Network Protocols: Myths, Missteps, and Mysteries

Radia Perlman
EMC

radia@alum.mit.edu

Network Protocols

- A lot of what we all “know”

Network Protocols

- A lot of what we all “know”….is not true!!

How networking tends to be taught

- Memorize these standards documents, or the arcane details of some implementation that got deployed
- Nothing else ever existed
  - Except possibly to make vague, nontechnical, snide comments about other stuff

My philosophy on teaching (and books)

- Look at each conceptual problem, like how to autoconfigure an address
- Talk about a bunch of approaches to that, with tradeoffs
- Then mention how various protocols (e.g., IPv4, IPv6, Appletalk, IPX, DECnet, …) solve it

But some professors say…

- Why is there stuff in here that my students don't "need to know"?

Standards…

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

Standards Bodies…

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

What about “facts”?

- What if you measure A vs B?
- What are you actually measuring?...one implementation of A vs one implementation of B
- ***So don’t believe something unless you can figure out a plausible property of the two protocols that would make that true***

Buzzwords

- Useful for impressing customers
- They don’t necessarily mean anything.
- They traumatize most engineers
- People who are sure they know the definition disagree, so…
- Be *way* more specific when talking between engineers, or when thinking about a problem

Think Critically

- Don’t believe everything you hear
  - Or even read in textbooks
- Don’t repeat things you don’t understand!

This field is really confusing

- "Common knowledge"
  - Need IP+Ethernet because IP is "layer 3" and Ethernet is "layer 2"
  - Security is built into IPv6, but is just an add-on to IPv4
  - SDN is revolutionary stuff

An example of something confusing

The story of Ethernet

- What is Ethernet?
- How does it compare/work with IP?
- People talk about “layer 2 solutions” vs “layer 3 solutions”. What’s that about?

So, first we need to review network “layers”

- ISO credited with naming the layers
- They defined 7 layers
- It’s just a way of thinking about networks

Perlman's View of ISO Layers

- 1: Physical

Perlman's View of ISO Layers

- 1: Physical
- 2: Data link: (neighbor to neighbor)

Perlman's View of ISO Layers

- 1: Physical
- 2: Data link: (neighbor to neighbor)
- 3: Network: create path, forward data (e.g., IP)

Perlman's View of ISO Layers

- 1: Physical
- 2: Data link: (neighbor to neighbor)
- 3: Network: create path, forward data (e.g., IP)
- 4: Transport: end-to end (e.g., TCP, UDP)

Perlman's View of ISO Layers

- 1: Physical
- 2: Data link: (neighbor to neighbor)
- 3: Network: create path, forward data (e.g., IP)
- 4: Transport: end-to end (e.g., TCP, UDP)
- 5 and above:

Perlman's View of ISO Layers

- 1: Physical
- 2: Data link: (neighbor to neighbor)
- 3: Network: create path, forward data (e.g., IP)
- 4: Transport: 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?

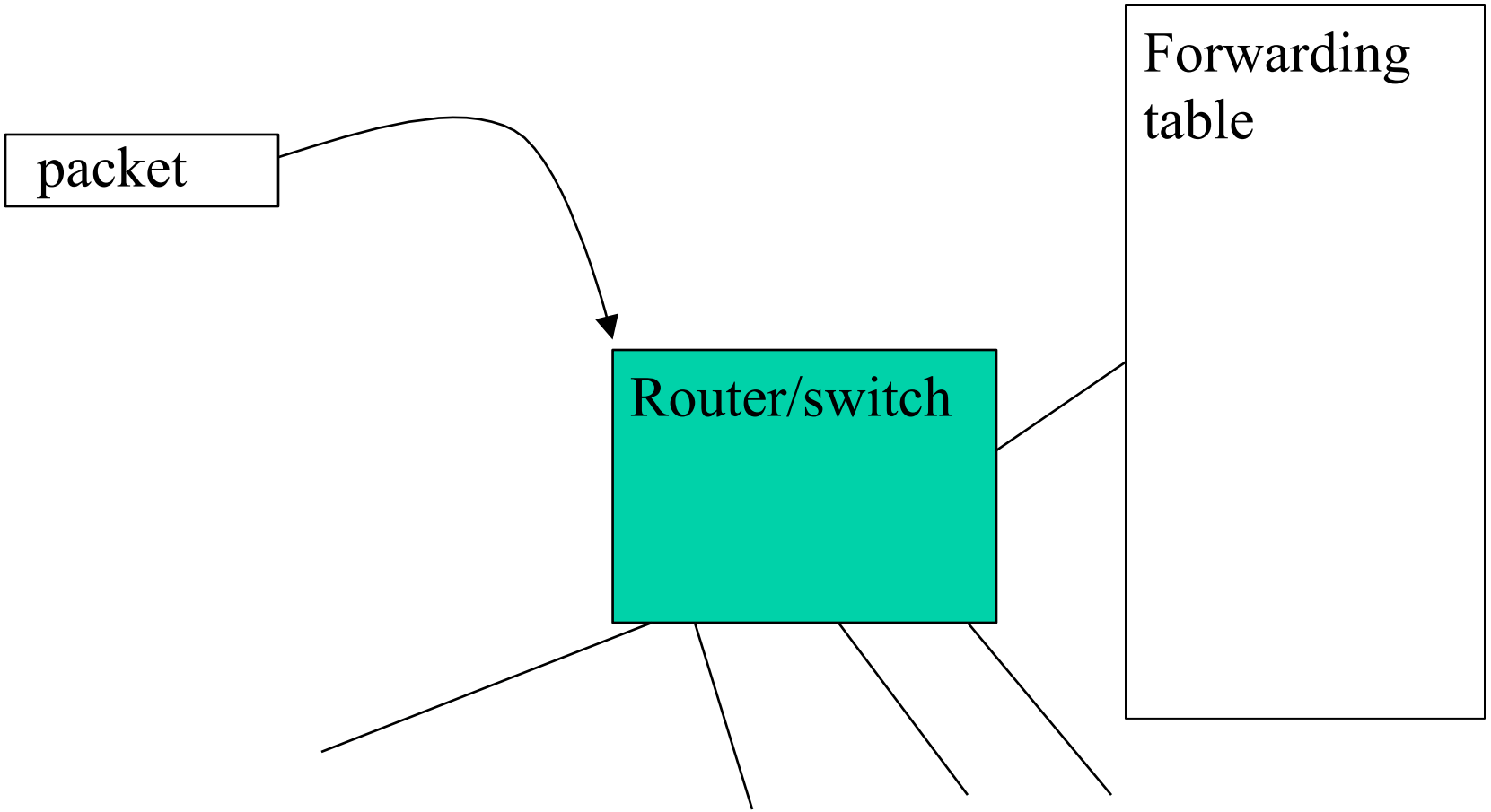
So…why are we forwarding Ethernet packets?

- Ethernet was intended to be layer 2
- Just between neighbors – not forwarded
- What exactly is Ethernet?
- No way to understand it without seeing the history

Back then…

- I was the designer of layer 3 of DECnet
  - the routing protocol I designed was adopted by ISO and renamed IS-IS
- Layer 3 calculates paths, and forwards packets
- Layer 2 just marked beginning and end of packet, and checksum (links between two nodes)

Router/Bridge/Switch

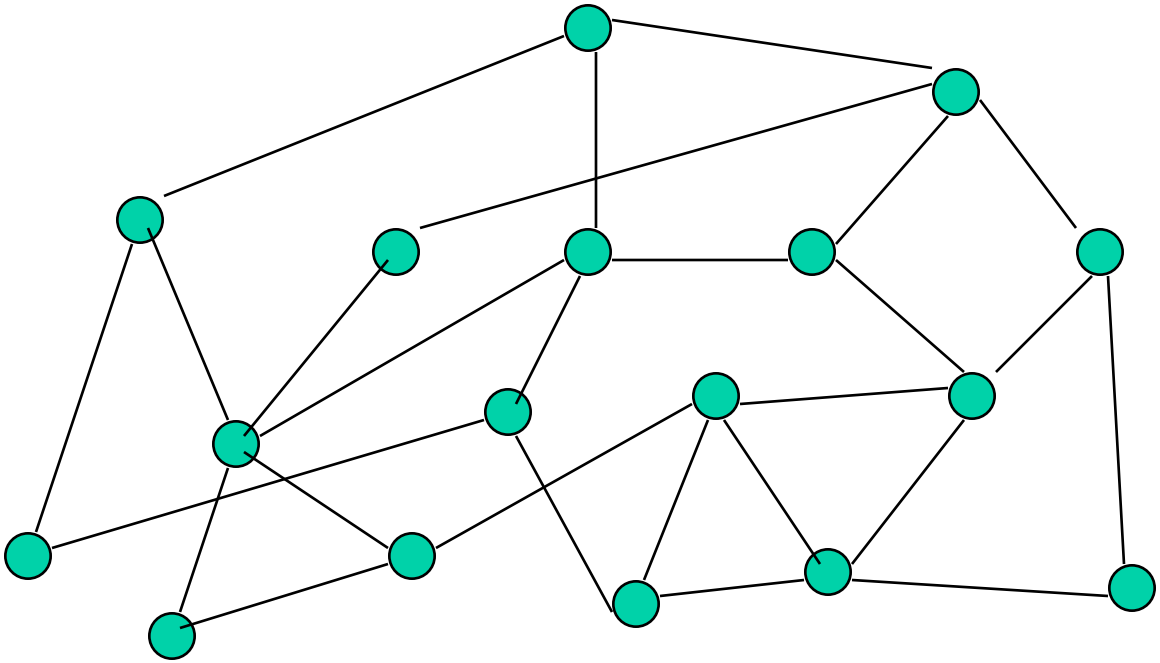


Computing the Forwarding Table

Computing the Forwarding Table

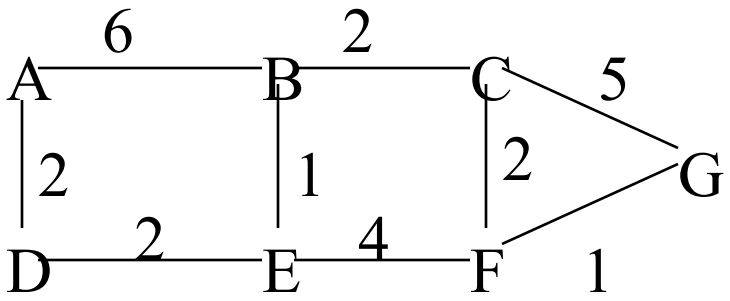
- Could be done with a central node
  - ATM, Infiniband, …
- Or with a distributed algorithm

