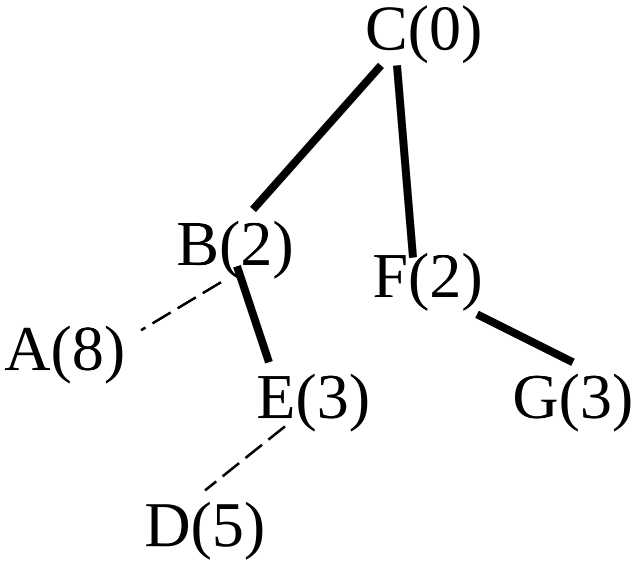
Make shortest TENT solid

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	



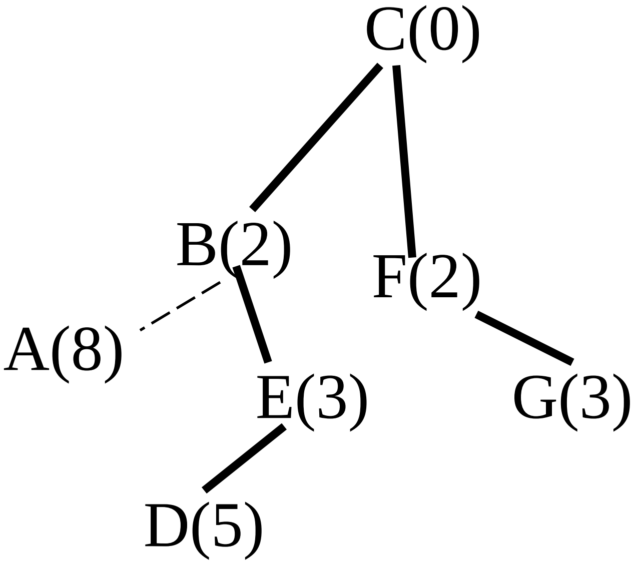
Look at newest tree node’s LSP

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	



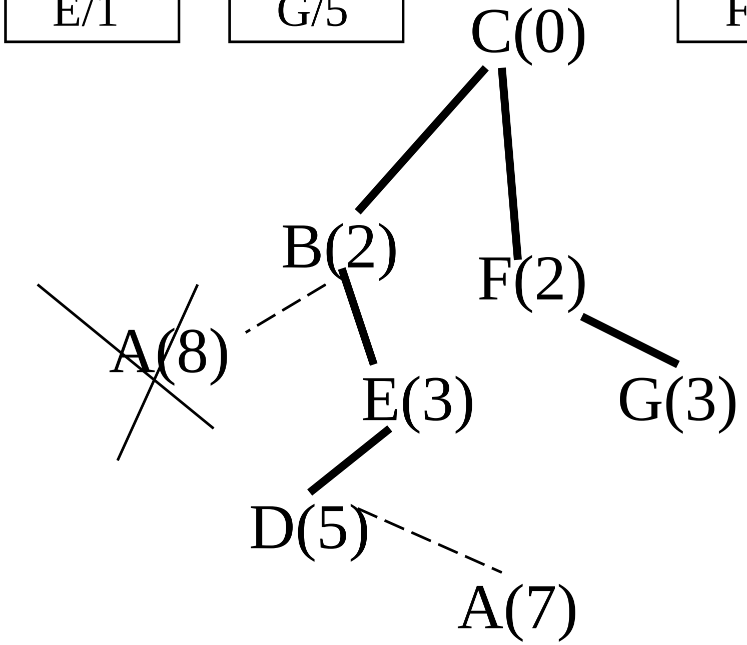
Make shortest TENT solid

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	



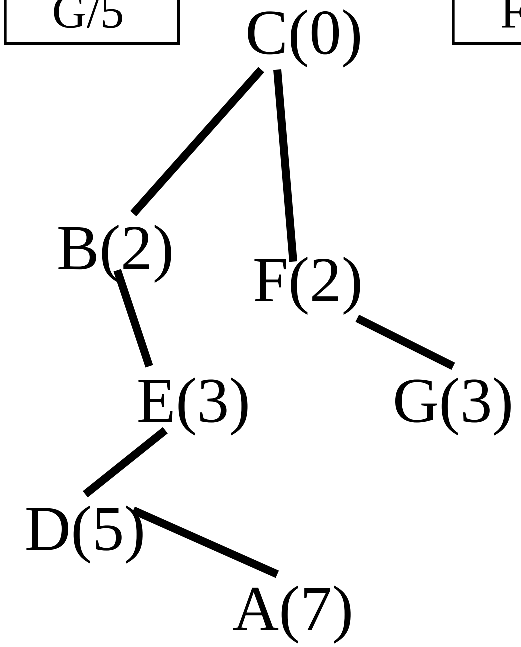
Look at newest node’s LSP

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	



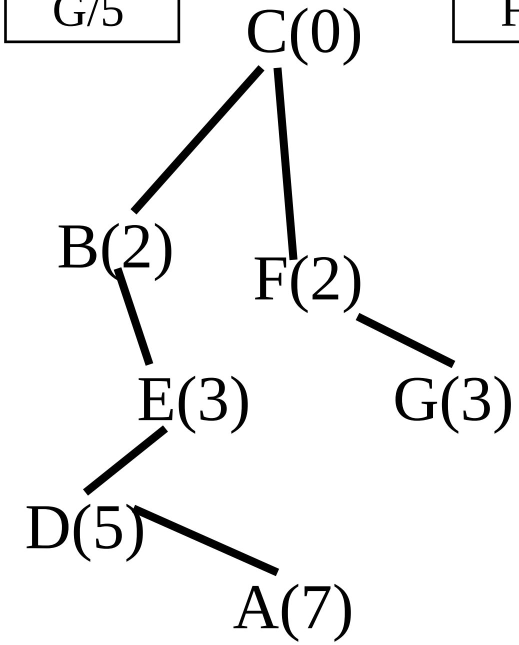
Make shortest TENT solid

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	

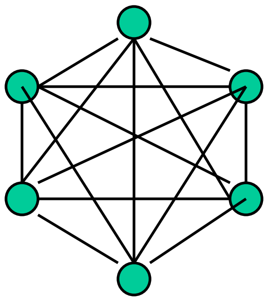


Another enhancement of link state

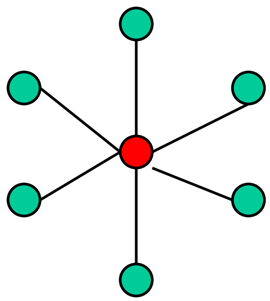
- Pseudonodes
- Since routing algorithm is proportional to the number of links
- If an Ethernet with 100s of nodes were considered fully connected, the link state database would be too large

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

So back to distributed algorithm vs central fabric manager

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?

How the world should be…

- Layer 2 is supposed to be between neighbors – not forwarded “at layer 2”
- Layer 3 is supposed to be forwarded
- And layer 3 should be allowed to get things out of order
- And layer 4 should number things, and put them back in order

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?

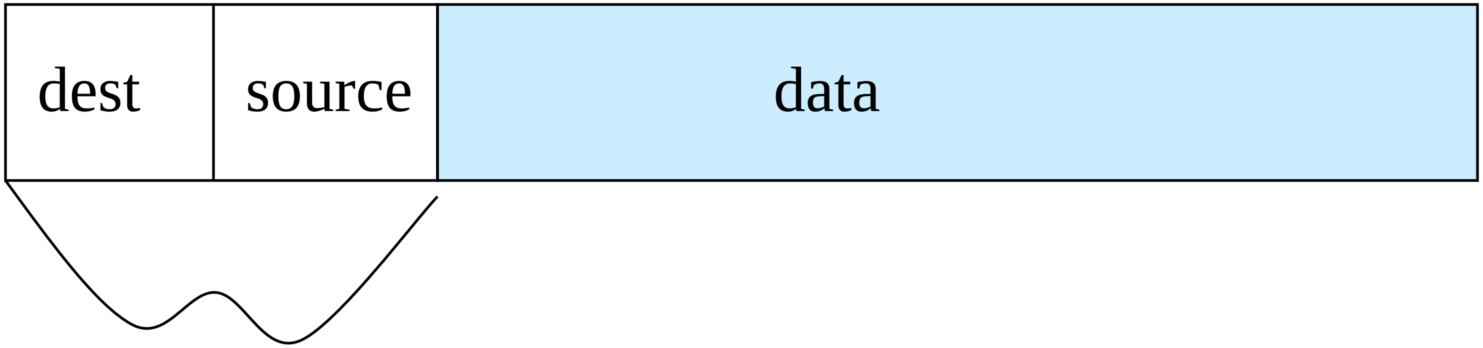
- Ethernet was intended to be layer 2
- Just between neighbors – not forwarded
- What exactly is Ethernet?

Let’s start in the early 1980’s

- 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

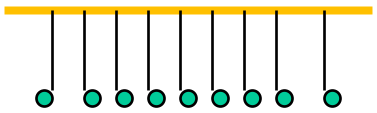
Plus stuff like protocol type and VLAN

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)

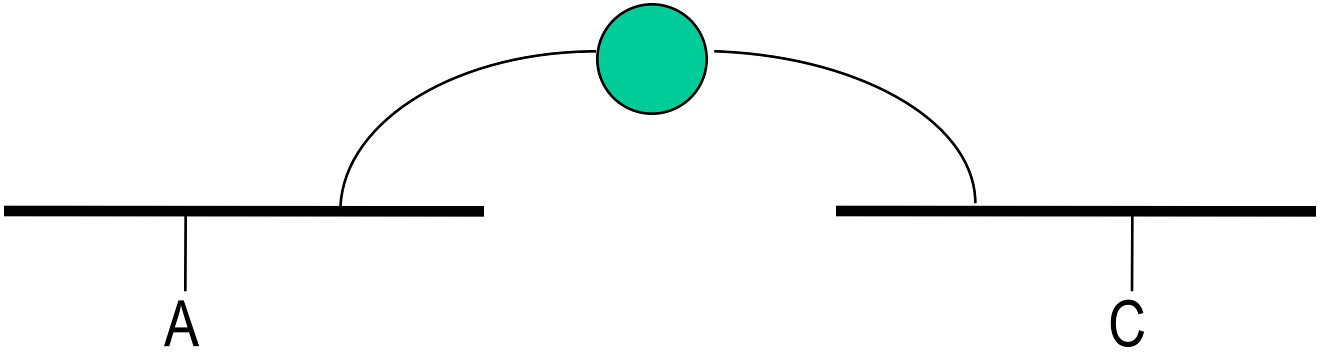
But Ethernet hasn’t been CSMA/CD for decades

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)

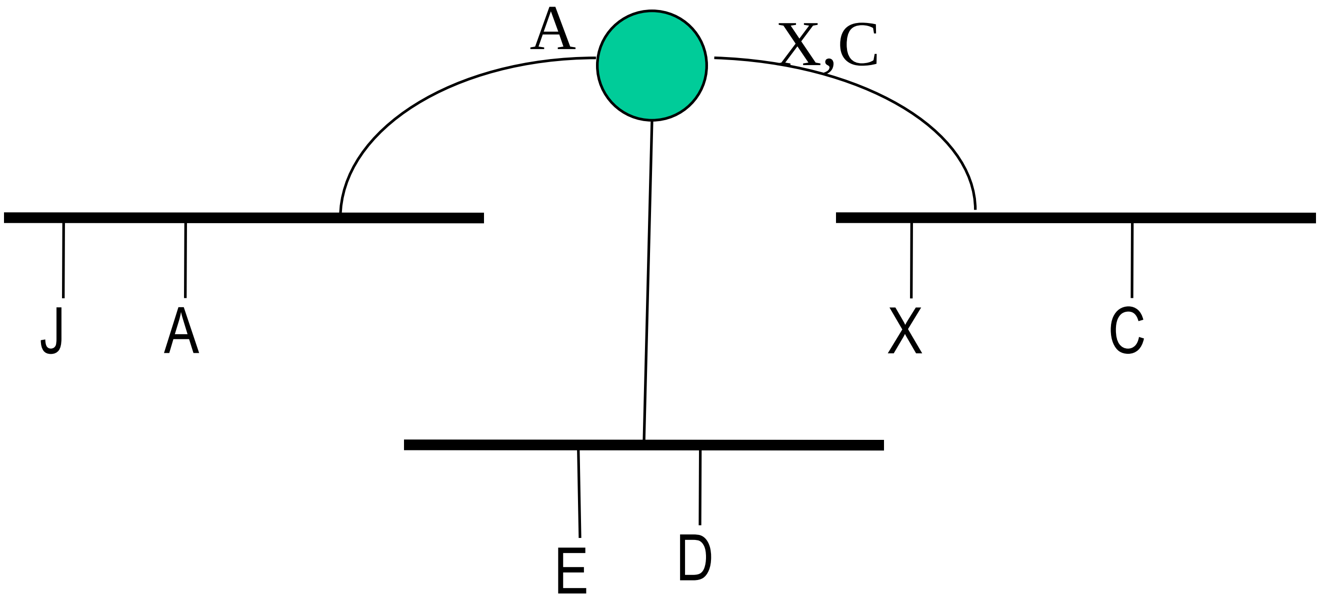
Need something that will sit between two Ethernets, and let a station on one Ethernet talk to another



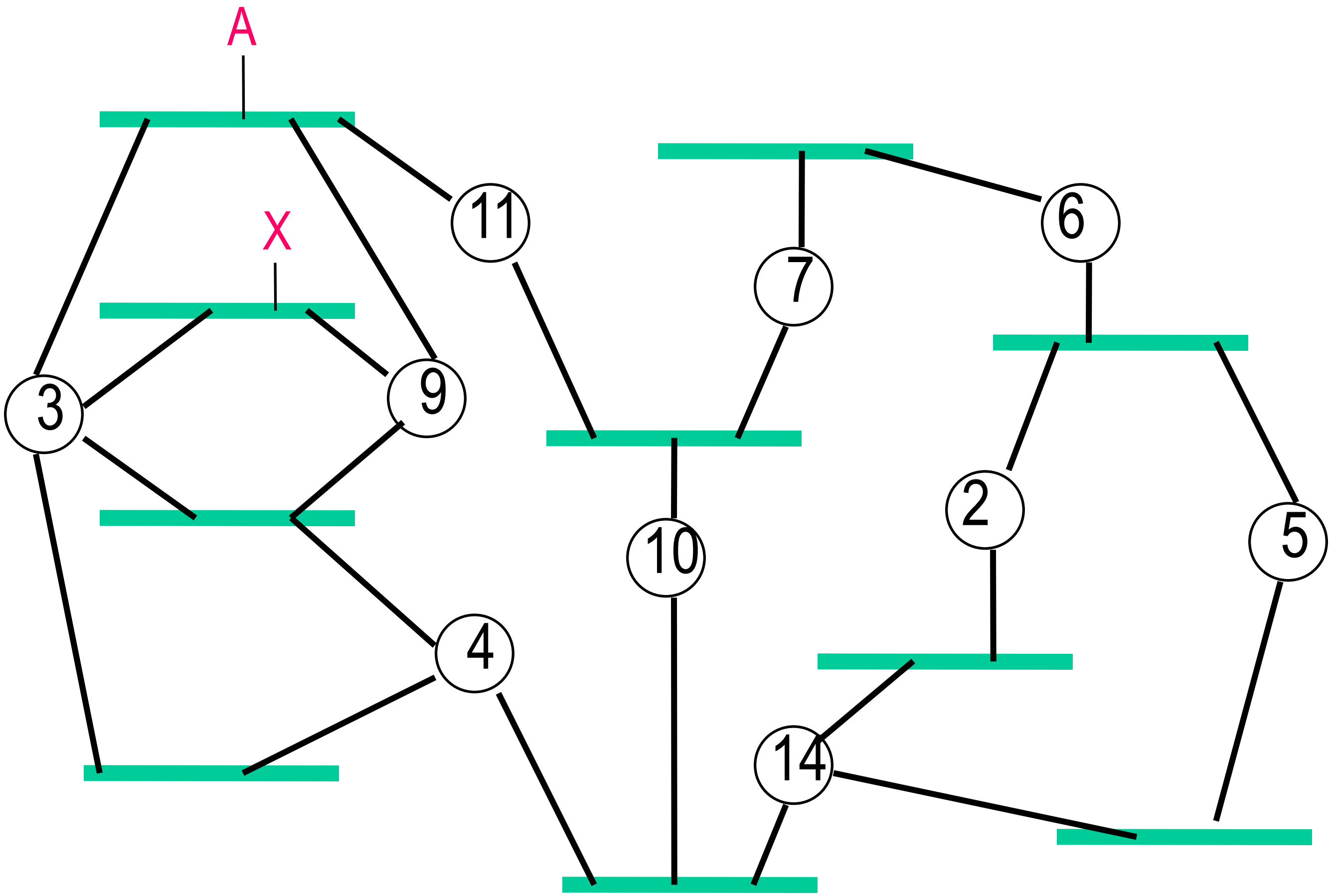
Without modifying the endnode, or Ethernet packet, in any way

The basic concept

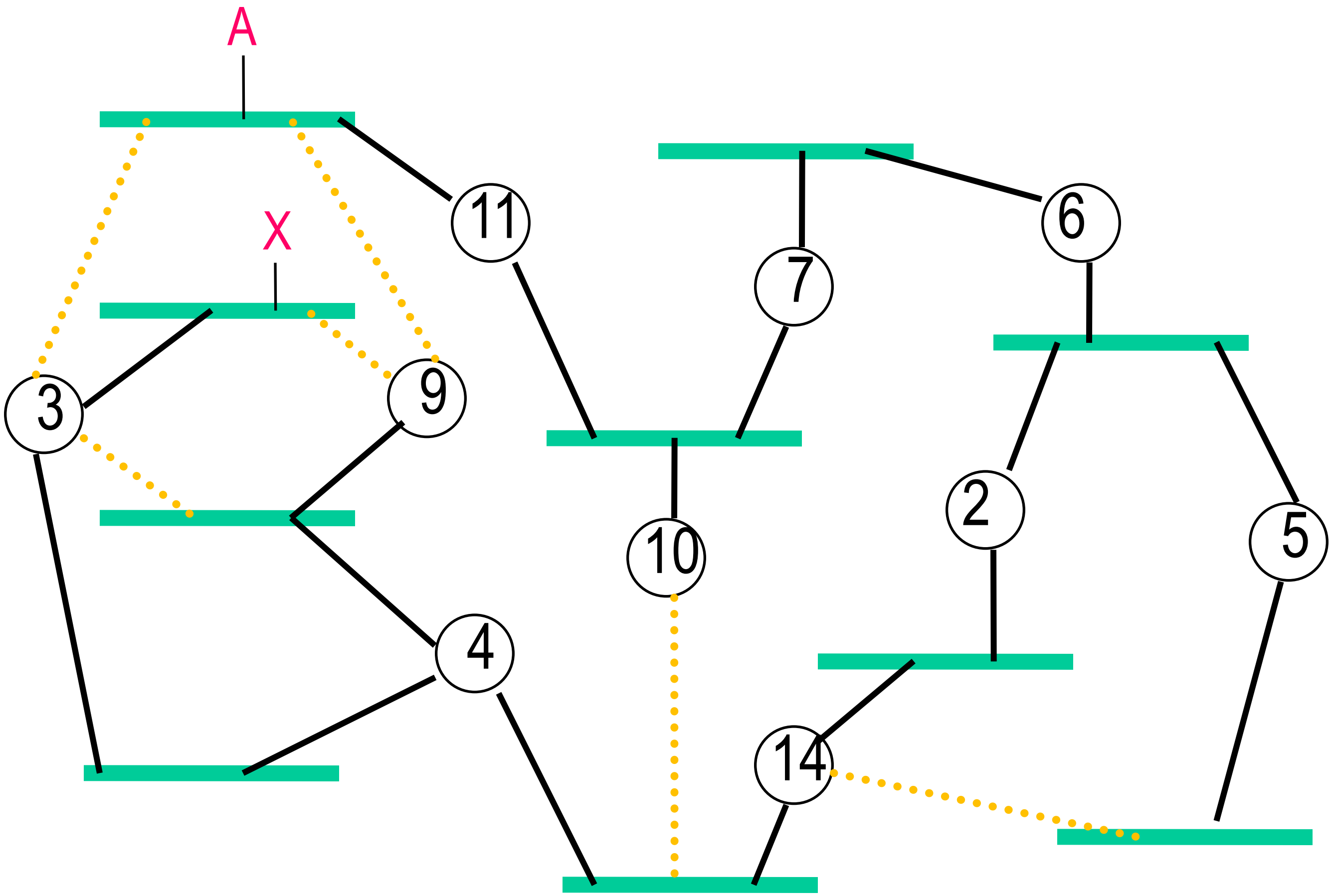
- Bridge just listens promiscuously, and forwards to each other port when the ether is free
- Learn (Source=S, input port). Once learned, if see a packet with destination=S, know where to forward it (rather than “all the ports”)
- This requires a tree (no loops) topology


