Spanning Tree Protocol
How to allow redundancy (loops) in the link layer topology.

Brad Smith
Reminders

• Guest lectures
  – Thursday 5/21 – Jim Warner (Senior Network Engineer) on cable plant
  – Tuesday 5/26 - Prof Garcia-Luna on DUAL/EIGRP and current research.

• Dates
  – RIP lab due Sunday
  – Intra-domain routing quiz and STP exercise Thursday
  – Dordal Section 2.5 for STP
Projects

• Status report – e-mail to me (Brad) by “midnight”
  – Due Tuesday, 5/19 (2 weeks)
  – Summary of how things are going... specifically any problems you’ve run into.

• Volunteers for early presentations
  – 10 min presentation, 5 min Q&A
  – 5 Tuesday (6/2), 6 Thursday (6/4), 10 Thursday (6/9... finals slot)
  – Tue/Thu presentations get special grading consideration... volunteers?

• Deliverables
  – Presentation – 10 mins w/ 5 mins for questions
  – Turn in – by “midnight” the day of our final slot (June 9th)
    • Slides from presentation
    • Paper describing
      – Technology covered in the lab
      – Lessons learned
    • Lab, answer key, netref content
Ethernet

• Media Access Control
  – Original Ethernet – CSMA/CD
  – Repeaters, hubs, bridges, and switches

• **Problem:** one broadcast domain per switch.
  – **Solution:** Virtual LANs (VLANs)

• **Problem:** loops in the topology.
  – **Solution:** spanning-tree protocol (STP)
Ethernet

• Media Access Control
  – Original Ethernet – CSMA/CD
  – Repeaters, hubs, bridges, and switches

• Problem: one broadcast domain per switch.
  – Solution: Virtual LANs (VLANs)

• Problem: loops in the topology.
  – Solution: spanning-tree protocol (STP)
Redundancy is Good

- Selective forwarding improves bandwidth utilization
  - Multiple simultaneous transmissions
- Also allows for improved robustness through redundant paths
- Challenge is redundancy means means loops
L2 Loops - Flooded unicast frames

- Bridge loops can occur any time there is a redundant path or loop in the bridge network.
- The switches will flip flop the bridging table entry for Station A (creating extremely high CPU utilization).
- Bridge Loops can cause:
  - Broadcast storms
  - Duplicate Ethernet frames
  - MAC address table instability
Unknown Unicast

Switch Moe learns Kahn’s MAC address.

SAT (Source Address Table)
Port 4: 00-90-27-76-96-93

Thanks to Rick Graziani @ Cabrillo for this animation.
Unknown Unicast

Destination MAC is an unknown unicast, so Moe floods it out all ports.

SAT (Source Address Table)
Port 4: 00-90-27-76-96-93

Thanks to Rick Graziani @ Cabrillo for this animation. 00-90-27-76-5D-FE
Unknown Unicast

Switch Larry records the Source MAC of the frame twice.

Thanks to Rick Graziani @ Cabrillo for this animation.
Unknown Unicast

Switch Larry floods the unknown unicast out all ports, except the incoming port.

SAT (Source Address Table)
Port 4: 00-90-27-76-96-93

SAT (Source Address Table)
Port A: 00-90-27-76-96-93

Thanks to Rick Graziani @ Cabrillo for this animation. 00-90-27-76-5D-FE
Unknown Unicast

Switch Moe receives the frame, changes the MAC address table with newer information and floods the unknown unicast out all ports.

Thanks to Rick Graziani @ Cabrillo for this animation.
Unknown Unicast

And the cycle continues!

SAT (Source Address Table)
Port 4: 00-90-27-76-96-93
Port 1: 00-90-27-76-96-93

Thanks to Rick Graziani @ Cabrillo for this animation.

00-90-27-76-5D-FE
Spanning Tree – Only for Loops

• Loops may occur in your network as part of a design strategy for redundancy.

• STP is not needed if there are no loops in your network.

• However, DO NOT disable STP!

• Loops can occur accidentally from network staff or even users!