Distributed Routing Algorithms



Distributed Routing Protocols

- Rtrs exchange info
- Use it to calculate forwarding table



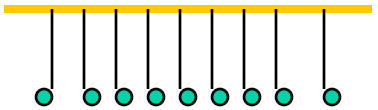
A	B	C	D	E	F	G
B/6	A/6	B/2	A/2	B/1	C/2	C/5
D/2	C/2	F/2	E/2	D/2	E/4	F/1
	E/1	G/5		F/4	G/1	

Back to history

- I was doing layer 3
- Then along came Ethernet

Original 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



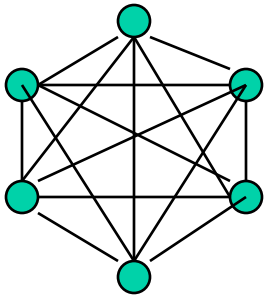
- Lots of papers about goodput under load only about 60% or so because of collisions
- Limited in # of nodes (maybe 1000), distance (kilometer or so)

I saw Ethernet as a new type of link

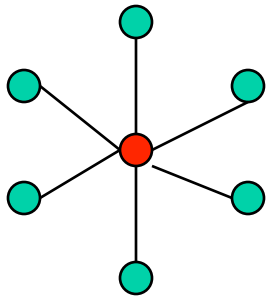
- I had to modify the routing protocol to accommodate this type of link
- For instance, the concept of “pseudonodes” and “designated routers” so that instead of n^2 links, it’s n links with n+1 nodes

Pseudonodes

Instead of:



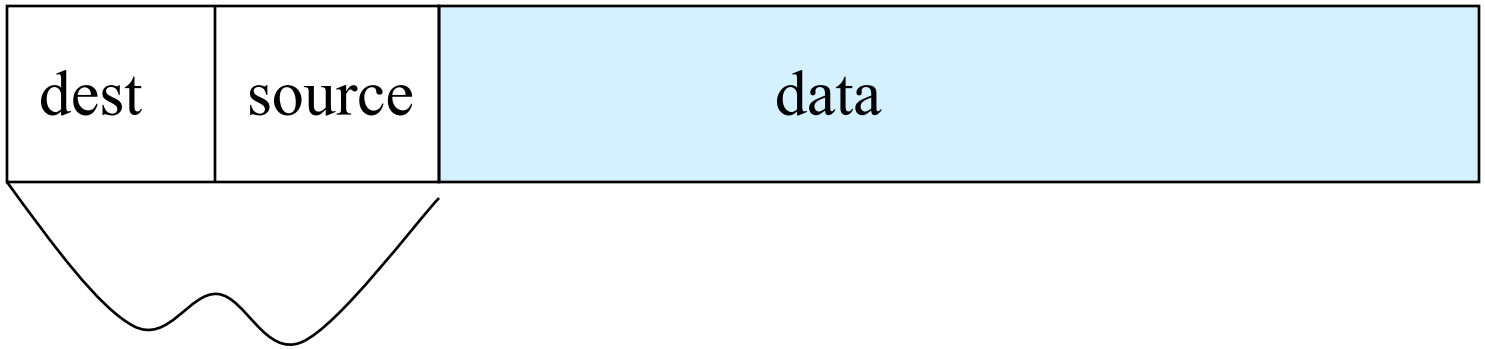
Use pseudonode



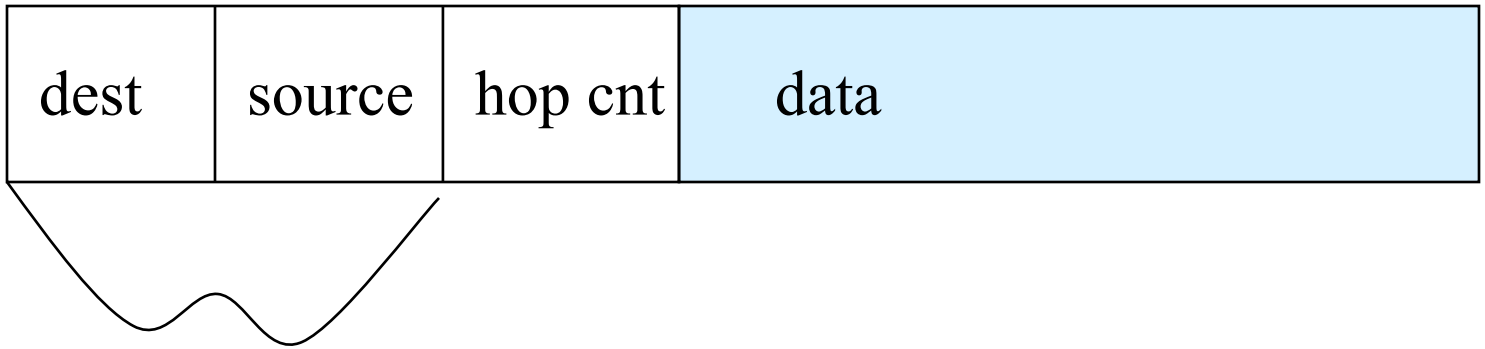
But Ethernet was a link in a network, not a network

- I wish they'd called it "Etherlink"

Ethernet packet



Layer 3 Packet



It's easy to confuse Ethernet with layer 3

- It looks sort of the same
- No hop count field…
- Flat addresses (no way to summarize a bunch of addresses in a forwarding table)
- But it never occurred to the Ethernet inventors that anyone would be forwarding an Ethernet packet

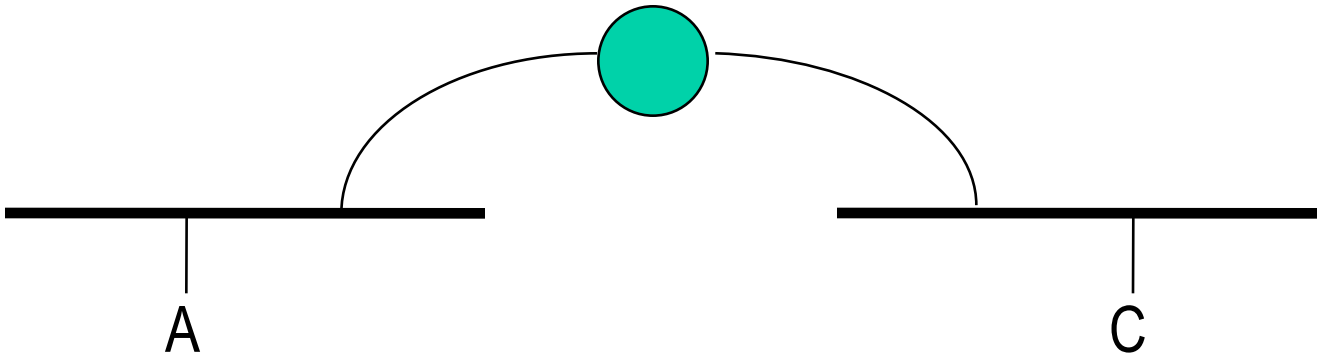
So…why are we forwarding Ethernet packets?

How Ethernet evolved from CSMA/CD to spanning tree

- People got confused, and thought Ethernet was a network (layer 3) instead of a link (layer 2)
- Built apps on Ethernet, with no layer 3
- Router can't forward without the right envelope
- I tried to argue…

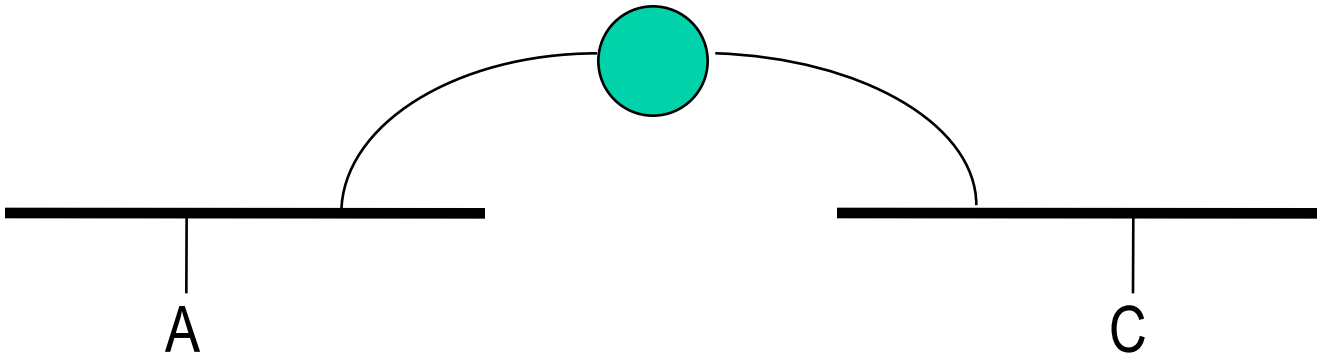
Problem Statement (from about 1983)

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



Problem Statement (from about 1983)