A
X,C
J
A
X
C
E
D

Physical Topology



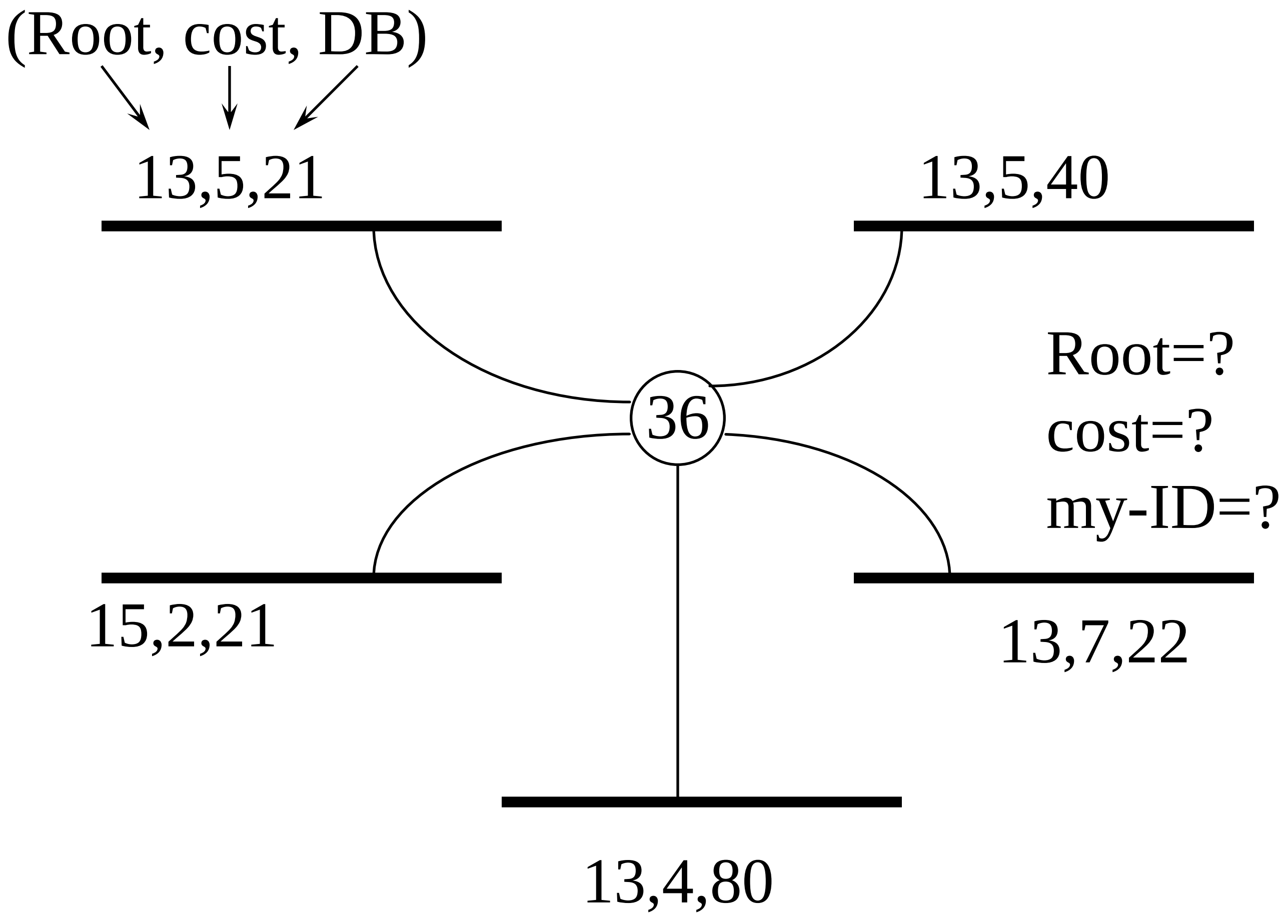
Pruned to Tree



Basic Spanning Tree Algorithm

- Bridge ("DB" or "Designated Bridge") that can transmit "best Hello" on the link periodically transmits a Hello (BPDU):
  - Root ID (actually priority.ID)
  - distance to Root
  - DB ID (priority.ID)
  - other stuff
- "Best Hello" is numerically smallest
  - Root ID | distance | my ID | port ID

Finding cost to Root



(Root, cost, DB)

13,5,21

13,5,40

36

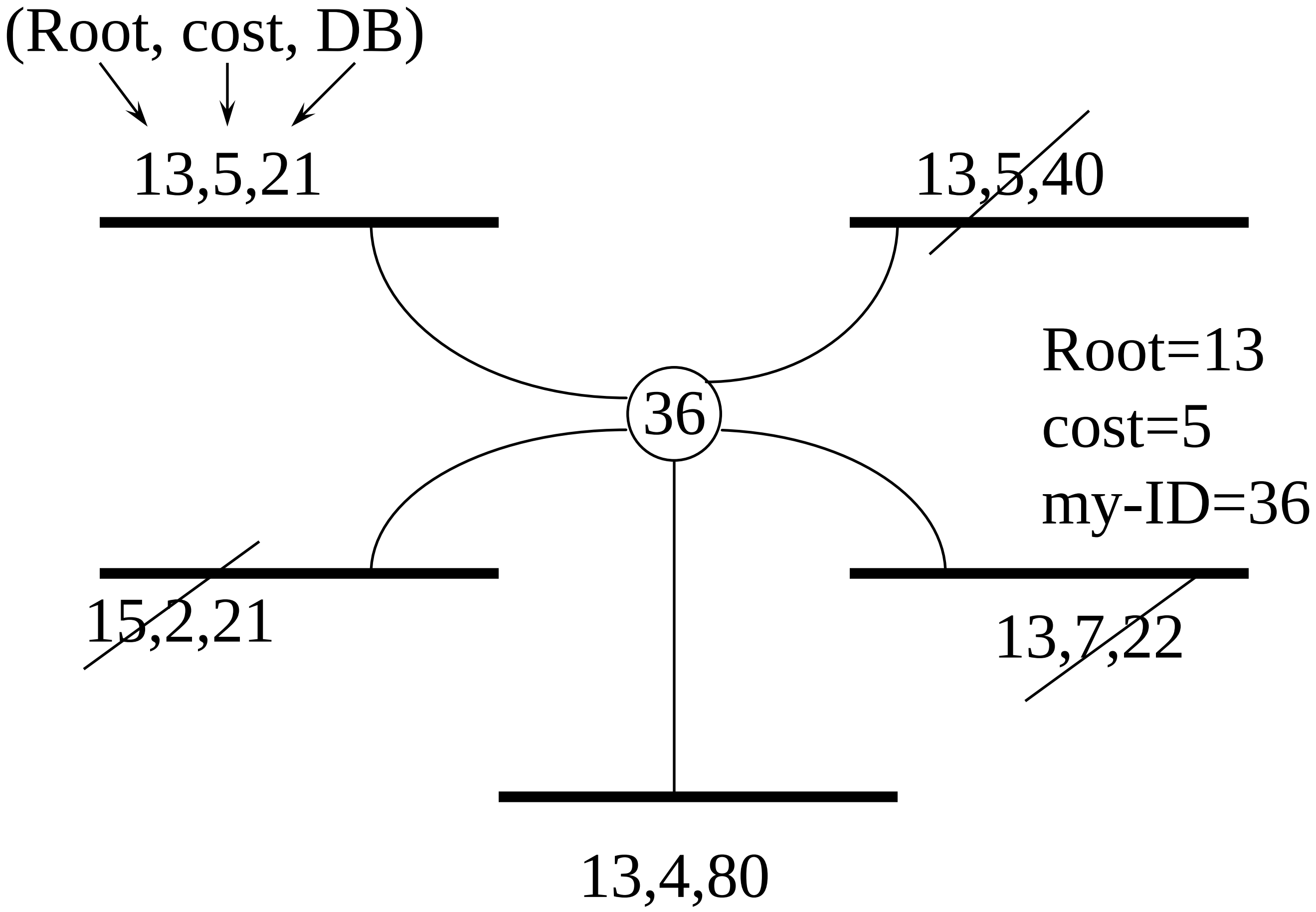
Root=?
cost=?
my-ID=?

15,2,21

13,7,22

13,4,80

Finding cost to Root



(Root, cost, DB)

13,5,21

~~13,5,40~~

36

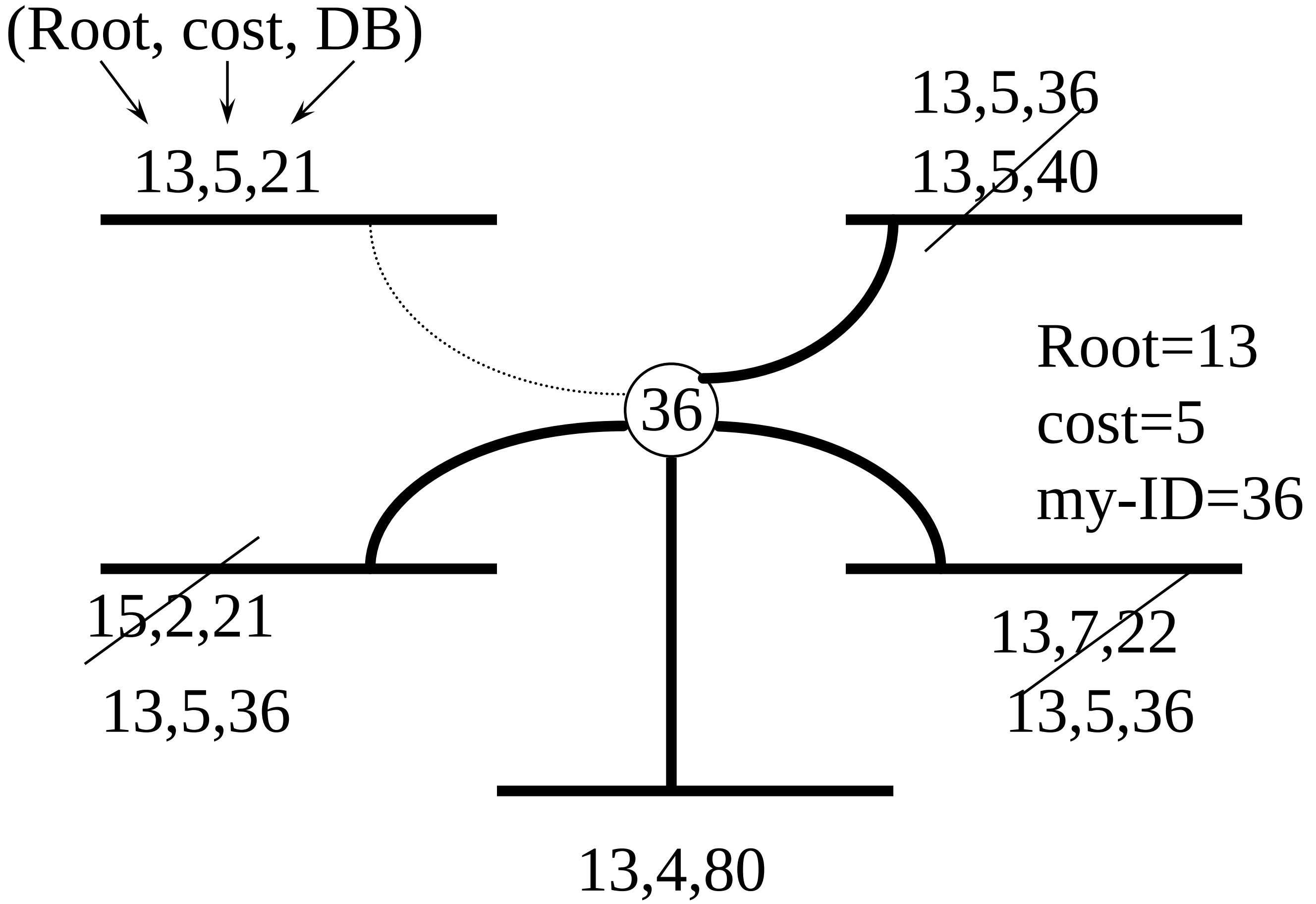
Root=13
cost=5
my-ID=36

~~15,2,21~~

~~13,7,22~~

13,4,80

In tree: if DB or Root port



(Root, cost, DB)

13,5,21

13,5,36

~~13,5,40~~

36

Root=13
cost=5
my-ID=36

~~15,2,21~~

13,5,36

~~13,7,22~~

13,5,36

13,4,80

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.

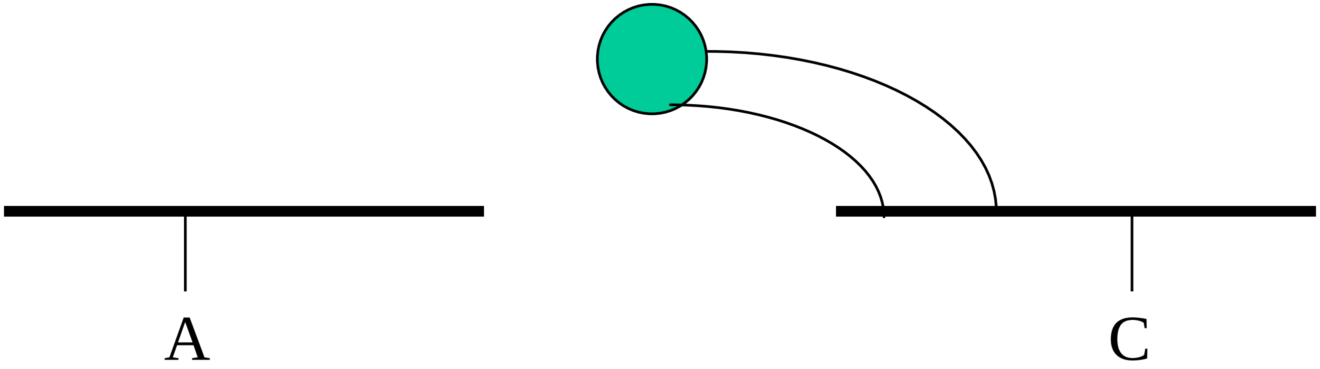
A mesh is made by folks like me.
Then bridges find a spanning tree.

Radia Perlman

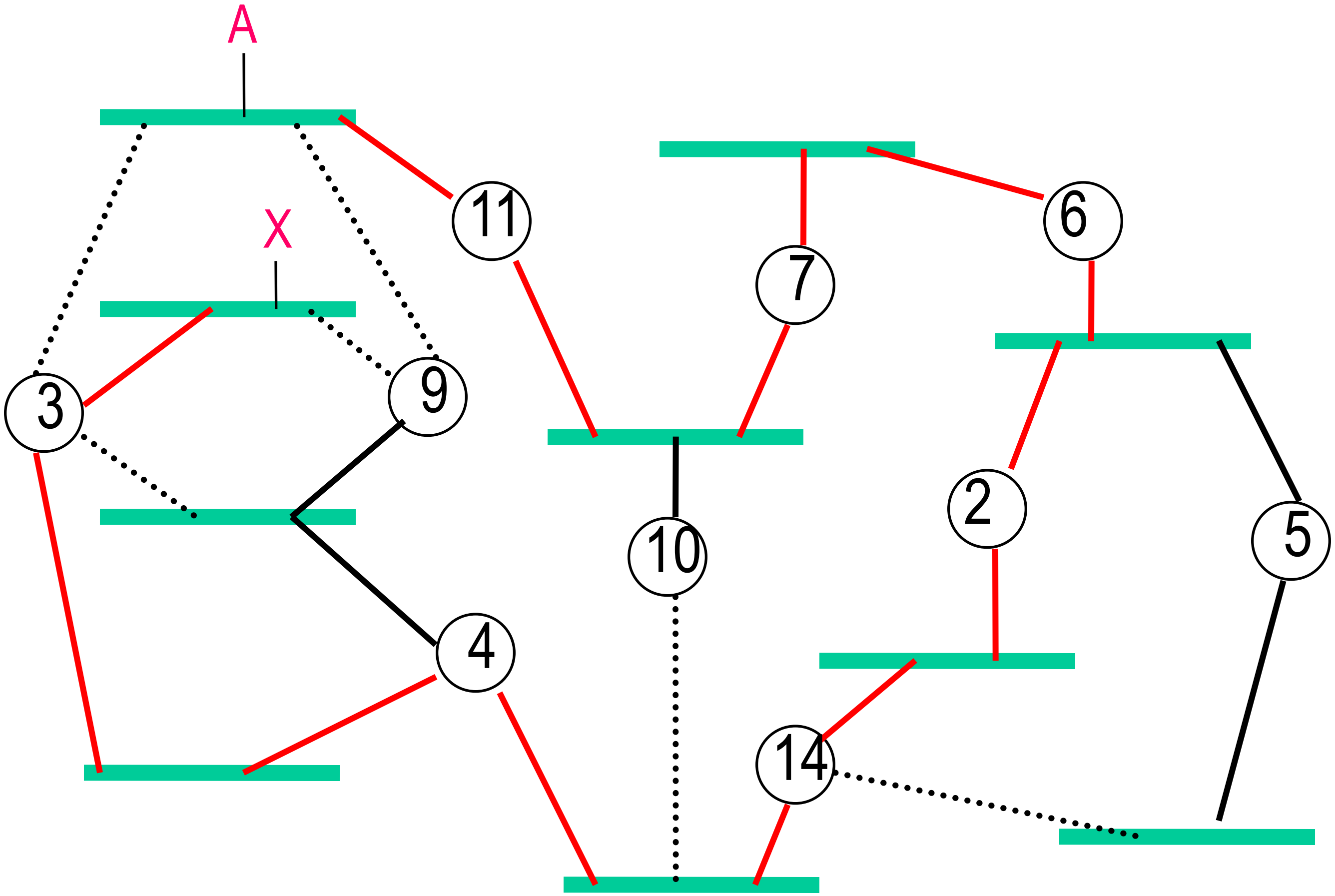
Bother with spanning tree?

- Maybe just tell customers “don’t do loops”
- But loops allow for backup paths
- And mistakes can be made…
- 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

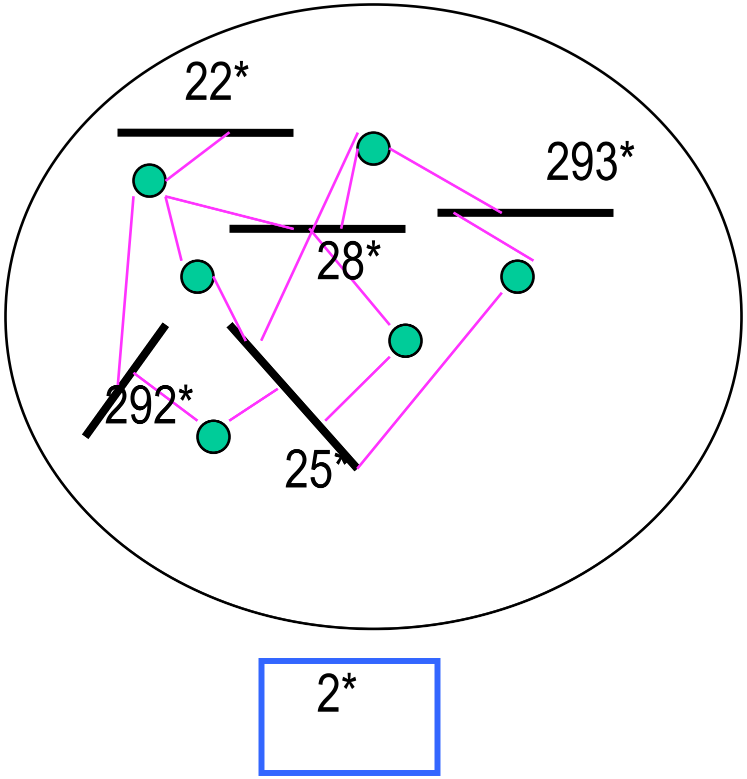
Layer 3 doesn’t have to work that way!

- 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

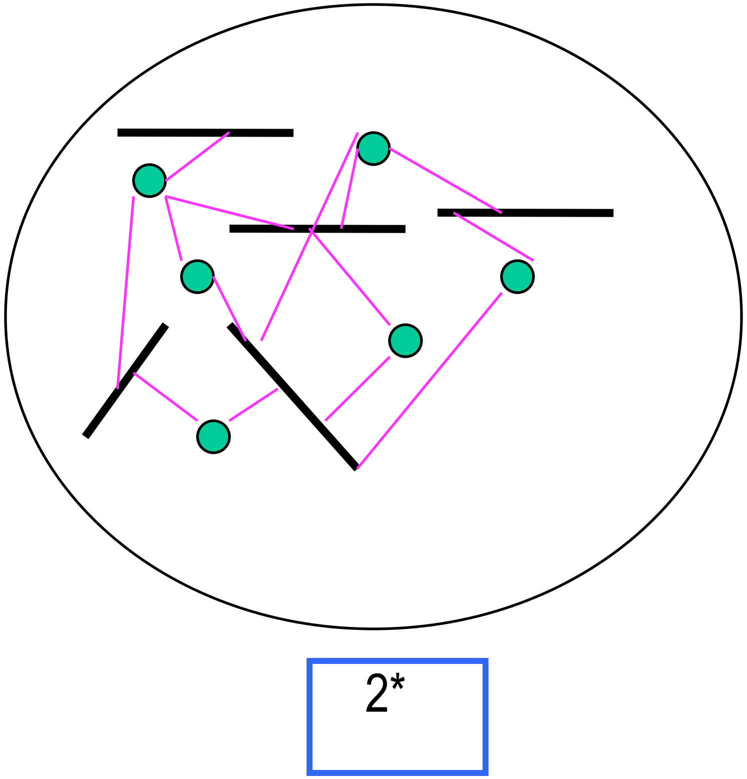
14 bytes	6 bytes
Prefix shared by all nodes in large cloud	Endnode ID

Hierarchy

One prefix per link (like IP)