Two users interconnecting the switches in their cubicles.
DO NOT disable STP! 😊

- One port on SoE switch, one on campus switch.
- STP and DHCP leaked between domains.
- Ports either mis-labeled or not labeled
  - In the chase
  - On the switches
- Switch config showed ports as empty, but were enabled
- Even STP couldn’t help in this situation😊!
Introduction

• Spanning Tree Protocol approach is to avoid loops by building a tree
  – Frames received on a tree port are forwarded out all other tree ports

• What does each switch need to know to implement a tree?
  – What is the root of the tree
  – Which of its ports are on the tree

• What are the characteristics of these ports?
  – The port a neighbor reports the “best” path to the root on ("root port")
  – The ports I report the “best” path to the root on ("designated ports")

• What is left... ports that are not on the tree
  – “Blocked ports”

• First constraint
  – *There is exactly one designated port on each segment (next slide)*

• Why do we need a root port if it has no significance in how traffic is handled?
“Exactly one designated port...”
“Exactly one designated port...”

Should be like this...
“Exactly one designated port...”
“Exactly one designated port...”

But instead is like this...
“Exactly one designated port...”

Frame comes in from a station...
“Exactly one designated port...”

Forwarded out other tree ports...
“Exactly one designated port...”

Forwarded out other tree ports...
“Exactly one designated port...”

Forwarded out other tree ports...
“Exactly one designated port...”

Forwarded out other tree ports...
“Exactly one designated port...”

Forwarded out other tree ports...
“Exactly one designated port...”

And so on...
Spanning Tree Protocol

- STP computes a tree that covers the full network graph.
- Exchanges Bridge Protocol Data Units (BPDUs) that describe paths.
- Defines an ordering over BPDUs that ranks shortest path to a “root” switch as “better”
- Uses this ordering to identify, for each switch, the ports it should
  – forward traffic between, and
  – ignore traffic on.

- Which ports are on the tree?
Goal (Invariants)

- Ports on the tree are
  - **Root ports**: the port on the switch with the shortest path to the root.
  - **Designated ports**: ports on the switch where the switch has the shortest path to the root.

- Each segment has one designated port

- **Each switch has at most 1 root port**
  - Which switch will have no root ports?

- What is the error in this picture?
How This Works...

- BPDUs describe a path

- Imagine BPDU’s have two fields:
  - Root switch ID
  - Path cost to Root switch from sender

- Define ordering between two BPDUs as
  - BPDU with smallest Root switch ID is “better”
  - If the Root switch ID’s are equal, the BPDU with lowest root path cost is “better”

- Choose path with “better” BPDU

- D receives
  - [A,2] and [B,0] from B
  - [A,1] and [C,0] from C

- Which path to A does D select?
Real BPDUs

• BPDUs need more fields...
  – Root Bridge ID
  – Path Cost to Root Bridge
  – Sender BID
    – (Sender) Port Priority
    – (Sender) Port ID

• What happens if both B and C think their port on link B-C is designated?

• How does the “Sender BID” field solve this?

• Sender BID is the tie-breaker
Real BPDUs

• BPDUs need even more fields...
  – Root Bridge ID
  – Path Cost to Root Bridge
  – Sender BID
  – (Sender) Port Priority
  – (Sender) Port ID

• What do the port fields do? – 1st Scenario
  – Provides control and determinism
  – B marks port y designated and z blocked
  – Port Priority allows admin to control segment chosen by B (e.g. primary 10gig with 1gig backup)
  – Port ID provides determinism... selection of a port among those with same priority is not random.
Real BPDUs

- BPDUs need even more fields...
  - Root Bridge ID
  - Path Cost to Root Bridge
  - Sender BID
  - (Sender) Port Priority
  - (Sender) Port ID

- What do the port fields do? – 2nd Scenario
  - Provides control and determinism
  - B will mark all ports as designated
  - C will mark z as blocked and y as designated
  - Port Priority allows admin to control segment chosen by C (e.g. primary 10gig with 1gig backup)
STP – The Goal

*Identify the ports that form the switches into a shortest-path spanning tree.*
STP – The Approach

• Switches exchange messages describing their best *paths* to the root

• These messages are called *Bridge Protocol Data Units*
  – Root ID, Root Path Cost, Sender ID, Port Priority, Port ID
  – Set by *sending* switch (Sender ID, Port Priority and ID of sender)!
  – Sorted by fields, in order (Root ID, then Path Cost, etc.).

• **Root** port is port with “best” path to root (smallest BPDU) *on switch*
  – Root switch doesn’t have a root port.

