Spanning Tree (802.1D)

Spanning Tree is a managed system where each port is in a forwarding or blocking state. Blocking state means the port doesn't process any frames except STP messages- the switch physically receives the frame on blocked port, but ignores it. If topology changes (a switch goes down), STP convergence updates port states accordingly. This system has a "root bridge" and uses bridge IDs for each switch containing a priority # and MAC address. Distance to the root switch/bridge are measured as a "root cost" from each switch and port, and governs the forwarding and blocking of ports, and ultimately, the paths Layer 2 stuff can take around the network.

The Root Bridge/ Root Switch and the System's Basics

- The root switch is always the designated switch on all directly connected segments
- All ports on the root switch are designated ports, in a forwarding state
- Nonroot switches have a root port (with lowest cost to the root switch); if directly connected, it's the root port.
- Each nonroot switch has at least one designated port (DP) to the other non-root switches.
- Similarly, a designated switch is chosen by lowest "root cost" among other paths to root, other non-root switches connect to the root through their neighboring designated switch.
- Throughout this system of non-root switches, up/up ints become either DPs or blocked
- The designated port's job is to forward STP messages (BDPUs) back and forth from the root

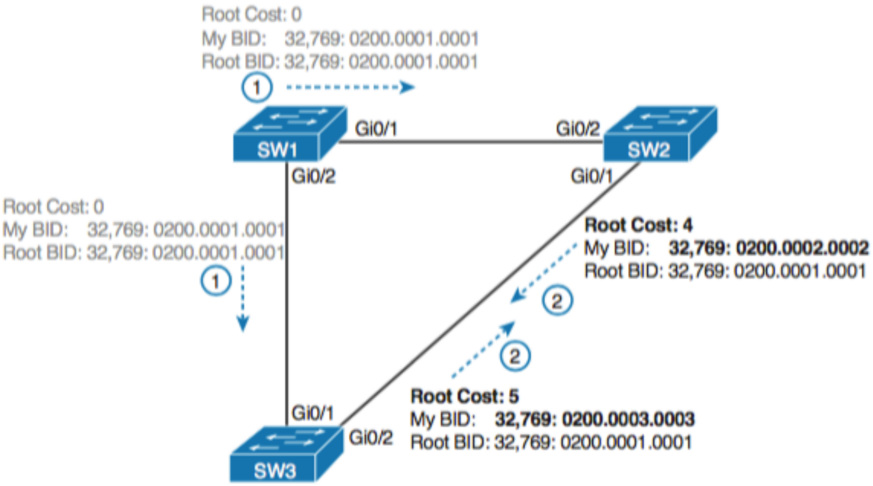
Electing the Root Switch [more explanation in Bridge ID (BID) section]

Root switch has the lowest value priority field in their BID

If SW1 Priority = 4096 and SW2 Priority = 8192, then SW1 is Root

If a priority tie occurs, switch with lowest MAC address field in BID is Root

If SW1 Pri= 4096 & MAC= 0200.xxxx.xxxx and SW2 Pri= 4096 & MAC= 0911.xxxx.xxxx, SW1 is Root


Root Cost: 0
My BID: 32,769: 0200.0001.0001
Root BID: 32,769: 0200.0001.0001
1
SW1
Gi0/1
Gi0/2
SW2
Gi0/2
Gi0/1
Root Cost: 0
My BID: 32,769: 0200.0001.0001
Root BID: 32,769: 0200.0001.0001
1
Root Cost: 4
My BID: 32,769: 0200.0002.0002
Root BID: 32,769: 0200.0001.0001
2
2
Root Cost: 5
My BID: 32,769: 0200.0003.0003
Root BID: 32,769: 0200.0001.0001
Gi0/1
Gi0/2
SW3

Bridge Protocol Data Units (BPDU) exchange info among switches

Hello messages are a BPDU containing:

- Root Bridge ID, Sender’s Bridge ID and root cost
- Timer values on the root switch [hello, MaxAge, and forward delay timers]

Configuration BPDUs originate from Root Bridge. These are the majority of BPDUs on a healthy network
Topology Change Notification (TCN) BPDUs sent to the Root Bridge to alert that active topology has changed. (notification and acknowledgement subtypes)