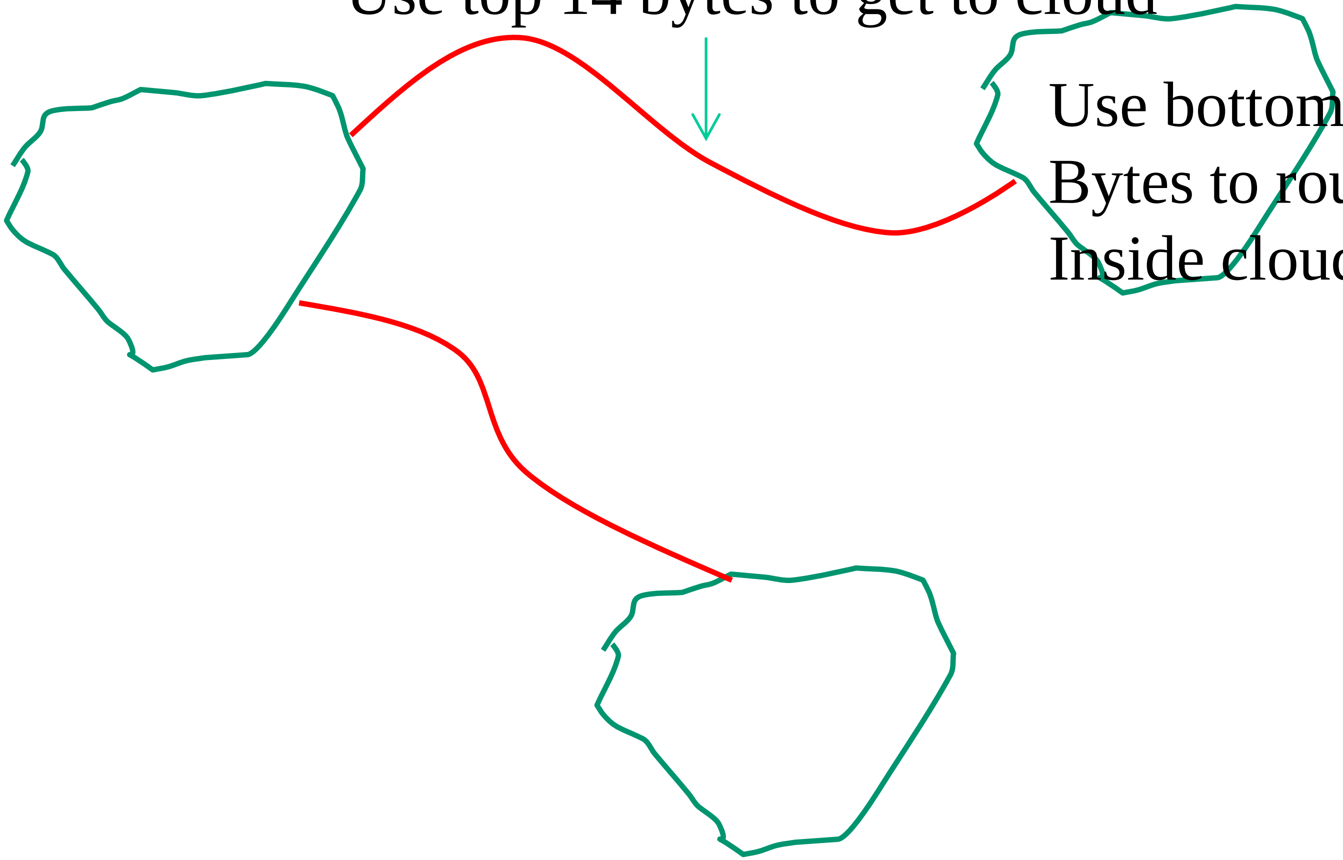


One prefix per campus



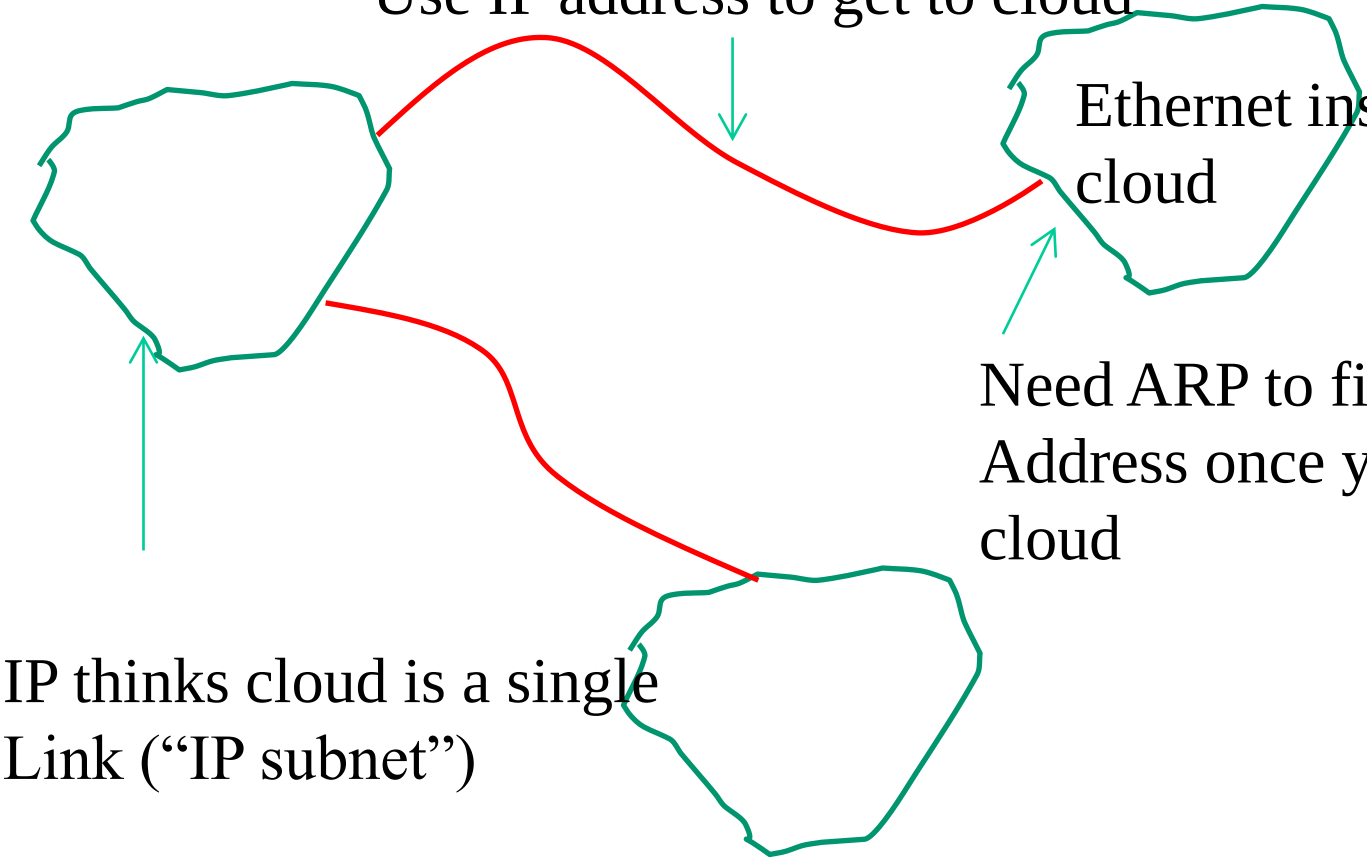
CLNP

Use top 14 bytes to get to cloud



Use bottom 6
Bytes to route
Inside cloud

IP+Ethernet



Use IP address to get to cloud

Ethernet inside cloud

Need ARP to find Ethernet Address once you get to the cloud

IP thinks cloud is a single Link (“IP subnet”)

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

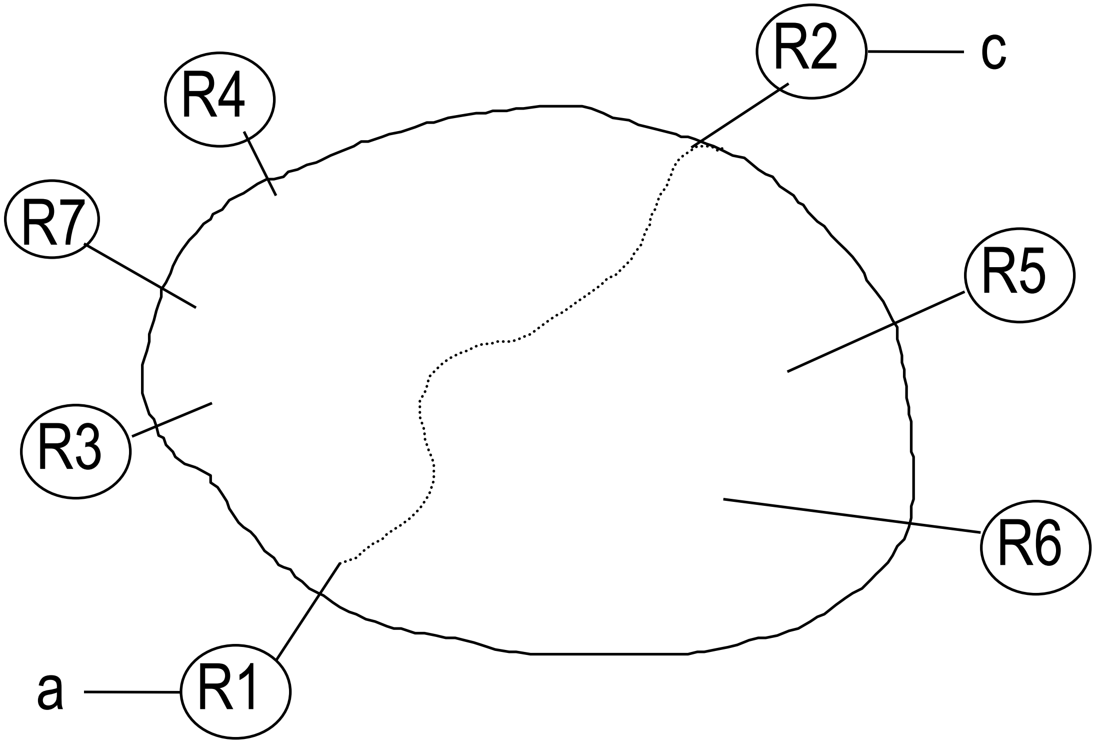
TRILL

- 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

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

Basic TRILL concept



Basic TRILL concept

- TRILL switches find each other (perhaps with bridges in between)
- Calculate paths to other TRILL switches
- First TRILL switch tunnels to last TRILL switch

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

Table of Endnodes

- (IP, MAC, egress nickname)
- Traditionally: flooding
  - IP → MAC done by flooding ARP
  - MAC → egress done by flooding
- But it doesn’t have to be that way

Alternatives

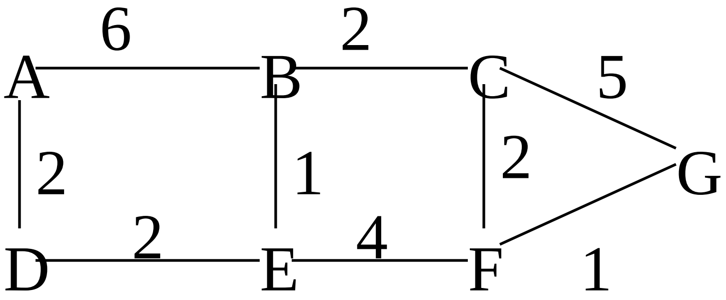
- Configuration of location of all endnodes
- Querying a directory rather than flooding when talking to a new endnode
    - Usually there is something (e.g., DHCP server) that knows where everything is
    - You could have mirrors of that
    - It's not a big enough database to worry about anything fancy
    - If something moves, directory can remember who asked

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.,
  - Hop Count
  - Shortest paths
  - Multiple paths (exploit parallelism)
  - Traffic engineering
- Extra header has easy-to-look-up addresses (dense)

Run “Link State Protocol”

- 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	

Specifically, IS-IS

- IS-IS was the protocol for DECnet/CLNP
- Very little has changed since 1980’s
- OSPF kind of copied it
- IS-IS better for TRILL because
  - IS-IS runs on layer 2; does not depend on IP
  - Its encoding is very flexible; add “TLV” fields
  - Simpler, more scalable

Network of RBridges (TRILL switches)

- All the RBridges know how to reach all the other RBridges
- But don’t know anything about endnodes

Why link state?

- Since all switches know the complete topology, easy to compute lots of trees deterministically (we’ll get to that later)
- Easy to piggyback “nickname allocation protocol” (we’ll get to that later)

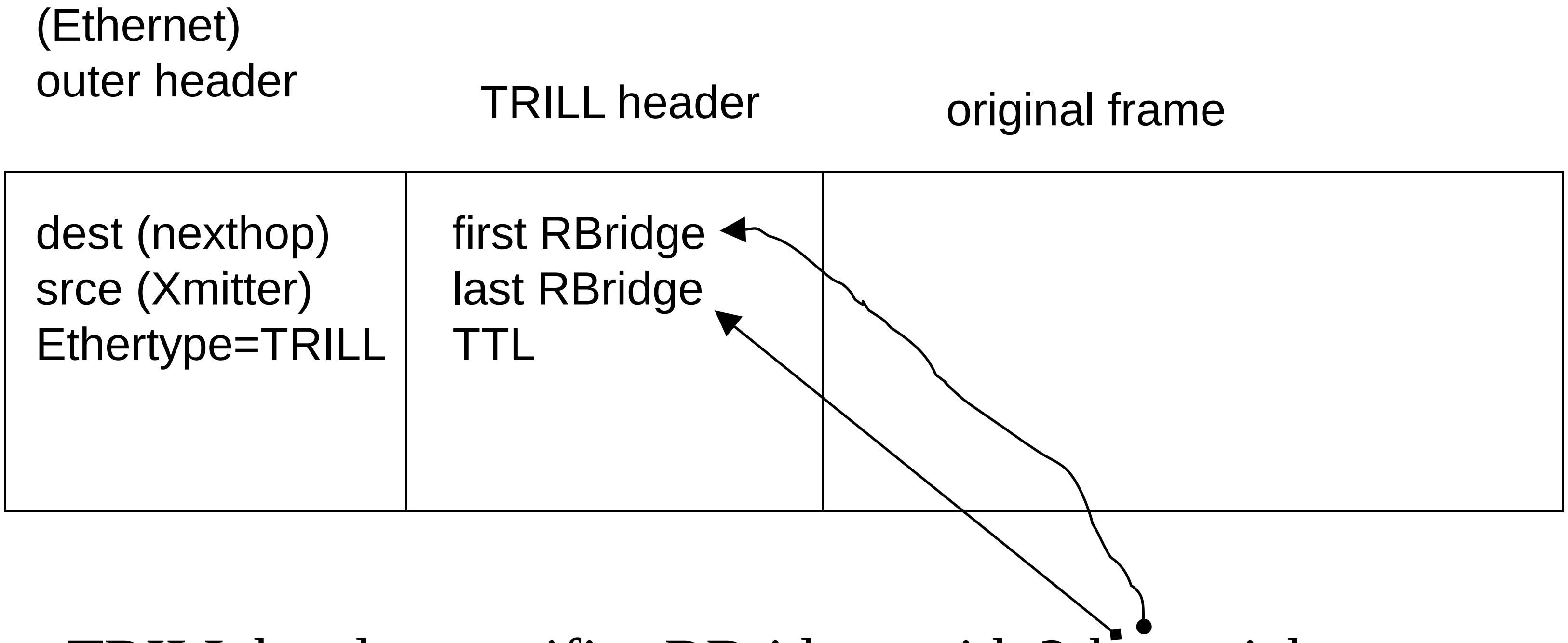
Routing inside campus

- First RB encapsulates to last RB
  - So header is “safe” (has hop count)
  - Inner RBridges only need to know how to reach destination RBridge
- Still need tree for unknown/multicast
  - But don’t need spanning tree protocol – compute tree(s) deterministically from the link state database

Details

- What the encapsulated packet looks like
- How R1 knows that R2 is the correct “last RBridge”

Encapsulated Frame



(Ethernet)
outer header

TRILL header

original frame

dest (nexthop)
srce (Xmitter)
Ethertype=TRILL

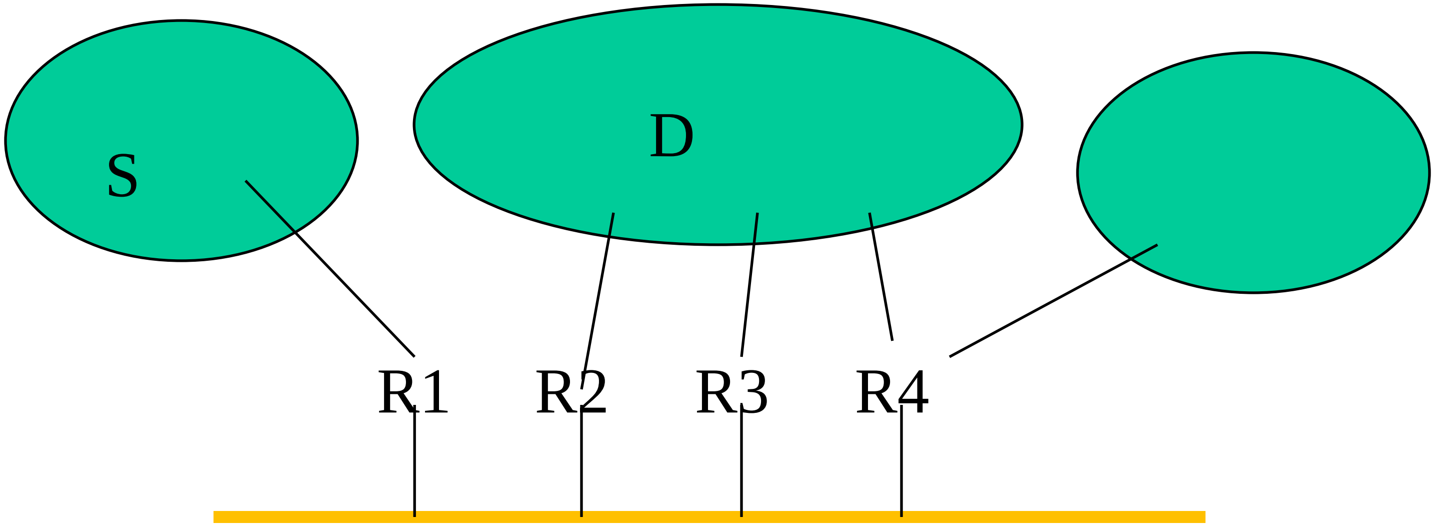
first RBridge
last RBridge
TTL

TRILL header specifies RBridges with 2-byte nicknames

What’s the outer Ethernet header for?

- To work well with switches inside spanning tree island
- In case there’s a shared link, to specify who the next recipient is

Outer Ethernet Header on Shared Links



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

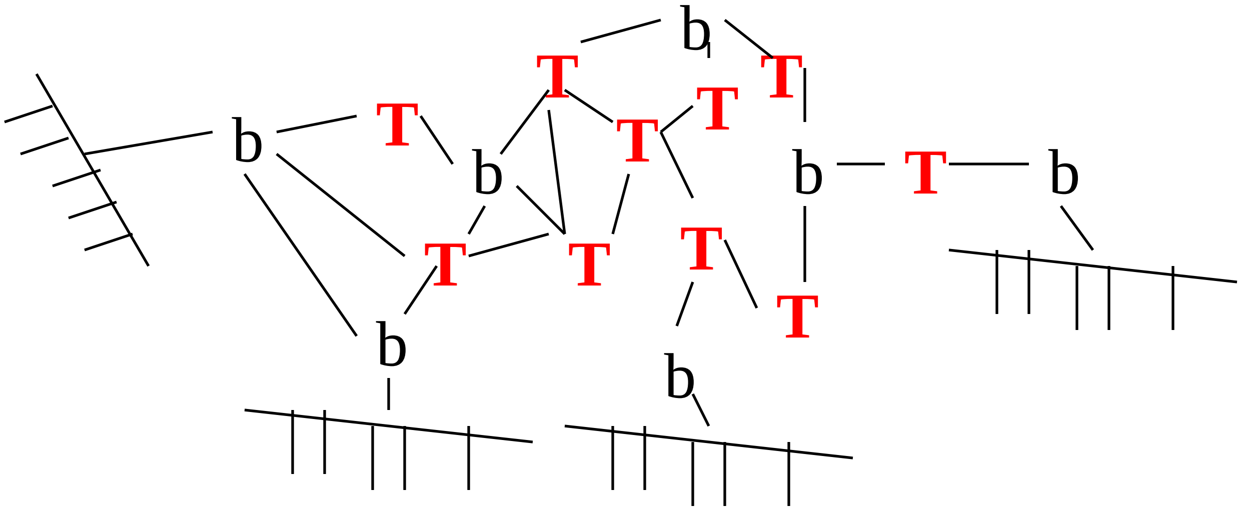
- Switches inside cloud don’t need to know about all the endnodes…
  - Forwarding table size of # of switches
- The outer header is like a layer 3 header, and can use all the layer 3 techniques, e.g.,
  - Shortest paths
  - Multiple paths (exploit parallelism)
  - Traffic engineering

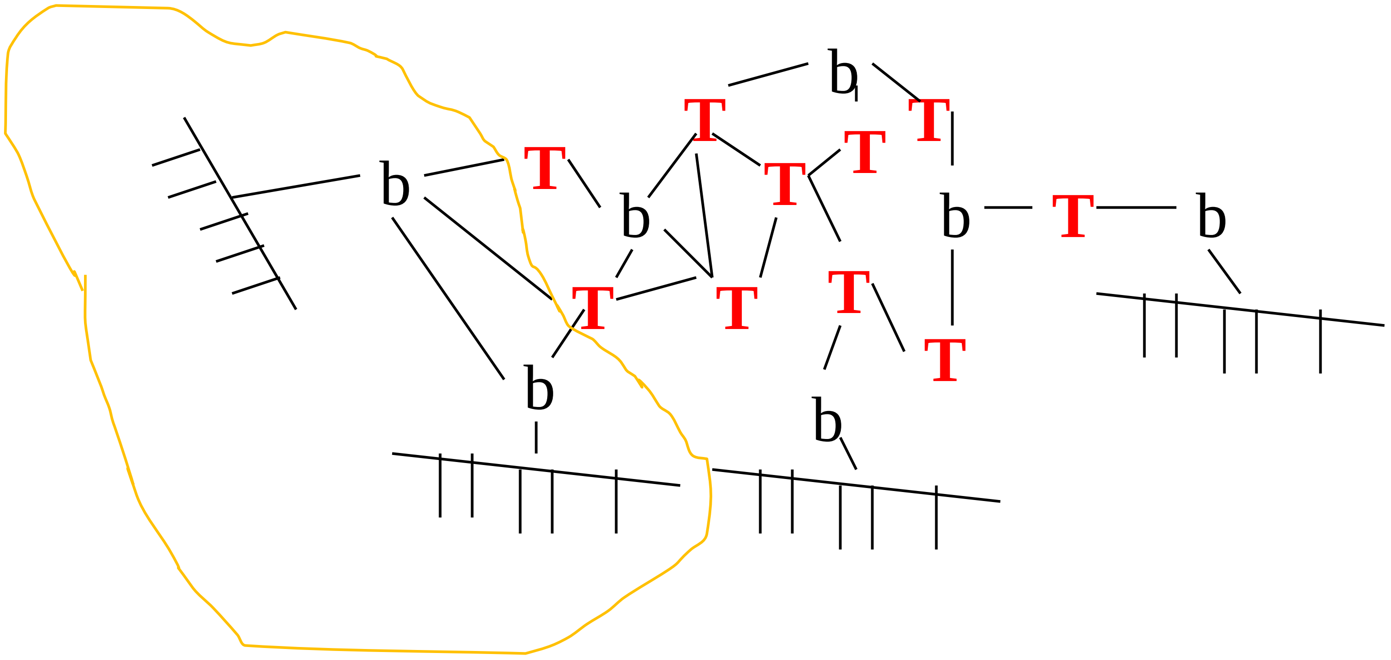
2-byte Nicknames

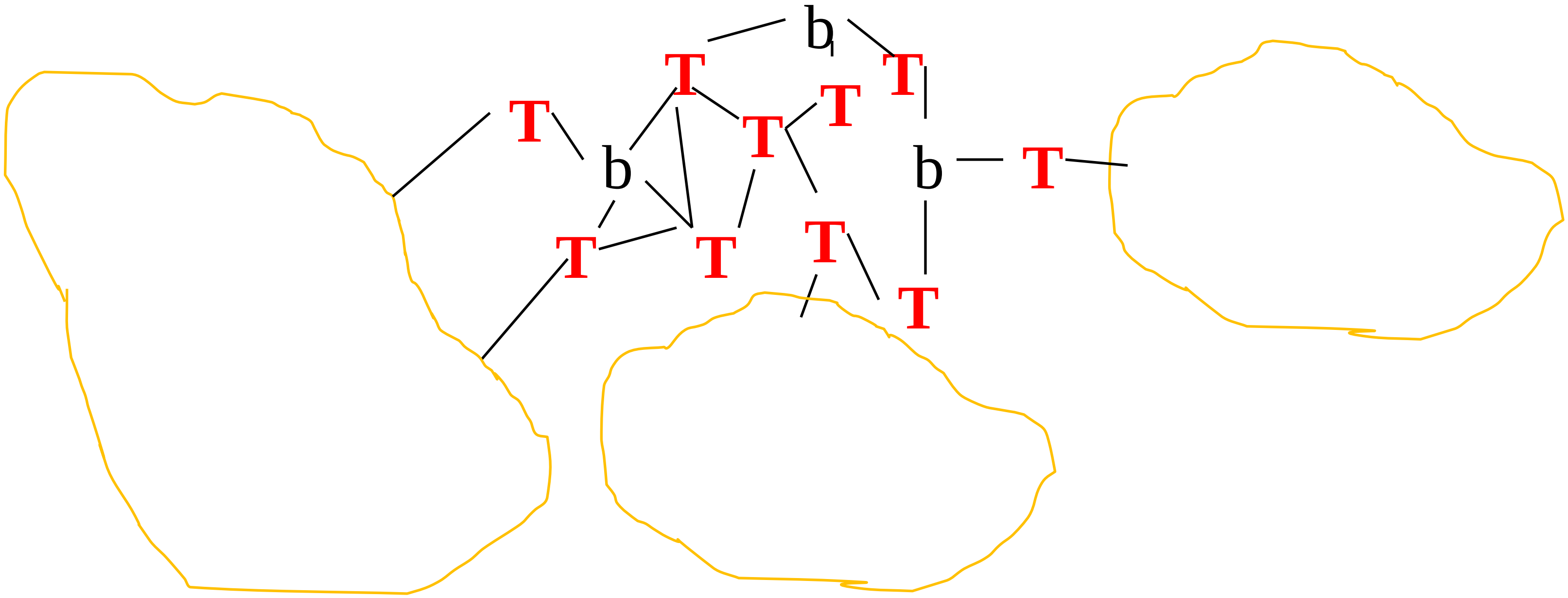
- Saves hdr room, faster fwd’ing
- Dynamically acquired
  - Choose unused #, announce in LSP
  - If collision, IDs and priorities break tie
  - Loser chooses another nickname
  - Configured nicknames higher priority
- Is 64,000 switches enough?