• **Designated** port is port with “best” path to root (smallest BPDU) *on link*

• All other ports are **Blocked**

• The **Root** and **Designated** ports form the switches into a tree.
  – Each *segment* has *exactly* 1 designated port and *0 or more* root ports
  – Each *switch* has *at most* 1 root port and *0 or more* designated ports
STP – The Algorithm

State:
• A list of “Port BPDUs” set to the smallest BPDU sent/received on a port.

Initialization:
• Send BPDU(Root=Me, Cost=0, Sender=Me, ...) on each port and update Port BPDUs

On receipt of a BPDU:
• If BPDU < current Port BPDU, update Port BPDU
• Add link cost to BPDU cost
• For all ports where BPDU < current Port BPDU (sort by Root then Cost then Sender...),
  — Send: BPDU(Root=BPDU Root, Cost=BPDU Cost, Sender=Me, ...)
  And update the Port BPDU.

Periodically update port status:
• If smallest (Port BPDU + link cost) on switch was not sent by me, label port Root
• If Port BPDU was sent by me, label port Designated
• Label all other ports Blocked

On receipt of data frame on Root or Designated port:
• Send out all other Root or Designated ports

Not complete... topology changes!
Root vs Designated Port Decision

• Different BPDU’s used for Root vs. Designated port selection
  – Port BPDU used for Designated port decision
  – Port BPDU + link cost used for Root port decision

• Reflects difference in Root vs. Designated port designation
  – Designated port is the best port on a link...
    • ...therefore it doesn’t consider the link’s cost
  – Root port is the best port on a switch...
    • ...therefore it does consider the link’s cost
Port BPDUs

• All port BPDU’s on a segment are the same!

• 2 common mistakes
  – Add link cost on receipt of BPDU vs transmit
  – Use receiving port ID vs sender port ID
  – Special cases of different Port BPDUs on one segment...
Add link cost on transmit

State:
• A list of “Port BPDUs” set to the smallest BPDU sent/received on a port.

Initialization:
• Send BPDU(Root=Me, Cost=0, Sender=Me, ...) on each port and update Port BPDUs

On receipt of a BPDU:
• If BPDU < Port BPDU, update Port BPDU
• Add link cost to BPDU cost
• For all ports where BPDU < Port BPDU (sort by Root then Cost then Sender...), send:
  BPDU(Root=BPDU Root, Cost=BPDU Cost, Sender=Me, ...)
  And update the Port BPDU.

Periodically update port status:
• If smallest (Port BPDU + link cost) on switch was not sent by me, label port Root
• If Port BPDU was sent by me, label port Designated
• Label all other ports Blocked

On receipt of data frame on Root or Designated port:
• Send out all other Root or Designated ports

Not complete... topology changes!
Correct – add link cost on transmit
Correct – add link cost on transmit

A sends BPDU.

Switches and Ports:
- Switch A: Port 1: [A,0,A,1], Port 2: [A,0,A,2]
- Switch B: Port 1: [A,0,A,1]
- Switch C: Port 2: [A,0,A,2]
Correct – add link cost on transmit

B & C receive BPDUs.
Correct – add link cost on transmit

B & C send BPDUs.
Correct – add link cost on transmit

C receives BPDU.
Correct – add link cost on transmit
Correct – add link cost on transmit

[Diagram showing network topology with labels and costs]

Switch A
- Port 1: \([A,0,A,1]/D\)
- Port 2: \([A,0,A,2]/D\)

Switch B
- Port 1: \([A,0,A,1]/R\)
- Port 2: \([A,5,B,2]\)

Switch C
- Port 1: \([A,0,A,2]/R\)
- Port 2: \([A,5,B,2]\)
Correct – add link cost on transmit

Switch A | Switch B | Switch C
--- | --- | ---
Port 1 | [A,0,A,1]/D | [A,0,A,1]/R | [A,0,A,2]/R
Port 2 | [A,0,A,2]/D | [A,5,B,2]/D | [A,5,B,2]/B

RootID, RootPathCost, SenderID, PortID)/Status

(A,0,A,1) Root
(A,0,A,2) Designated
(A,5,B,2) Blocked
Correct – add link cost on transmit

Port BPDUs the same on each segment!
Incorrect – add link cost on receive

![Diagram showing network topology and port costs.

