Myths, Missteps, and Folklore in Network Protocols

Radia Perlman
(radia.perlman@sun.com)
Why this talk?

• Learn from mistakes
• Understand oddities in today’s protocols
• Counteract “religion” aspect
  – Mark Twain: “It’s not what you don’t know that’ll get you. It’s what you do know that ain’t true”
• Be provocative. Start lively discussion
• Pet peeve: teach networking as a science, not like a trade school
Myths

• Technology that is most successful is the best technically
• Everything in standards makes sense
• If a conference or journal accepts paper A and rejects paper B, that means paper B is not as good as paper A
Example

- Sometimes you need to know the history in order to understand something
Topic 1: Bridges, Routers, and Switches, Oh My!

• Myths
  – bridges were designed before routers, because they are/were simpler
  – switches are a technology newer than bridges that make bridges obsolete
A bit of background

- ISO’s OSI Reference Model, network layers:
  - 1: physical
  - 2: neighbor-to-neighbor
  - 3: talk across multi-hop path
  - 4: end-to-end
A bit of background

- ISO’s OSI Reference Model
  - 1: physical
  - 2: neighbor-to-neighbor
  - 3: talk across multi-hop path
  - 4: end-to-end
  - 5 and above: boring
A bit of background

• ISO’s OSI Reference Model
  – 1: physical
  – 2: neighbor-to-neighbor
  – 3: talk across multi-hop path
  – 4: end-to-end
  – 5 and above: boring
• layer 1 relay: repeater
• layer 2 relay: bridge
• layer 3 relay: router
OK: so what is layer 2?

• If I ran the world
  – no such thing as layer 2 relay
• My discovery of true definition of “a layer 2 protocol”
OK: so what is layer 2?

- If I ran the world
  - no such thing as layer 2 relay
- My discovery of true definition of “a layer 2 protocol”
  - anything designed by a committee whose charter is to design a layer 2 protocol
So, where did bridges come from?
In the beginning...

- There were routers
- I was DECNnet routing architect
- Layer 3 requires endnode cooperation
  - put an envelope on the data, with source, destination, hop count, etc
  - Do additional things (e.g., ARP in IP)
  - Have an address that reflects topology
Then along came the EtherNET

- Fine technology
- But it’s not a network! It’s a link in a network
- Ethernet is multiaccess, so requires envelope with source and destination
- But it’s not a layer 3 protocol, e.g.,
  - addresses have no structure
  - no hop count
  - technology doesn’t scale
Confusion

• People designed things to work directly on Ethernet. Left out layer 3
• Press got excited about “Which will win? DECnet or Ethernet?”
• Explaining was hopeless
Problem Statement

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

- Listen promiscuously
- Learn location of source
- Forward based on learned location of destination
Basic bridge idea

Listen promiscuously
Learn location of source
forward based on learned location of destination
Need loop-free topology

• Can’t learn location of source if it’s in multiple directions
• loops are a disaster (no hop count, exponential proliferation)
Bridge loops

S

B1 B2 B3
Need spanning tree algorithm

- Finds loop-free spanning subset of current topology
A talks to X
Problems

• But suboptimal routing
  – Can’t have pairwise optimal with spanning tree
  – Concentrates traffic
• Also, temporary loops scary
  – conservative about turning on loops (I was REAL scared of temporary loops)
• And inherently fragile
  – Lost messages turn *on* links
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.
Bridges without Spanning Tree?

• Implementers wanted bridges as simple as possible. “Don’t allow loops”
• Felt a little bad about forcing them to do STP
• … Until, first customer site
First customer site
Bridge success

• Simple, fast, reliable, plug-and-play
• Routers have gotten better. Will bridges go away?
• No for subtle reason: IP needs address per link.
• Layer 3 doesn’t have to work that way
  – CLNP and DECnet
  – Bottom level of routing is a whole cloud with the same prefix
  – Routing is to endnodes inside the cloud
Hierarchy

One prefix per link

One prefix per campus
Advantages of lots of links in a prefix

- Zero configuration of routers inside campus
- Nodes can move within campus and keep address
- Address space doesn’t need to be chopped up
Plug for TRILL working group in IETF

- TRILL = TRansparent Interconnection of Lots of Links
- Use layer 3 routing, and encapsulate with a civilized header
- But still look like a bridge from the outside
R-Bridges/TRILL

• Compatible with today’s bridges and routers
• Like routers, terminate bridges’ spanning tree
• Like bridges, glue LANs together to create one IP subnet (or for other protocols, a broadcast domain)
• Like routers, optimal paths, fast convergence, no meltdowns
• Like bridges, plug-and-play
RBridging layer 2

- Link state protocol among Rbridges (so know how to route to other Rbridges)
- Like bridges, learn location of endnodes from receiving data traffic
- But since traffic on optimal paths, need to distinguish originating traffic from transit
- So encapsulate packet
“Layer 2” routing

• Think of it just like routers
  – At each hop, add a layer 2 header to get to the next hop RBridge
  – The “destination” is the egress RBridge
• First RBridge
  – Looks up destination endnode, finds egress RB
  – Adds shim header, forwards to egress RB
• Egress RBridge decapsulates
Rbridging
Encapsulation Header

<table>
<thead>
<tr>
<th>S=Xmitting Rbridge</th>
<th>hop count</th>
<th>original pkt (including L2 hdr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D=Recving Rbridge</td>
<td>In/out RBridge</td>
<td></td>
</tr>
</tbody>
</table>

pt=“transit”

Outer header: new p-t: otherwise, ordinary Ethernet hdr
Shim: hop count, ingress and egress RBridge
To make shim shorter, dynamically acquire 2-byte RBridge nicknames
Flooded traffic

• Some traffic needs to be sent to all links
  – Unknown destinations
  – Multicast traffic

• Could use a single spanning tree
  – Spanning tree computed from link state database (not separate spanning tree protocol)

• But we decided on per-ingress trees..And actually a bit more flexible than that
Other subtleties

• VLANs
• Shared VLAN learning
• IP multicast
• ARP/ND optimization
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 effectively.
   Without a common spanning tree.