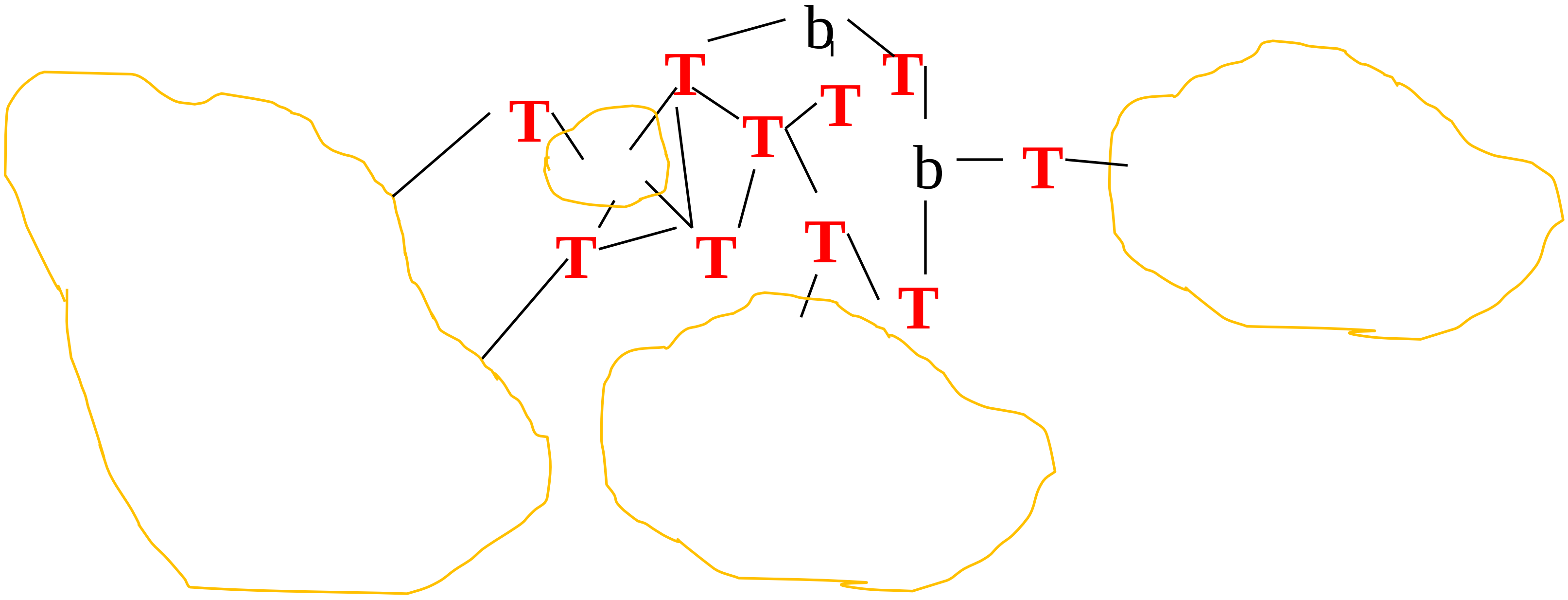
Form network of TRILL switches

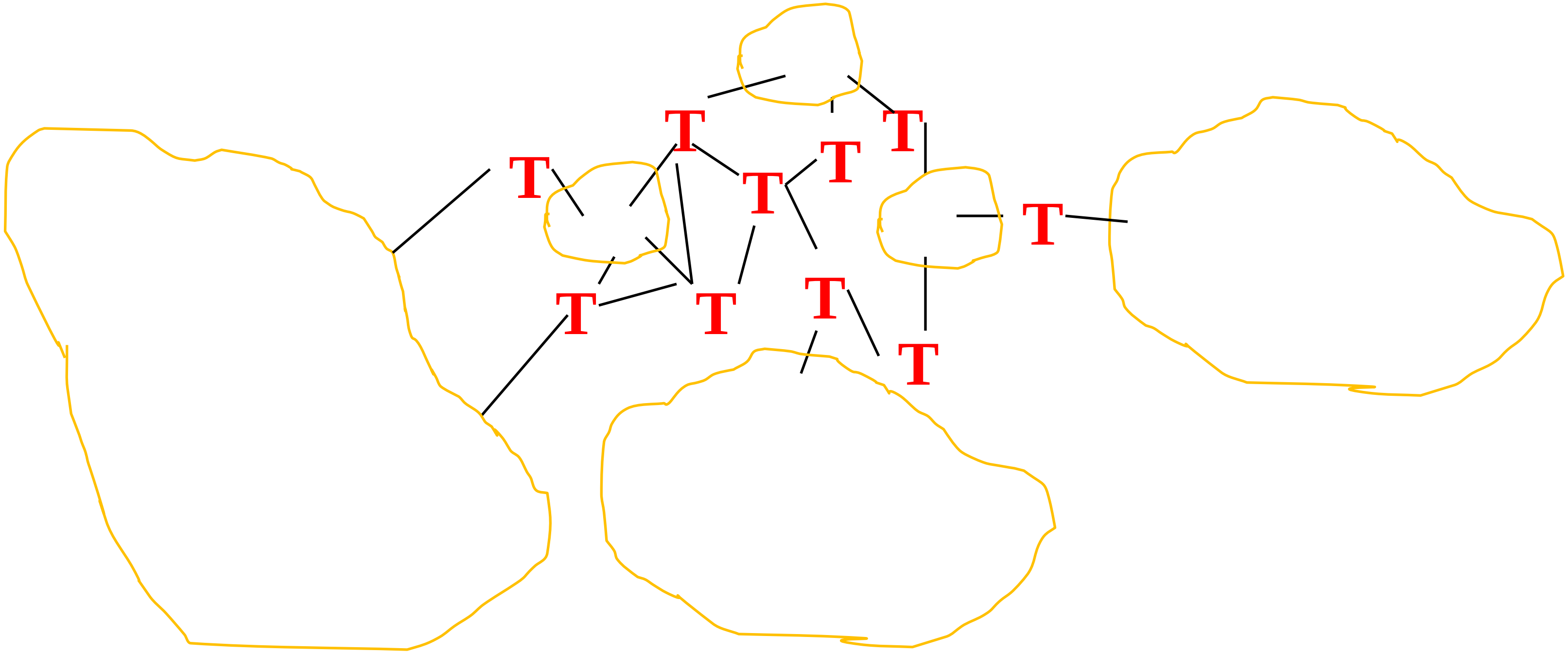
- TRILL switches find each other if:
  - Directly connected with pt-to-pt
  - Both connected to same Ethernet island
- Do “link state protocol” among TRILL switches to calculate paths to other TRILL switches

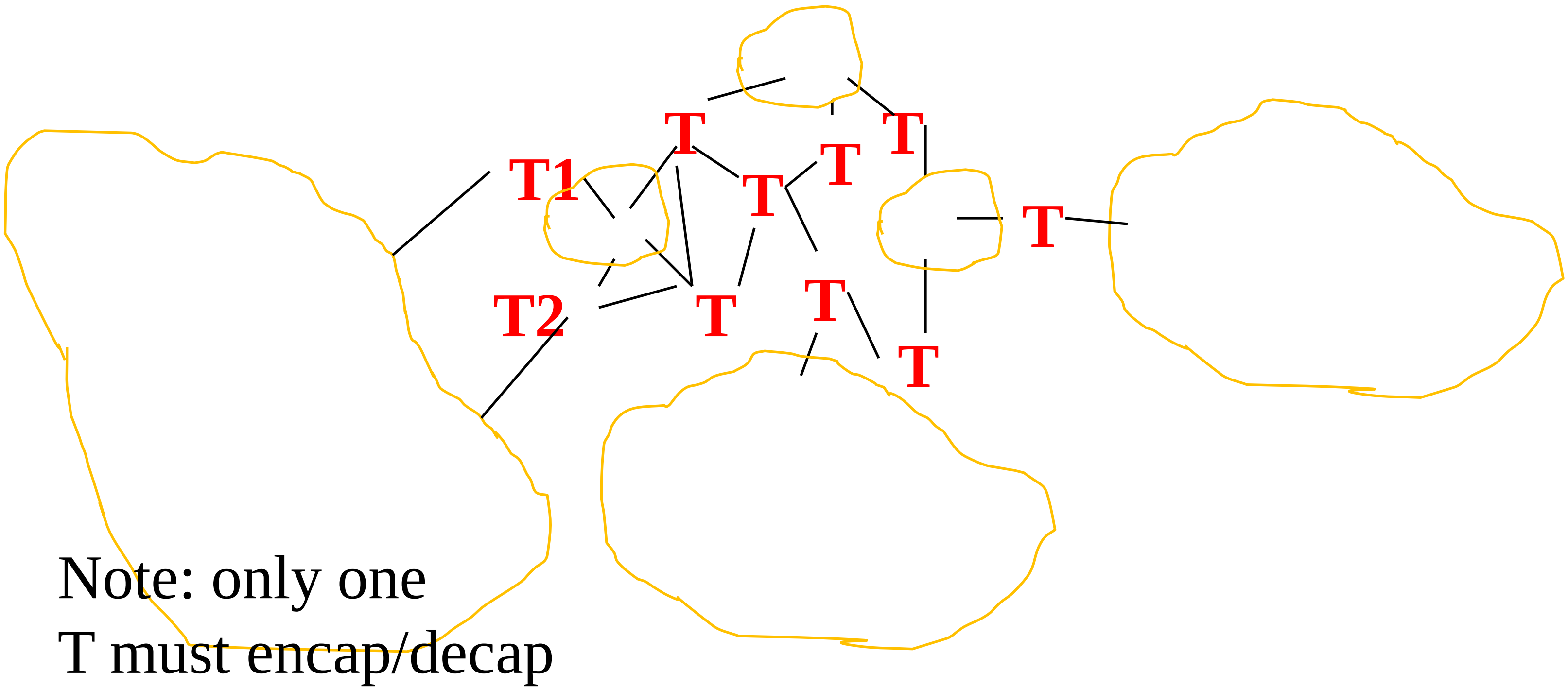

b
T
T
T
T
b
T
b
T
b
T
b
T
T
T
b
T
b


b
T
T
b
T
T
T
T
b
T
b
T
T
T
T
b
b
T
b


b
T
T
T
T
T
b
T
b
T
T
T
T
T
T


b
T
T
T
T
T
b
T
T
T
T
T
T


T
T
T
T
T
T
T
T
T
T
T



Note: only one
T must encap/decap
So T1 and T2 must
Find each other and
coordinate

How does R1 know that R2 is the correct “last RBridge”?

- Currently….If R1 doesn’t, R1 sends packet through a tree
- When R2 decapsulates, it remembers (ingress RBridge, source MAC)

How does R1 know that R2 is the correct "last RBridge"?

- Currently….If R1 doesn't, R1 sends packet through a tree
- When R2 decapsulates, it remembers (ingress RBridge, source MAC)
- Note: If we'd used MPLS as encapsulation header, we wouldn't have that information (1st RBridge) to learn from

Other possibilities

- Configuration of (MAC addresses, location) into switches
- Directory listing (IP, MAC, switch location)
  - Consulted by first switch, or hypervisor, or VM, or application
  - No reason endnode couldn’t encapsulate into TRILL header, using switch’s nickname as “first switch” – or, pretend to be a switch and get a nickname

Directory

- Could act as the DHCP server (knows (IP, MAC) because it hands them out..can learn switch location based on encapsulated DHCP request
- But what if MAC moves? Short DHCP leases?
  - Or, there is a fabric manager that determines VM moves…it will know and can tell directory
- Could remember who requested an entry, and tell them if info changes

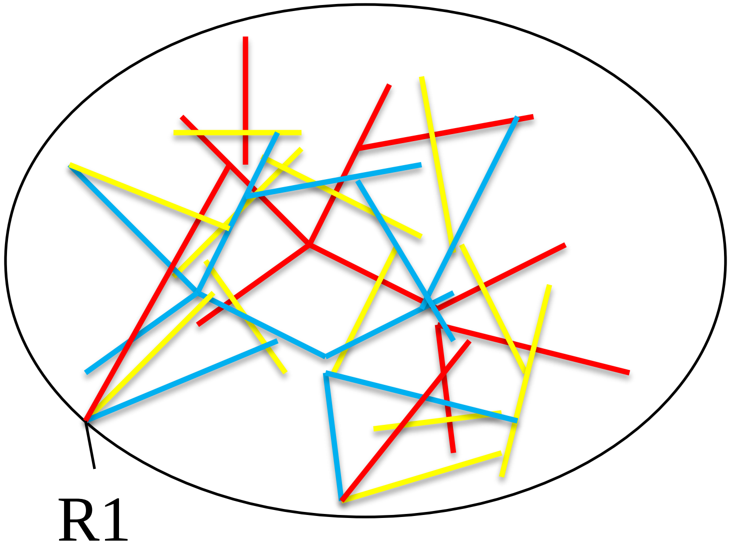
Use of “first” and “last” RBridge in TRILL header

- For Unicast, obvious
  - Route towards “last” RBridge
  - Learn location of source from “first” RBridge
- For Multicast/unknown destination
  - Use of “first”
    - to learn location of source endnode
    - to do “RPF check” on multicast
  - Use of “last”
    - To allow first RB to specify a tree
    - Campus calculates some number of trees

TRILL and Multicast

- For spreading multicast traffic around, campus computes several trees
- “Last TRILL switch” field in TRILL header specifies which tree to send on
- Traffic filtered in the core based on VLAN, and IP multicast addresses

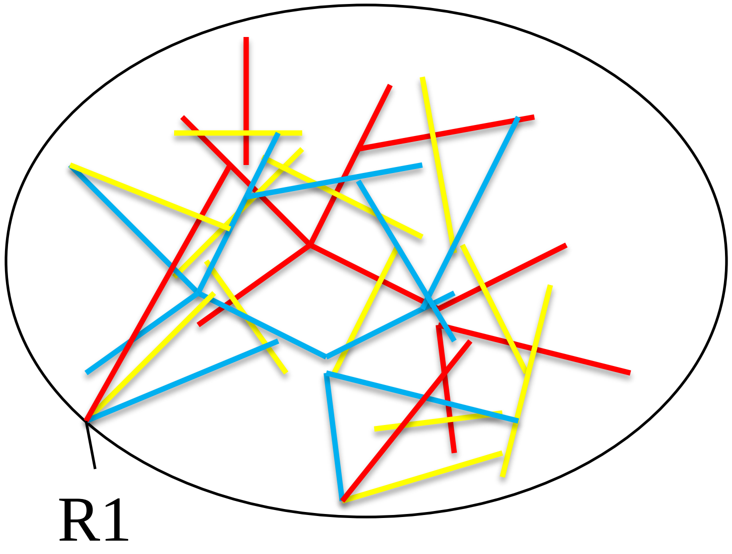
Multiple trees for multicast



R1 specifies which tree
(yellow, red, or blue)

Multiple trees for multicast

Which tree	1st switch	hops	Original Ethernet packet



R1 specifies which tree
(yellow, red, or blue)

Orthogonal concept

Recently, a bunch of similar things invented

- NVGRE, VXLAN, …

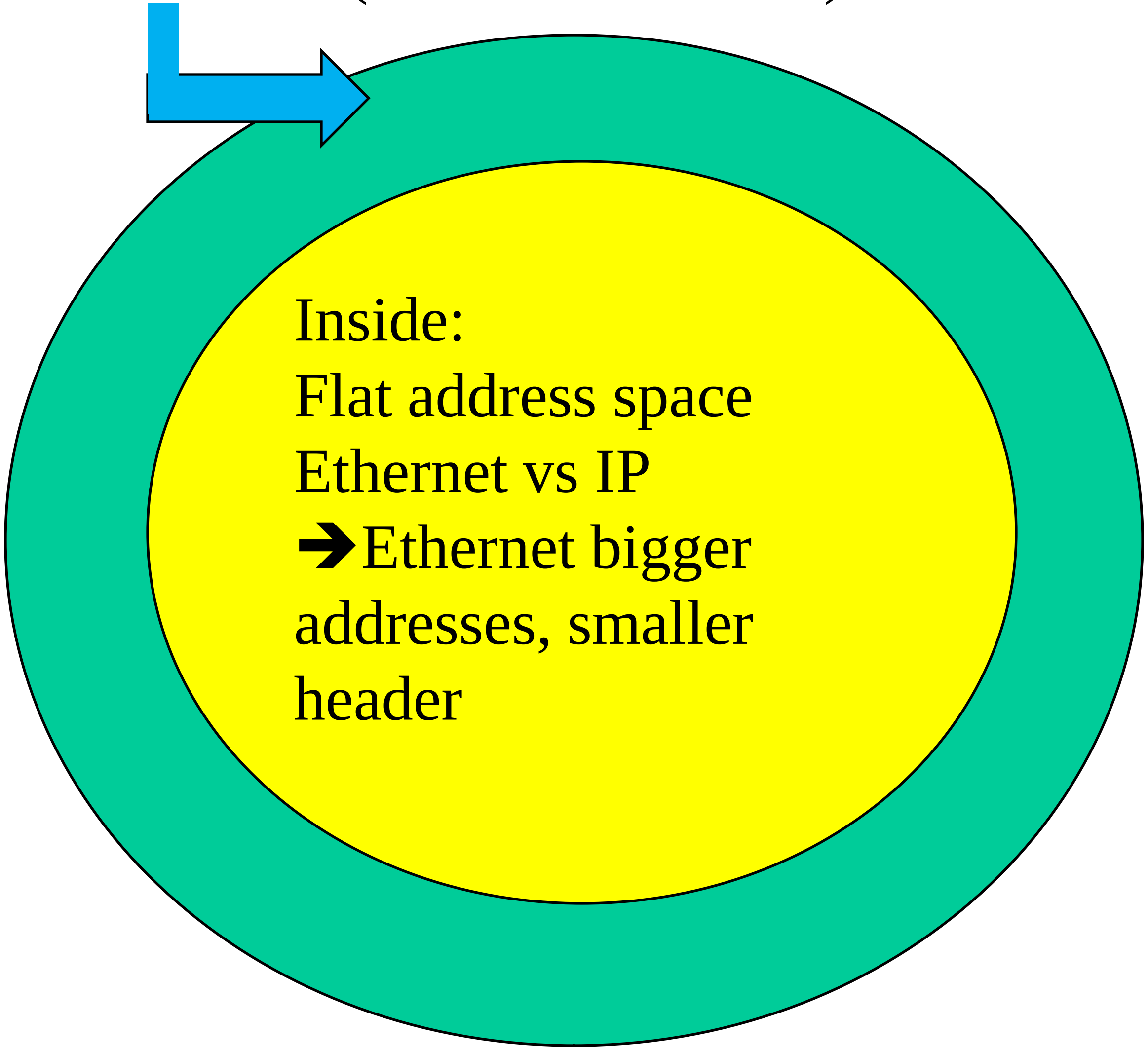
Who encapsulates/decapsulates?

- 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

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

Outer header: TRILL is 6 bytes, autoconfigured, vs IP+UDP/GRE+stuff (VXLAN/NVGRE)



Inside:
Flat address space
Ethernet vs IP
➔Ethernet bigger addresses, smaller header

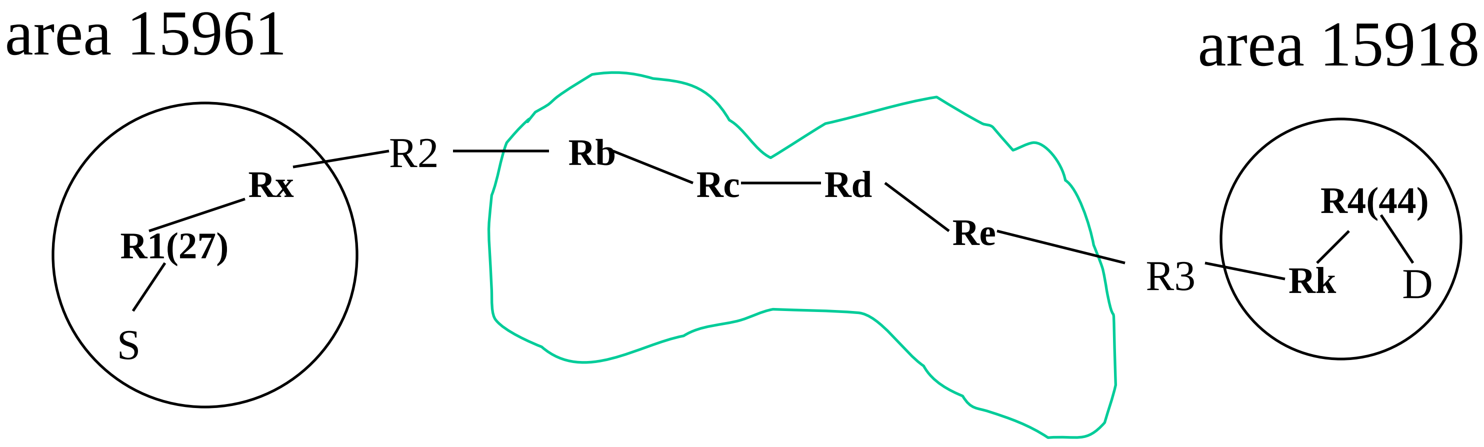
What does encapsulation header address?

