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

Brad Smith
Logistics

• Projects
  • Feedback...
  • Status reports due Tuesday, 5/23 (2 weeks)... summary of how things are going... specifically any problems you’ve run into.

• Thursday – class in E2 180 (the “Simularium”)
  – Spanning Tree Protocol exercise and quiz

• Sunday
  – RIP lab due

• Next Tuesday
  – Inter-Domain Routing and BGP lecture

• Readings
  • Dordal: BGP – Chapter 10
  • “BGP Routing Policies in ISP Networks” (class web site)
Class on Thursday in E2 180 (the “Simularium”)
Project Deliverables

- Presentation… 10 mins w/ some time for questions
  - Soon I’ll be asking for volunteers for early presentations (Tue/Thu of the last week)… *get special grading consideration*. Think about it…

- Turn in… by the day of our final slot (Thursday, June 15th)
  - Slides from presentation
  - Paper describing
    - Technology covered in the lab
    - Lessons learned
  - Lab, answer key, netref content

- Remember… links to command references are on web site…
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 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
- Flooding in link layer causes exponential explosion in frames...
Unknown Unicast

Switch Moe learns Kahns’ 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.
Unknown Unicast

Switch Larry records the Source MAC of the frame twice.

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

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

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.

00-90-27-76-96-93

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😊!
STP – The Goal

Identify the ports that form the switches into a shortest-path spanning tree.
Goal of STP is to build a tree...

- 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*
Goal is to build a tree...

- 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*
Goal is to build a tree...

- 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*

No loops!

```
    Root
    +---
    |   +---
    |   |   +---
    |   |   |   No loops!
    |   |   +---
    |   +---
    +---
```
How identify “on-tree” ports?

• Pick the root of the tree.

• Select “on-tree” ports relative to this root...
  – The port where a neighbor has the “best” path to the root
    • The ROOT port
  – The ports where I have the “best” path to the root
    • DESIGNATED ports

• What is left?
  – Ports that are not on the tree... BLOCKED ports
How identify “on-tree” ports?

The port where a neighbor has the “best” path to the root.
How identify “on-tree” ports?

The ports where I have the “best” path to the root
How identify “on-tree” ports?

The ports that are left...
How compute root & on-tree ports?

- By exchanging messages called *Bridge Protocol Data Units (BPDUs)*
- BPDUs describe the *best path* from the *sender* to the *root switch*
  - Root ID, Root Path Cost, Sender ID, Port Priority, Port ID
  - Set by *sending* switch (i.e. Sender ID, Port Priority and ID of sender)!
- BPDUs compared by fields, in order (Root ID, then Root Path Cost, etc.).
- Start by switches assuming they are root
  - Send [Root ID = **ME**, Root Path Cost = 0, Sender = **ME**, Port Priority, Port ID]
- Each switch saves best BPDU heard (received or sent) on each port
  - Called *Port BPDU*
- **REMEMBER:** BPDU describes best path from SENDING switch
How compute root & on-tree ports?

• 2 cases for port BPDU
  – I sent it - I am best switch on that LAN segment
    • Port marked DESIGNATED
  – I didn’t send it - another switch is best on that LAN segment
    • If such a port has minimum \textit{port BPDU cost + link cost} mark it as \textbf{ROOT}
    • Otherwise mark it as \textbf{BLOCKED} (not on-tree)

• 2 cases for switches
  – No \textbf{ROOT} port – I am root switch and \textit{my root path distance is 0!}
  – One \textbf{ROOT} port – I am not root switch and \textit{my root path distance is root port BPDU cost + the link cost for the root port.}

• Traffic is forwarded out \textbf{ROOT} and DESIGNATED (“on-tree”) ports.

• \textit{Why do we need a root port if it has no significance in how traffic is handled?}
How This Works...

• BPDUs describe a path

• Imagine simple BPDU’s with just 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] from B
  – [A,1] from C

• Which path to A does D mark as root port?
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? – 1\textsuperscript{st} Scenario
  – Without Port Priority/ID
    • Both x and y marked designated... \textit{problem!}
  – With Port Priority/ID
    • B marks port y designated and z blocked (\textit{why?})
  – Provide control and determinism
    • Port ID provides \textit{determinism}... selection of a port among those with same priority is not random.
    • Port Priority allows admin to \textit{control} segment chosen \textit{by B} (e.g. primary 10gig with 1gig backup)
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
  – Without Port Priority/ID
    • Both ports on B designated, both ports root on C..?
  – With Port Priority/ID
    • B will mark all ports as designated
    • C will mark z as blocked and y as root
  – Provides control and determinism
    • Port Priority allows admin to control segment chosen by C (e.g. primary 10gig with 1gig backup)
Interesting Invariants

• How many root ports on a switch?
  – 0 (root switch) or 1

• How many root ports on a segment?
  – 0 (all connected switches have better paths) or more

• How many designated ports on a switch?
  – 0 (better switches on all non-root ports) or more

• How many designated ports on a segment?
  – Exactly 1
    – This is what guarantees no looping!
“Exactly one designated port...”