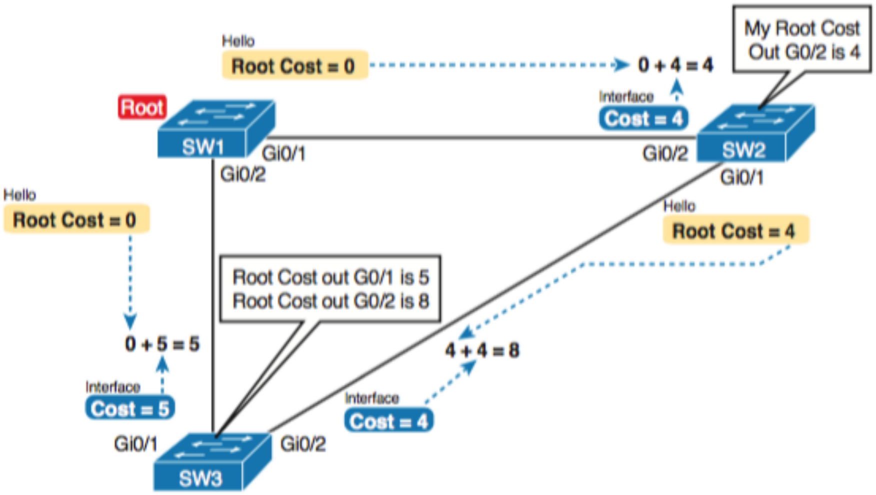
Timers

Hello - 2 seconds default - Time period between recieving hellos created by the root
MaxAge- 10x hello timer (20 sec default) -How long switch waits after no hello BDPUs, before changing states
Forward delay- 15 secs - time interface goes from blocking to forwarding in each of the listening, learning states

Electing Designated Ports - the port with the lowest-cost hello on a LAN segment.

(Hello msgs have a cost of 0 when they leave the root switch) Cost by line type, 2nd # is "long path cost":

10 Mbps	100	2000000
100 Mbps	19	200000
1 Gbps	4	20000
10 Gbps	2	2000
100 Gbps		200



Both SW2 and SW3 list their respective cost to the root switch (cost 4 on SW2 and cost 5 on SW3). SW2 lists the lower cost, so SW2's Gi0/1 port is the designated port on that LAN segment. ***The order is lowest root cost, then lowest BID, then lowest neighbor port priority, then finally, lowest neighbor internal port number***

50-second convergence delay: STP has the interface in both *learning and listening states* for a time equal to the forward delay timer, which defaults to 15 seconds each. A convergence event that causes an interface to change from blocking to forwarding requires 30 sec to transition those PLUS the MaxAge to first move from blocking

Temporary states when going from blocking to forwarding:
- Listening- doesn't forward- does MAC table cleanup
- Learning- also doesn't forward- learn the MAC address of received frames

State	Forwards?	Learns MACs?	Transitory or Stable State?
Blocking	No	No	Stable
Listening	No	No	Transitory
Learning	No	Yes	Transitory
Forwarding	Yes	Yes	Stable
Disabled	No	No	Stable

Rapid STP (IEEE 802.1w - RSTP): (is backward-compatible w/ non-RSTP switches)
- Calls blocking "discarding", no listening state. Only has learning, forwarding or discarding states
- Hello time = 2 secs, Max Age = 6 secs (3 x hello, 3 missed BDPUs)
- ***improves convergence from about 50 seconds to about 10 seconds***
- adds two port types allowed to immediately enter the forwarding state rather than passively wait for the network to converge. In addition to forwarding, root, designated, and disabled:
    - Alternate port - best alternate path to the root bridge if there is a failure on the designated port
    - Backup port. Applies to scenarios with a hub. Antiquated.

Check Before Connecting New HW
When plugged into the network, switches send out hello BPDUs listing their own BID as the root BID, then, if it hears a hello that lists a better (lower) BID, that switch stops advertising itself as root and starts forwarding the superior hello. This is how an unauthorized or improperly vetted device can screw up a STP topology- if one shows up with a lower BID, convergence spreads it as a new root switch and everything goes haywire.

PortFast
- Allows immediately change from blocking to forwarding. No listening or learning states
- For endpoint/edge devices (PCs) - NOT bridges/switches
- *Don't put on interfaces receiving BPDU (not another switch) - w/o BDPU Guard can create loops*
- If voice VLAN set, PortFast turns on automatically- will stay on even if voice VLAN is unset

BPDU Guard
- BDPU Guard disables a port if any BPDUs are received on it and prevents PortFast problems
- STP opens up the following security exposures:

    Attacker adds switch with low STP priority and becomes the Root Switch
    Attacker could plug into multiple ports/switches, become root, & forward (or TCPDump) LAN traffic
    Users can harm the LAN when they connect a non-STP switch (can be counted in elections, etc.)

