Myths, Missteps, and Folklore in Protocol Design

Radia Perlman

Sun Labs, East Coast

radia.perlman@sun.com

Outline

- Bridges, Routers, and Switches, Oh My!
- What's with IP Multicast?
- What's with IPv6?
- ARPANET flooding
- BGP
- Compatible Parameters
- Forward Compatibility
- How not to get any advantage from a public key system

Why This Talk?

- Learn from mistakes
- Understand oddities in today's protocols
- Counteract "religion" aspect

  "*It's not what you don't know that'll get you. It's what you do know that ain't true*"

  .....Mark Twain
- Be provocative. Start lively discussion.

Bridges and Routers and Switches, Oh My!

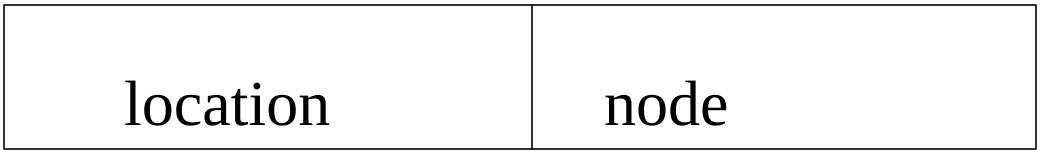
- ISO’s OSI Reference Model
  - layer 1: physical
  - layer 2: neighbor-neighbor
  - layer 3: talk across cloud via a multi-hop path
  - layer 4: end-to-end
  - layer 5 and above: boring
- Repeater: layer 1 relay
- Bridge: layer 2 relay
- Router: layer 3 relay
- OK. Why is there such a thing as a layer 2 relay?

Definition of Layer 2 Protocol:

Anything defined by a committee whose charter is to standardize a layer 2 protocol.

What does a (connectionless) layer 3 protocol look like?

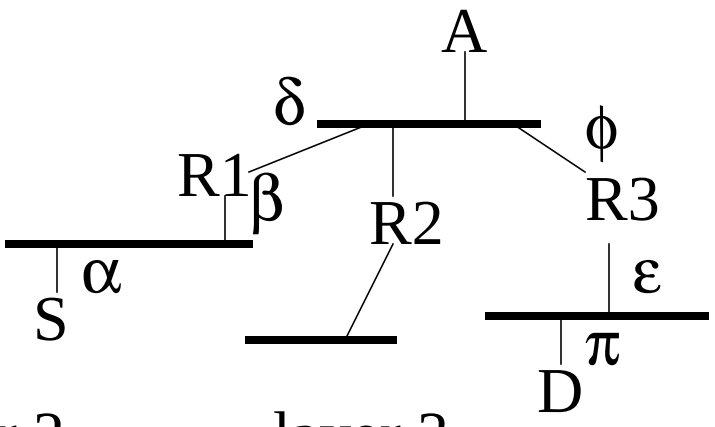
- put “envelope” on your data that includes destination, source address and hop count
- Layer 3 addresses usually hierarchical



- With point-to-point links, only need one set of addresses: ultimate source, and ultimate destination
- With multi-access links like Ethernet, need “next hop” also

Ethernet

- Need “next hop” address in addition to ultimate destination



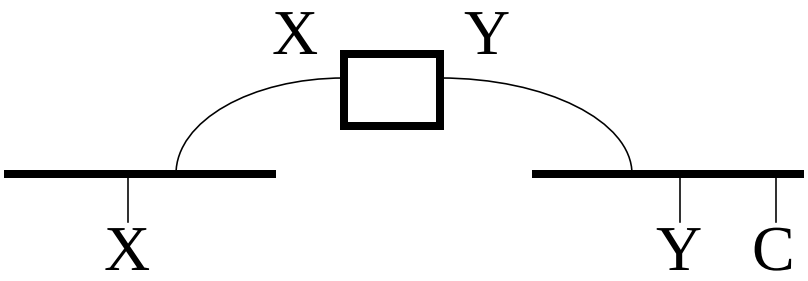
layer 2	layer 3	
src=α dst=β	src=S dst=D	data
src=δ dst=ϕ	src=S dst=D	data
src=ε dst=π	src=S dst=D	data

- Also required rethinking routing algorithm (e.g., Designated Routers, pseudonodes)

What are Bridges?