- 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

Extensions

- Expanding the number of VLANs (called “fine-grained labels) to 24 bits (actually, that’s done now)
- IS-IS hierarchy
  - Various proposals for nicknames:
    - Unique nicknames: (area | nickname)
      - Each “area” gets a block of nicknames
    - Aggregated Nickname: Rewrite nickname field at level 1/level 2 boundary

How aggregated nicknames work



S transmits packet to D

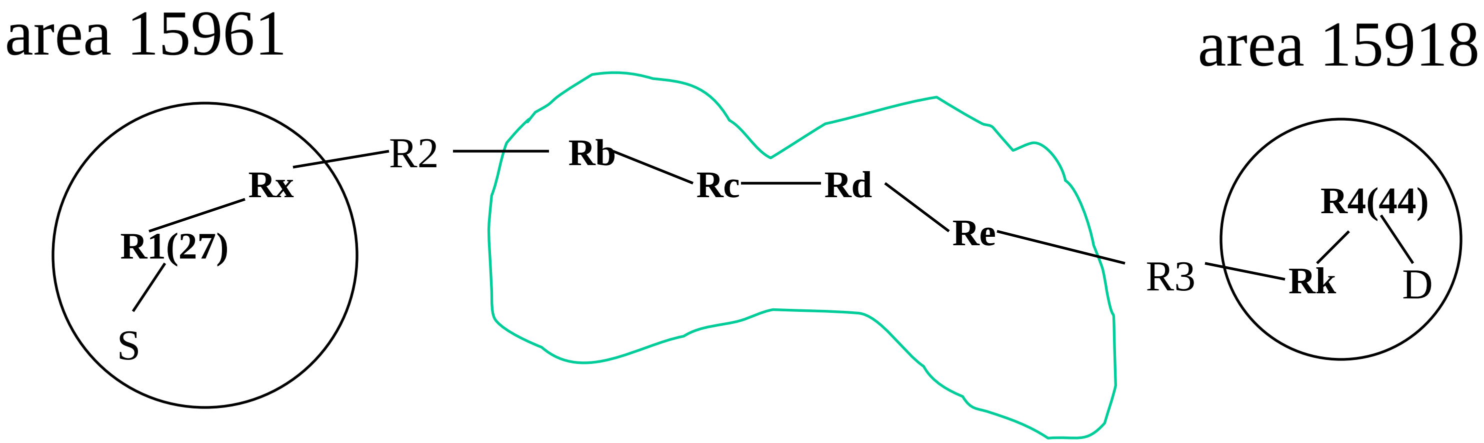
R1 encapsulates with TRILL header

ingress=27, egress=15918

R2 replaces ingress with 15961

R3 replaces egress with 44

How aggregated nicknames work



S transmits packet to D
R1 transmits ingress=27, egress=15918
R2 transmits ingress=15961, egress=15918
R3 transmits ingress=15961, egress=44

What if D is unknown?

- If unknown by R2, need to do multi-area multidestination frame
- If R1 did know the right area, but R3 doesn’t know D, then R3 needs to turn it into “unknown unicast” within D’s area

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

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

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

New concept: Self-Stabilization

New concept: Self-Stabilization

- Even if glitches occur (e.g., sick switch injects bad messages), once bad nodes get disconnected, network should return to normal

New concept: Self-Stabilization

- Even if glitches occur (e.g., sick switch injects bad messages), once bad nodes get disconnected, network should return to normal
- Important because unlike PCs, networks don’t have a “restart” button

Rant

- Windows 8 did not get rid of of “restart”!
- They just hid it…
  - – Swoop gesture from right
  - – Touch “settings”
  - – Touch “power”
  - – Voila! There’s the restart button!

Back to self-stabilization

- Seems easy…
- Two examples of non-self-stabilizing protocols
  - Original ARPANET
  - Spanning tree

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

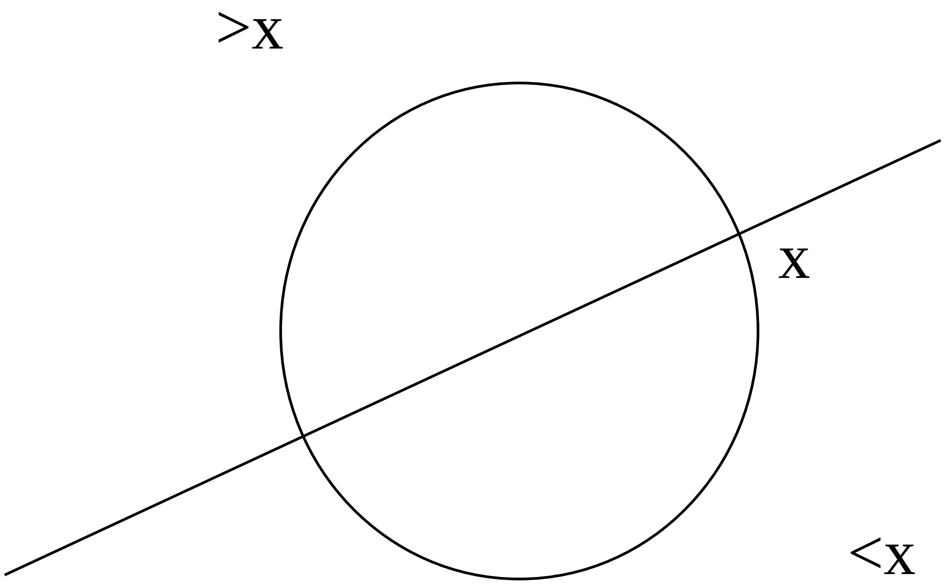
Sequence Number Handling

- 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)

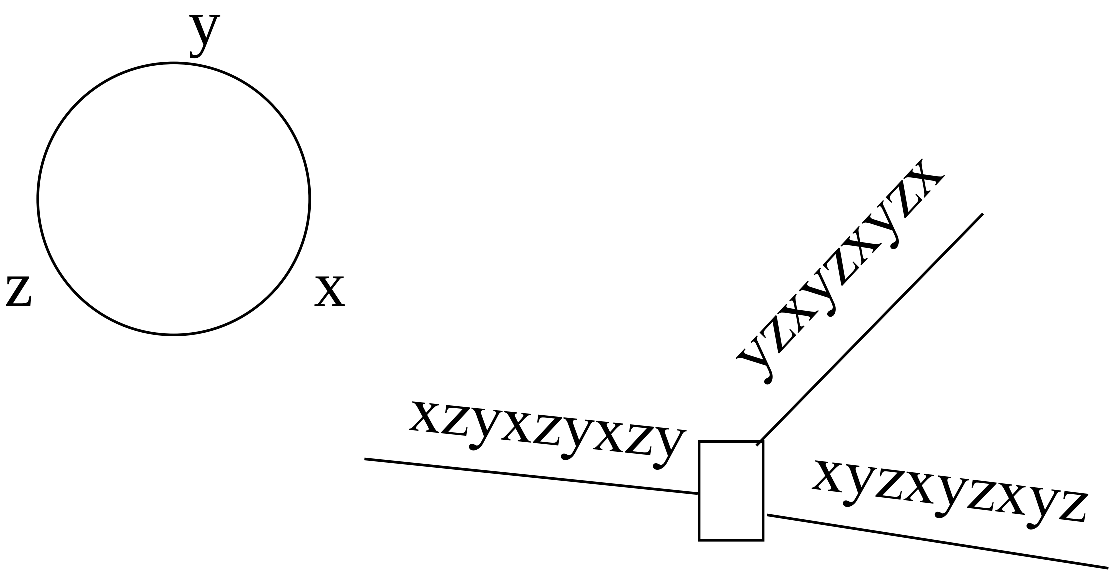
Arithmetic in circular space



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

2nd example: Spanning tree Protocol

2nd example: Spanning tree Protocol

- Forwarding Ethernet packets is really dangerous
  – No hop count
  – Exponential proliferation of packets on shared links

When should you use a link?

- In layer 3, if you hear “I’m still here” messages from a neighbor
  - Otherwise you do NOT use the link

When should you use a link?

- In layer 3, if you hear “I’m still here” messages from a neighbor
  - Otherwise you do NOT use the link
- In contrast, with spanning tree, if you do NOT hear from a “more qualified” neighbor bridge, you assume you must forward onto the link

So…suppose you can’t keep up with wire speed

- You throw away incoming packets before even checking if they are spanning tree messages
- You might throw away too many from your neighbor…and conclude you’re the only bridge on the link!

So…suppose you can’t keep up with wire speed

- You throw away incoming packets before even checking if they are spanning tree messages
- You might throw away too many from your neighbor…and conclude you’re the only bridge on the link!
- If you couldn’t keep up with incoming traffic before, wait until there is looping traffic…with no hop count!

In my defense

- My spec (for Digital) said “You must engineer your bridge to be able to process all incoming messages”
- But IEEE took that out

Topic if time

Data: Making it be there when you want it, and go away when you want it gone

Radia Perlman

(radia@alum.mit.edu)

Problem

- Traditional tradeoff between
  - Robust recovery after disaster: make lots of copies
  - Assured delete: need to find all copies and delete them
- We will simultaneously solve both problems: Allow robust recovery, and assured delete
- Paper: "File System Design with Assured Delete", NDSS 2007, Radia Perlman

Note

- This is about storage in a cloud or file server
- Once the file is supposed to be gone (from the cloud/server) it will be unrecoverable (from the cloud/server or any backups that the cloud/server keeps)
- But this isn’t “DRM”. If an authorized client machine accesses the data and stores it in the clear, or prints a copy…this doesn’t solve that

This talk

- Two types of assured delete
  - Expiration date
  - On-demand, individual files
- Both are simple and practical
- But the on-demand is a bad idea! (I’ll explain why, after explaining how to do it)

Expiration time

- When create data, put (optional) “expiration date” in metadata
- After expiration, file must be unrecoverable, even though backups will still exist

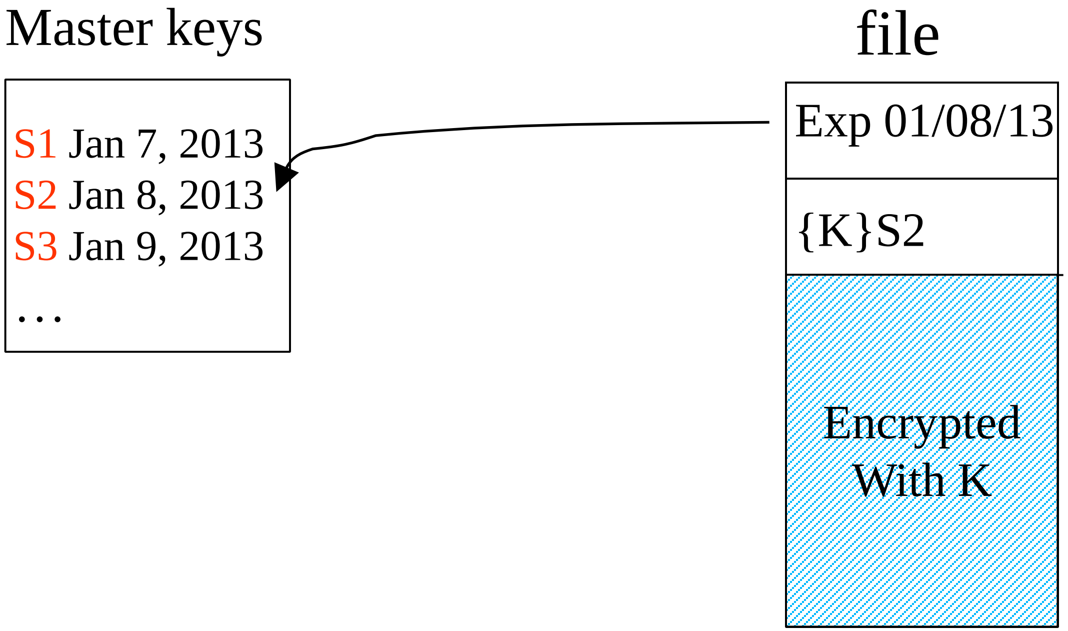
Obvious approach

- Encrypt the data, and then destroy keys
- But to avoid prematurely losing data, you'd have to make lots of copies of the keys
- Which means it will be difficult to ensure all copies of backups of expired keys are destroyed

First concept: Encrypt all files with same expiration date with the same key

File system with Master keys

Master keys: Secret keys (e.g., AES)
generated by file system
Delete key upon expiration



Master keys

S1 Jan 7, 2013
S2 Jan 8, 2013
S3 Jan 9, 2013
…

file

Exp 01/08/13

{K}S2

Encrypted
With K

How many keys?

- If granularity of one per day, and 30 years maximum expiration, 10,000 keys

So…how do you back up the master keys?

Imagine a service: An "ephemerizer"

- creates, advertises, protects, and deletes public keys
- "ephemerize keys" on backup by encrypting with ephemerizer public key
- To recover from backup: file system asks Ephemerizer to decrypt

Ephemerizer publicly posts

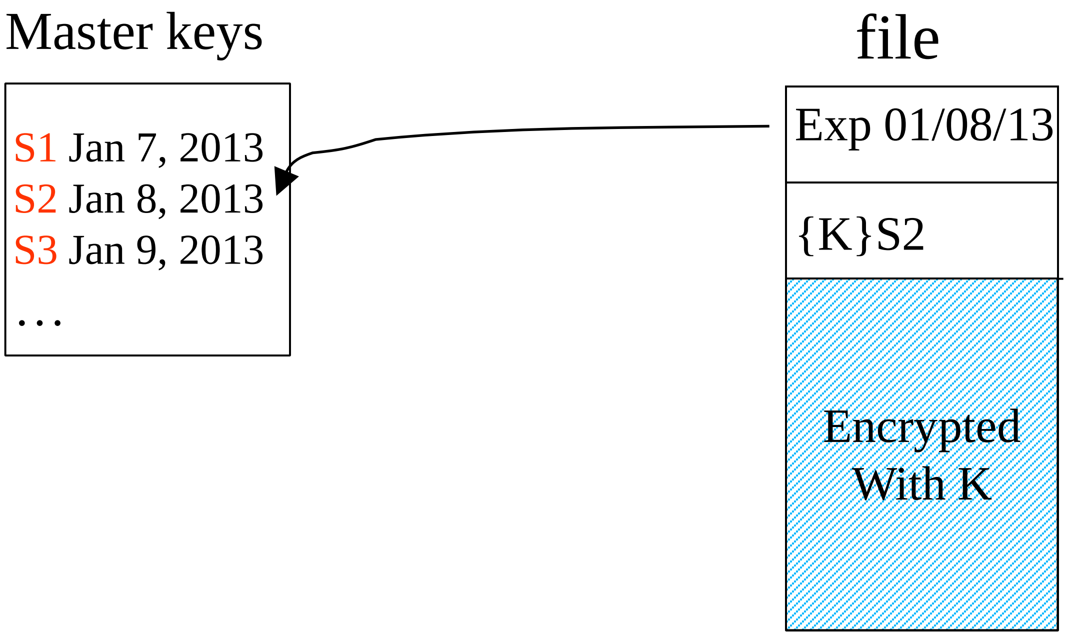
Jan 7, 2013: public key $P_{\text{Jan7of2013}}$
Jan 8, 2013: public key $P_{\text{Jan8of2013}}$
Jan 9, 2013: public key $P_{\text{Jan9of2013}}$
Jan 10, 2013: public key $P_{\text{Jan10of2013}}$

etc

One permanent public key P certified through PKI
Signs the ephemeral keys with P

File system with Master keys

Master keys: Secret keys (e.g., AES) generated by file system



Master keys

S1 Jan 7, 2013
S2 Jan 8, 2013
S3 Jan 9, 2013
…

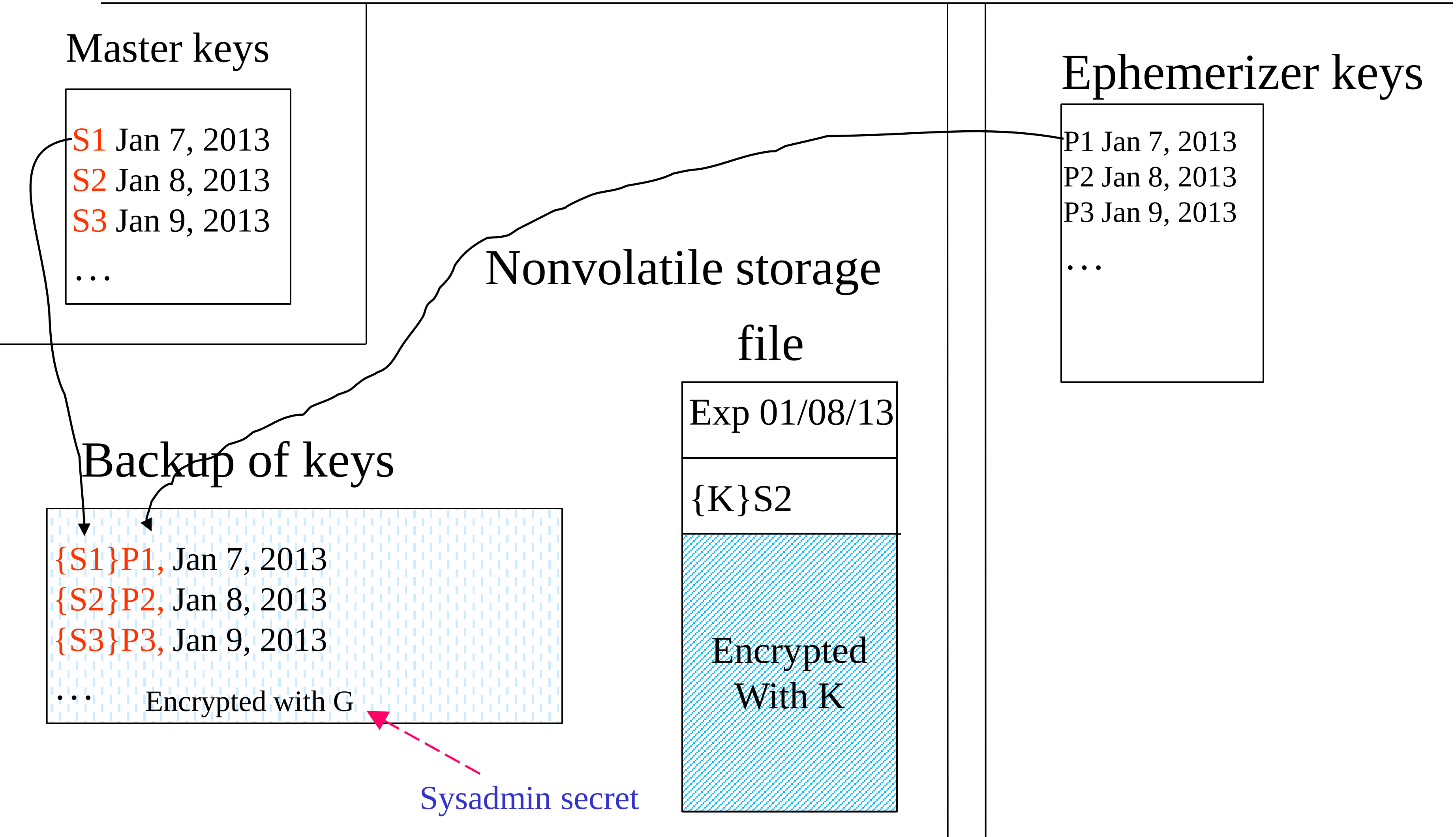
file

Exp 01/08/13

{K}S2

Encrypted
With K

Backup of Master Keys



Master keys

S1 Jan 7, 2013
S2 Jan 8, 2013
S3 Jan 9, 2013
…

Ephemerizer keys

P1 Jan 7, 2013
P2 Jan 8, 2013
P3 Jan 9, 2013
…

Nonvolatile storage

file

Exp 01/08/13

{K}S2

Encrypted
With K

Backup of keys

{S1}P1, Jan 7, 2013
{S2}P2, Jan 8, 2013
{S3}P3, Jan 9, 2013
…

Encrypted with G

Sysadmin secret

Notes

- Only talk to the ephemerizer if your hardware with master keys dies, and you need to retrieve master keys from backup
- Not every time you open a file!!
- Ephemerizer really scalable:
  - Same public keys for all customers (10,000 keys for 30 years, one per day)
  - Only talk to a customer perhaps every few years…to unwrap keys being recovered from backup

But you might be a bit annoyed at this point

But you might be a bit annoyed at this point

- Haven’t we simply pushed the problem onto the ephemerizer?
- It has to reliably keep private keys until expiration, and then reliably delete them

Two ways ephemerizer can “fail”

- Prematurely lose private keys
- Fail to forget private keys

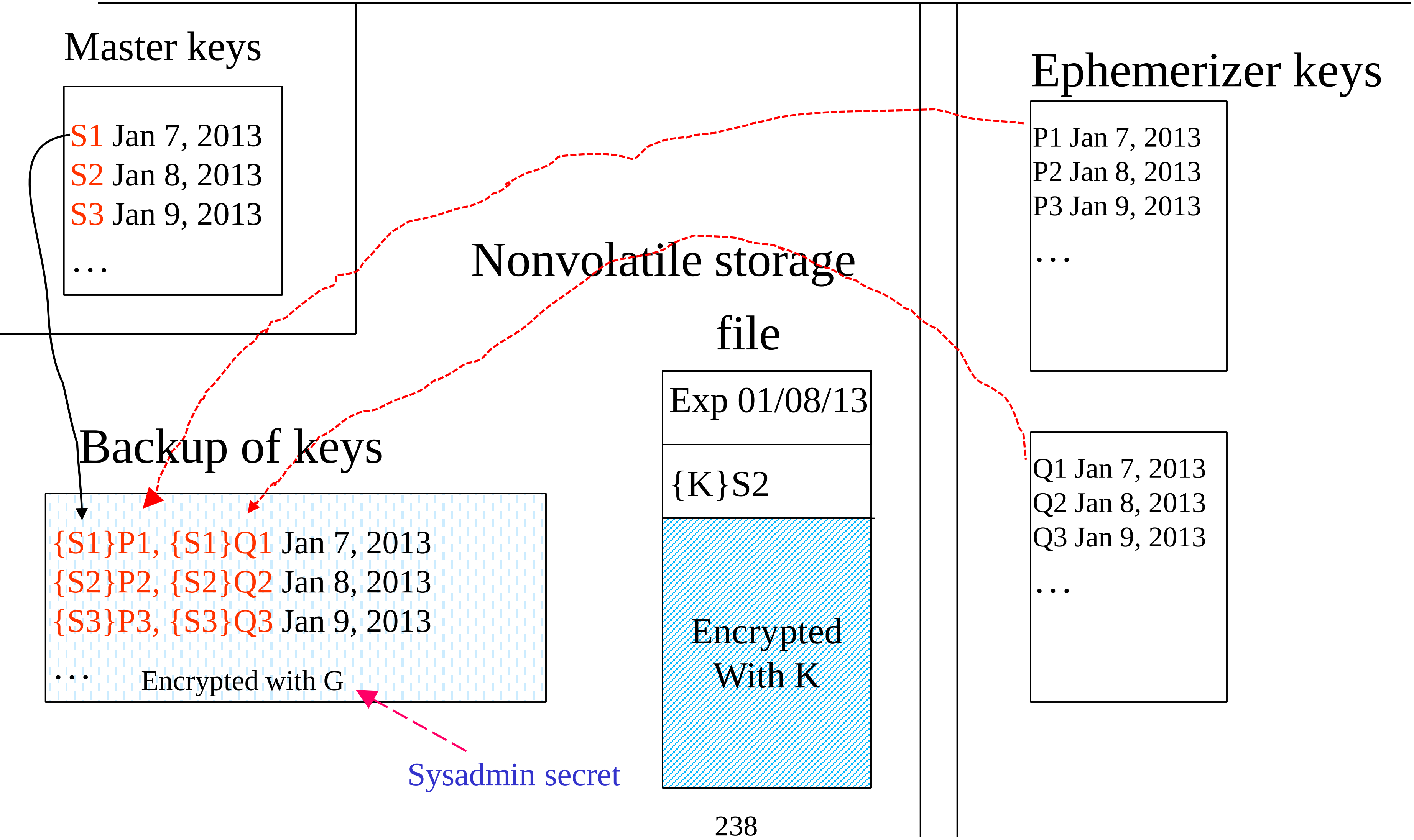
The reason why it’s not just pushing the problem

- We will allow an ephemerizer to be flaky, and lose keys
- An honest ephemerizer should not make copies of its ephemeral private keys
- So…wouldn’t it be a disaster if it lost its keys when a customer needs to recover from backup?