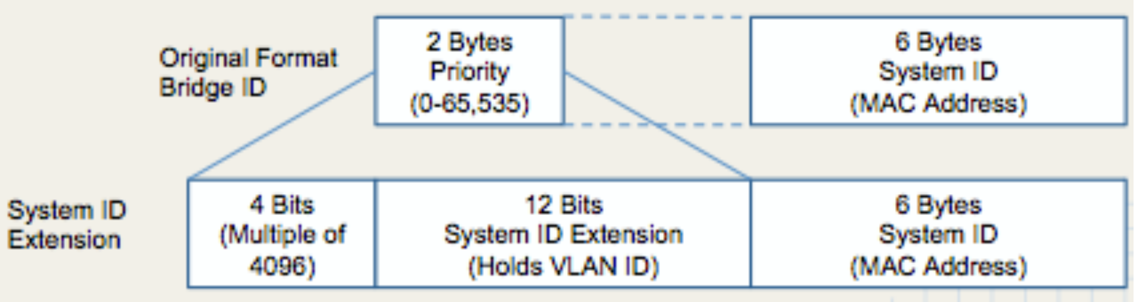
The Bridge ID - { [Priority(4096-61440) + VLAN ID(1-4094)] + the MAC address}

STP Bridge ID (BID) is a unique 8-byte value- basically a priority field slapped onto a MAC address

- a 2-byte priority field The last 12 bits holds a VLAN ID 0-4096 (it's just basically a 16-bit number)
- a 6-byte system ID - the 48-bit MAC address

The 2-byte Priority field is separated into 2 parts

4-bit priority field; the part that can be changed with the "priority" directive
12-bit *system ID extension* - basically, just space for the VLAN ID)


Original Format Bridge ID
2 Bytes Priority (0-65,535)
6 Bytes System ID (MAC Address)
System ID Extension
4 Bits (Multiple of 4096)
12 Bits System ID Extension (Holds VLAN ID)
6 Bytes System ID (MAC Address)

When the bits are written out, those leading 4 bits reside starting at the 4096 bit (the 13th bit), so each number starting there would naturally be a multiple of 4096. Since VLAN IDs can range from 1 to 4094, it all makes perfect sense and looks a lot less complicated like this:

[Priority 4096-61440 + VLAN 1-4094] + tack on the MAC address.
Means, [0000 + 000000000000] + your MAC address
0000 - [111111111111] <---- 4095 in binary (enough bits reserved to plop a "priotrity" onto)
We configure the priority field below in square brackets:
[1000] - 000000000000 <---- Default base priority of 32768 in binary
[0110] - 000000000000 <---- Secondary root bridge designation number in binary
[????] - 000000000000 <-----you can set a multiple here for a priority (from 4096 x ? up to 61440)

The Easy Way

- spanning-tree vlan vlan-id **root primary** [auto-set this vlan's priority low enough to become root now]

Cisco switches use a default base priority of 32,768
If current root base priority > 24,576, make base priority 24,576.
If 24,576 or lower, sets base priority to highest multiple of 4096 that results in becoming root.
For our numbers below, 28,672 is 0110 000000000000

- spanning-tree vlan vlan-id **root secondary** [again, this auto-sets it for us]

A priority value worse than the primary switch but better than all the other switches.
Sets the switch's base priority to 28,672 *regardless* of the current root's priority value.
For our numbers below, 28,672 is 0111 000000000000

The Hard Way

It is much easier to just designate using root primary and secondary to avoid doing it yourself. If you insist on doing "spanning-tree vlan vlan-id priority x":

- spanning-tree vlan vlan-id **priority** x

A switch configured with VLANs 1 through 4, with a default base priority of 32,768, has a default STP priority of 32,769 in VLAN 1, 32,770 in VLAN 2, 32,771 in VLAN 3, and so on. So, you can view the 16-bit priority as a base priority (as configured on the **spanning-tree vlan** vlan-id **priority** x command) plus the VLAN ID."