- Myth: Bridges were invented before routers.
- Natural assumption. Bridges are simpler.
- But no! Routers were first
- People got confused by EtherNET. It's a link in a network. Not a "network"!
  - It puts addresses on pkts, just like layer 3
  - But addresses are not hierarchical
  - No hop count
  - Technology doesn't scale (distance, # nodes)
- People thought it REPLACED layer 3
- I lost the battle. People implemented protocols (like LAT) without layer 3
- Bridges were a kludge to work without the cooperation of the endnodes

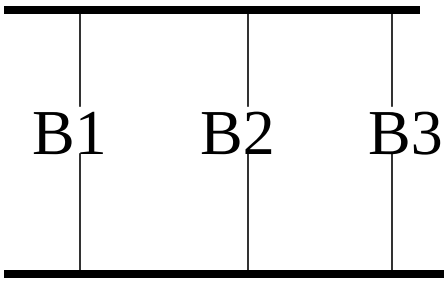
Basic idea of bridges



- listen promiscuously
- learn source's location
- forward based on learned location of destination

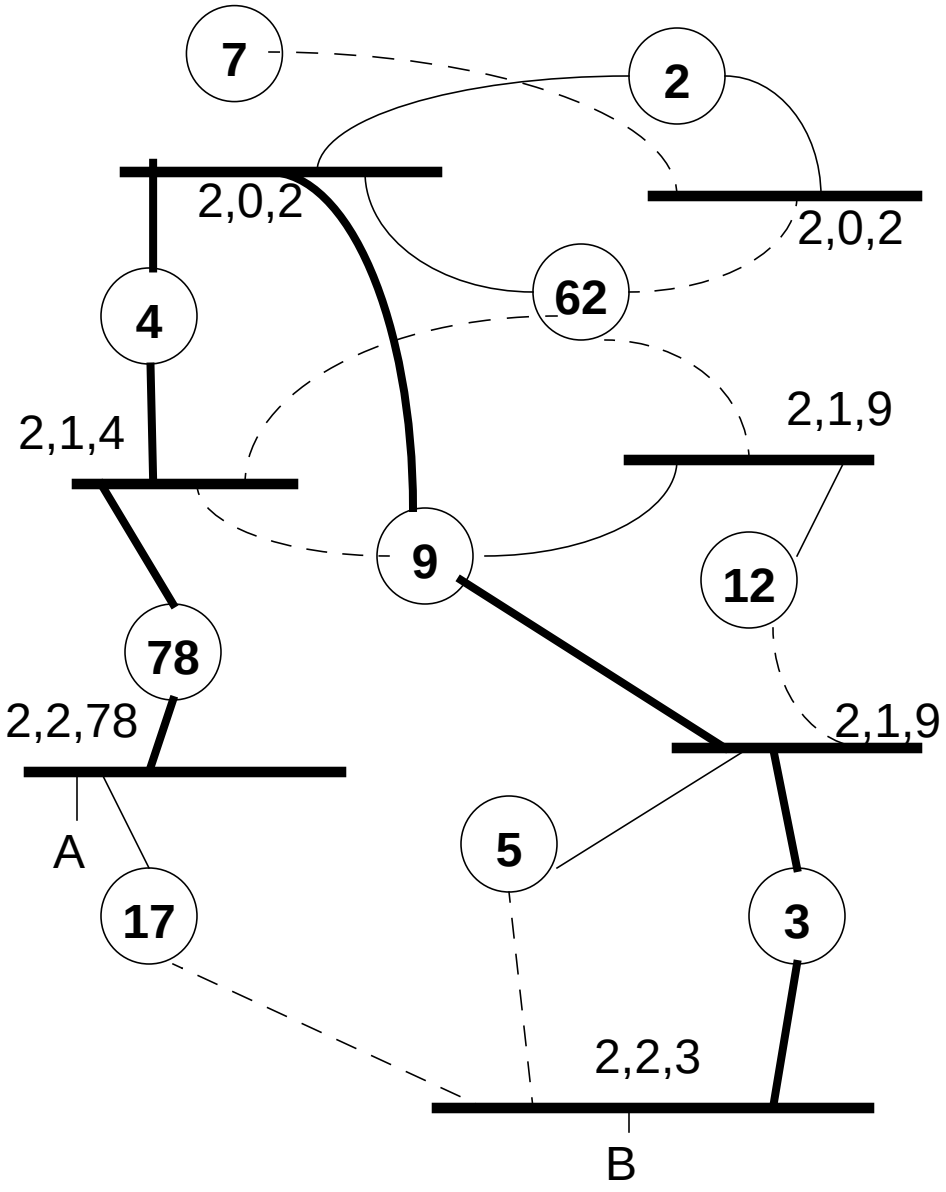
Problem

- Need loop-free topology
  - Can’t learn location of source if it is reachable from multiple directions
  - loops are a disaster (no hop count, exponential proliferation)



- So, they needed the Spanning Tree Algorithm
  - Since loops are a disaster, have to be very conservative about turning on links
  - Tree-topology means traffic concentration, suboptimal paths

Spanning Tree Algorithm



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.

Why were bridges so popular?

