Chapter 4
STP Enhancements - For Performance & Stability
NET3011 – 17W

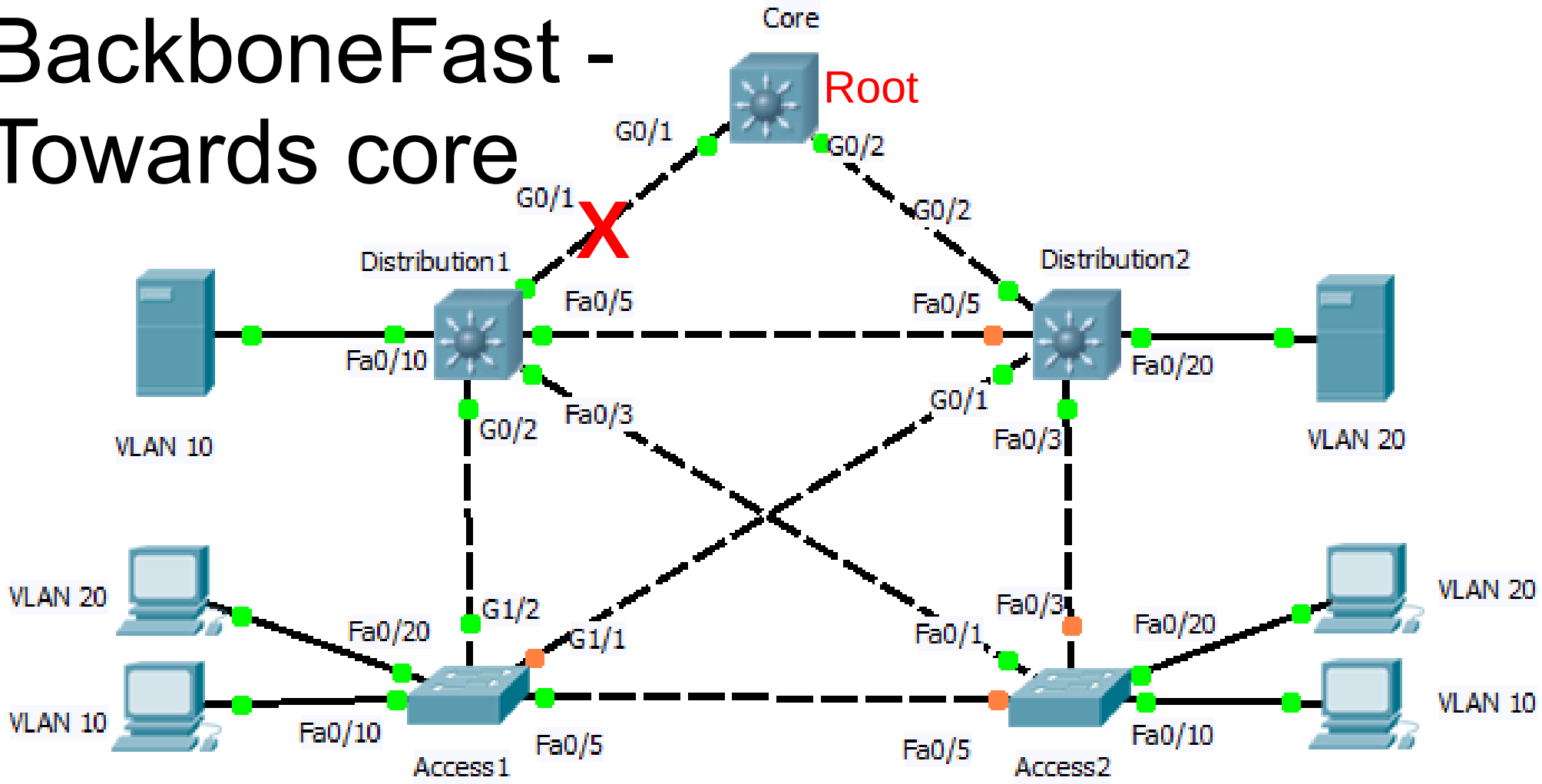
Enhancements - Basics

- STP was very well suited to the existing networks at the time it was invented (low bandwidth & CPU overhead), but networks are faster now
- Enhancements are needed in multiple areas:
  - **speed**: 3 methods make convergence *fast*(er)
  - **performance**: with extra speed comes potential instability; need to *guard* against problems
  - **other** options: supplement or supplant STP
- Rapid Spanning Tree = STP + 3 speed-ups (mostly)
- "What if …? Would that mean …?" has lead to tweaks for specific scenarios.