The reason why it's not just pushing the problem

- You can achieve arbitrary robustness by using enough "flaky" ephemerizers!
  - Independent ephemerizers
  - Independent public keys

Use multiple ephemerizers!



Master keys

S1 Jan 7, 2013
S2 Jan 8, 2013
S3 Jan 9, 2013
…

Nonvolatile storage

Ephemerizer keys

P1 Jan 7, 2013
P2 Jan 8, 2013
P3 Jan 9, 2013
…

file

Exp 01/08/13

{K}S2

Encrypted
With K

Q1 Jan 7, 2013
Q2 Jan 8, 2013
Q3 Jan 9, 2013
…

Backup of keys

{S1}P1, {S1}Q1 Jan 7, 2013
{S2}P2, {S2}Q2 Jan 8, 2013
{S3}P3, {S3}Q3 Jan 9, 2013
…

Encrypted with G

Sysadmin secret

Summarizing

- Only need ephemerizer after a disaster
- Ephemerizer really scalable: millions of customers; rarely talks to any of them
- Ephemerizer only needs 10,000 public keys (one per day, 30 years) regardless of number of customers
- Easy to build an ephemerizer; generate private keys and do decryptions in protected hardware, never making copies

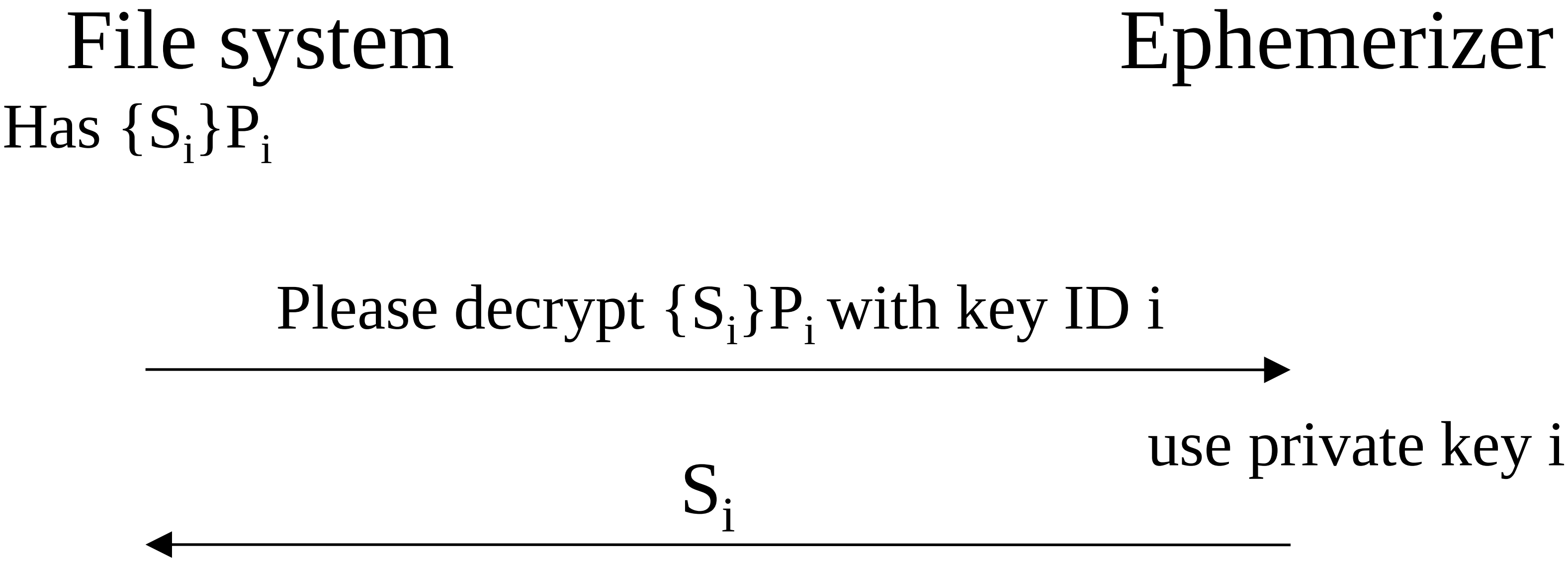
What if ephemerizer doesn’t destroy private key when it should?

- Then the file system can use a quorum scheme (k out of n ephemerizers)
  - Break master key into n pieces, such that a quorum of k can recover it
  - Encrypt each piece with each of the n ephemerizers’ public keys

Asking ephemerizer to decrypt

- Cool protocol for asking the ephemerizer to decrypt
  - Which gives no information to the ephemerizer!

What we want to accomplish



File system

Ephemerizer

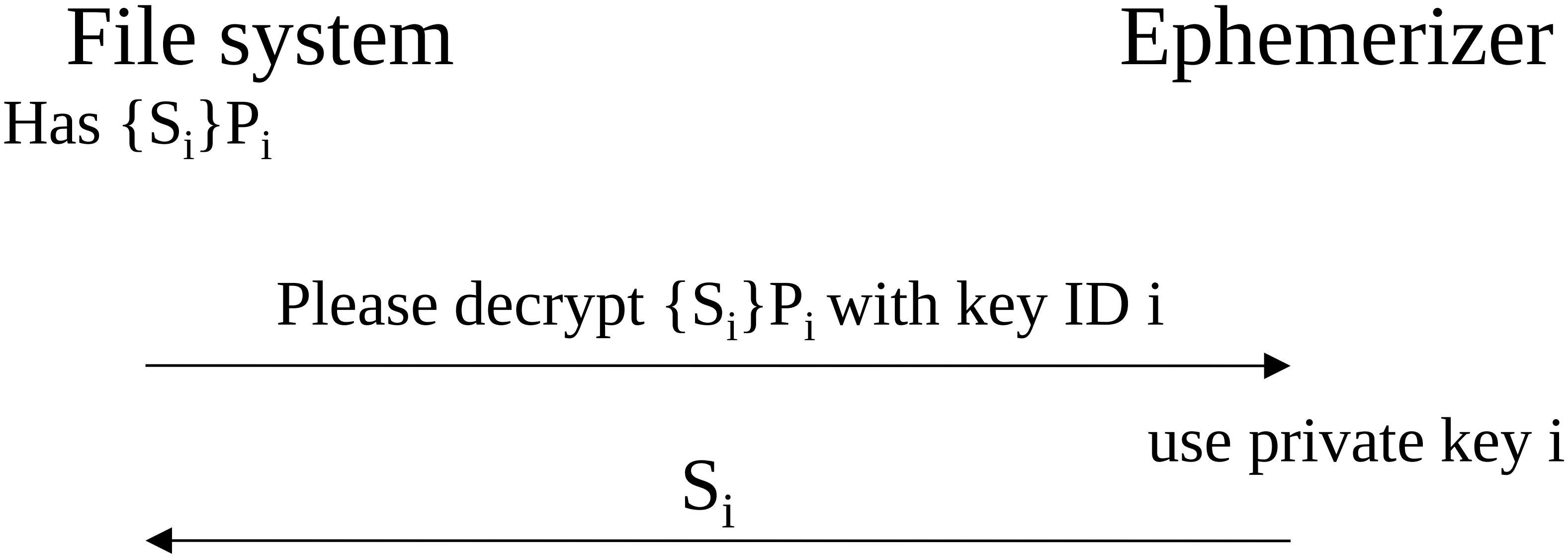
Has $\{S_i\}P_i$

Please decrypt $\{S_i\}P_i$ with key ID i

use private key i

S_i

What we want to accomplish



File system

Ephemerizer

Has $\{S_i\}P_i$

Please decrypt $\{S_i\}P_i$ with key ID i

use private key i

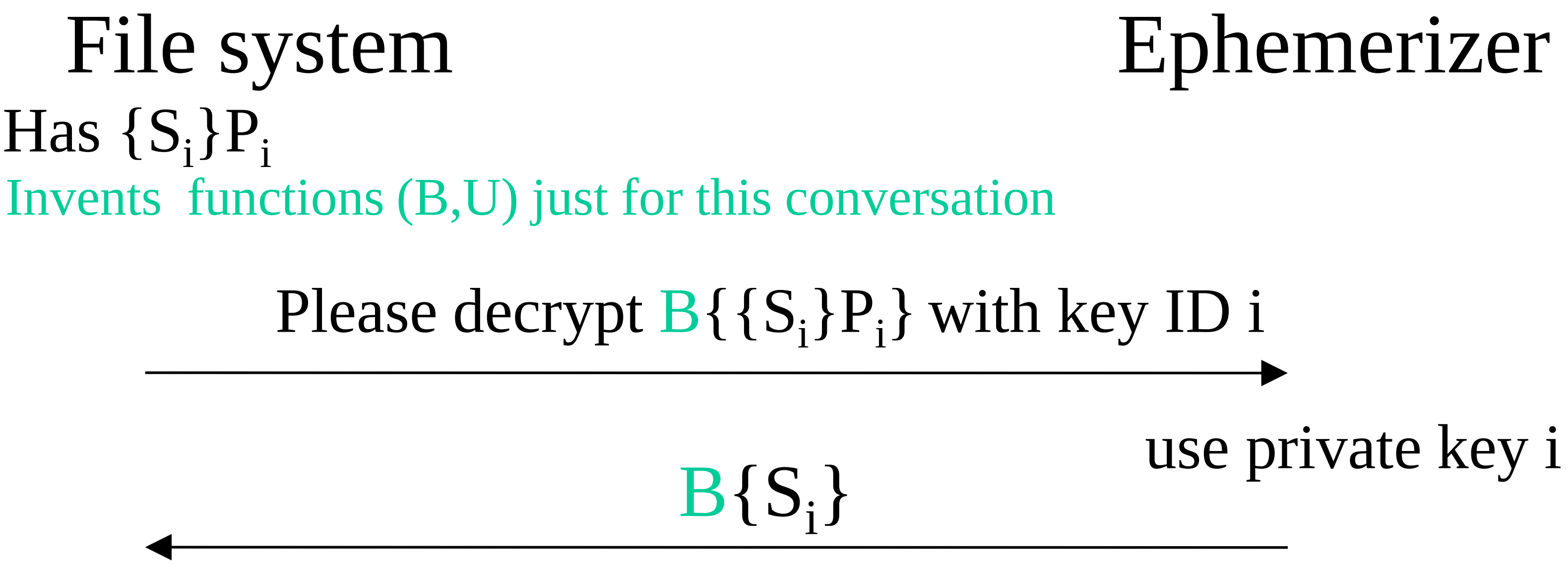
S_i

But we don’t want the Ephemerizer to see S_i

We’ll use “blind decryption”

- Same basic idea as blind signatures
- FS wants Ephemerizer to decrypt {Si}Pi with its private key #i
  - … Without seeing what it is decrypting
- FS chooses inverse functions blind/unblind (B, U)
- encrypts (blinds) with Blind Function, which commutes with Ephemerizer’s crypto
- Then applies U to unblind

Using Blind Decryption



File system

Ephemerizer

Has $\{S_i\}P_i$

Invents functions (B,U) just for this conversation

Please decrypt $B\{\{S_i\}P_i\}$ with key ID i

use private key i

$B\{S_i\}$

File system applies U to get S_i

Ephemerizer only sees $B\{S_i\}$

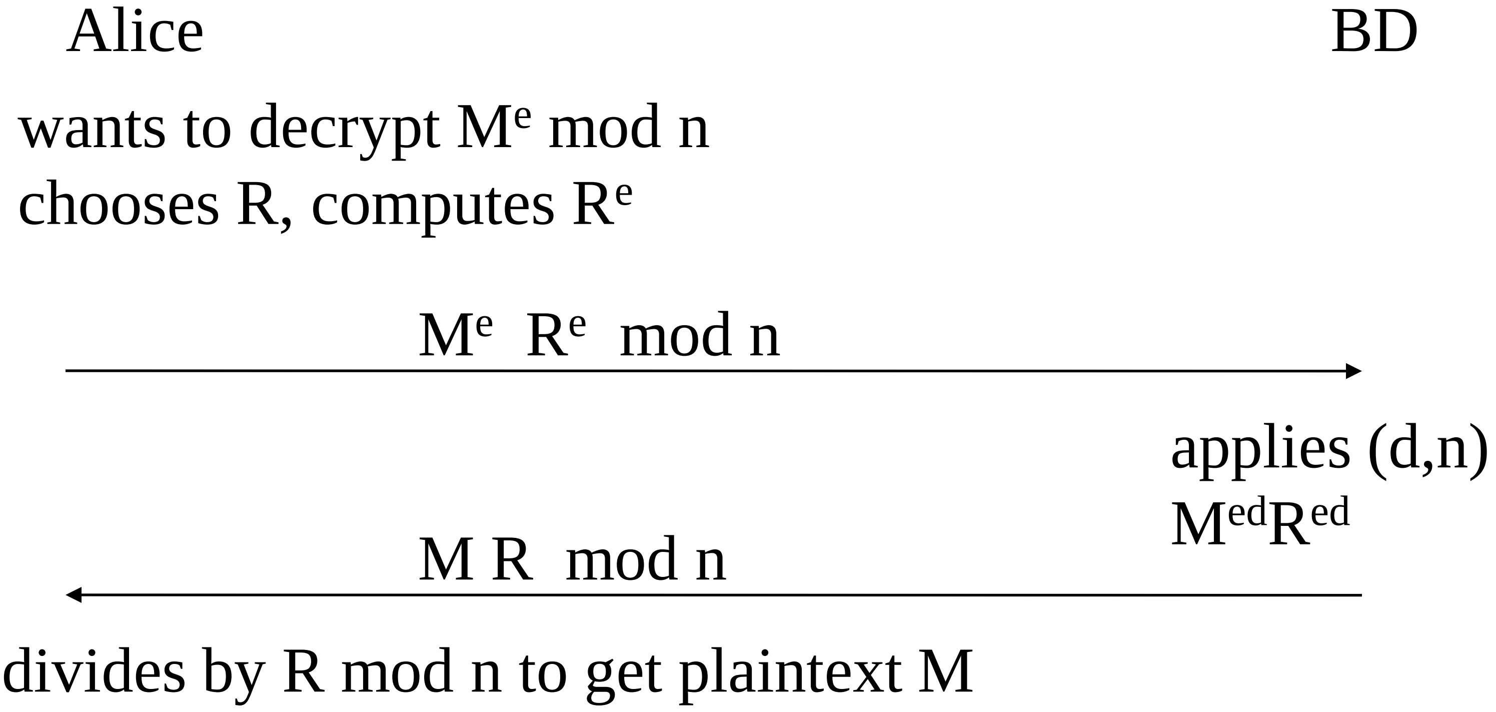
Non-math fans can take a nap

For you math fans…

Quick review of RSA

- Public key is (e,n). Private key is (d,n), where e and d are “exponentiative inverses mod n”
- That means X^{ed} mod n=X
- Encrypt X with public key (e,n) means computing X^e mod n

Blind Decryption with RSA, BD’s RSA PK=(e,n), msg=M



Alice

BD

wants to decrypt M^e mod n
chooses R, computes R^e

$M^e \; R^e$ mod n

applies (d,n)
$M^{ed}R^{ed}$

M R mod n

divides by R mod n to get plaintext M

Other functions work

- For instance, there’s a Diffie-Hellman version that works with elliptic curves
- But for intuition, enough to see one function…

Properties of our protocol

- Ephemerizer gains no knowledge when it is asked to do a decryption
- Protocol is really efficient: one IP packet request, one IP packet response
- No need to authenticate either side
- Decryption can even be done anonymously

OK, non-math fans can wake up now

Because of blind decryption

- The customer does not need to run its own Ephemerizers, or really trust the Ephemerizers very much
- Ephemeral key management can be outsourced

Running an ephemerizer

- A customer *could* run some of its own ephemerizers—they should be fairly inexpensive and easy to manage
- But a customer might not be able to have enough of them in enough geographic locations for true robustness during disasters
- So it’s nice to use really remote ones if necessary

Outer encryption on ephemerized backup keys

- We need a global secret G
- Otherwise, anyone that got the encrypted backups could ask the Ephemerizer to decrypt
- G could be something like a sysadmin password, held in the head of multiple system administrators

To recover from backup

- To retrieve the state of the file system after a disaster you need:
  - G
  - The encrypted backups of the keys
  - The encrypted backups of the data
  - Help from the ephemerizer

Interaction with Ephemerizer

- Only need to bother Ephemerizer once after a crash, for each expiration time (i.e., 10,000 decryptions)
- Rather than every time file system opens a file
- But we can actually make it only one decryption after a crash!

Another optimization

- Since the S’s are in a sequence…
- Make them derivable from each other, like with a one-way hash (or have file system store each successive S encrypted with previous S)
- That way, after a crash, only have to talk to Ephemerizer once!

Sometimes you don’t know when you want to delete a file

Examples

- Company severs relations with a client; destroys all files
    - Keys can be nested; use time-based keys in that client's folder
- Being sued; not allowed to delete anything; make makeup with custom key, then destroy custom key after suit done; key expirations will revert to time
- Spy: ship captured; tell ephemerizer not to decrypt with custom key

Custom Keys

- The ephemerizer’s time-based key can be shared by many clients
- With custom keys, you need to have the ephemerizers keep a key for each of your custom classes for you, so you can tell it when to delete it (or when to apply special authorization to use it for decryption)

Custom Keys

- Hopefully there would be a manageable number of these

Note: This isn’t DRM!

- DRM requires tamper-proof reader
- Ephemerization does not make DRM easier or harder

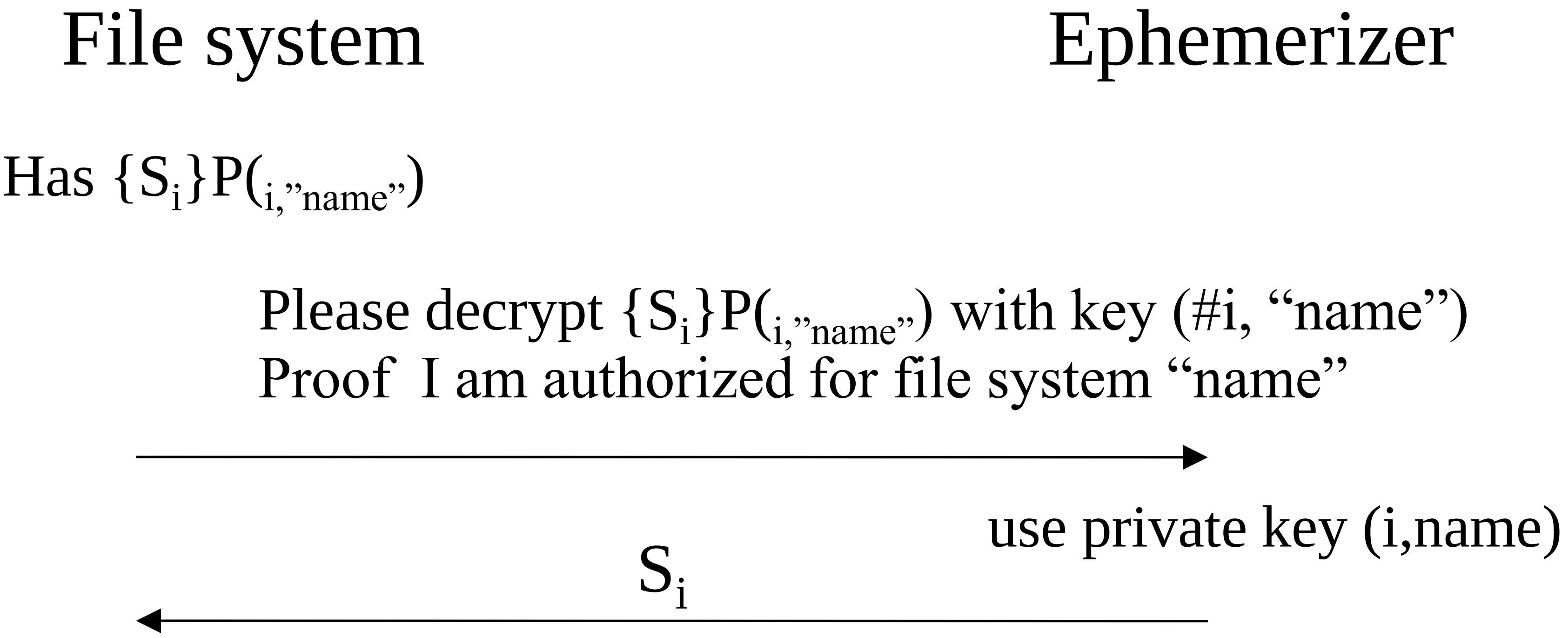
A subtle enhancement

- What if you fire system admin Fred?
- He might be able to find a backup of your keys, and he knows G
- Solution: Ephemerizer’s key for Jan 1 isn’t a single key, it’s a family of keys, some function of
  - file system owner name
  - Advertised Ephemerizer parameters for that date’s “public key”

Name-based public keys

- To “ephemerize” “Radia”’s January 1 key, take the ephemerizer’s advertised value P, parameterize it with “Radia” to get a public key $P_{\text{Radia/Jan1}}$
- Encrypt with public key $P_{\text{Radia/Jan1}}$
- If you need ephemerizer to decrypt, you have to also prove you authorized to speak for “Radia”
- That credential can be revoked through the PKI

Name-based Public Keys



File system

Ephemerizer

Has $\{S_i\}P(_{i,\text{''name''}})$

Please decrypt $\{S_i\}P(_{i,\text{''name''}})$ with key (#i, “name”)
Proof I am authorized for file system “name”

use private key (i,name)

S_i

To reduce slide clutter I left out blinding

- But of course you still need blinding
- In addition to being able to parameterize a key pair with a name

What functions work?