0001:000000000001 = 4097
0010:000000000001 = 8193
0100 = 16384, 1000 = 32768

Vlan 1 would be 32769, Vlan 2 32770, Vlan 10 32778, Vlan 100 32868
1111= 61,440   <--- if there are more than one bit turned on, things get weird. One trick is to pretend the VLAN-ID part is all 1's so it flattens out to multiples of 4096 only, when trying to calculate stuff)

Final word: If binary makes you unconfortable, forget it. Use "root primary" and "root secondary"

Determining the Root Switch and the Root Port on Nonroot Switches

Use **show spanning-tree**, **show spanning-tree root** to rule out any switches that have an RP, because root switches do not have an RP. **show spanning-tree** identifies the local switch as root directly: "This switch is the root". In **show spanning-tree root**, the RP column is empty if the local switch is the root.
Using **show spanning-tree vlan** x on a few switches, and recording the root switch, RP, and DP ports can quickly show you most STP facts. Chase the RPs. If starting with SW1, and SW1's G0/1 is an RP, try the switch on the other end of SW1's G0/1 port.

STP Tiebreakers When Choosing the Root Port

The three tiebreakers are, in the order: lowest neighbor bridge ID, lowest neighbor port priority, finally, lowest neighbor internal port number. Only root paths that tie are considered when thinking about tiebreakers.
Figure 2-8 shows that SW3 is not root and that its two paths to reach the root tie with their root costs of 8. The first tiebreaker is the lowest neighbor's BID. SW1's BID value is lower than SW2's, so SW3 chooses its G0/1 interface as its RP.

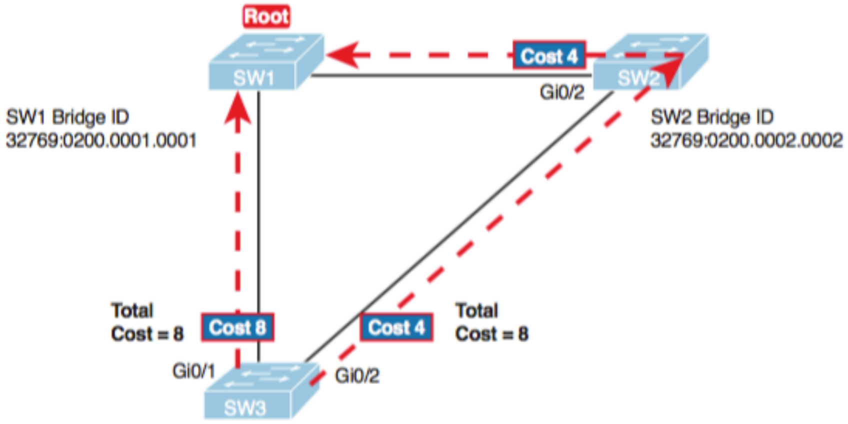


Figure 2-8 *SW3's Root Cost Calculation Ends in a Tie*

The last two RP tiebreakers come into play only when two switches connect to each other with multiple links, as shown in above. In that case, a switch receives hellos on more than one port from the same neighboring switch, so the BIDs tie.

So, SW2 becomes root, and SW1 needs to choose its RP. SW1's root cost over each path will tie at 19. SW2 sends hellos over each link to SW1, so SW1 cannot break the tie based on SW1's neighbor BID because both list SW2's BID. So, SW1 has to turn to the other two tiebreakers.