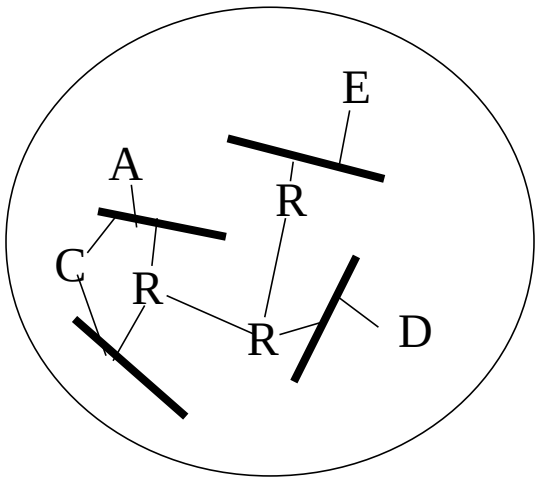
- Routers of the time weren’t multiprotocol
- Protocols like LAT and NETBIOS without layer 3
- Simple, high performance, rock solid, and cheap

Why haven't they gone away?

- Router price/performance has improved
- IP is now pretty much ubiquitous, so multiprotocol and no-layer-3 is not an issue
- Bridges still more plug-and-play, and cheaper
- With IP, you need an address per "link".
  - People want to allow a campus in which nodes can move and keep address
  - Can do that with some other layer 3 protocols (CLNP, DECnet). Not IP
  - Bridges allow the equivalent of CLNP "level 1 routing" for IP, but with worse routes (spanning tree)

CLNP (ISO's version of IP)

- Prefix isn't "link", it's "area"
- Address: ("area", "node")
- "Node" is 6-bytes. Autoconfigure using MAC
- "Area" is the rest of the address, and works based on longest matching prefix (like IP).
- Within area, rtrs keep track of individual nodes



- So within area, optimal routing, can move and keep address, have multiple links, ...

Switches

- Ethernet used to be a long bus
- Easier to wire, more robust if star topology: one huge multi-port repeater with pt-to-pt links

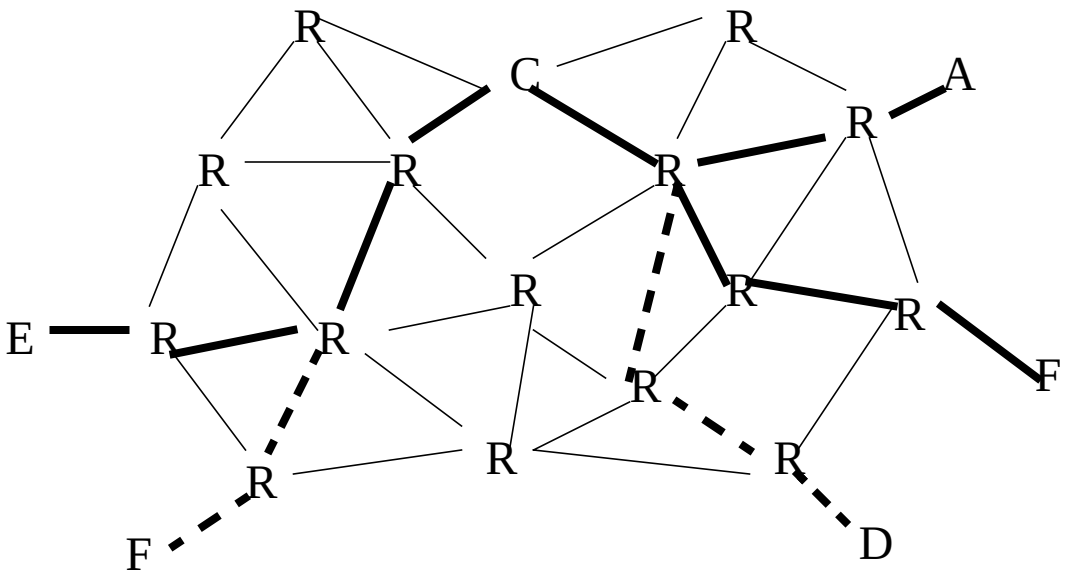
- If store and forward rather than repeater, then can have more aggregate bandwidth (A to B can simultaneously use whole bandwidth while C and D talk)
- Can cascade devices
- Should therefore do spanning tree
- We've reinvented the (transparent) bridge!

Next Story: IP Multicast

- How did it get to be so complicated?
- It’s been being designed since 1988!

Multicast

- It doesn't have to be hard!
- ATM had "point-to-multipoint" VCs



  - Create VC=25, destination=E
  - Add destination=F to VC 25
  - Add destination=D to VC 25
- IP people wanted "join" initiated by member, not root. That's OK.

How IP Multicast Could Work

- Have member request Root to add them to the tree. Then it would work just like ATM.
- Alternatively, if you want the join to go the other way:
  - Group defined by (Root=R, G)
  - Send “Join (R,G)”
  - Join message routed towards R
- Create tree
- This wouldn’t have been worth zillions of conference papers and PhD theses.