- Switch A
  - Port 1
  - Port 2
- Switch B
  - Port 1
- Switch C
  - Port 1

<table>
<thead>
<tr>
<th>Port</th>
<th>Switch A</th>
<th>Switch B</th>
<th>Switch C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Port 1 cost
- Port 2 cost

Root
Designated
Blocked

Port 1
Port 2
Incorrect – add link cost on receive

A sends BPDU.
Incorrect – add link cost on receive

B & C receive BPDUs.
Incorrect – add link cost on receive

B & C send BPDUs.
Incorrect – add link cost on receive

C drops B’s BPDU because [A,5,C,2] < [A,9,B,2]!
Incorrect – add link cost on receive

Diagram:

- Switch A
  - Port 1: [A,0,A,1]/D
  - Port 2: [A,0,A,2]/D
- Switch B
  - Port 1: [A,5,A,1]
  - Port 2: [A,5,B,2]
- Switch C
  - Port 1: [A,5,A,2]
  - Port 2: [A,5,C,2]

Legend:
- Root
- Designated
- Blocked

[RootID, RootPathCost, SenderID, PortID]/Status

Blue = Error
Incorrect – add link cost on receive

![Diagram of network with ports and status codes]

**Blue** = Error

<table>
<thead>
<tr>
<th>[RootID, RootPathCost, SenderID, PortID]/Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Switch A</strong></td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Port 1</td>
</tr>
<tr>
<td>Port 2</td>
</tr>
</tbody>
</table>
Incorrect – add link cost on receive

Blue = Error

[RootID, RootPathCost, SenderID, PortID]/Status

<table>
<thead>
<tr>
<th>Port 1</th>
<th>Switch A</th>
<th>Switch B</th>
<th>Switch C</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A,0,A,1]/D</td>
<td>[A,5,A,1]/R</td>
<td>[A,5,A,2]/R</td>
<td></td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]/D</td>
<td>[A,5,B,2]/D</td>
<td>[A,5,C,2]/D</td>
</tr>
</tbody>
</table>
Incorrect – add link cost on receive

Port BPDUs differ on each segment!

LOOP!
Use *sender port ID*

- Switches exchange messages describing their best *paths* to the root.
- These messages are called **Bridge Protocol Data Units**
  - Root ID, Root Path Cost, Sender ID, Port Priority, Port ID
  - *Set by sending switch* (Sender ID, Port Priority and ID of sender)!
  - Sorted by fields, in order (Root ID, then Path Cost, etc.).
- **Root** port is port with “**best**” path to root (smallest BPDU) *on switch*
  - Root switch doesn’t have a root port.
- **Designated** port is port with “**best**” path to root (smallest BPDU) *on link*
- All other ports are **Blocked**
- The **Root** and **Designated** ports form the switches into a tree.
Correct – Sender Port ID

Switch A

<table>
<thead>
<tr>
<th>Port 1</th>
<th>Port 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Switch B

<table>
<thead>
<tr>
<th>Port 1</th>
<th>Port 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Port 1: 5
Port 2: 5

Switches:
- A
- B

Root
Designated
Blocked
Correct – Sender Port ID

A sends BPDUs.

<table>
<thead>
<tr>
<th>[RootID, RootPathCost, SenderID, PortID]/Status</th>
<th>Switch A</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>[A,0,A,1]</td>
<td></td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]</td>
<td></td>
</tr>
</tbody>
</table>
Correct – Sender Port ID

B receives BPDUs.

<table>
<thead>
<tr>
<th>[RootID, RootPathCost, SenderID, PortID]/Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Switch A</strong></td>
</tr>
<tr>
<td>Port 1</td>
</tr>
<tr>
<td>Port 2</td>
</tr>
</tbody>
</table>
Correct – Sender Port ID

[RootID, RootPathCost, SenderID, PortID]/Status

<table>
<thead>
<tr>
<th></th>
<th>Switch A</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>[A,0,A,1]/D</td>
<td>[A,0,A,2]</td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]/D</td>
<td>[A,0,A,1]</td>
</tr>
</tbody>
</table>
Correct – Sender Port ID

Switch A

Port 1: [A,0,A,1]/D
Port 2: [A,0,A,2]/D

Switch B

Port 1: [A,0,A,2]/B
Port 2: [A,0,A,1]/R

[RootID, RootPathCost, SenderID, PortID]/Status

A

1

2

B

1

2

5

5

Root

Designated

Blocked
Correct – Sender Port ID

Port BPDUs the same on each segment!