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

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

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

But if 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...
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 transmission/reception of a BPDU:
• If BPDU < current Port BPDU, update Port BPDU
• Add link cost to BPDU cost (BPDU’)
• 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 Port BPDU was sent by me, label port Designated
• If smallest (Port BPDU + link cost) on switch was not sent by me, label port Root
• 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...
    • ...the cost used in selecting it is my root path cost
    • ...therefore it does consider the link’s cost
Port BPDU Invariant

• 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 BPDU's 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 transmission/reception of a BPDU:
- If BPDU < current Port BPDU, update Port BPDU
- *Add link cost to BPDU cost (BPDU’)*
- 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 Port BPDU was sent by me, label port Designated
- If smallest (Port BPDU + link cost) on switch was not sent by me, label port Root
- 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

A

Switch A  Switch B  Switch C

Port 1

Port 2
Correct – add link cost on transmit

A sends BPDU.
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

Switch A | Switch B | Switch C
---|---|---
Port 1 | [A,0,A,1]/D | [A,0,A,1] | [A,0,A,2]
Port 2 | [A,0,A,2]/D | [A,5,B,2] | [A,5,B,2]
Correct – add link cost on transmit

A          B          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]   [A,5,B,2]

[RootID, RootPathCost, SenderID, PortID]/Status
Correct – add link cost on transmit

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]/D

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

[RootID, RootPathCost, SenderID, PortID]/Status
Correct – add link cost on transmit

Port BPDUs the same on each segment!

NO LOOP!
Incorrect – add link cost on receive

Switch A | Switch B | Switch C
--- | --- | ---
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

Switch A

Root: A
Designated: B
Blocked: C

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

Switch B

Root: A
Designated: B
Blocked: C

Port 1: [A,5,A,1]
Port 2: [A,5,B,2]

Switch C

Root: A
Designated: B
Blocked: C

Port 1: [A,5,A,2]
Port 2: [A,5,C,2]

Blue = Error

[RootID, RootPathCost, SenderID, PortID]/Status

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

![Network Diagram]

- **Root**
- **Designated**
- **Blocked**

**RootID**, **RootPathCost**, **SenderId**, **PortID**/Status

<table>
<thead>
<tr>
<th></th>
<th>Switch A</th>
<th>Switch B</th>
<th>Switch C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>[A,0,A,1]/D</td>
<td>[A,5,A,1]/R</td>
<td>[A,5,A,2]/R</td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]/D</td>
<td>[A,5,B,2]</td>
<td>[A,5,C,2]</td>
</tr>
</tbody>
</table>

**Blue** = Error
Incorrect – add link cost on receive

Blue = Error

A

Switch A Switch B Switch C
Port 1 [A,0,A,1]/D [A,5,A,1]/R [A,5,A,2]/R
Port 2 [A,0,A,2]/D [A,5,B,2]/D [A,5,C,2]/D
Incorrect – add link cost on receive

[RootID, RootPathCost, SenderID, PortID]/Status

<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>[A,0,A,1]/D</td>
<td>[A,5,A,1]/R</td>
<td>[A,5,A,2]/R</td>
</tr>
<tr>
<td>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>

Blue = Error

Port BPDUs differ on each segment!

LOOP!
How compute root & on-tree ports?

• By exchanging messages called *Bridge Protocol Data Units (BPDUs)*

• BPDUs describe the **best path** from the sender to the **root switch**
  – Root ID, Root Path Cost, Sender ID, Port Priority, Port ID
  – Set by *sending* switch (i.e. Sender ID, Port Priority and ID of sender)!

• BPDUs compared by fields, in order (Root ID, then Root Path Cost, etc.).

• Each switch saves best BPDU heard on each port
  – Called *Port BPDU*
  – Port BPDU set to best BPDU sent/received on port

• **REMEMBER:** BPDU describes best path from SENDER switch
Correct – Sender Port ID

A 1 5 5 B
2 1

Switch A
- Port 1: Blocked
- Port 2: Designated

Switch B
- Port 1: Designated
- Port 2: Blocked

Legend:
- Red Circle: Root
- Green Circle: Designated
- Black Circle: Blocked
Correct – Sender Port ID

A sends BPDUs.

[RootID, RootPathCost, SenderID, PortID]/Status

<table>
<thead>
<tr>
<th>Port</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
Correct – Sender Port ID

B receives BPDUs.

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

Switch A | Switch B
---|---
Port 1  | [A,0,A,1]  | [A,0,A,2]
Port 2  | [A,0,A,2]  | [A,0,A,1]
Correct – Sender Port ID

![Diagram of network topology with ports and 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

A 1 5 5 2
B 2

[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]/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>

Root
Designated
Blocked
Correct – Sender Port ID

Port BPDU's 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

Switch B

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

A – Switch A
B – Switch B

[A,0,A,1] – Port 1
[A,0,A,2] – Port 2
Incorrect – Sender Port ID

B receives BPDUs.

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

![Diagram showing switching between Switch A and Switch B with ports 1 and 2, and statuses Root, Designated, and Blocked.]

<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,1]</td>
</tr>
<tr>
<td>Port 2</td>
<td>[A,0,A,2]/D</td>
<td>[A,0,A,2]</td>
</tr>
</tbody>
</table>

*Blue* = Error
Incorrect – Sender Port ID

\[
\begin{array}{c|c|c}
\text{Port 1} & \text{Switch A} & \text{Switch B} \\
\hline
[A,0,A,1]/D & [A,0,A,1]/R \\
\hline
\text{Port 2} & [A,0,A,2]/D & [A,0,A,2]/B \\
\end{array}
\]

Blue = Error
Incorrect – Sender Port ID

Port BPDUs differ on each segment!

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**

**VLAN0001**

Spanning tree enabled protocol ieee
Root ID  Priority   24577  
Address  000f.2490.1380
Cost     23
Port     1  (FastEthernet0/1)
Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

**Bridge ID**  Priority   32769  *(priority 32768 sys-id-ext 1)*
Address  0009.7c0b.e7c0
Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
Aging Time  300

<text omitted>

**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

**Bridge ID**  Priority   32778  *(priority 32768 sys-id-ext 10)*
Address  0009.7c0b.e7c0
Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
Aging Time  300
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.

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
• 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)
Questions?