IP Multicast API

- Design axiom: It should look *exactly* like Ethernet
  - anyone join group without permission
  - anyone send to group without listening
  - 28-bit address space
- Reality
  - no way to do this efficiently
  - address allocation alone a nightmare
  - Ethernet multicast used for a small number of pre-defined groups, for bootstrapping.
  - IP multicast different applications: conference call, crowded web-site, interactive games. Tens of millions of groups, geographically dispersed.
  - API shouldn’t be the same!

API “Cast in concrete”

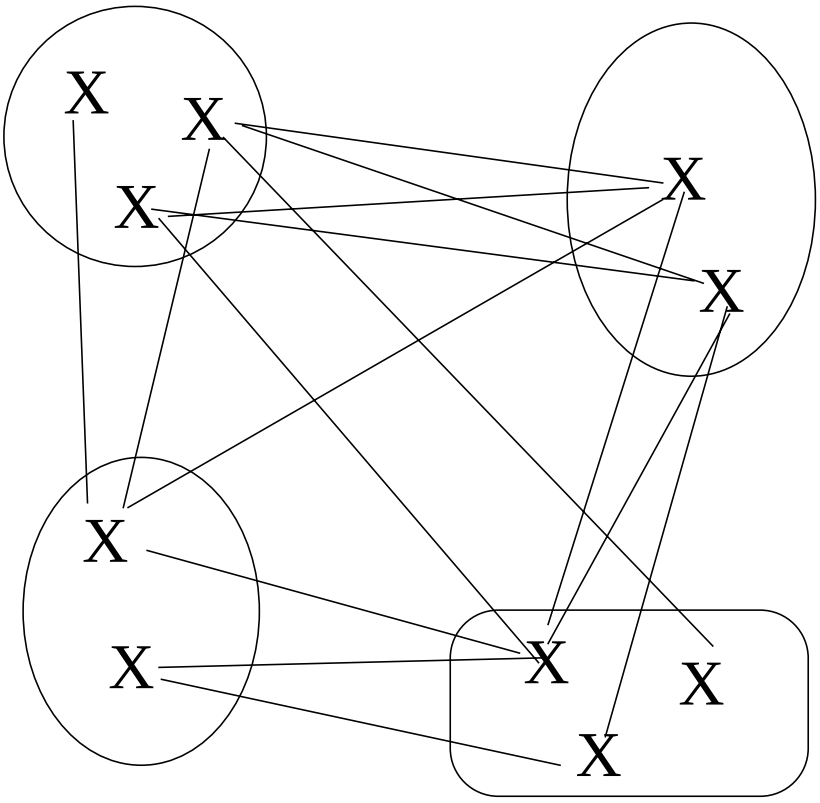
- Group names globally unique “class D IP addresses” (top 4 bits=1110)
- Endnodes inform local router “I want to join G” using IGMP protocol
- Everything else “details that the world could work out later”
- So API was like cement overshoes

12 Years of IP Multicast Design

- DVMRP/PIM-DM: Flood and prune
  - Scaling: periodic flooding of tens of millions of groups, prune state: remember every (S,G) pair each neighbor is *not* interested in
- MOSPF: Routing algorithm informs routers of location of all receivers for all groups
- CBT: Create a bidirectional tree to a root. OK, but root undefined.
- PIM: Make CBT immensely complex
  - advertise currently alive Root candidates
  - hash G to select from {Root candidates}
  - Dynamically form per-source trees, if source sending “enough traffic”
  - Shared tree is unidirectional

12 Years of IP Multicast Design, Cont'd

- MSDP: configure root-candidates to know about each other. If you're the root of tree G, and S transmits, flood info to all other root-candidates in the Internet that (S,G) was active in the last minute!



“Simple Multicast”

- Root-centric. Choose Root. Ask Root for address, which is 8 bytes: (Root,G).
  - addresses trivial to administer
  - easy for routers to know who the root is
  - addresses plentiful
- Picking root
  - crowded web site: that site should be root
  - newsgroup: choose something, if it’s down, newsgroup might be down for awhile. Or have backup group
  - videoconference: any of the participants, perhaps with backup group
- After resisting with truly ugly politics, mostly now doing this. Using unidirectional trees, and still using separate IGMP-like protocol