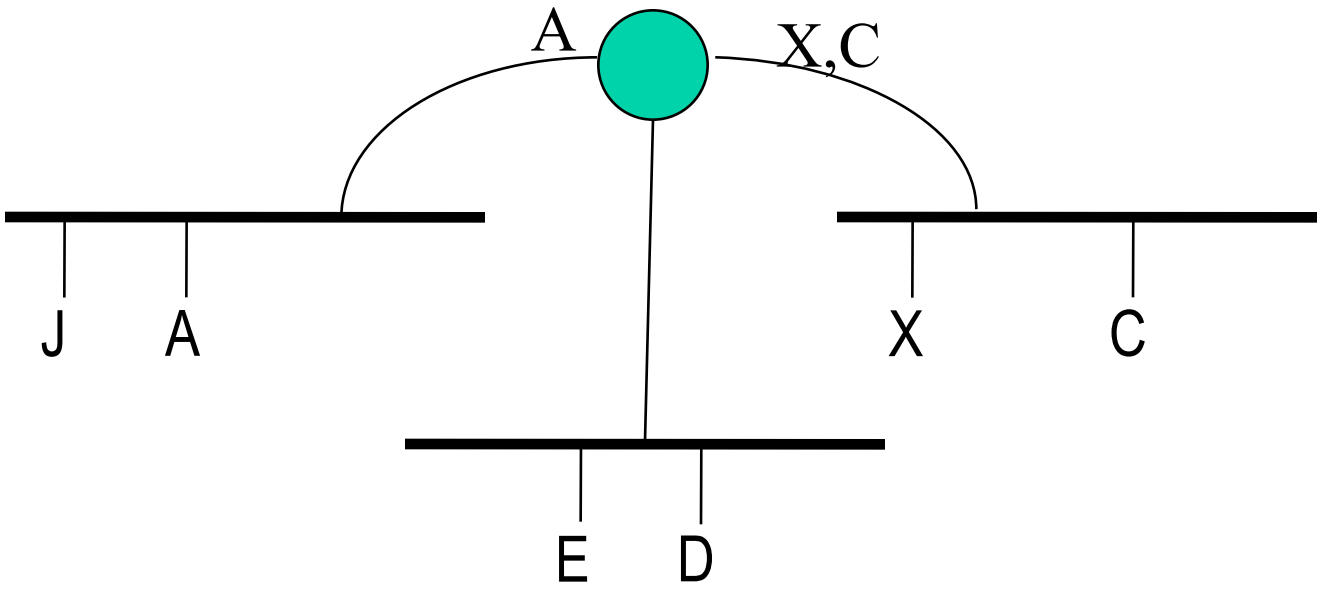
Need something that will sit between two Ethernets, and let a station on one Ethernet talk to another



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

Basic concept

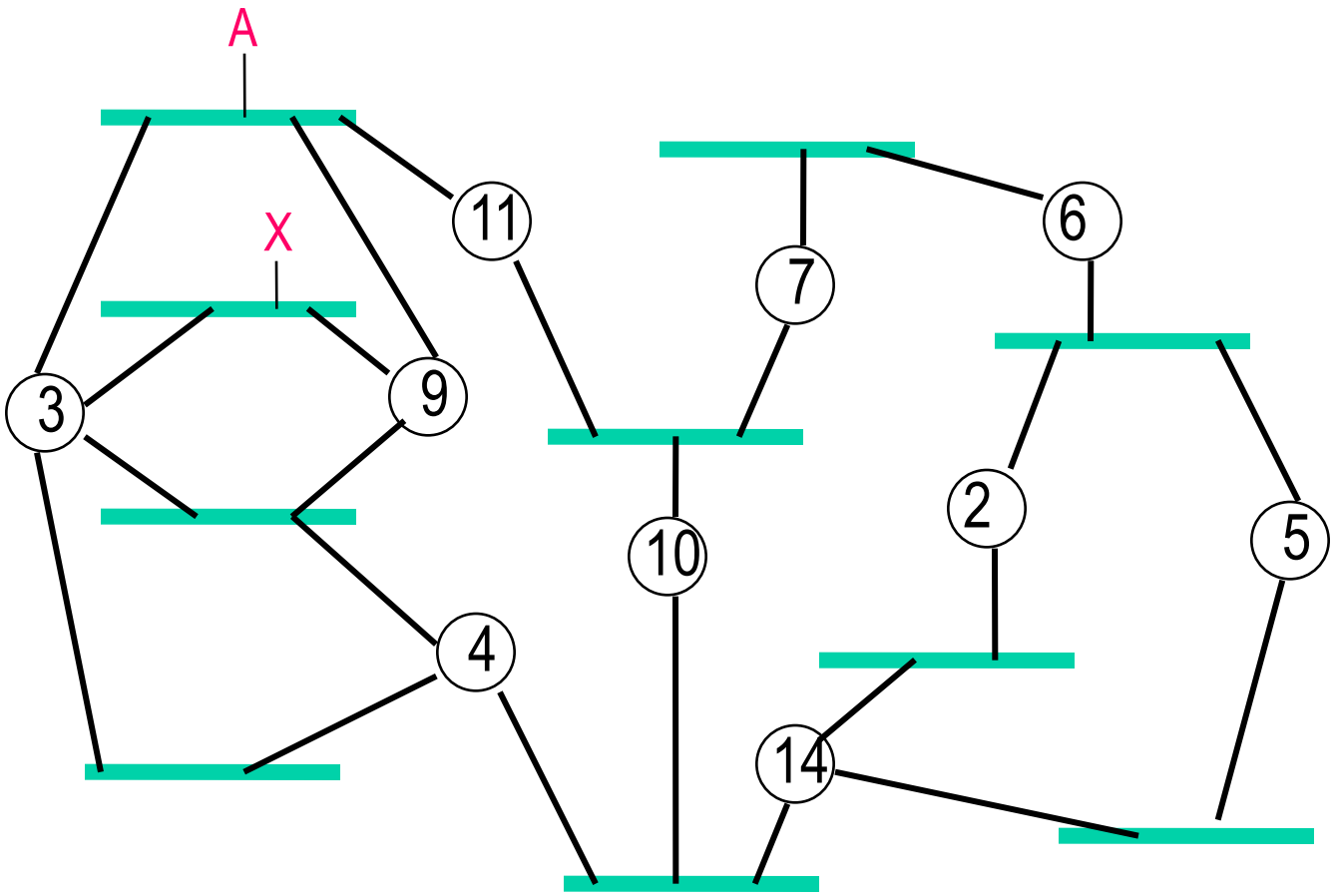
Listen promiscuously
Forward on other ports, based on learned (source, port)



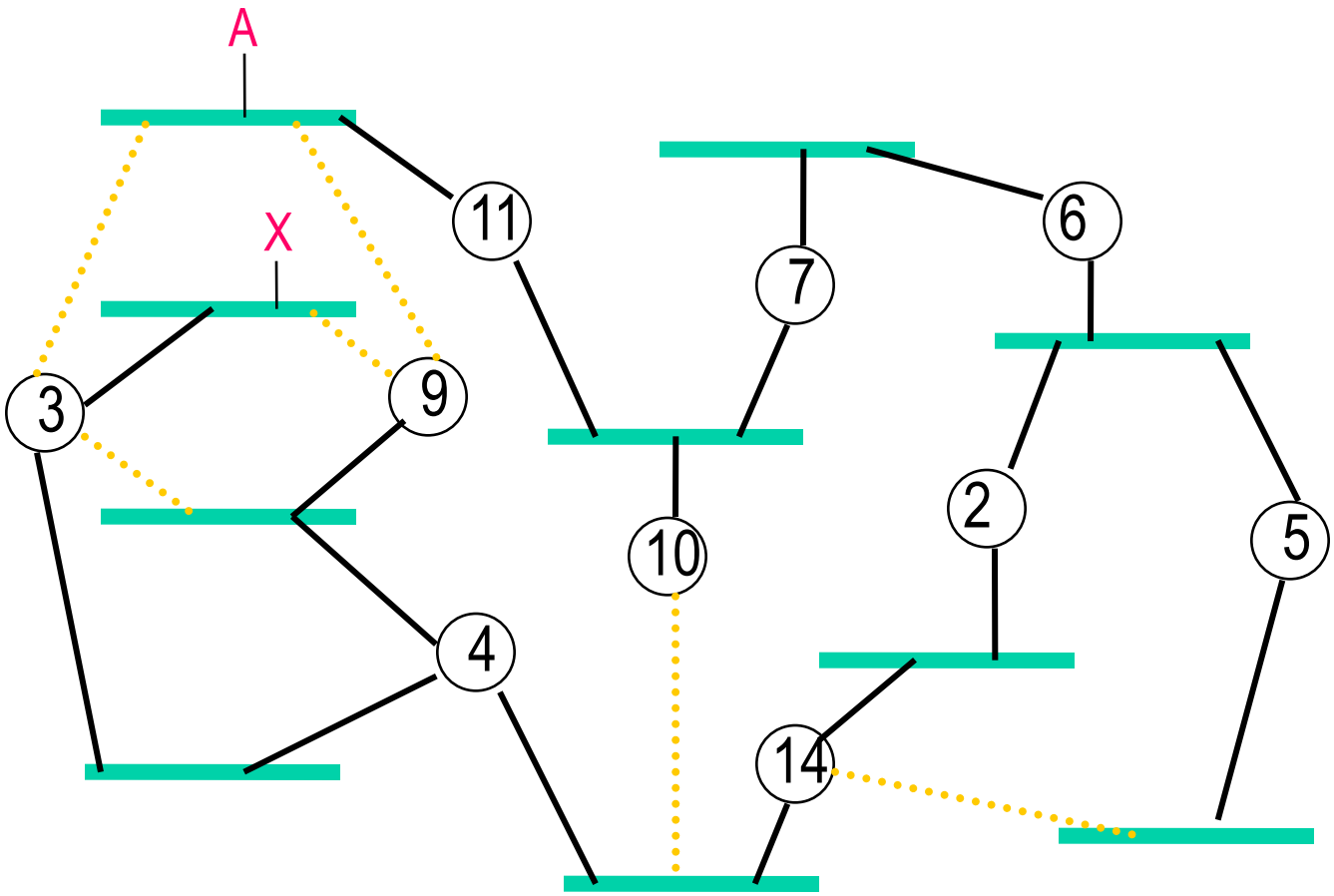
How about require physical tree topology?