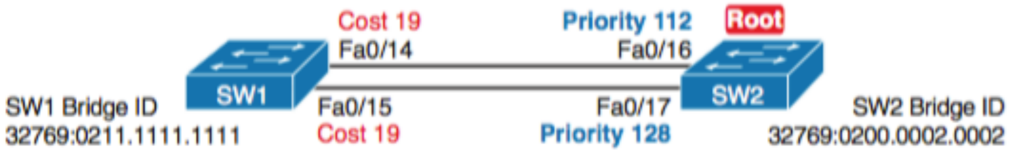


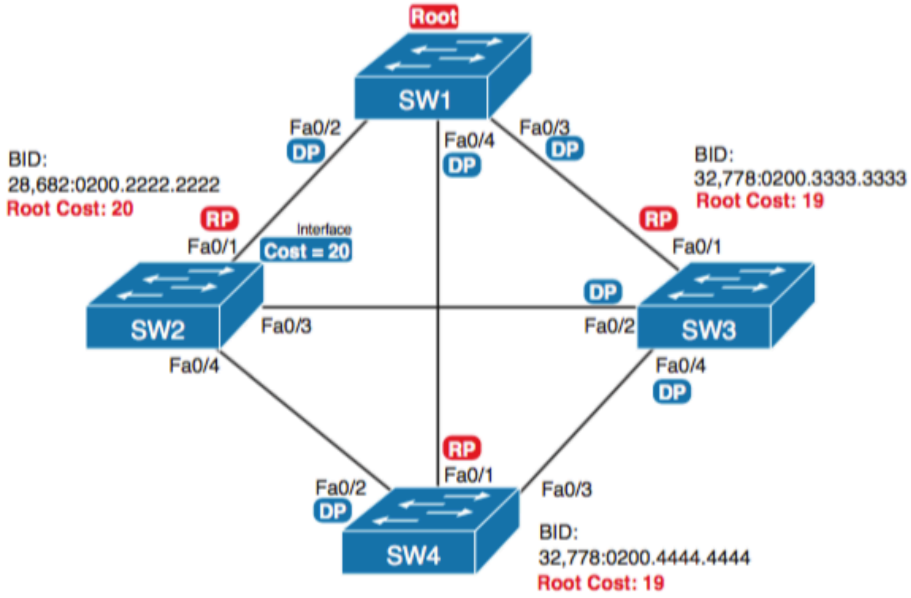
Figure 2-9 *Topology Required for the Last Two Tiebreakers for Root Port*

The next tiebreaker is configurable: the neighboring switch's port priority on each neighboring switch interface. Cisco switch ports default to a setting of 128, with a range of values from 0 through 255, with lower being better (as usual). Here, SW2's F0/16 was manually set with **spanning-tree vlan 10 port-priority 112**. SW1 learns that the neighbor has a port priority of 112 on the top link and 128 on the bottom, so SW1 uses its top (F0/14) interface as the root port.

If the port priority ties, which it often does due to the default values, STP relies on an internal port numbering on the neighbor. Cisco switches assign an internal integer to identify each interface on the switch. The nonroot looks for the neighbor's lowest internal port number (as listed in the hello messages) and chooses its RP based on the lower number.

With Fa0/1 having the lowest number, then Fa0/2, then Fa0/3, and so on- SW2's Fa0/16 would have a lower internal port number than Fa0/17; SW1 would learn those numbers in the hello; and SW1 would use its Fa0/14 port as its RP. (In real life, most engineers would put these two links into an EtherChannel)

This figure notes the root, RPs, and DPs and each switch’s least cost to reach the root over its respective RP.



Remember the order of the criteria:
Lowest root cost, lowest BID, lowest neighbor port priority, lowest neighbor internal port number
Focus on the segments that connect the nonroot switches for a moment.
SW2–SW4 segment: SW4 wins because of its root cost of 19, compared to SW2’s root cost of 20.
SW2-SW3 segment: SW3 wins because of its root cost of 19, compared to SW2’s root cost of 20.
SW3-SW4 segment: SW3 and SW4 tie on root cost, both with 19. SW3 wins due to its better (lower) BID value.
SW2 loses and does not become DP on the links to SW3 and SW4 even though SW2 has the better (lower) BID value. The first DP criteria is the lowest root cost, and SW2’s root cost happens to be higher than SW3’s and SW4’s.

----

show spanning-tree vlan Bridge and RootID info, timers; port roles for local ports, switch role for device.
show spanning-tree vlan 10 root - lists root’s BID for each VLAN, local switch’s root cost and root port
show spanning-tree vlan 10 bridge - lists the local switch’s BID, MAC, timers
debug spanning-tree events

spanning-tree portfast - **spanning-tree bpduguard enable** (to be run on individual interfaces)
Default is no portfast and no BPDU Guard. Change the opposite to make all on the new default:
spanning-tree portfast default - **spanning-tree portfast bpduguard default** (Global config)
spanning-tree portfast disable - **spanning-tree bpduguard disable** (Run per interface config)

Problems?