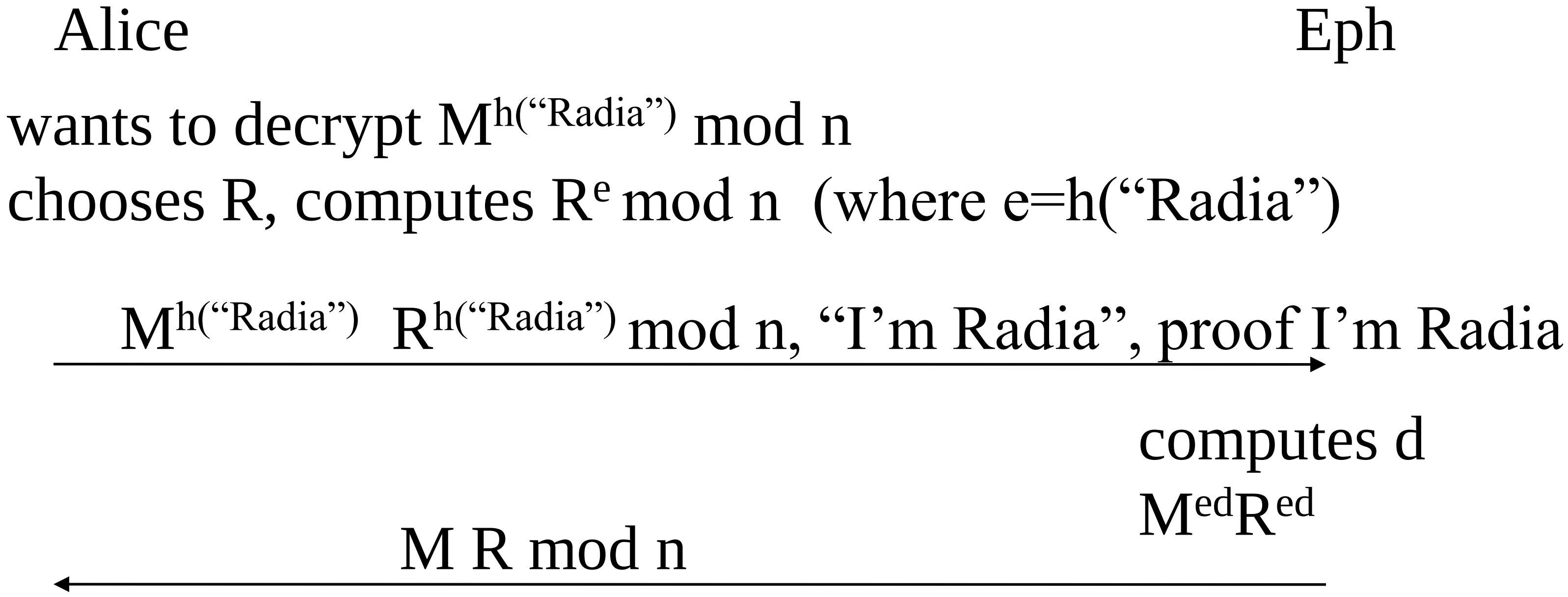
- The math of IBE (identity based encryption) (with separate “domain parameters” for each date)
- The Diffie-Hellman blindable math we omitted from these slides
- RSA variant which probably works…no known flaws (by me)…but easiest to explain…and maybe one of you can prove whether it is secure

Non-math fans can take a nap

RSA-based key blindable parameterizable families

- Jan 1 “public key” isn’t (e,n): it’s just “n”
- Ephemerizer advertises: (date, n)
- The RSA encryption key “Radia” uses for that date is (public exponent=h(“Radia”), n)
    - $S^{h(\text{“Radia”})}$ mod n
- The private key (known to the Ephemerizer) is knowledge of the factors of n (which enables the ephemerizer to compute exponentiative inverses mod n)

Blind Decryption with parameterized RSA



Alice

Eph

wants to decrypt $M^{h(\text{“Radia”})}$ mod n

chooses R, computes R^{e} mod n (where e=h(“Radia”)

$M^{h(\text{“Radia”})}$ $R^{h(\text{“Radia”})}$ mod n, “I’m Radia”, proof I’m Radia

computes d

$M^{ed}R^{ed}$

M R mod n

divides by R to get plaintext M

OK, non-math fans can wake up now

Another interesting (I hope) issue

- How to build an ephemerizer out of a dirt-cheap smart card
  - With limited storage, but attached to general purpose computer
- Smart card remembers two secret keys: current one and “next one”: K_n and K_{n+1}
- It generates public key pairs, encrypts the private key with K_n, and stores it on computer

How to forget one of the private keys

- Read in each private key (except the one you want to delete), one by one
- Decrypt with K_n
- Encrypt with K_{n+1}
- Store $\{S\}K_{n+1}$
- Forget K_n
- Generate K_{n+2}

New (small) topic

What if laws keep changing?

- Rather than file system keeping track of law that says this type of file must be retained for n years
- Have ephemerizer key based on (creation date, legal class) rather than expiration date
- Have ephemerizer destroy private key at the appropriate time

What if you get sued?

- You aren’t allowed to delete anything until the suit resolves
- So, you ask each ephemerizer to make you a (single) custom key (so if 3 ephemerizers, public keys P,Q,Z)
- You do a backup of all the (unexpired) master keys, “ephemerized” with P, Q, Z.
- After the suit is resolved, tell the ephemerizers to discard the private keys P, Q, Z.
- And the master keys go back to the original expirations

People kept wanting “on-demand” delete

- And I kept arguing that it was not useful, and wouldn’t be scalable

People kept wanting “on-demand” delete

- And I kept arguing that it was not useful, and wouldn’t be scalable
- But then I realized how to do it

People kept wanting “on-demand” delete

- And I kept arguing that it was not useful, and wouldn’t be scalable
- But then I realized how to do it
- And think it’s a really bad idea

People kept wanting “on-demand” delete

- And I kept arguing that it was not useful, and wouldn’t be scalable
- But then I realized how to do it
- And think it’s a really bad idea
- And it’s useful to see both how to do it, and why it’s a bad idea

On demand delete

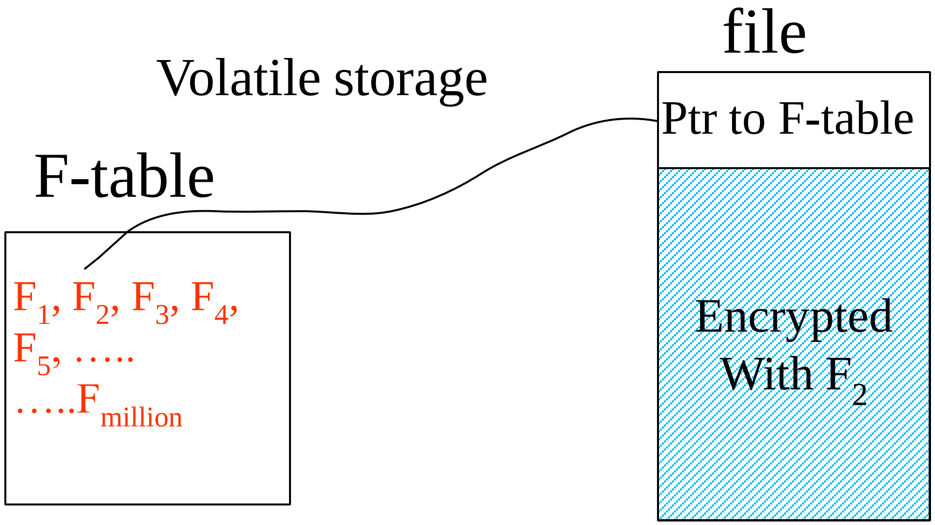
On-demand delete

- The previous design assumes
  - Key manager keeps one key for each expiration time
  - At file creation, you have to know its expiration
- What if you want to do on-demand delete?
- But then you wouldn’t be able to share keys…if you throw away a key, all files encrypted with the same key go away
- On the surface, seems much harder

Ephemerizer state

- Needs to keep two public keys for each customer file system
  - current public key
  - previous public key

File system with F-table



Volatile storage

F-table

F_1, F_2, F_3, F_4, F_5, …..

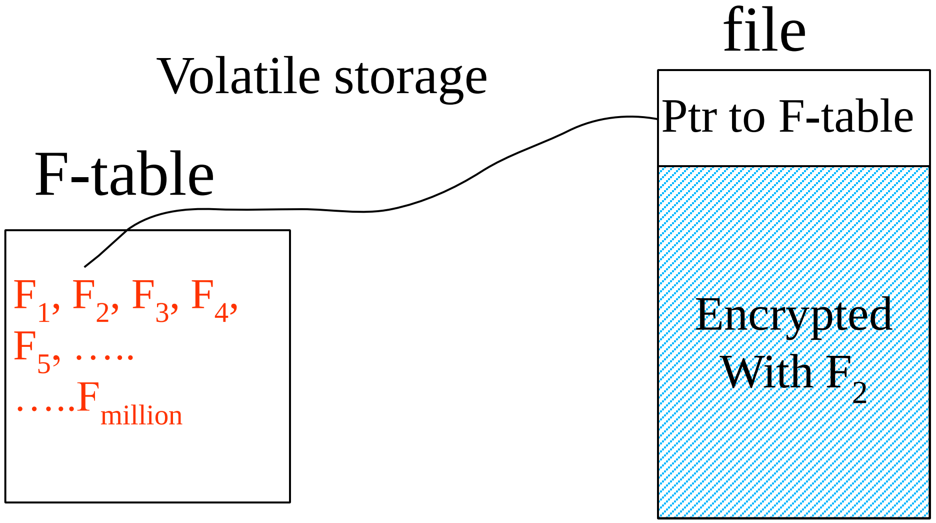
…..$F_{million}$

file

Ptr to F-table

Encrypted With F_2

File system with F-table



Volatile storage

F-table

F_1, F_2, F_3, F_4, F_5, …..

…..$F_{million}$

file

Ptr to F-table

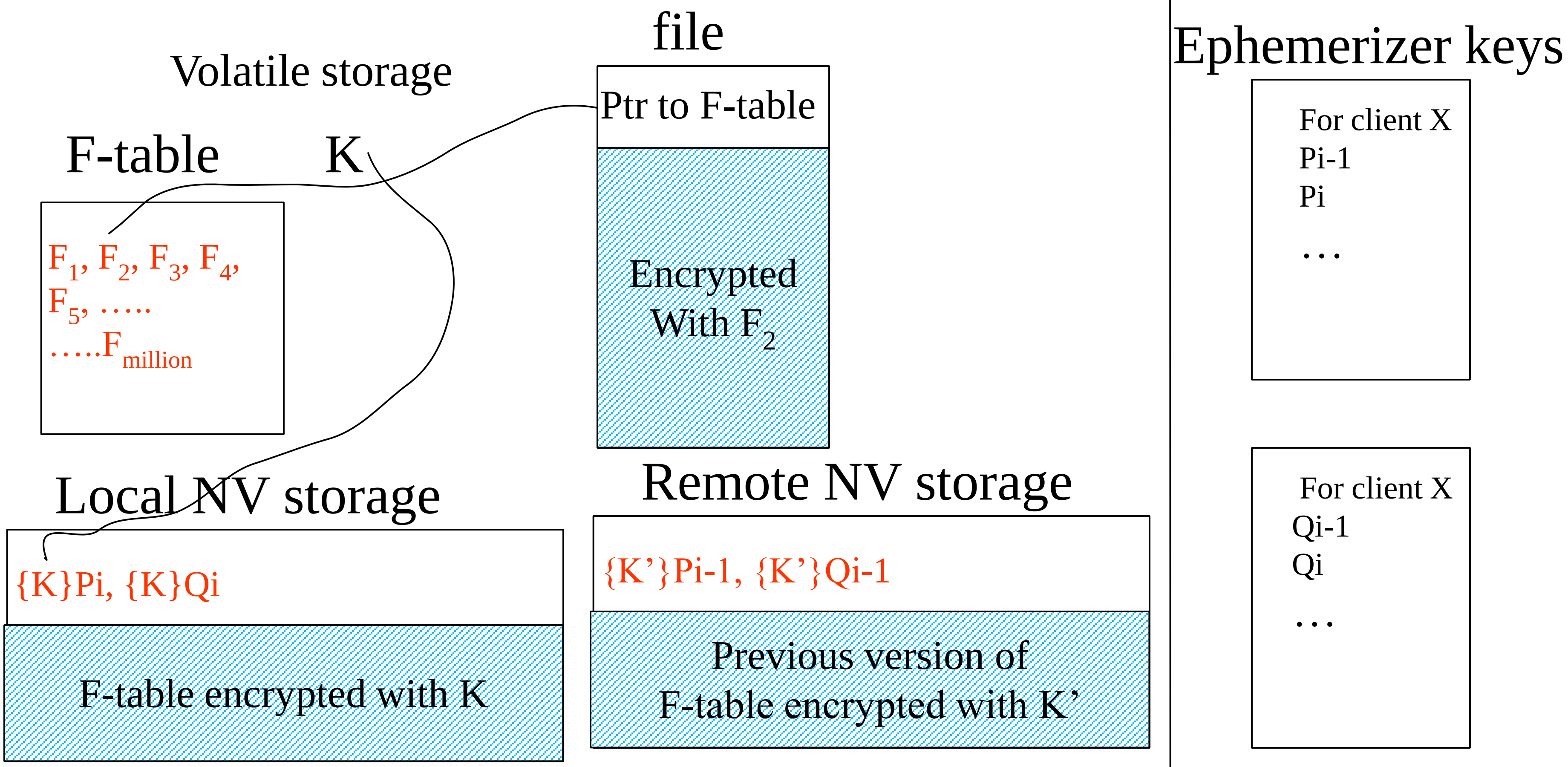
Encrypted With F_2

Modify F-table when you assure-delete a file

Or create a new file

F-table has key for each file…if a million files, a million keys

File system with F-table



Volatile storage

F-table

K

F_1, F_2, F_3, F_4, F_5, …..

…..$F_{million}$

file

Ptr to F-table

Encrypted With F_2

Ephemerizer keys

For client X
Pi-1
Pi
…

Local NV storage

{K}Pi, {K}Qi

F-table encrypted with K

Remote NV storage

{K'}Pi-1, {K'}Qi-1

Previous version of F-table encrypted with K'

For client X
Qi-1
Qi
…

In theory…

- Ephemerizer could roll over keys on a schedule
- But then a week-long power failure could be a disaster

So what’s wrong?

My concern

- Suppose you change P’s every week
- Suppose you find out that the file system was corrupted a month ago
- And that parts of the F-key database were corrupted, without your knowledge
- You can’t go back

Why isn’t pre-determined expiration time as scary?

- If file system is not corrupted when a file is created, and the file is backed up, and the S-table is backed up, you can recover an unexpired file from backup
- Whereas with the on-demand scheme, if the file system gets corrupted, all data can get lost

Summary

- Use multiple independent ephemerizers with independent keys; ephemerizer keys need not be backed up
- Keys can be nested (expiring keys in a folder with custom keys)
- Not DRM (completely orthogonal to DRM)
- Time-based is probably the most useful

Back to main presentation

Another topic

- An example of where adopting something from a different standards organization was the wrong thing

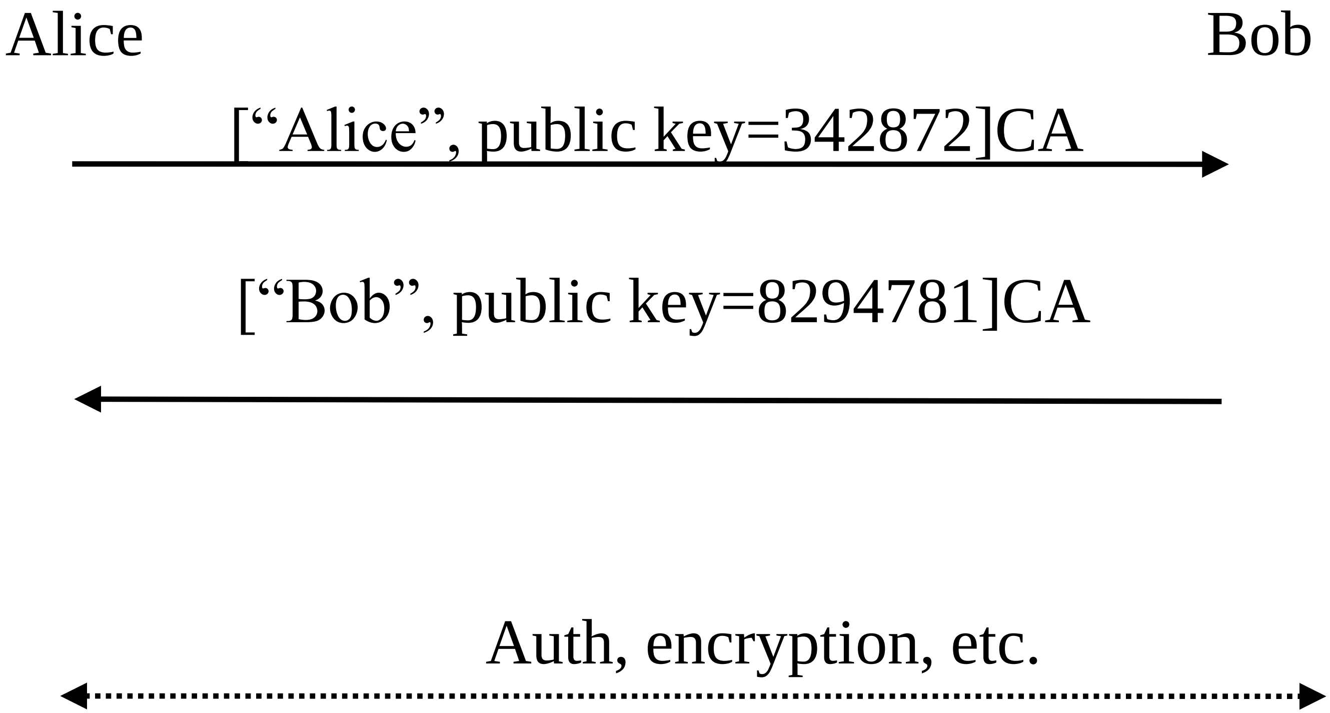
Example: PKIX (certificate format)

First some background

What’s a certificate?

- Public keys
  - Two numbers (public, private)
  - If Bob knows Alice’s public key, and Alice knows her private key, she can prove to Bob she is Alice
- How does Bob know Alice’s public key?
- Bob trusts something known as a CA (Certification authority) to sign a statement that “Alice’s public key is 297483927489”

Key Distribution - Public Keys



Alice

Bob

[“Alice”, public key=342872]CA

[“Bob”, public key=8294781]CA

Auth, encryption, etc.

Certificate Format

- So, a certificate maps a name to a public key
- IETF’s PKIX working group decided to base certificates on X.509 (an ITU standard)
- Why should it matter?

The problem with X.509

- X.509 maps an X.500 name to a public key
- What’s an X.500 name?
- A perfectly reasonable hierarchical namespace
  - Example: C=countryname, O=organizationname, OU=organizationunitname, CN=commonname

So, X.509 would have been fine…

- If Internet protocols (and Internet users) were using X.500 names
- But they don’t…they use DNS names like labs.examplecompanyname.com

- So what good is something that maps some string that the application (and user) is unfamiliar with, to a public key?

Example

- Human types “foo.com” (a DNS name embedded in a URL)
- Site sends certificate with an X.500 name

Example

- Human types “foo.com” (a DNS name embedded in a URL)
- Site sends certificate with an X.500 name
  - C=US
  - O=AtticaPrison
  - OU=DeathRow
  - OU=ParticularlyVilePrisoners
  - CN=Horrible Person

Example

- Human types “foo.com” (a DNS name embedded in a URL)
- Site sends certificate with an X.500 name
- One strategy used by some implementations: Ignore name in certificate, but validate the math of the signature

Example

- Human types “foo.com” (a DNS name embedded in a URL)
- Site sends certificate with an X.500 name
- One strategy used by some implementations: Ignore name in certificate, but validate the math of the signature
- What security does that give?

Example

- What security does that give?
- The warm fuzzy feeling that SOMEONE paid Verisign for a certificate…

There are (at least) 3 kludgy ways of encoding a DNS name into a PKIX cert

- Invent a new category “DC” instead of “C” or “O” or “OU”, for “domain component”, and encode something like labs.intel.com as
  - DC=com; DC=intel; DC=labs
- Use the “alternate name” field in the PKIX certificate to put the DNS name
- Use the bottom of the X.500 name (the “common name”; “CN=“) to put in the DNS name

Are 3 ways better than one?

- No!!!
- Suppose a CA enforces that you own the DNS name in the “common name” field, but allows you to put whatever you want into the alternate name
- This is a security problem waiting to happen

Human names: Another difficult problem

- I’d love humans to be authenticated with certificates and keys
- But what do we do for names?
- If you work at a big company, you likely have the problem of figuring out which John Smith you want to talk to….
  - john.smith or johnny.smith, or john.q.smith

And all the math in the world won’t help us really know what/who to trust

- Imposters have posed as airline pilots, doctors
- Or even had proper credentials (e.g., Bernie Madoff)
- And we don’t tend to search for things by DNS names anyway…

Another topic

Another topic

- “The Internet is great for keeping everyone informed”

Another topic

- “The Internet is great for keeping everyone informed”
- Alternative view: Too many choices allows us to focus on just what we want to hear

Another topic

- “The Internet is great for keeping everyone informed”
- Alternative view: Too many choices allows us to focus on just what we want to hear
- Also, anyone can say anything, and will have an audience who will believe them
- Actually scary…polarizes society

Some things that can’t possibly work

Some things that can’t possibly work ….but do!

Some things that can’t possibly work ….but do!

- Wikipedia

Some things that can’t possibly work ….but do!

- Wikipedia
- Internet search

Some things that can’t possibly work ….but do!

- Wikipedia
- Internet search
- Online shopping

Some things that can’t possibly work ….but do!