- What about miscabling?
- What about backup paths?
- So…spanning tree algorithm
  - Allowing any physical topology
  - Pruning to a loop-free topology for sending data

Physical Topology



Pruned to Tree



Algorhyme

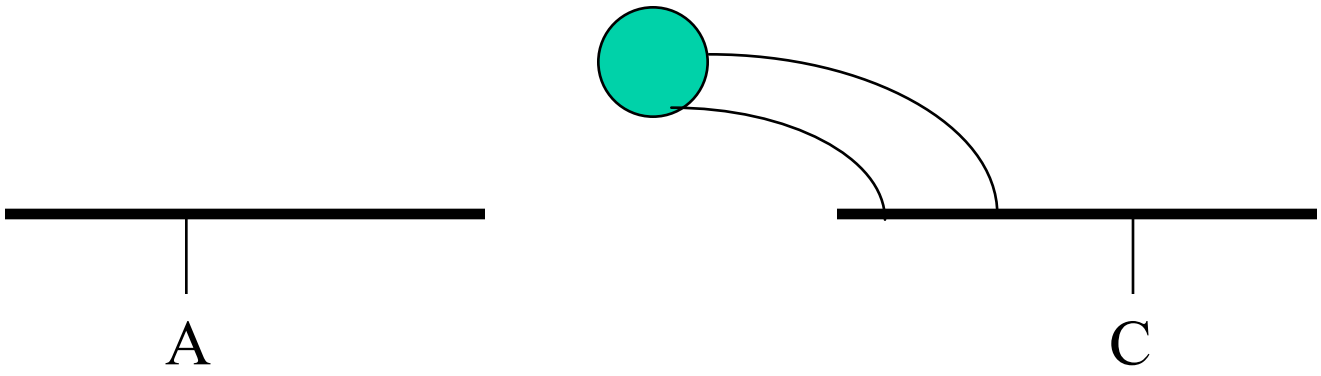
I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree which must be sure to span
So packets can reach every LAN.
First the root must be selected,
By ID it is elected.
Least cost paths from root are traced,
In the tree these paths 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



CSMA/CD died long ago

- A variant is used on wireless links
- But wired Ethernet quickly became pt-to-pt links, and spanning tree
- So “Ethernet” today has nothing to do with all the papers about CSMA/CD

Rant: New words for no reason

Switches

- One day I was told “Nobody cares about bridges anymore…the new thing is switches”

Switches

- One day I was told “Nobody cares about bridges anymore…the new thing is switches”
- OK…what’s a switch?

Switches

- One day I was told “Nobody cares about bridges anymore…the new thing is switches”
- OK…what’s a switch?
  - More ports?
  - Faster?
  - In hardware?

Why not get rid of Ethernet and use only IP?

- Original problem: no layer 3 in the network stack, and multiple layer 3 protocols (IP, Appletalk, IPX, DECnet)
- World has converged to IP as layer 3, and it's in the network stacks

Why not get rid of Ethernet and use only IP?

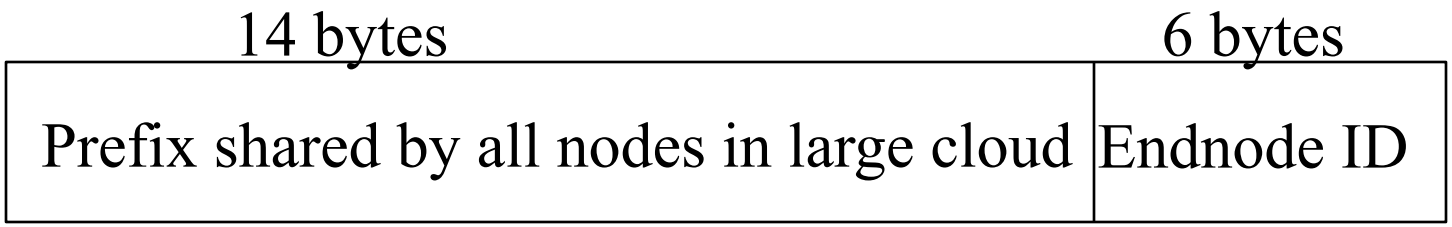
- Just put your data in a layer 3 envelope!
- Hook entire network together with layer 3
- On a point-to-point link, you don't need 6-byte layer 2 addresses…you don't need any layer 2 addresses!
- Problem: an annoying property of IP

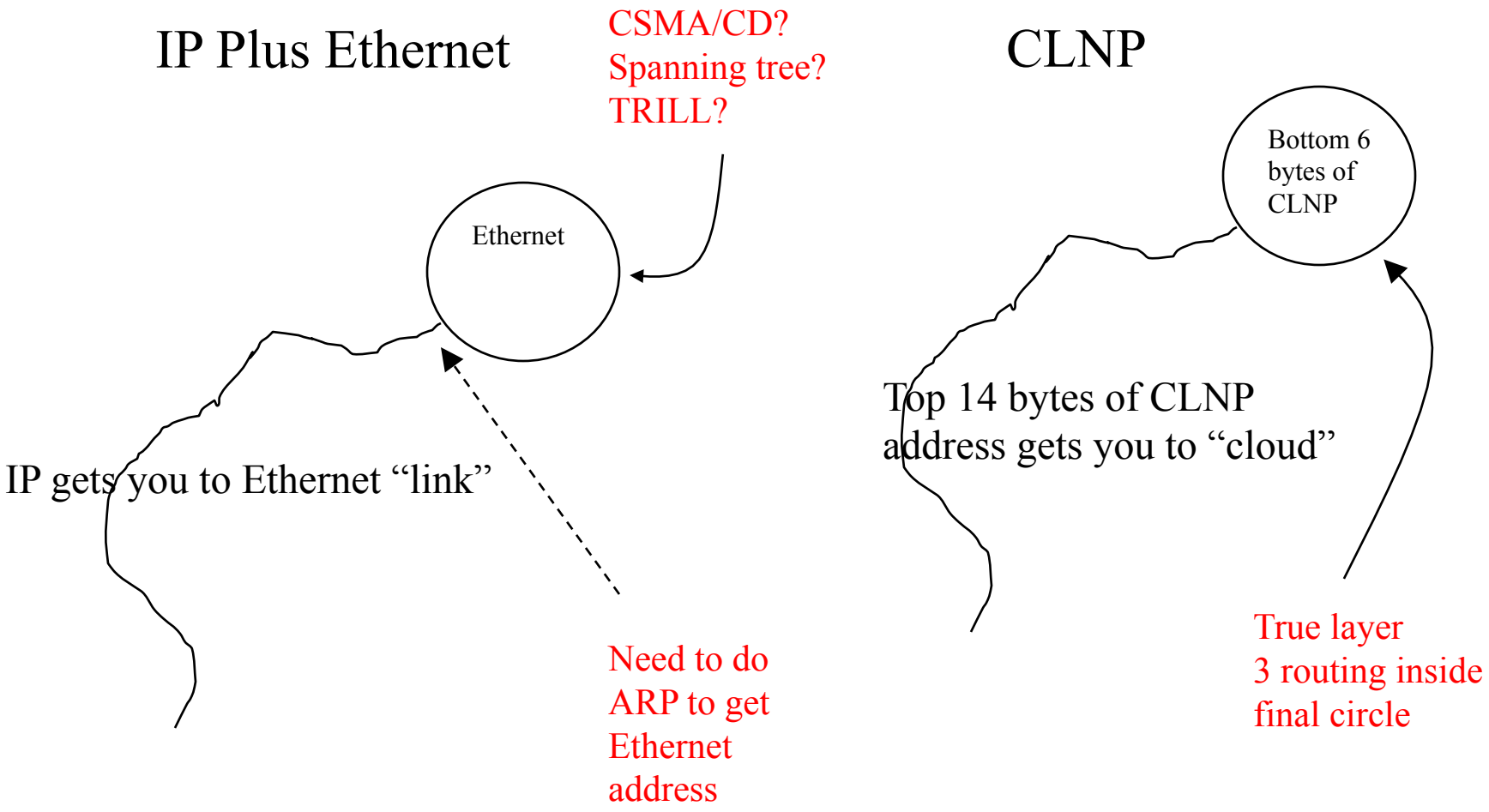
What's wrong with IP?

- 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 IP address changes

Layer 3 doesn't have to work that way!

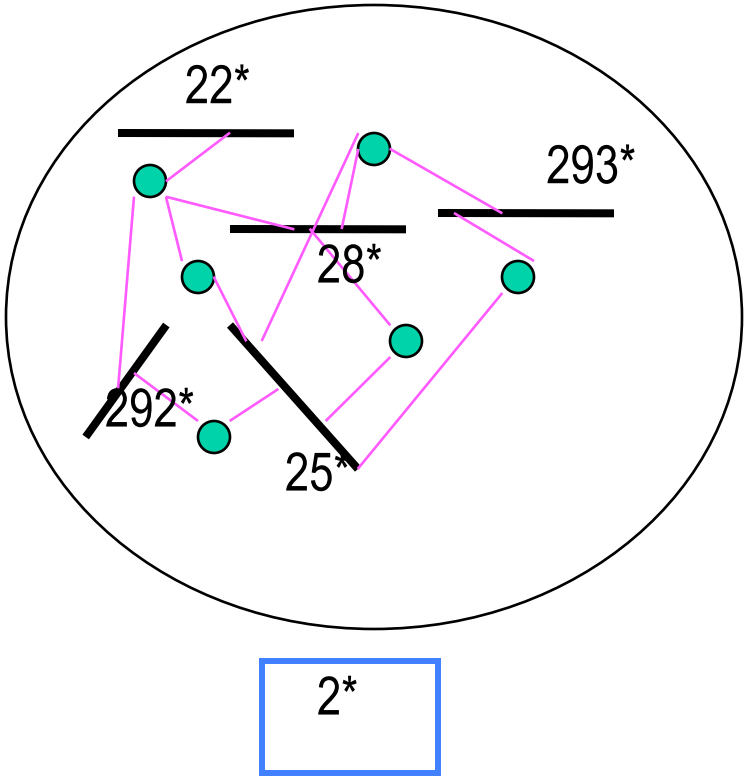
- CLNP / DECnet...20 byte address
  - Bottom level of routing is a whole cloud with the same 14-byte prefix
  - Routing is to 6 byte ID inside the cloud
  - Enabled by "ES-IS" protocol, where endnodes periodically announce themselves to the routers