- If there is no evidence of STP cost config on the interface, check the current actual speed as well. Cisco switches choose STP cost defaults based on the current speed, not adjustments set by speed configs you set

UplinkFast: pg 197 - UplinkFast: Access Layer Uplinks
BackboneFast pg 198 - BackboneFast: Redundant Backbone Paths

Commands for Spanning Tree - PortFast - BDPUGuard - UplinkFast - BackboneFast

spanning-tree vlan-id	Enable STP.
spanning-tree mode { **pvst | rapid-pvst | mst** }	Set the STP mode.
spanning-tree vlan vlan-number **root primary**	Changes this switch to the root.
spanning-tree vlan vlan-number **root secondar y**	Sets this switch's STP base
spanning-tree [**vlan** vlan-id] {**priority** priority}	Changes the bridge priority of this switch for the specified VLAN.
Switch(config-if)# **spanning-tree** [**vlan** vlan-number] **cost** cost	Changes the STP cost .
Switch(config-if)# **spanning-tree** [**vlan** vlan-number] **port-priority** priority	Changes the STP port priority in that VLAN (0 to 240, in increments of 16).
spanning-tree vlan vlan-id **priority** bridge-priority	Set bridge priority.
spanning-tree vlan vlan-id **root** {**primary | secondary**} [**diameter** diameter]	Set root bridge (macro).
spanning-tree [**vlan** vlan-id] **hello-time** seconds **spanning-tree** [**vlan** vlan-id] **forward-time** seconds **spanning-tree** [**vlan** vlan-id] **max-age** seconds	Set STP timers.
show spanning-tree	STP info all VLANs, ports summarized
show spanning-tree detail	STP info all VLANs, ports detailed.
show spanning-tree interface interface-id	Lists STP info for the specified port
show spanning-tree vlan vlan-id	Lists STP info for the specified VLAN
show spanning-tree [vlan vlan-id] **summary**	View the total number of switch ports currently in each of the STP states.
show spanning-tree [**vlan** vlan-id] **root**	Show a VLAN's root bridge ID, root port, and root path cost.
show spanning-tree [**vlan** vlan-id] **bridge**	Show this switch's bridge ID and STP timers
show spanning-tree interface type number **portfast**	1-line status message about PortFast
channel-group channel-group-number **mode** {**auto | desirable | active | passive | on**}	Enables EtherChannel.
show etherchannel [chan-grp-#] {**brief | detail | port | port-channel | summary**}	Info on state of EtherChannels
debug spanning-tree events	Causes the switch to provide informational messages about changes in the STP topology
spanning-tree portfast (disable)	Enables/disables PortFast.
spanning-tree bpduguard enable | disable	Enables/disables BPDU Guard
spanning-tree portfast default	Changes switch default for PortFast to enabled.
spanning-tree portfast bpduguard default	Changes switch default for BPDU Guard to enabled.
spanning-tree uplinkfast [**max-update-rate** pkts-per-second]	Set UplinkFast on a switch.
spanning-tree backbonefast	Set BackboneFast on a switch.
show spanning-tree uplinkfast	Show the STP UplinkFast status.
show spanning-tree backbonefast	Show the STP BackboneFast status.
show spanning-tree summary	Display global BPDU Guard, BPDU filter, and Loop Guard states.
show spanning-tree interface type mod/num [**detail**]	Look for detailed reasons for inconsistencies.
show spanning-tree inconsistentports	List ports labeled in an inconsistent state.
show udld [type mod/num]	Display the UDLD status on one or all ports.
udld reset	Reenable ports that UDLD aggressive mode errdisabled.

Task	Global Command Syntax	Interface Command Syntax
Enable Root Guard.	--	Switch(config-if)# **spanning- tree guard root**
Enable BPDU Guard.	**spanning-tree portfast bpduguard default**	Switch(config-if)# **spanning- tree bpduguard enable**
Enable Loop Guard.	**spanning-tree loopguard default**	Switch(config-if)# **spanning- tree guard loop**
Enable UDLD.	**udld** **{enable | aggressive | message time seconds}**	Switch(config-if)# **udld** **{enable | aggressive | disable}**
Enable BPDU filtering.	**spanning-tree bpdufilter default**	Switch(config-if)# **spanning- tree bpdufilter enable**

Switch(config-if)# **spanning-tree portfast** Define an edge port
Switch(config-if)# **spanning-tree link-type point-to-point** Override a port type

Switch(config)# **spanning-tree mode mst** Enable MST on a switch
Switch(config)# **spanning-tree mst configuration** Enter MST configuration mode
Switch(config-mst)# **name** name Name the MST region.
Switch(config-mst)# **revision** version Set the configuration revision number

spanning-tree mst instance-id **root {primary | secondary**} [**diameter** diameter] Set root bridge (macro).
spanning-tree mst instance-id **priority** bridge- priority Set bridge priority.
spanning-tree mst instance-id **cost** cost Set port cost.
spanning-tree mst instance-id **port-priority** port-priority Set port priority.