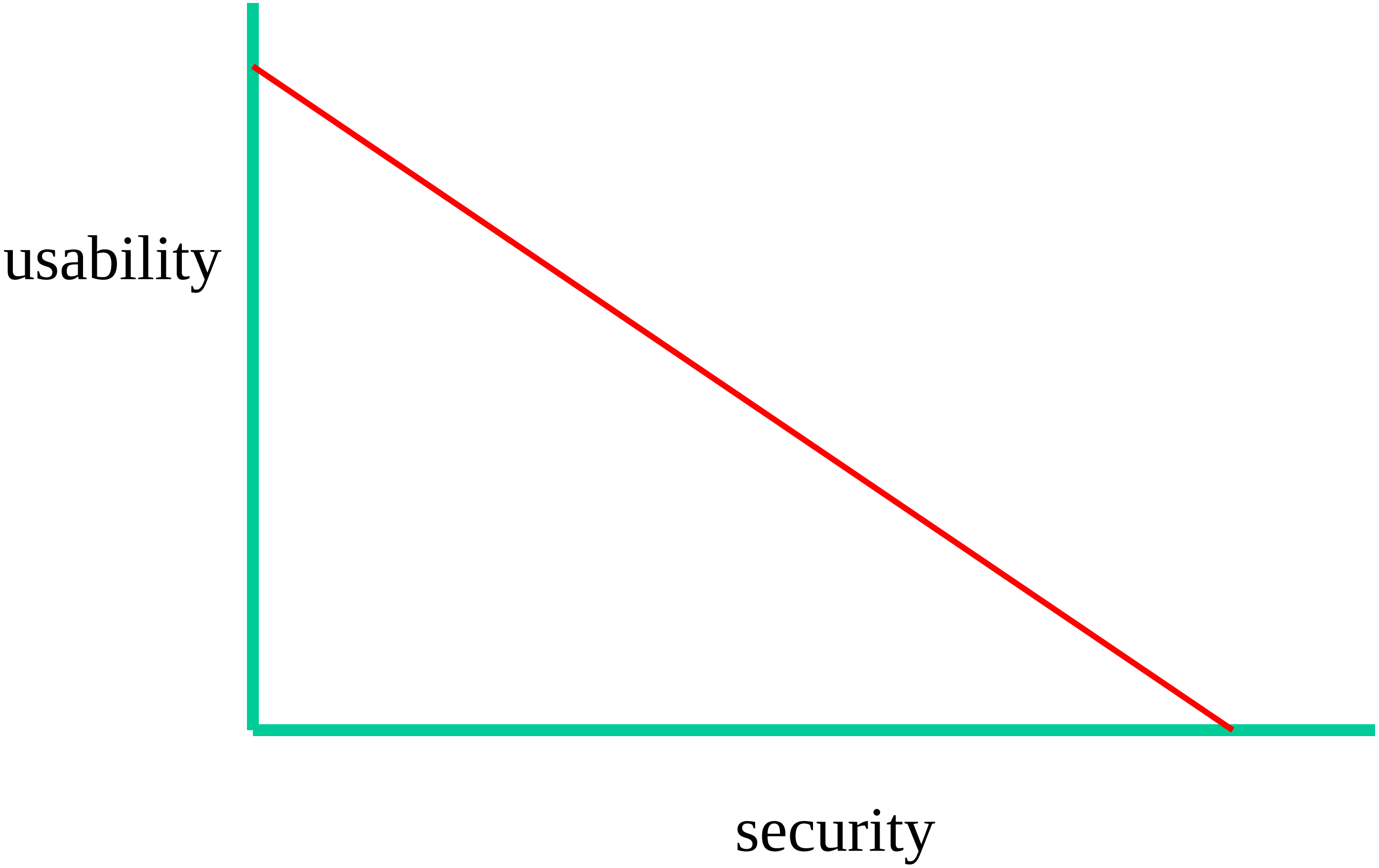
- Wikipedia
- Internet search
- Online shopping
- Ebay

And something that should be easy…

- User Authentication

It’s common to have to trade off usability vs security

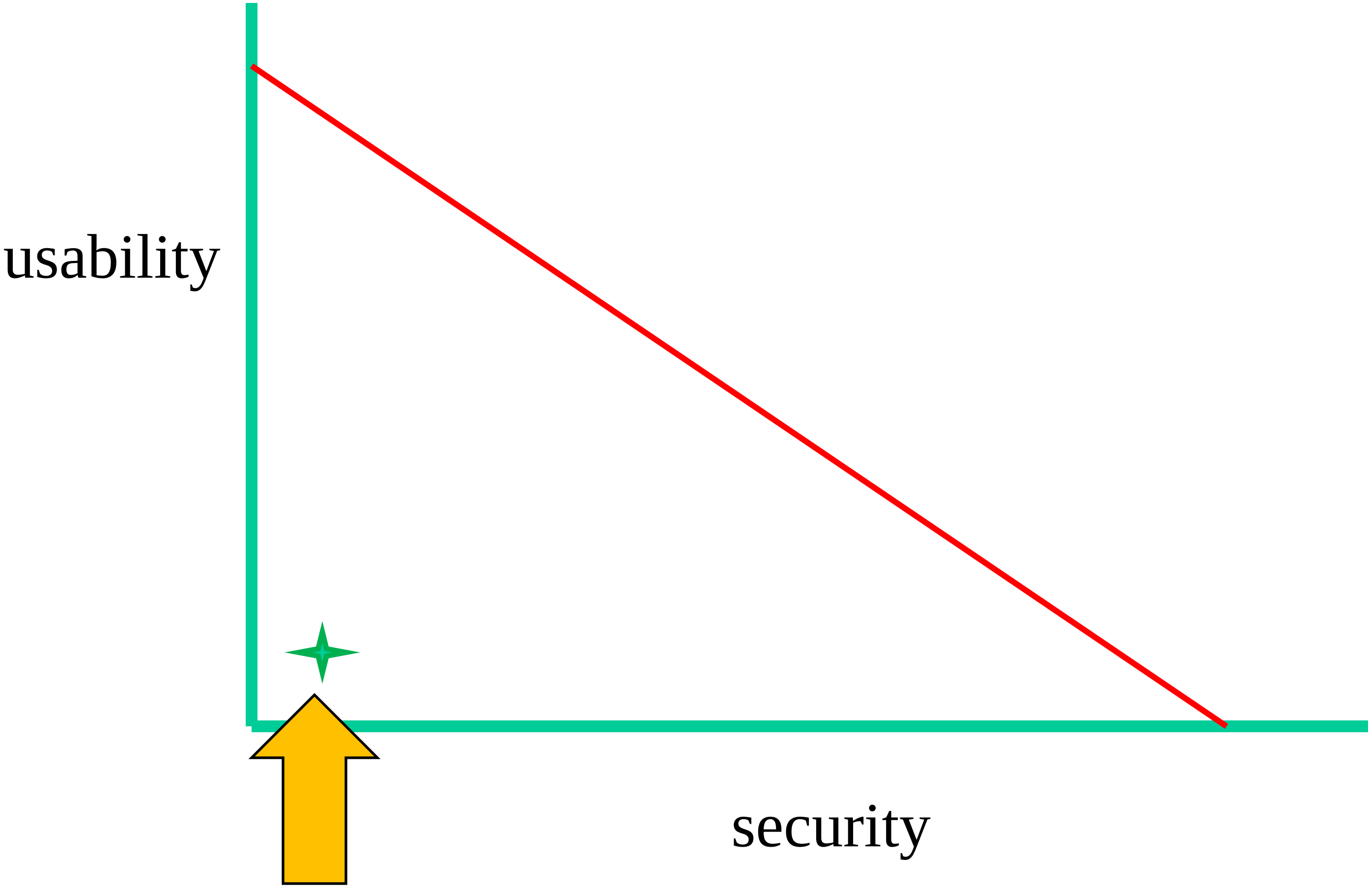
It’s common to have to tradeoff usability vs security



It’s common to have to tradeoff usability vs security

- But we’ve managed to make user authentication both
  - Maximally insecure, and
  - Maximally unusable

Unusable and insecure!



User authentication

- Every site has different rules for usernames and passwords
  - At least n characters, no more than x characters, must have at least one letter, one number, one special character, must not contain anything but letters and numbers, ….

User authentication

“Sorry, but your password must contain an uppercase letter, a number, a haiku, a gang sign, a hieroglyph, and the blood of a virgin.” (unknown author)

Annoying rules that add nothing to security

- Must change password at least every n days
- Must not reuse a password
- If you forget your password, you can’t reset it to the one you were using (that you temporarily forgot)
- These sorts of rules actually lower security!
- But they are written into “best practices”, so companies must do them

User authentication

- I do not want to hear…

User authentication

- I do not want to hear…

  “We need better user training”

Humans

- “Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed, but they are sufficiently pervasive that we must design our protocols around their limitations.”
  - Network Security: Private Communication in a Public World

Single Signon

- That is how things ought to work
- “The Network is the Computer”
- One strategy (I don’t like) is an “identity provider”
  - Alice authenticates to identity provider X
  - When Alice wants to talk to server S, she requests X to send her a short term “token”
    - “X vouches that whoever sends you this letter is Alice”

Problems with Identity Provider Approach

- Security problems
  - (Identity Provider can impersonate you anywhere)
  - Stolen token allows thief to impersonate you
- Trust problem (is there really one IdProv that will be trusted by all users, all servers?)
- Availability problem (if it is down, user can’t talk to anyone)
- Privacy problem (IdProv knows every site you visit)

Is there a solution?

- I think so…
- For instance…
- User has smart card with private key
- When user sets up an online account, instead of providing “username/password”, sends public key
- You don’t need any certificates!

Most of the time…

- All you are proving is that you are the same user that created the account

Things to consider

- User needs to be able to use multiple different devices
- If smart card is stolen, has to be unusable by thief, and replaceable to real person
- I think these can be solved

Just a bit of ranting…

Buying something

- Scenario: Buy something from a merchant you haven’t bought from recently
- All prepared with your info, credit card, etc.
- It asks you for your email address…

You’re a returning user!

- Type your username and password!

You’re a returning user!

- Type your username and password
- Of course you can’t remember it, so…

You’re a returning user!

- Type your username and password
- Of course you can’t remember it, so…
  - you manage to find “recover username”

You’re a returning user!

- Type your username and password
- Of course you can’t remember it, so…
  - you manage to find “recover username”
  - suddenly you are in a Monty Python movie
    - Answer the following questions three:
      - Telephone number
      - Address
      - Mother’s maiden name

My wish for a New Rule

- It should be no more onerous to be a returning user than a new user

Security questions for password/username recovery

- Favorite sports team
- 2nd grade teacher’s name
- Pet’s name
- Father’s middle name
- My middle name

My Wish for a New Rule

- Security questions must be specifiable by the user
- I’d say “or selectable from a very large list”, but I’m sure they can come up with an arbitrarily long list of questions I can’t answer

Keeping customer information

- I do not want to do “single click ordering”
- I do not mind typing in my address
- I do not mind typing in my credit card number
- Merchants insist on keeping all of this information
- And eventually this information gets stolen

My Wish for a New Rule

- After a merchant is paid, any subset of information about a customer (including all of the information) must be expunged by the merchant at the customer's request

Some security research I've done

- Assured delete
  - The ability to store things in a cloud or on a server with an "expiration date"
  - The cloud can make arbitrary many copies so the data doesn't get accidentally lost
  - After the expiration date, it's unrecoverable, even though the backups still exist
  - No performance overhead, easy to manage, easy to build

Some security research I’ve done

- Networks resilient to malicious failure of trusted components
  - Malicious switch can lie in routing alg, flood the network with garbage, do everything perfectly but discard data from one source, …
  - My thesis (1988) showed how to build a modest sized network (say 1000 nodes), that guaranteed 2 nodes can communicate provided at least one honest path connects them
- Recent work: Extending the guarantees to large hierarchical networks

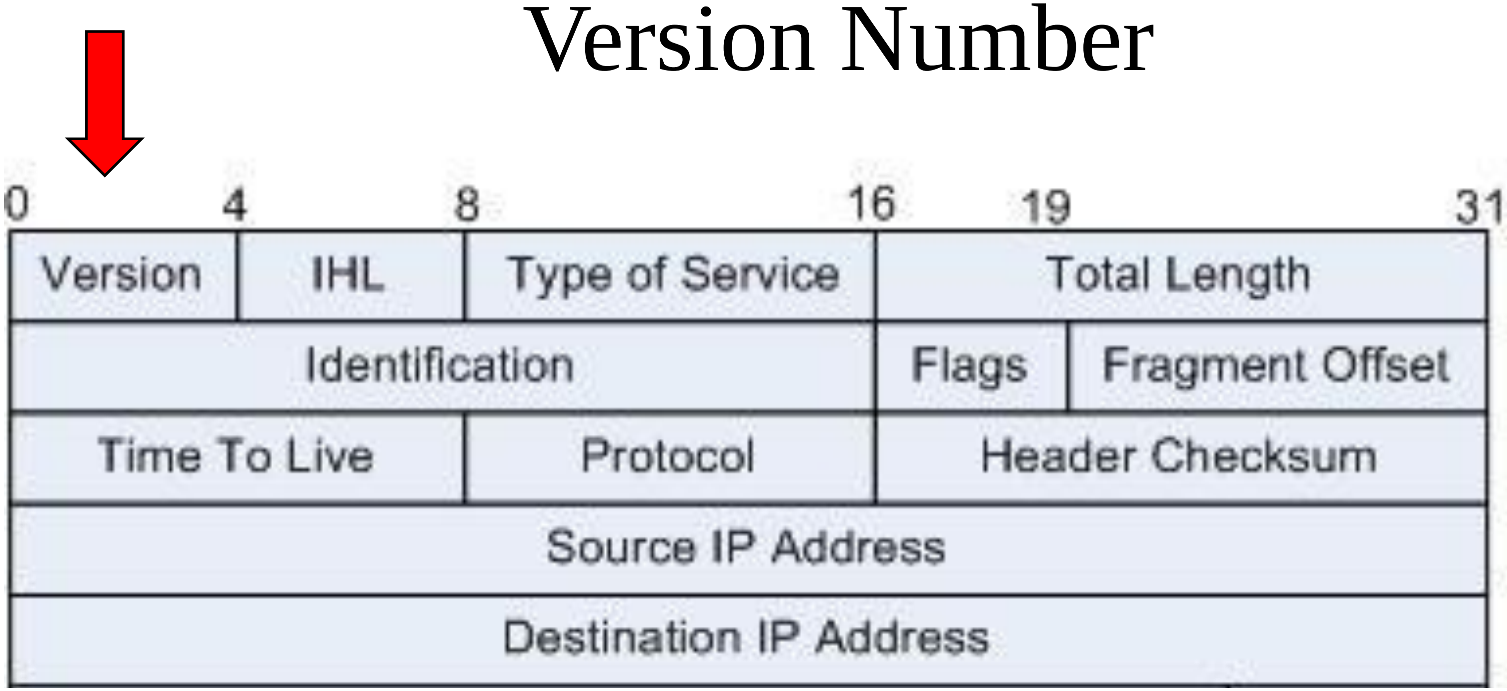
Another topic…infected machines

- There are so many ways to get your machine infected
- In the old days…this was done by bored teenagers looking for attention
- Now it’s criminals
  - Real business for controlling bots and launching DDOS attacks (Distributed Denial of Service), sending spam, having lots of computation for password cracking, etc.
  - Criminals don’t want you to know your machine is infected

Protocol Folklore

- Obvious stuff everyone gets wrong

Version Number



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

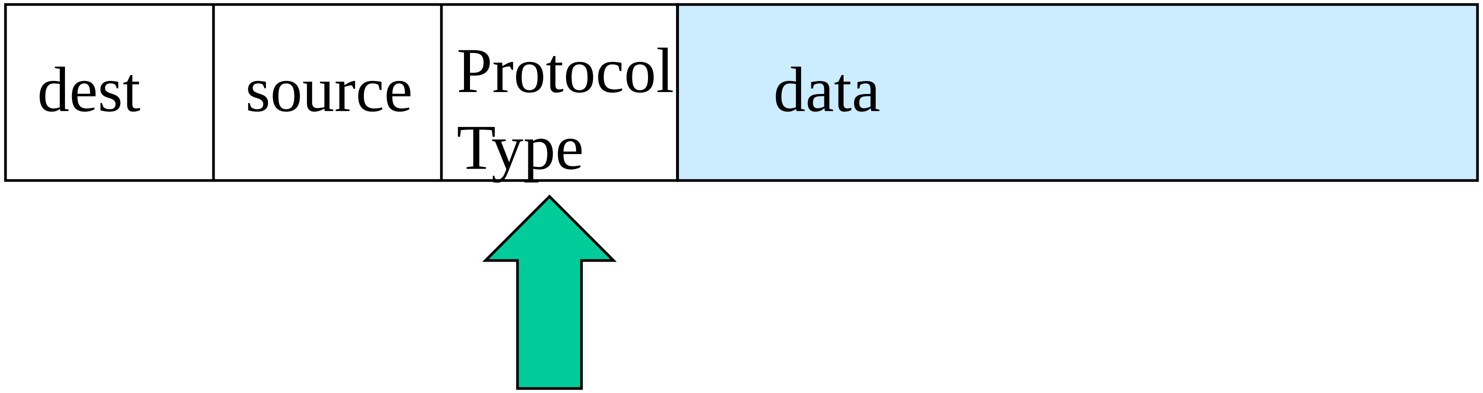
What’s a Version Number?

- Philosophical question:
  - what is “new version” vs “new protocol”?

What I think makes sense

- Envelope says what the protocol is (how to parse the packet)
  - Ethernet: Ethertype
  - IP: Protocol Type
  - TCP/UDP: port

Ethernet Protocol Type



dest

source

Protocol Type

data

What I think makes sense

- Envelope says what the protocol is (how to parse the packet)
- If differentiate based on protocol type, then it’s a new protocol
- If differentiate based on version number, then it’s a new version of the same protocol

If differentiate based on version number

- You can’t just say “write this value into this field
- You have to say “Look at the version number, and if it’s not your version, then drop the packet”!

Version

- Nobody seems to do this right
- IP, IKEv1, SSL don’t say 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…

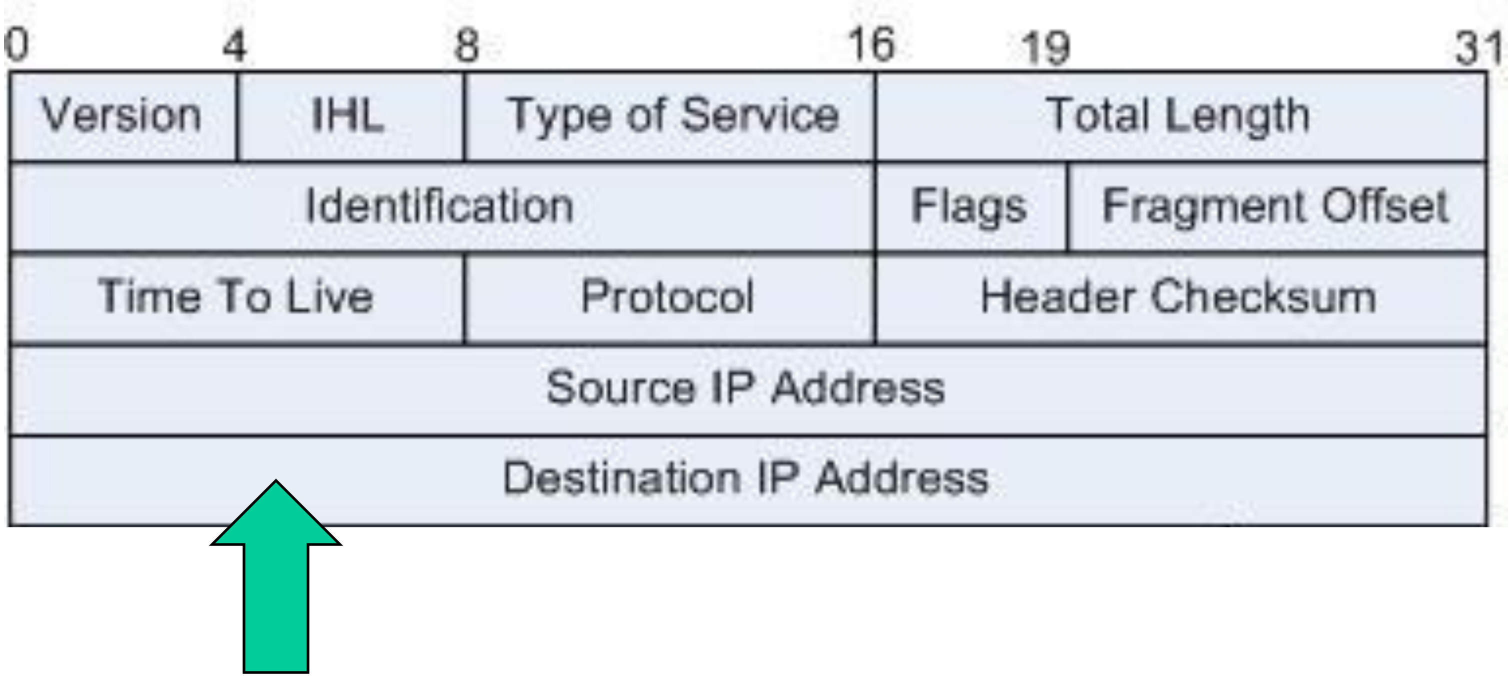
Example: Hello Timer

- 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

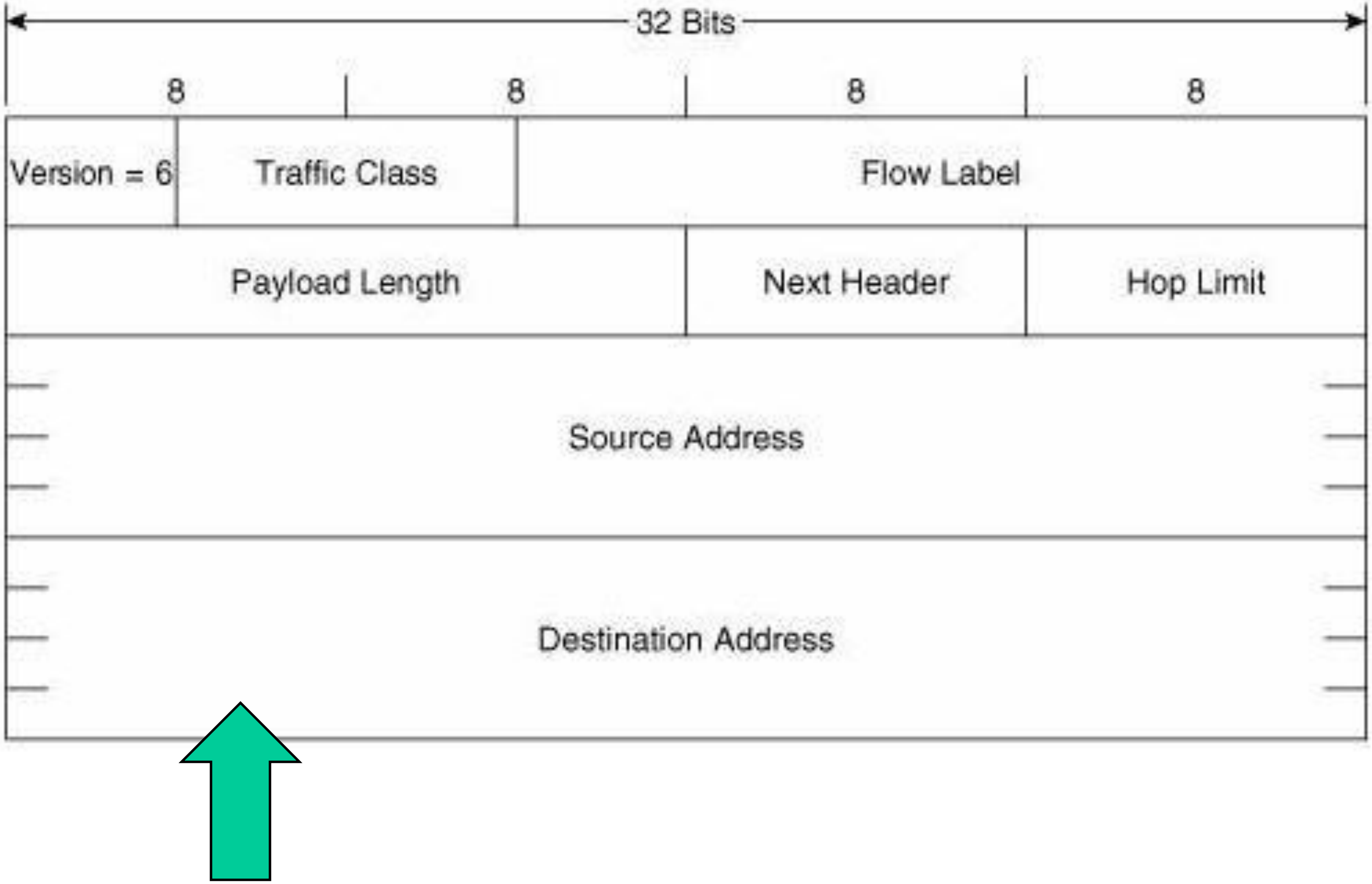
- Store-and-forward vs cut-through
- Cut through can start after the forwarding decision is made
- What field do you need to see for forwarding decision?

IPv4 header



<table>
<tr><th>0</th><th>4</th><th>8</th><th>16</th><th>19</th><th>31</th></tr>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="3">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragment Offset</td></tr>
<tr><td colspan="2">Time To Live</td><td>Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
<tr><td colspan="6">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

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

Thank you!