Quote

- *There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies....*

C.A.R. Hoare

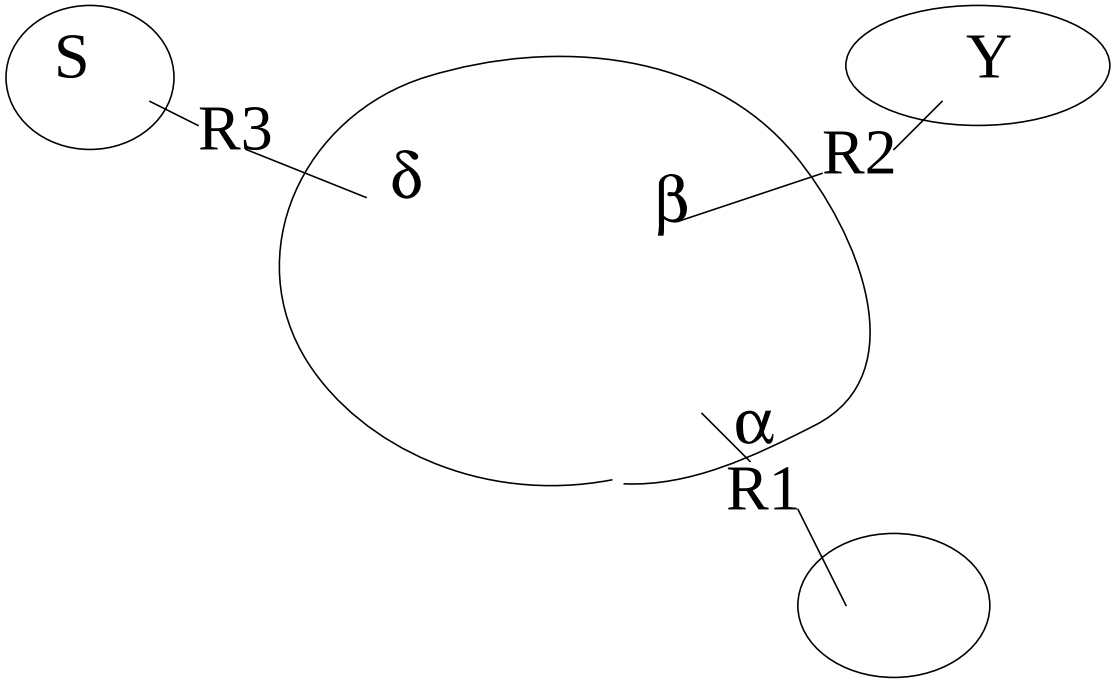
Next Story: IPv6

- Why should it have taken 9 years (and counting) to design IPv6?
- What are the brilliant new ideas that took 900 or so man-years (and thousands of pages of specifications) to design?
- Reality: IPv6 is just a(nother) simple connectionless layer 3 protocol: put source, destination address, and hop count, on your data
- IAB, in 1992, realized IP addresses were too small
- They recommended moving to CLNP, which was just like IP but with bigger addresses
- Furthermore, it was a mature standard, implemented by all the major vendors

Politics

- Some vocal IETF'ers said, "Can't replace IP with ISO!". Instead invent new thing, IPv6 (wound up with 16-byte addresses)
- CLNP had features superior to IPv6 (level 1 routing instead of bridging, big enough address so top level could be E.164 address)
- IPv6 myths
  - It's not a "new protocol". It's just a new version of IP
  - Security is built into IPv6. It's just an add-on to IPv4

Nice Feature of CLNP Addresses



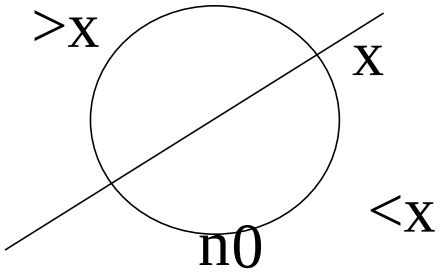
- Suppose E.164 addressing within backbone
- S's address would be δ.S. Y's would be β.Y.
- R2 and R3 don't need configuration, routing protocol or anything. By magic, no overhead, S and Y can talk.

Why Hasn't IPv6 been deployed?

- Much more difficult now. Internet much bigger
- IPv4, out of necessity, had some important improvements:
  - CIDR (more sensible allocation of addresses)
  - DHCP (dynamic configuration, almost as easy as CLNP's)
  - NAT (network address translation) allows an entire site to share a small number (maybe 1) global IP addresses
- IPv6 still being designed. Meets several slots every IETF, and interim meetings as well!

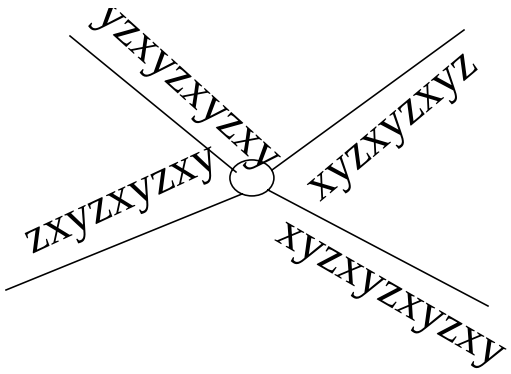
Next Story: Unstable Protocols are bad

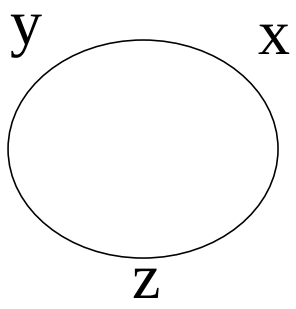
- ARPANET Flooding
- Link state routing: Flood LSP: your ID and a list of your links (neighbor, cost)
- If see “new” LSP (higher sequence number), flood. Else ignore.
- Sequence number wraps around



The Great Disaster

- ARPANET incident — symptom: net didn't work





- Turned out a sick router, Fred, generated LSPs with random sequence numbers before dying.
- x < y < z < x
- Queues of all routers filled with LSPs from Fred
- The more work routers did, the more LSPs in the system

How do you fix a network?