<table>
<thead>
<tr>
<th>[RootID, RootPathCost, SenderID, PortID]/Status</th>
<th>Switch A</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>[A,0,A,1]/D</td>
<td>[A,0,A,2]/B</td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]/D</td>
<td>[A,0,A,1]/R</td>
</tr>
</tbody>
</table>

A controls port.
Incorrect – Sender Port ID

Switch A

<table>
<thead>
<tr>
<th>Port 1</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Switch B

<table>
<thead>
<tr>
<th>Port 1</th>
<th>Switch A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Root

Designated

Blocked
Incorrect – Sender Port ID

A sends BPDUs.

[RootID, RootPathCost, SenderID, PortID]/Status

<table>
<thead>
<tr>
<th></th>
<th>Switch A</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>[A,0,A,1]</td>
<td></td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]</td>
<td></td>
</tr>
</tbody>
</table>
Incorrect – Sender Port ID

B receives BPDUs.

<table>
<thead>
<tr>
<th>RootID, RootPathCost, SenderID, PortID/Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>[RootID, RootPathCost, SenderID, PortID]/Status</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Switch A</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>[A,0,A,1]</td>
<td>[A,0,A,1]</td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]</td>
<td>[A,0,A,2]</td>
</tr>
</tbody>
</table>

Root
Designated
Blocked
Incorrect – Sender Port ID

<table>
<thead>
<tr>
<th>[RootID, RootPathCost, SenderID, PortID]/Status</th>
<th>Switch A</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1 [A,0,A,1]/D</td>
<td>[A,0,A,1]</td>
<td>[A,0,A,1]</td>
</tr>
<tr>
<td>Port 2 [A,0,A,2]/D</td>
<td>[A,0,A,2]</td>
<td>[A,0,A,2]</td>
</tr>
</tbody>
</table>

Blue = Error
Incorrect – Sender Port ID

[RootID, RootPathCost, SenderID, PortID]/Status

<table>
<thead>
<tr>
<th></th>
<th>Switch A</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>[A,0,A,1]/D</td>
<td>[A,0,A,1]/R</td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]/D</td>
<td>[A,0,A,2]/B</td>
</tr>
</tbody>
</table>

Blue = Error
Incorrect – Sender Port ID

Port BPDUs differ on each segment!

<table>
<thead>
<tr>
<th></th>
<th>Switch A</th>
<th>Switch B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>[A,0,A,1]/D</td>
<td>[A,0,A,1]/R</td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]/D</td>
<td>[A,0,A,2]/B</td>
</tr>
</tbody>
</table>

Blue = Error

A doesn’t control port.
Summary - STP Invariants

• On a given segment, *all port BPDUs are the same!*

• As a result,
  – Each **segment** has **exactly 1** designated port and **0 or more** root ports
  – Each **switch** has **at most 1** root port and **0 or more** designated ports
Bridge ID

- 802.1D BID includes
  - 2 bytes priority
- Per VLAN Spanning Tree (PVST) includes
  - 4 bits priority and
  - 12 bits Extended System ID (VLAN)
## Priority = Priority (Default 32,768) + VLAN

Access2# `show spanning-tree`

<table>
<thead>
<tr>
<th>VLAN0001</th>
<th>Spanning tree enabled protocol ieee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root ID</td>
<td>Priority 24577</td>
</tr>
<tr>
<td>Address</td>
<td>000f.2490.1380</td>
</tr>
<tr>
<td>Cost</td>
<td>23</td>
</tr>
<tr>
<td>Port</td>
<td>1 (FastEthernet0/1)</td>
</tr>
<tr>
<td>Hello Time</td>
<td>2 sec   Max Age 20 sec Forward Delay 15 sec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bridge ID</th>
<th>Priority</th>
<th>(priority 32768 sys-id-ext 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>0009.7c0b.e7c0</td>
<td></td>
</tr>
<tr>
<td>Hello Time</td>
<td>2 sec   Max Age 20 sec Forward Delay 15 sec</td>
<td></td>
</tr>
<tr>
<td>Aging Time</td>
<td>300</td>
<td></td>
</tr>
</tbody>
</table>