Set STP timers.
spanning-tree mst hello-time seconds
spanning-tree mst forward-time seconds
spanning-tree mst max-age seconds

STP and RTSP Port Path Cost values

Data rate	**STP 802.1D-1998**	**RSTP 802.1W-2004**
Base Metric	**1 Gigabit per sec**	**2 Terabit per sec**
Calculation	**1,000,000/(*N* Kb/s)**	**20,000,000,000/ (*N* Kb/s)**
	(1998 is nonlinear)	
	(1990 was linear)	
<=100 Kb/s		200,000,000
1 Mbit/s		20,000,000
4 Mbit/s	250	5,000,000
10 Mbit/s	100	2,000,000
16 Mbit/s	62 (was 63)	1,250,000
45 Mbit/s	39 (was 22)	444,445
100 Mbit/s	19 (was 10)	200,000
155 Mbit/s	14 (was 6)	129,032
622 Mbit/s	6 (was 2)	32,154
1 Gbit/s	4 (was 1)	20,000
2 Gbit/s	3	10,000
10 Gbit/s	2	2,000
100 Gbit/s		200
1 Tbit/s		20
10 Tbit/s		2

802.1W-2004: Limiting the range of the Path Cost parameter to 1–200 000 000 ensures that the accumulated Path Cost cannot exceed 32 bits over a concatenation of 20 hops
The grey cells indicate values presented by Cisco in the Switch 300-115 Official Cert Guide and are not in the IEEE 802.1W-2004.
Cisco also says in the CIEE 5.0 Routing and Switching book the short format is Cisco's, and not IEEE.
Indeed, in IEEE 802.1W-2004, it is the long format which remains (ed.: I am thankful since a nonlinear format is nonsensical)

Switching: EtherChannel

Cisco's Port Aggregation Protocol (PAgP) and Link Aggregation Control Protocol (LACP - 802.3ad)
- Combines multiple parallel segments of equal speed into an EtherChannel (single interface)
- Multiple segments of equal speed up to 8 links. If one fails, the rest stay up.
- Prevents STP convergence when only a single failure occurs; with at least one up, no STP convergence
- Aggregated links provide some load balancing on switches (and better use of bandwidth)

<u>Create the port-channel interface, exit, then enter interface config mode for ports and add them with a mode:</u>

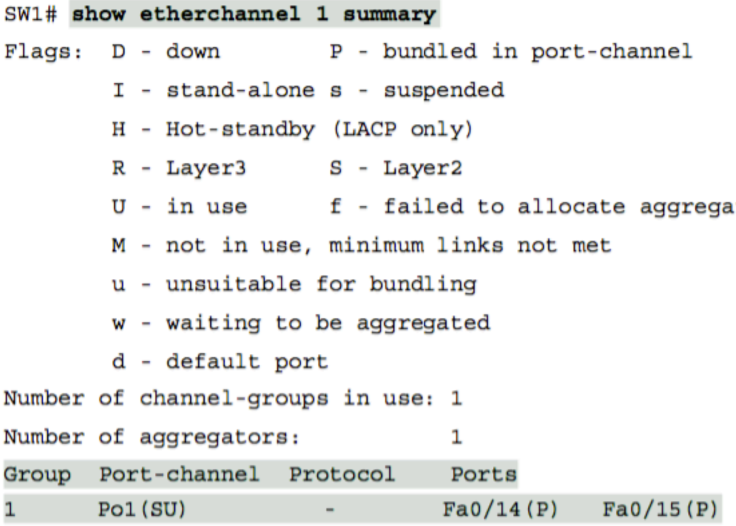
```
Cat2950(config)# interface port-channel 1   <exit>
Cat2950(config)# interface fa0/2
Cat2950(config-if)# channel-group 1 mode on
```

PAgP: **channel-group 1 mode {desirable | auto}**
LACP: **channel-group 1 mode {active | passive}**
Manually run with **channel-group** 1 **mode on** (has no encapsulation; set "on" on both ends)
- One side must be either desirable or active to get the other end to begin negotiations (like trunks)
- Number must remain the same on the current device. Another device could call it 4 or 5

Three terms as synonyms: EtherChannel, PortChannel, and Channel-group.
The **channel-group** configuration command
show etherchannel shows status of a “PortChannel” (instead of EtherChannel or Channel-group)
show spanning-tree also says Port-channel