Ray Perlner
What are switches?

- Myth: Ethernet is wildly successful
- Reality: Ethernet (CSMA/CD) doesn’t exist anymore!
- Panel: bus (Ethernet) vs ring (vs star)
Stars
Plug star into another star
Stars

- Start with multi-port repeater
- Then notice that hub can be smarter. Store and forward, learn location of sources.
- Then notice can cascade them.
- Should run spanning tree.
- We’ve reinvented the bridge!
- “Switched Ethernet”: pt-to-pt links with bridges!
- Ports don’t need to be the same speed!
New topic: What’s with IPv6?

• Why should it have taken over a decade to design?
• What’s really new?
• IAB, in 1992, realized IP addresses were too small, and said we should replace IP with CLNP, ISO’s “Connectionless Network Protocol”
• CLNP is like IP, with 20 byte addresses
• At the time, CLNP fully implemented, mature standard, enthusiasm in Europe
IPv6 reality

• Much harder to change Internet to bigger addresses now than in 1992
  – Internet much larger
  – More “mission-critical”

• Less incentive now
  – delay necessitated inventions like DHCP, NAT
IPv6 myths

• It’s not a replacement of IP. It’s just a “new version” of IP

• Security is built into IPv6. It’s just an add-on to IPv4
New Topic: Bad “Framework”
Multicast

- Ethernet: falls out of technology
- ATM: create VC. “Add member”
IP Multicast

• Idea: make it look “just like Ethernet”
  – globally unique multicast addresses
    • IP address 32 bits, top 4 bits=1110
  – anyone can request to listen. anyone can send without being a member

• So, start out with unchangeable “model”
  – signalling protocol to inform local rtr to send G
Problem: Can’t be implemented

• various attempts:
  – flood and prune
    • send all data everywhere, in case someone in Albania wants to listen
    • if not interested, send “prune”
    • keep track of all (S,G) pairs nbr NOT interested in
  – MOSPF
    • routers keep track of all listeners for all groups
IP Multicast attempts

• Tree building like with ATM
  – send join towards Root
  – create tree

• Problems:
  – who is Root for G?
    • unscalable intradomain protocol to select a Root-candidate for G
  – how to administer addresses
IP Multicast

- So, came up with unscalable complex intradomain
- Then MSDP to piece domains together
How IP Multicast should look

• Two types
  – finding something (low bandwidth, can’t set up tree). Just flood with RPF
  – conference call, etc. Find host H. Build tree to H. Have address of group be (H,G), where G only has to be unique to H
New topic: simple things people screw up

• Parameters. Need to set in running net!
  – better if plug and play
  – hopefully not there because committee couldn’t decide on a value
  – Sometimes parameters must be compatible:
    • hello timer vs dead timer
  – IS-IS strategy: tell nbr
  – OSPF: refuse to talk unless equal!
  – Spanning tree: use root bridge’s values
Forward compatibility

• TLV encoding: so can define new fields, and have old implementations skip them
  – L must be consistent
  – BOOTP “vendor specific” field
Version number

• What’s the difference between a new protocol and a new version of a protocol?
Version number

• What’s the difference between a new protocol and a new version of a protocol?
• For instance, is IPv6 a new version of IP, whereas CLNP would have been “replacing it”?
What’s a “different” protocol

• Reasonable definition: If has a different Ethertype
• New version: share the same Ethertype
• If you ever make the format incompatible, change the version number
• Which means you can’t just specify what you set the field to, but that you should drop something with an unrecognized version number
• Some fancy things you could do (major|minor; bit for “I could support a higher version)
Version numbers, reserved fields

• Lots of protocols don’t say what to do
• So difficult to do new version
• SSL
  – version 3 totally redid packet format
  – moved version number field!
  – So have to send version 2 (0.2) msg with 3 (3.0) in version number field
  – version 2 nodes happy to accept version 768!
Version numbers, reserved fields

- Lots of protocols don’t say what to do
- So difficult to do new version
- IPv4
  - Just says “set the field to 4”
  - So you can’t send an IPv6 packet to an IPv4 node
  - So IPv6 is a new protocol, not a new version of IPv4
So you’d assume they’ve learned their lesson for IPv6

• No…the spec says “fill in this field as 6”
SSL

• Version 3 completely changed the format from version 2
SSL

• Version 3 completely changed the format from version 2
• And even moved the version field!
SSL

• Version 3 completely changed the format from version 2
• And even moved the version field!
• How can this work?
  – Have to send first packet in v2 format, specifying version number as 3
  – Ironically, field is 2 bytes: version 2 = (0,2)
  – Version 3 = (3,0)
  – So version 2 node sees version 768, doesn’t even blink!
Summary

• We need to have protocol designers humble enough to learn from previous protocols
• We need to learn from more than RFCs (which don’t give rationale, just operation)
• If things aren’t simple they won’t work
• Know what problem you’re trying to solve before you try to solve it!