VLAN0010

| Spanning tree enabled protocol ieee |
| Root ID  | Priority 4106                     |
| Address  | 000b.fd13.9080                    |
| Cost     | 19                                |
| Port     | 1 (FastEthernet0/1)               |
| Hello Time | 2 sec   Max Age 20 sec Forward Delay 15 sec |

<table>
<thead>
<tr>
<th>Bridge ID</th>
<th>Priority</th>
<th>(priority 32768 sys-id-ext 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>0009.7c0b.e7c0</td>
<td></td>
</tr>
<tr>
<td>Hello Time</td>
<td>2 sec   Max Age 20 sec Forward Delay 15 sec</td>
<td></td>
</tr>
<tr>
<td>Aging Time</td>
<td>300</td>
<td></td>
</tr>
</tbody>
</table>
Path Cost – Revised Spec (Non-Linear)

<table>
<thead>
<tr>
<th>Link Speed</th>
<th>Cost (Revised IEEE Spec)</th>
<th>Cost (Previous IEEE Spec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Gbps</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1 Gbps</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>100 Mbps</td>
<td>19</td>
<td>10</td>
</tr>
<tr>
<td>10 Mbps</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

• IEEE modified the most to use a non-linear scale with the new values of:
  – 4 Mbps 250 (cost)
  – 10 Mbps 100 (cost)
  – 16 Mbps 62 (cost)
  – 45 Mbps 39 (cost)
  – 100 Mbps 19 (cost)
  – 155 Mbps 14 (cost)
  – 622 Mbps 6 (cost)
  – 1 Gbps 4 (cost)
  – 10 Gbps 2 (cost)

• You can change the path cost by modifying the cost of a port.
• Exercise caution when you do this!
• BID and Path Cost are used to develop a loop-free topology.
• Coming very soon!
Spanning tree transitions each port through several different states.

- **Blocking** (loss of BPDU detected) (max age = 20 sec)
- **Listening** (forward delay = 15 sec)
- **Learning** (forward delay = 15 sec)
- **Forwarding**

From Blocking to Forwarding:
20 sec + 15 sec + 15 sec = 50 seconds
Spanning Tree Port States

- **Blocking**
  - Only receive BPDUs

- **Listening**
  - Send and receive BPDUs
  - Builds Root/Designated/Blocked assignments

- **Learning**
  - Receive user frames (Address Learning)
  - Populates Source Address Table

- **Forwarding**
  - Send and receive user frames
Review

• Redundancy
  – Is good because it provides robustness
  – Is bad because it can result in looping and, therefore, switch and link overload

• STP
  – Eliminates loops by computing a tree over the switches in a subnet
  – BPDUs describe path
  – BPDU ordering identifies “good” paths
  – Ordering used to classify bridge ports
    • **Root** port has shortest path to root bridge
    • **Designated** port has shortest path to root on a segment
    • All other paths labeled as **Blocked**
  – Frames are only accepted and transmitted on Root and Designated ports
Review

• BPDUs sorted by five fields (in order)...
  – Root Bridge ID
  – Path Cost to Root Bridge
  – Sender BID
  – (Sender) Port Priority
  – (Sender) Port ID

• Bridge ID includes Priority and “extended system ID” (VLAN ID) fields
TRILL

http://www.ethernetsummit.com/English/Collaterals/Proceedings/2013/20130403_A104_Eastlake.pdf
Layer 2 Security Features

- **DHCP Snooping**: Ports designated as "untrusted" are not permitted to send DHCP server messages. Alternately, unauthorized DHCP servers on "untrusted" ports cannot see client DHCP solicitations coming from other untrusted ports.
  - Protects against rogue DHCP servers (they either can’t send or receive DHCP messages).

- **Dynamic ARP inspection**: The switch builds a list of MAC addresses on each port by inspecting DHCP offers passing to the ports. Any ARP replies not matching a MAC address in the switch's lease table are dropped and not forwarded to the network.
  - Protects against ARP masquerading (IP-to-MAC mapping not seen in DHCP Offer).

- **IP Source guard**: The switch builds a list of IP addresses on each port by inspecting DHCP offers passing to the ports. Any packets not matching the IP address in the switch's lease table are dropped and not forwarded to the network. See RFC4388 and RFC6148 for a description of this functionality.
  - Protects against IP spoofing (using an address not previously assigned by DHCP).
The End