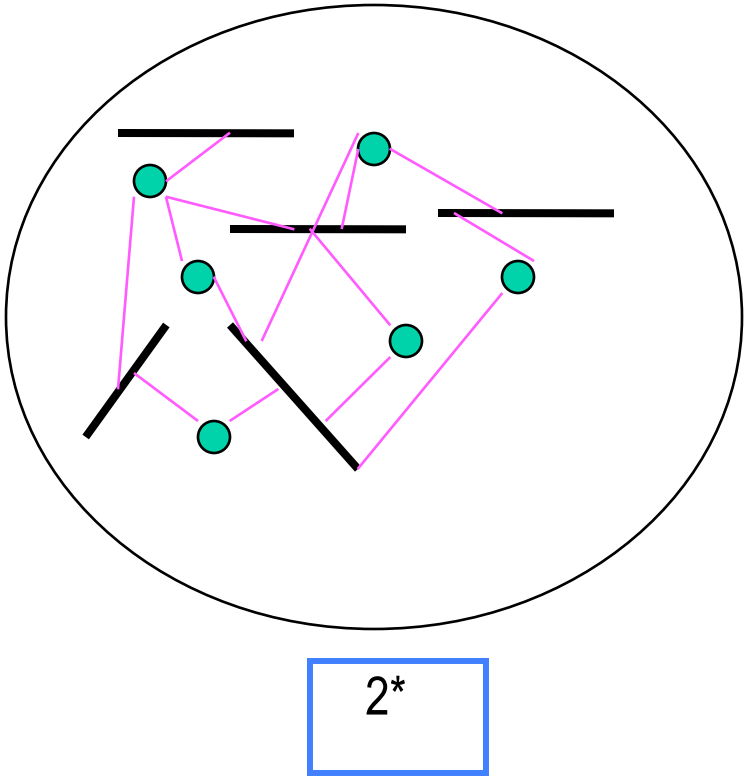

IP Plus Ethernet
CSMA/CD?
Spanning tree?
TRILL?
Ethernet
IP gets you to Ethernet "link"
Need to do
ARP to get
Ethernet
address
CLNP
Bottom 6
bytes of
CLNP
Top 14 bytes of CLNP
address gets you to "cloud"
True layer
3 routing inside
final circle

Hierarchy

One prefix per link (like IP)



One prefix per campus (like CLNP)



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 was deployed by all the vendors, TCP easily modified to run over CLNP
- IPv6 still not better than CLNP! (IPv6 also routes to a link, so will require Ethernet clouds, and ARP-like thing)

Somewhat neglected research areas

- Resilience despite malicious participants
- Usability

Malicious Participants

- All sorts of things can be subverted with a small number of malicious participants
  - "How a Lone Hacker Shredded the Myth of Crowdsourcing"
    - https://medium.com/backchannel/how-a-lone-hacker-shredded-the-myth-of-crowdsourcing-d9d0534f1731
- Things that shouldn't work (but seem to): wikipedia, ebay

Defense against malicious participants

- Problems and solutions are really diverse
- Three things I've looked at
  - PKI models that limit damage from a malicious CA
  - Network that guarantees A and B can talk provided there is at least one honest path between them (my thesis)
  - Reliable expiration of data

But with limited time, I can only talk about one or two

- I'll be around in the next couple of days if you want to hear about the others

What's a PKI?

- A way for Alice to securely know Bob's public key
- If Alice knows Bob's public key, he can authenticate
- It involves a trusted third party known as a CA (certification authority) that signs a document saying "I certify that 279284792837298 is Bob's public key"
- Bob might send cert(s) to Alice, or Alice might retrieve them from a directory

PKI Models

- Monopoly
- Oligarchy
- Anarchy
- Bottom-up

Monopoly

- Choose one universally trusted organization
- Embed their public key in everything that needs to verify certificates
- Make everyone get certificates from them
- Simple to understand and implement

Monopoly: What's wrong with this model?

- Monopoly pricing
- Getting certificate from remote organization will be insecure or expensive (or both)
- More widely it's deployed, harder to change the CA key to switch to a different CA
- That one organization can impersonate everyone

Oligarchy of CAs

- Come configured with 100 or so trusted CA public keys
- Usually, can add or delete from that set
- Eliminates monopoly pricing

What's wrong with oligarchy?

- Less secure!
  - Any of those organizations can impersonate anyone

Certificate Chains

- Allow configured CAs to issue certs for other public keys to be trusted CAs
- Configured CA's: "trust anchors"
- Accept chain of certs
  - Alice configured to trust X1
  - Bob has chain
    - "X1 says this is X2's key"
    - "X2 says this is X3's key"
    - "X3 says this is Bob's key"

Anarchy

- User personally configures trust anchors
- Anyone signs certificate for anyone else
- Public databases of certs (read and write)
- Problems
  - won't scale (too many certs, computationally too difficult to find path)
  - no practical way to tell if path should be trusted
  - (more or less) anyone can impersonate anyone

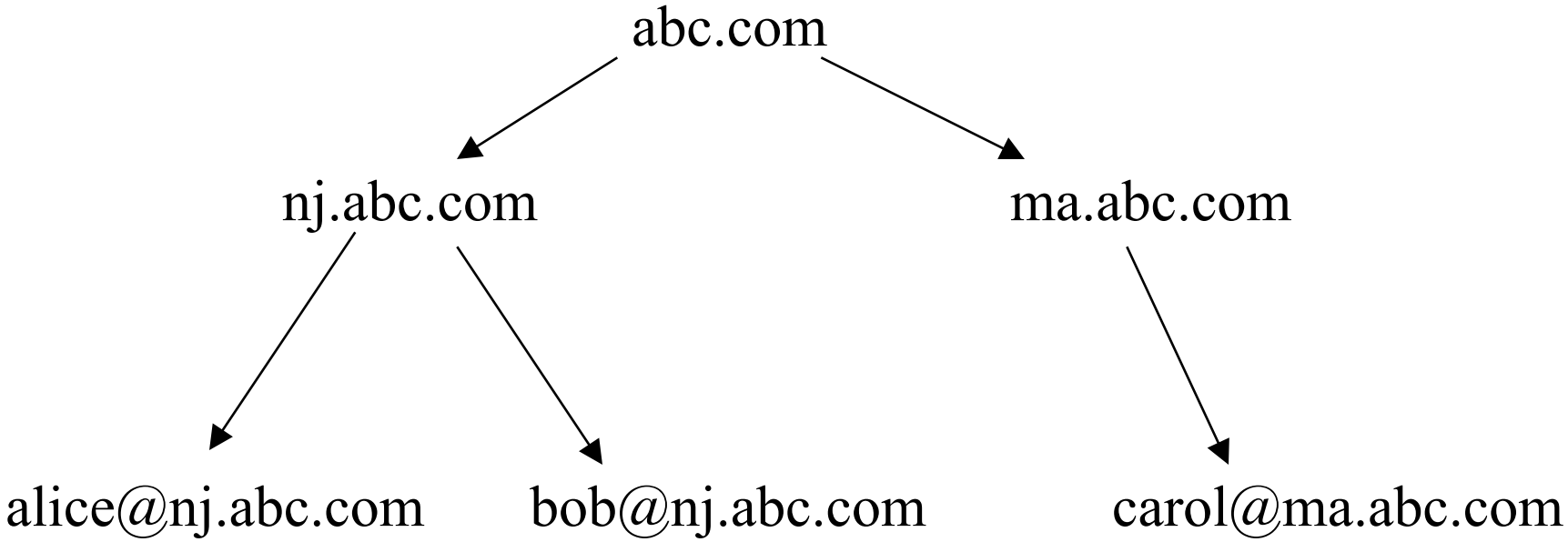
Now getting to recommended model

- CA trust isn't binary: "trusted" or "not"
- CA only trusted for a portion of the namespace
  - The name by which you know me implies who you trust to certify my key
    - Radia.perlman.emc.com
    - Roadrunner279.socialnetworksite.com
    - Creditcard#8495839.bigbank.com
  - Whether they are the same carbon-based life form is irrelevant

Need hierarchical name space

- Yup! We have it (DNS)
- Each node in namespace represents a CA

Top-down model



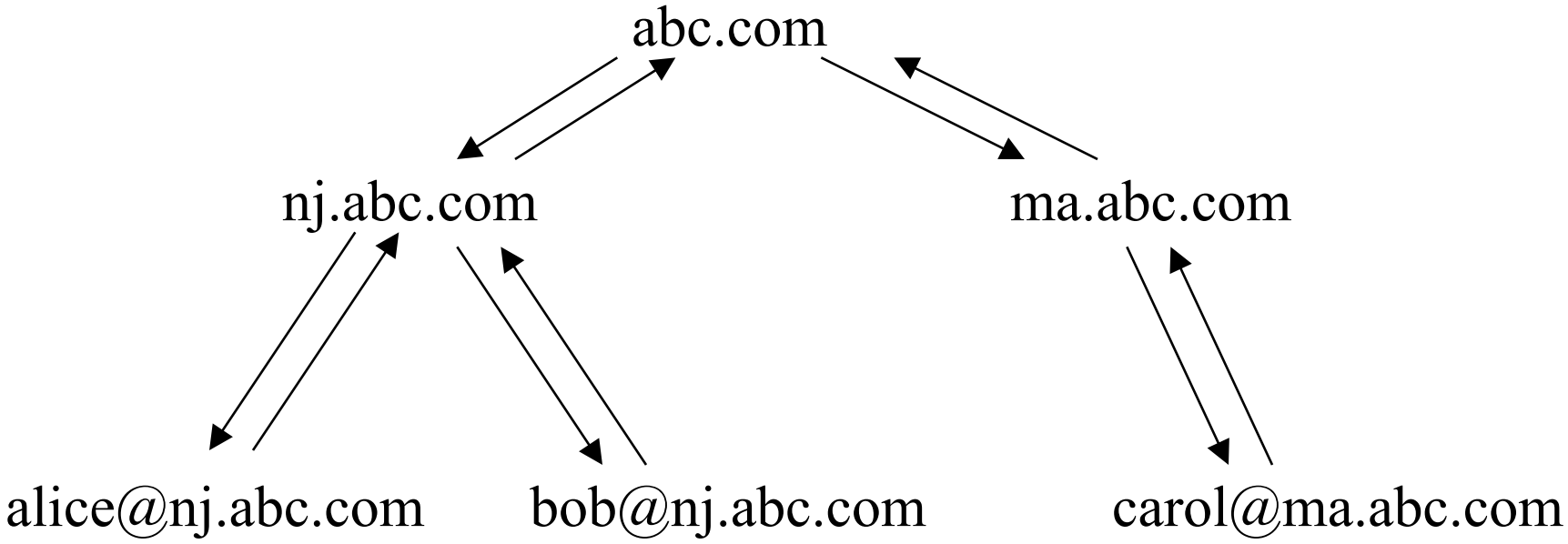
Top-down model

- Everyone configured with root key
- Easy to find someone's public key (just follow namespace)
- Problems:
  - Still monopoly at root
  - Root can impersonate everyone