- You depend on the net for diagnosing and fixing the net!
- The ARPANET people were incredibly lucky!
  - all rtrs identical hardware running identical software
  - Field service = protocol designers = protocol implementers
- Lessons:
  - This stuff is scary
  - Self-stabilization really important
  - Formal methods don’t necessarily work (this algorithm was proven correct!)
  - “Byzantine robustness” might be nice too

Next Story: Interdomain Routing

- What's a domain?
- Original interdomain routing protocol=EGP
- EGP merely reports "I can reach X". No costs!
- Reasoning:
  - routing too difficult for multivendor interoperability
  - within a domain, one vendor
  - something that isn't a "routing protocol" between domains
  - EGP is so broken nobody would call it a routing protocol, so it doesn't violate the axiom!

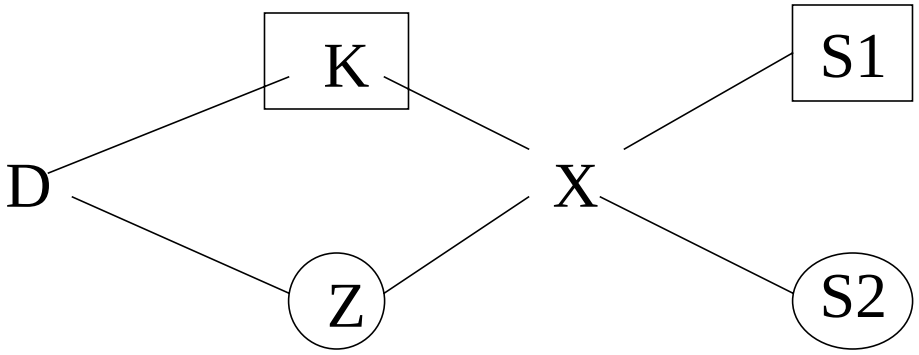
Policies

- Later “policies” became an excuse for interdomain protocols
- “Policies” means not all paths legal
  - only travel via commercial networks
  - I don’t want to route traffic between A and C
  - don’t go through Tennessee
- Why do we need two types of routing, one for inside a domain and one for between domains?
- It only makes sense if you believe all the following simultaneously:
  - policies not necessary within a domain
  - policies essential between domains
  - any protocol capable of handling policies would be too inefficient to work in the simplest case of policy=all paths legal

BGP

- “Normal” routing: minimize cost of path
- BGP
  - Neighbors tell each other the path they use (no metrics).
  - Choose path based on configured criteria
    . don’t ever use domain A
    . don’t use A if using B
    . only use A if no other path
  - Other configuration
    . don’t tell neighbor N you can reach D
    . tell neighbor N, but add extra hops to make path look less desirable

BGP Doesn't Support Many Reasonable Policies



- X makes a choice how to get to D
- Everyone behind X stuck with X's choice

BGP Problems

- Configuration intensive
- Unstable Policies
  - Policies that don’t converge
  - Policies that may converge, depending on order of events
  - Policies that will converge, but a link going down turns it into non-convergent
  - It’s NP-complete (i.e., intractably hard) to look at a set of policies and decide whether they’ll converge
- We’re probably stuck with BGP forever...

Compatible Parameters

- Important to be able to configure nodes, one at a time, and keep the network running
- Some parameters must be compatible (e.g., hello timer and dead timer)
- IS-IS
  - pairwise parameters reported in “Hellos”
  - area-wise parameters reported in LSPs
- bridges:
  - Root reports its values in spanning tree msgs
  - Everyone uses Root’s values
- OSPF
  - copied most of IS-IS, but got this part wrong. Report value. Refuse to talk if not identical

Forward Compatibility

- Allow for future
- Fields should be big enough (IPv4 address, packet ID, TCP sequence number)
- “Reserved” values: transmit as 0, ignore on receipt
- TLV (“type”, “length”, “value”) encoding
  - “length” must be consistent, so can skip unknown fields
  - BOOTP: “vendor specific” field
- Sequence number
  - must be always in same place
  - drop if sequence number too big
  - maybe have “Sequence number I support” in addition to “sequence number of this pkt”