<u>Misconfiguration of channel-group commands:</u>
- All members of a switch's specific etherchannel need to be using the same channel-group number.
- The neighboring switches can refer to that etherchannel as any number they want.
- If using the "on" keyword, it has to be on the other end too (and remember no encaps'ing).
- If using the "desirable" keyword, it's PAgP and the other must use either "desirable" or "auto".
- If using "active" keyword, it's LACP; the other must use either "active" or "passive".
- No "auto" or "passive" keyword on both switches! Then both wait on the other to begin negotiations.
- Don't mix LACP and PAgP commands on one end or the other- the protocols need to match.

show etherchannel [chan-grp-#] {**brief | detail | port | port-channel | summary**}



```
SW1# show etherchannel 1 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator
        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port
Number of channel-groups in use: 1
Number of aggregators:           1
Group  Port-channel  Protocol    Ports
1      Po1(SU)           -        Fa0/14(P)   Fa0/15(P)
```

The **show etherchannel summary** command. The D code letter means that the channel itself is down, with S meaning a Layer 2 EtherChannel. The bottom shows Portchannel (Po1) as L2 EtherChannel in a down state (SD) "Down" only refers to the EtherChannel state - the two physical interfaces are still connected (**sh int status**)

Configuration Checks for EtherChannel
Things need to match or be compatible. Watch for common sense stuff (no shutdown, speed, duplex)
- Operational access or trunking state (all must be access, or all must be trunks)
- On an access port, check the access VLAN;
- Trunk port? Check allowed VLAN list - **switchport trunk allowed** - also check the native VLAN
- Check STP interface settings (ie, cost)

For example, if you change the STP port cost on one of the interfaces but not the other you get this:

```
ERR_Disable: channel-misconfig (STP) error detected on Po1, putting Fa0/14 in err-disable state
ERR_Disable: channel-misconfig (STP) error detected on Po1, putting Fa0/15 in err-disable state
ERR_Disable: channel-misconfig (STP) error detected on Po1, putting Po1 in err-disable state
```

Make them match and it should bring everything back up again

EtherChannel Stuff from CCNP

Load balancing uses a hashing operation on either MAC or IP addresses and can be based solely on source or destination addresses, or both. To configure frame distribution for all EtherChannel switch links:

```
Switch(config)# port-channel load-balance method
```

The method is set with a global configuration command. It's set globally for the switch, not on a per-port basis.

Table 10-3 Types of EtherChannel Load-Balancing Methods

Method Value	**Hash Input**
src-ip	Source IP
dst-ip	Destination IP
src-dst-ip	Source and destination IP
src-mac	Source MAC
dst-mac	Destination MAC
src-dst-mac	Source and destination MAC
src-port	Source port number
dst-port	Destination port number
src-dst-port	Source and destination port number

- All hash operations are performed on bits, except src-dst combo methods use XOR
- Port number only specified in CCNP Switch book as available in models 4500, 6500

```
Switch# show etherchannel load-balance
        EtherChannel Load-Balancing Configuration:
                src-mac

                EtherChannel Load-Balancing Addresses Used Per-Protocol:
        Non-IP: Source MAC address
          IPv4: Source MAC address
          IPv6: Source MAC address
```

To verify efficiency of load-balancing method use the **show etherchannel port-channel** command

EtherChannel Troubleshooting Commands

Current EtherChannel status of each member port	**show etherchannel summary** **show etherchannel port**
Time stamps of EtherChannel changes	**show etherchannel port-channel**
Detailed status about each EtherChannel component	**show etherchannel detail**
Load-balancing hashing algorithm	**show etherchannel load-balance**
Load-balancing port index used by hashing algorithm	**show etherchannel port-channel**
EtherChannel neighbors on each port	**show** {**pagp | lacp**} **neighbor**
LACP system ID	**show lacp sys-id**

EtherChannel Configuration Commands

Select load-balancing method for switch.	**port-channel load-balance** method
Use a PAgP mode on an interface.	**channel-protocol pagp** **channel-group** number **mode** {**on |** {{**auto | desirable**} [**non-silent**]}}
Assign the LACP system priority.	**lacp system-priority** priority
Use an LACP mode on an interface.	**channel-protocol lacp** **channel-group** number **mode** {**on | passive | active**} **lacp port-priority** priority
Configure EtherChannel Guard	[**no**] **spanning-tree etherchannel guard misconfig**