Bottom-Up Model (what I recommend)

- Each arc in name tree has parent certificate (up) and child certificate (down)

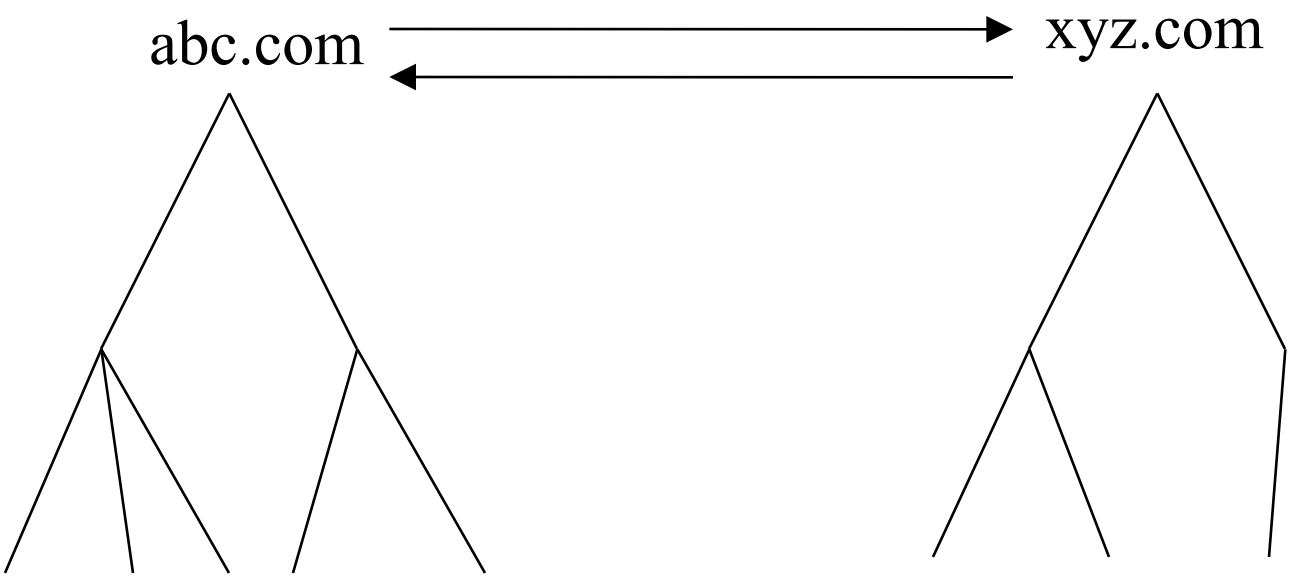
Within an organization



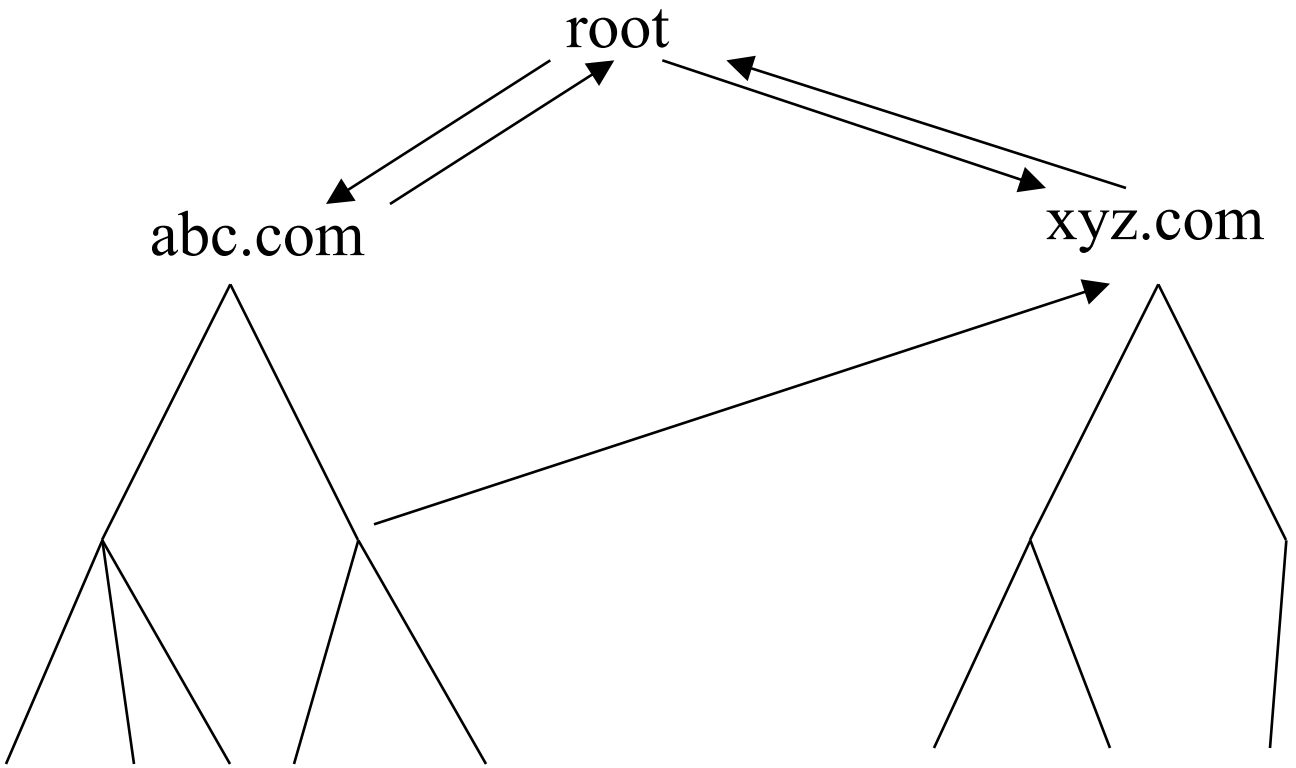
Need cross-certificates

- Cross-cert: Any node can certify any other node's key
  - So you don't have to wait for PKI for whole world to be created first
  - Can bypass hierarchy for extra security

Cross-links to connect two organizations



Cross-link for added security



Note: Crosslinks do not create anarchy model

- You only follow a cross-link if it leads to an ancestor of target name

Advantages of Bottom-Up

- For organization, no need to pay for certificates
- Security within your organization is controlled by your organization
- No single compromised key requires massive reconfiguration
- Easy to compute paths; trust policy is natural, and makes sense
- Malicious CA's can be bypassed, and damage contained

Another example of being resilient

Reliably store data, then reliably expire it

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

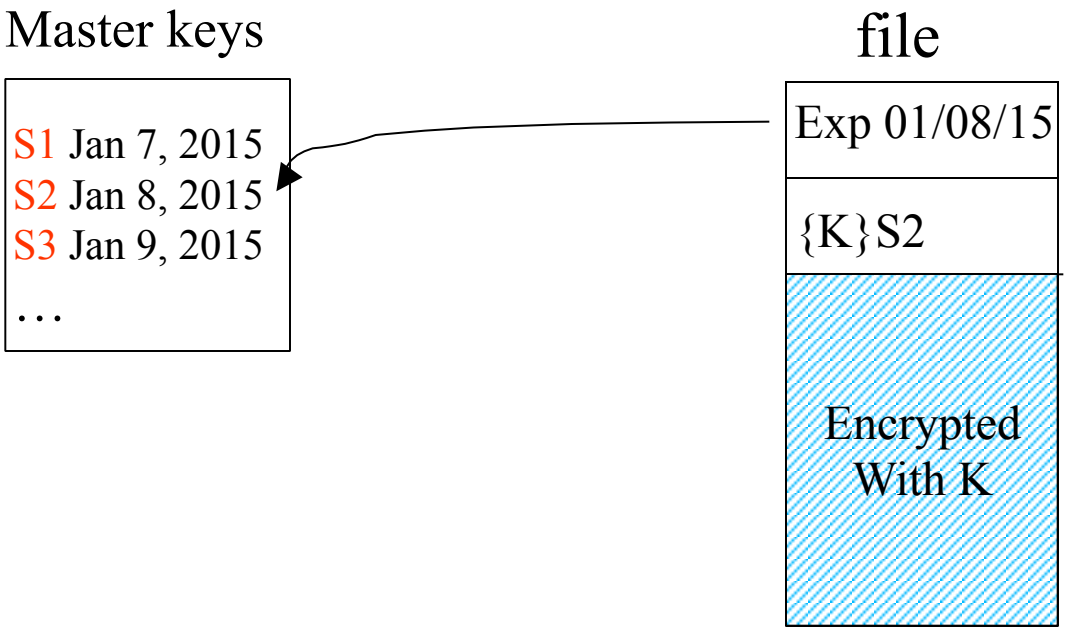
Obvious approach

- 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 storage system
Delete key upon expiration



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
- Storage system "ephemerizes" each master key on backup, by encrypting with (same expiration date) ephemerizer public key
- To recover from backup: storage system asks ephemerizer to decrypt