SSL Sequence Numbers

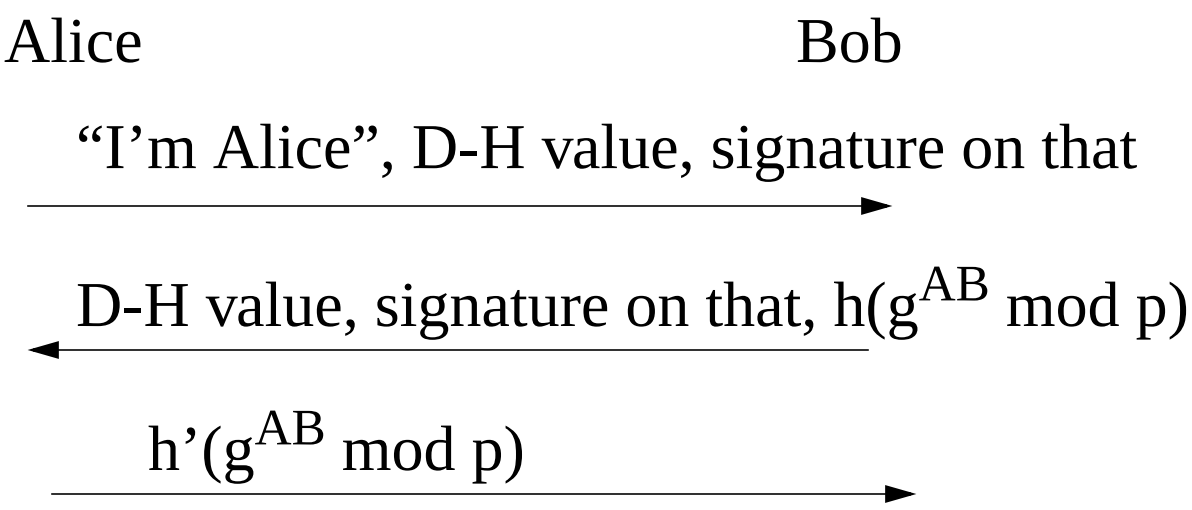
- 2-byte sequence number. v2 sets it to 0,2
- Version 3 redid packet format. Moved sequence number field! Sets it to 3,0
- V3 capable of talking either format must:
  - send initial pkt in v2 format (v3 would confuse v2 node) (until v2 interoperability is no longer an issue, if ever)
  - sequence number set to 3,0
  - So....v2 node gets pkt with sequence number=768, and it never occurs to it that pkt format might have changed in the intervening 766 versions!

Simplicity is Good

- How did IPSec get to complicated?
- Initially Photuris vs SKIP
- Either one would have been fine
- They selected....ISAKMP??!!
  - from NSA...if there's any organization less favored by IETF than ISO...
  - Not a protocol...a "meta-architectural framework"
  - Incomprehensible doc about inefficient, complex encodings
  - Certainly not something you could implement from
- Later IKE attempted to fill in details

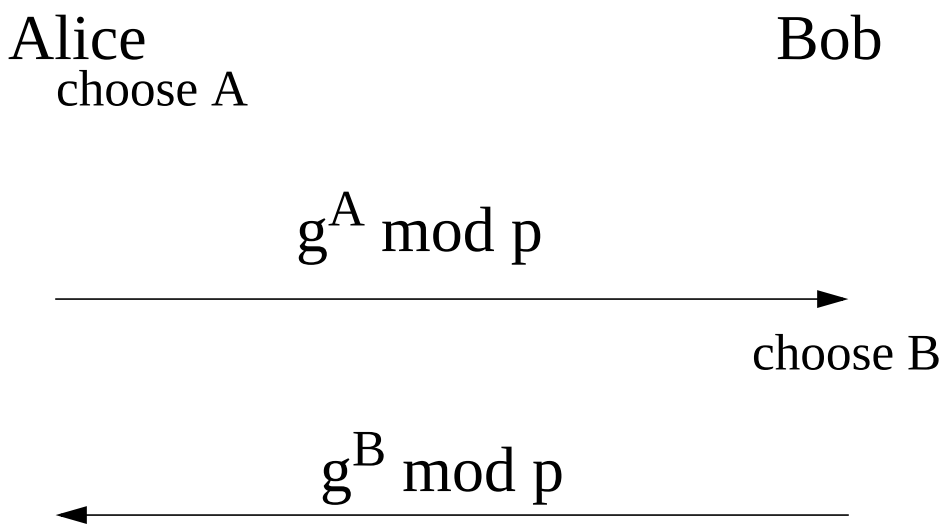
What it could have been

- Many ways of doing mutual authentication, and establish session key
- Here's one example:



Diffie-Hellman

- Assume the following is hard:
  - you know g, p, and g^X mod p. What's X?



shared secret = g^{AB} mod p

IKE

- Authentication based on some sort of secret, and get session key
- Two “phases” (no reason to have 2nd phase)
- First phase has 8 different protocols
  - “aggressive mode” (3 msgs)
  - “main mode” (6 msgs, also does identity hiding and parameter negotiation)
  - For each of those, protocol based on one of 4 keys types:
    - shared secret
    - public signature key
    - public encryption key (old way)
    - public encryption key (revised way)
- Nobody had the energy to really review this