Ephemerizer publicly posts

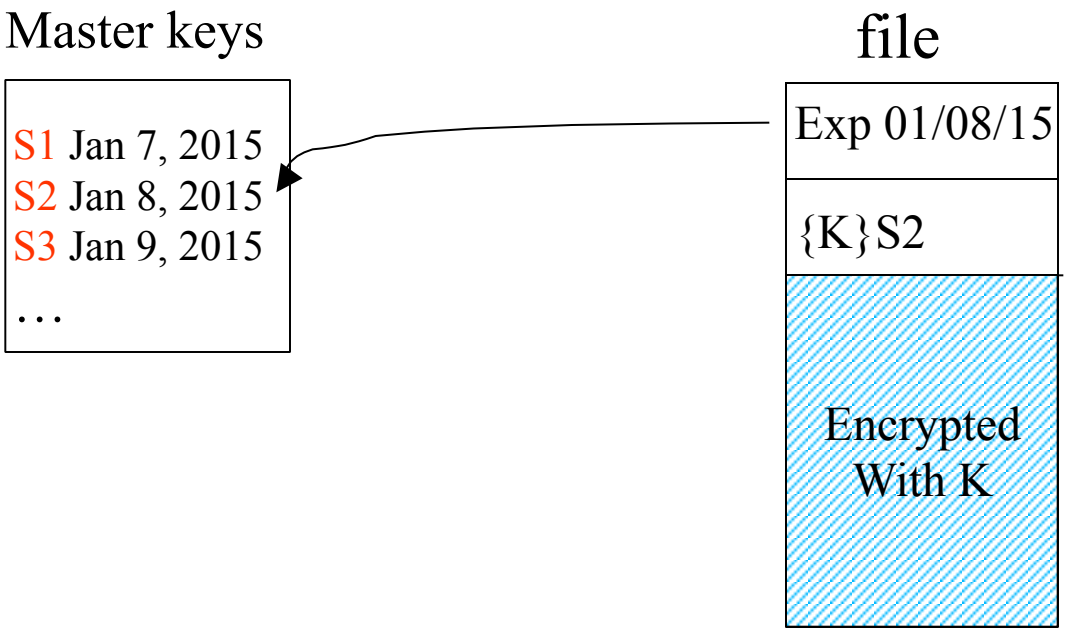
Jan 7, 2015: public key $P_{Jan7of2015}$
Jan 8, 2015: public key $P_{Jan8of2015}$
Jan 9, 2015: public key $P_{Jan9of2015}$
Jan 10, 2015: public key $P_{Jan10of2015}$

etc

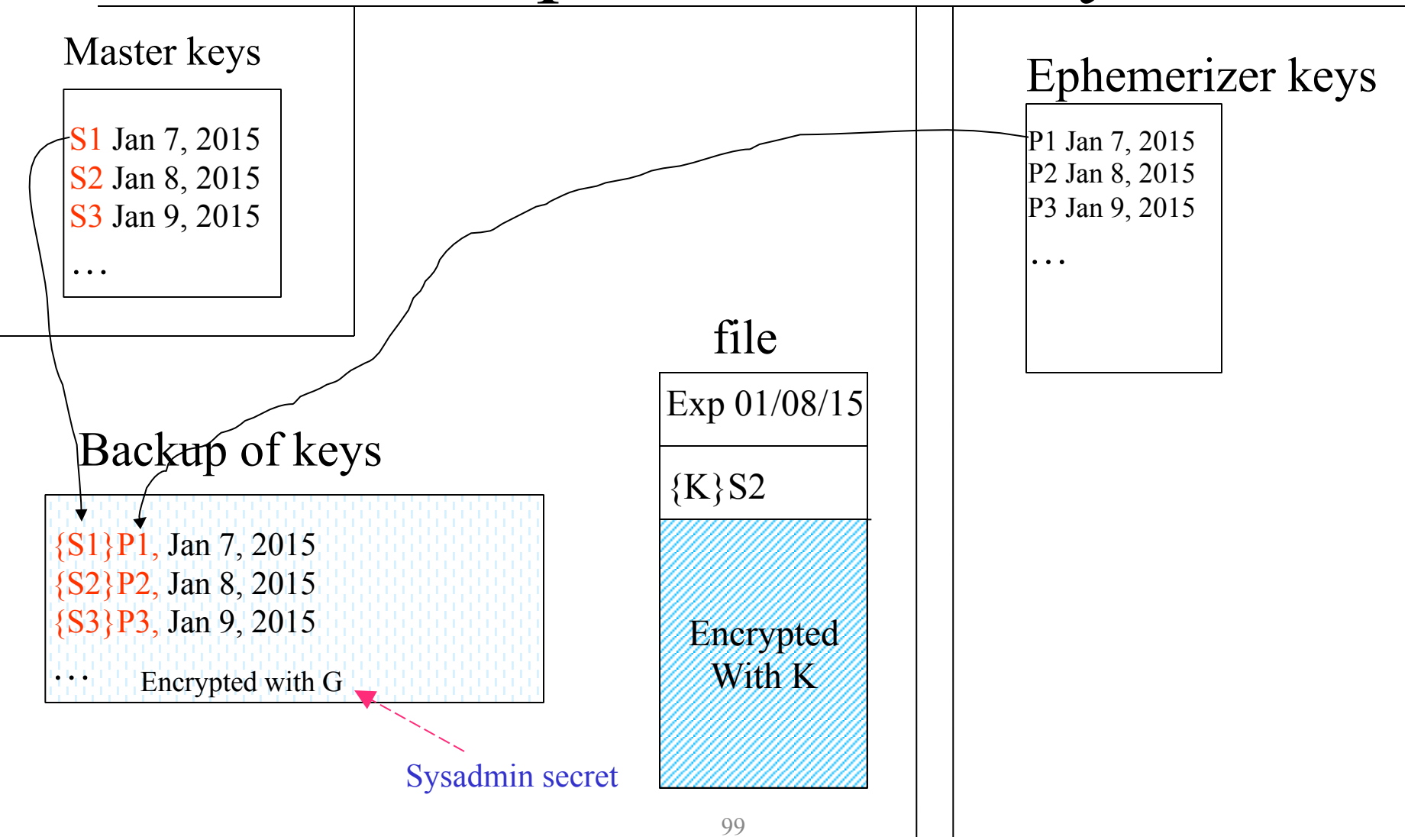
One permanent public key P certified through PKI
Signs the ephemeral keys with P

Storage system with Master keys

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



Backup of Master Keys



Notes

- Only talk to the ephemerizer if your hardware with master keys dies, and you need to retrieve master keys from backup
- 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

Two ways ephemerizer can “fail”

- Prematurely lose private keys
- Fail to forget private keys
- Let’s worry about these one at a time…first worry about losing keys prematurely

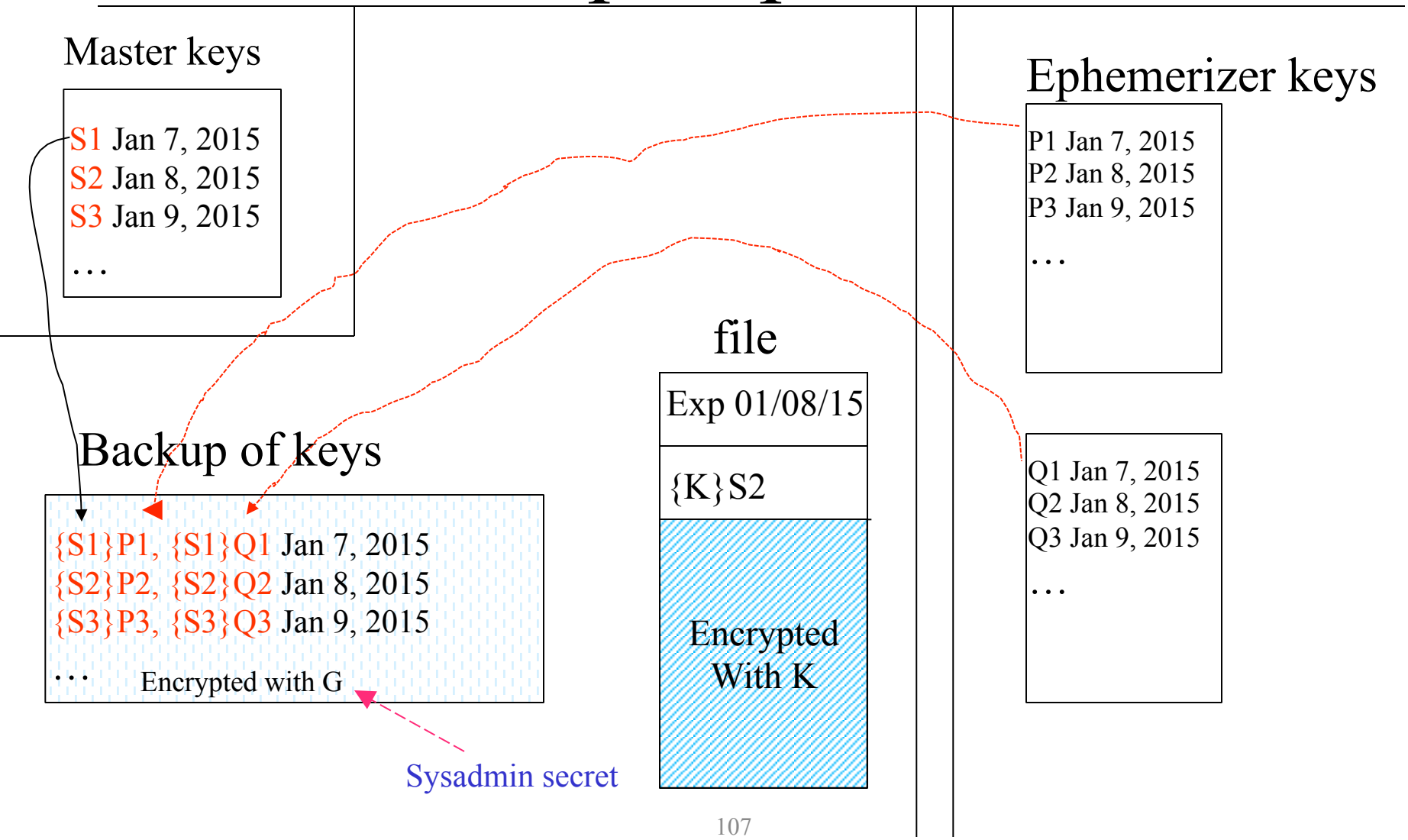
Losing keys prematurely

- We will allow an ephemerizer to be flaky, and lose keys
- Generate keys, and do decryption, on tamper-proof module
- 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
    - Different organizations
    - Different countries
    - Different continents
  - Independent public keys

Use multiple ephemerizers!



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

- Then the storage 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

If I had more time…

- A really cool protocol for asking the ephemerizer to decrypt
  - Super lightweight (one packet each direction, less computation than an SSL connection)
  - Gives the ephemerizer no information when it decrypts!
- After a disaster, only need to ask the ephemerizer for one decryption!
- And other cute little things…

Another neglected topic: usability

Usability

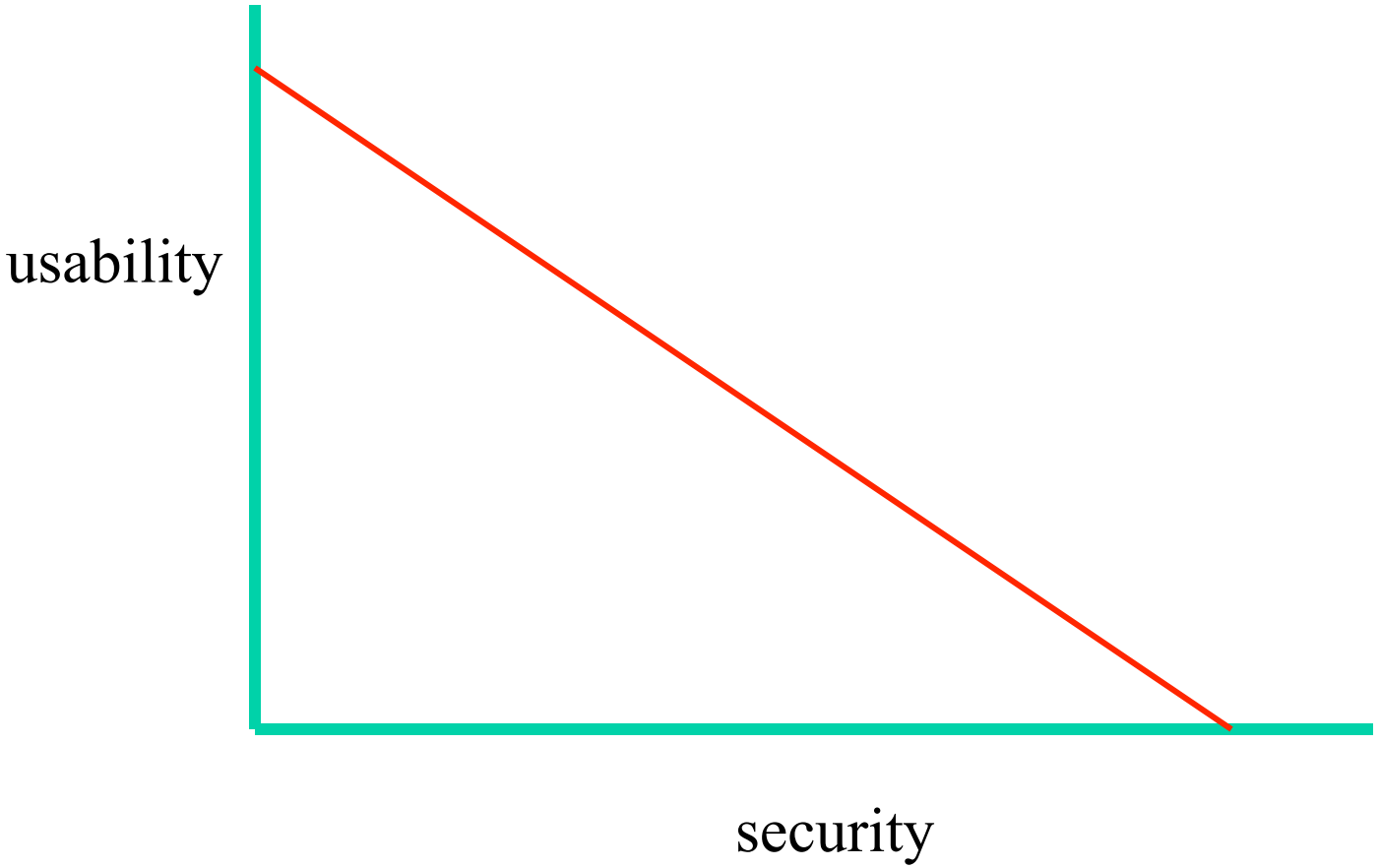
- Engineers should actually meet some humans, then they'd stop having programs ask questions like:

Usability

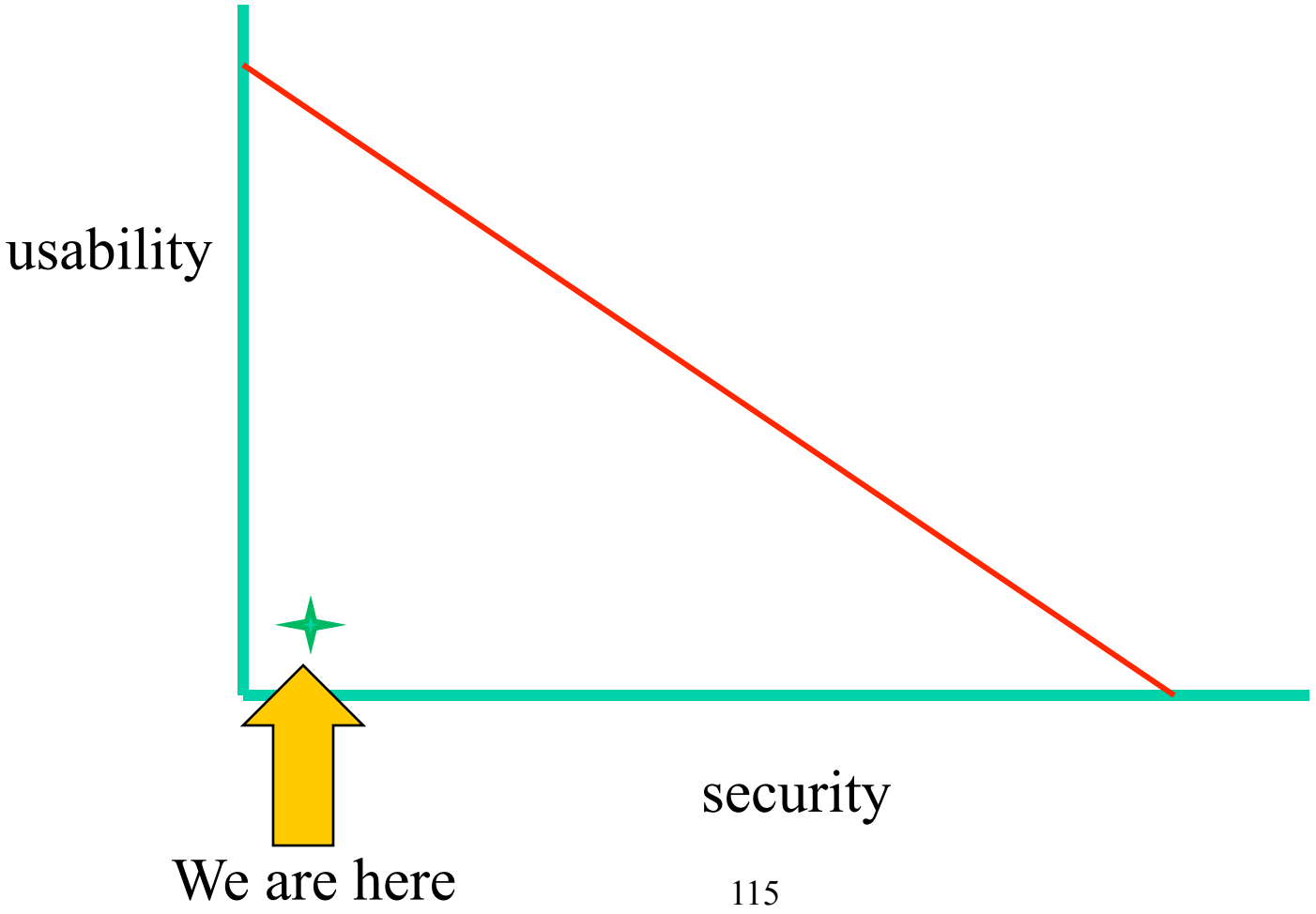
- Engineers should actually meet some humans, then they'd stop having programs ask questions like:
  - Do you want to display both the secure and insecure items?
  - Do you want POP or IMAP?

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

It's common to have to tradeoff usability vs security



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)

Can we do worse? Yes!!

- I had to set a password and got the message
  - "Your password does not meet our length, complexity, or history rules"
- It didn't even tell me the rules!!!

Security questions

- Who comes up with these?
  - Father's middle name
  - 2nd grade teacher's name
  - Veterinarian's name
  - Favorite sports team
  - My middle name

Why not let us create our own questions?

Annoying rules that add nothing to security

- Must change password at least every n days
- 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"

People

"*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

Protocols and Life

Protocols that don’t work

Protocols that don’t work

- “We’ll call you if there is a problem”
- “We’ll call you if we want to hire you”

Protocols that don’t work

- “We’ll call you if there is a problem”
- “We’ll call you if we want to hire you”
- “Tomorrow at 2 PM there will be a fire drill. Ignore all sounds and behavior of the staff at that time.”

Parting Thoughts

- What "wins out in the market place" isn't necessarily the best thing
- Like English, if existing stuff can do the job, or be tweaked, it's hard to replace it
- Don't believe (or repeat) things you can't understand…they are often false
- Know what problem you're solving before you try to solve it!