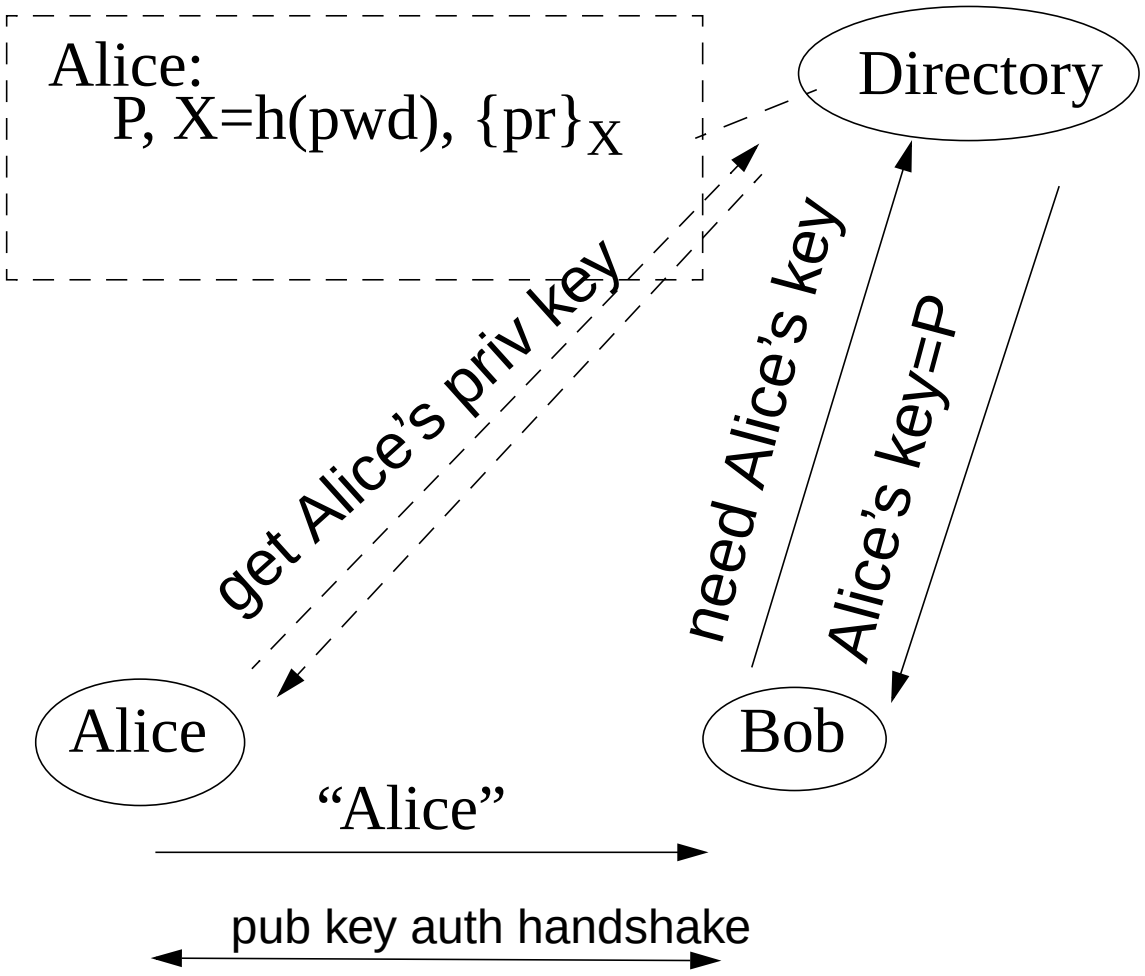
IKE, Secret Key variant

- When n choices, one is MUST
- For IKE, they chose main mode, pre-shared keys
- send ID in a key = f(shared key, other stuff)
- So....can’t decrypt to find out ID, unless you know who you’re talking to!
- OK...so spec says “ID must be IP address”
- So...why 6 message variant that hides IDs?
- Totally useless in “road warrior” case

How not to get any advantage from public keys

- What advantages should you get from PKI-type vs KDC-type system?
  - KDC is on-line, so vulnerable to network-based attacks
  - KDC must be available and give good performance at all times (so must be replicated, and all replicas physically secure)
  - KDC has database of all user secrets
- I'll tell you about a deployed system that had NO advantages over a secret-key system

System FOO


Alice:
P, X=h(pwd), {pr}X
Directory
get Alice's priv key
need Alice's key
Alice's key=P
Alice
Bob
"Alice"
pub key auth handshake

Summary

- The Internet has to be reliable now
- It has to be self-managing mostly
- Protocols have to be simple enough that multivendor implementations have some hope of working
- In the presence of failures, must at least be self-stabilizing
- Know the problem you're trying to solve before you try to solve it!