Speed-ups For Fast Convergence

- Three tweaks to the STP algorithm for special-case uses: how about looking at the core, distribution, and access layers?
- (Towards) Core = BackboneFast
  – Eliminate MaxAge delay; RLQ-Req/Ack msgs
- (Towards) Distribution = UplinkFast
  – Eliminate both Forward Delays; <1 sec failover
- (Towards) Access nodes = PortFast
  – Eliminate both Forward Delays, no TCN (and potential extra flooding); 0 sec startup

BackboneFast - Towards core




```
DLS1(config)#spanning-tree backbonefast
```

- Configured in global configuration mode and must be enabled on <u>all switches in the network</u>.
  - Makes use of special RLQ (Root Link Query) Req & ACK

Inferior BPDU

Normal BPDU

Bytes	Field
2	Protocol ID
1	Version
1	Message type
1	Flags
8	Root ID = Core
4	Cost of path
8	Bridge ID = Dist1
2	Port ID
2	Message age
2	Max age
2	Hello time
2	Forward delay

310P_126

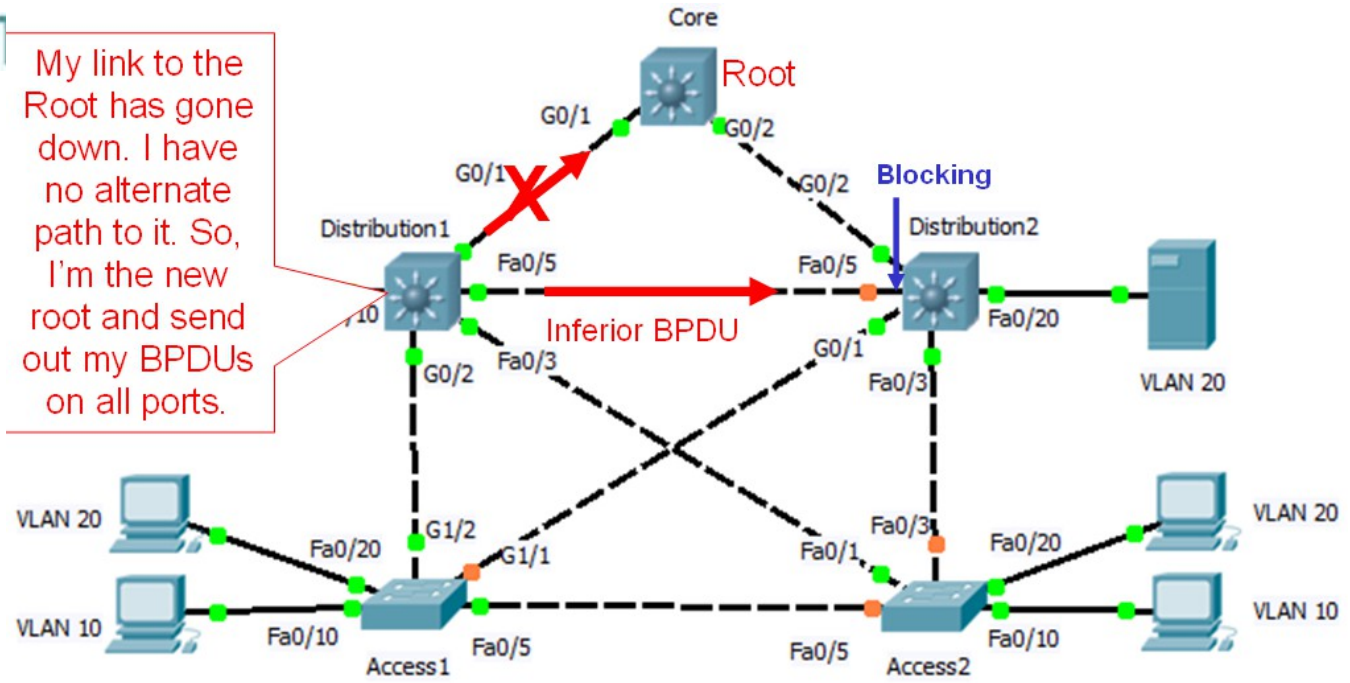
Inferior BPDU

2	Protocol ID
1	Version
1	Message type
1	Flags
8	Root ID = Dist1
4	Cost of path = 0
8	Bridge ID = Dist1
2	Port ID
2	Message age
2	Max age
2	Hello time
2	Forward delay

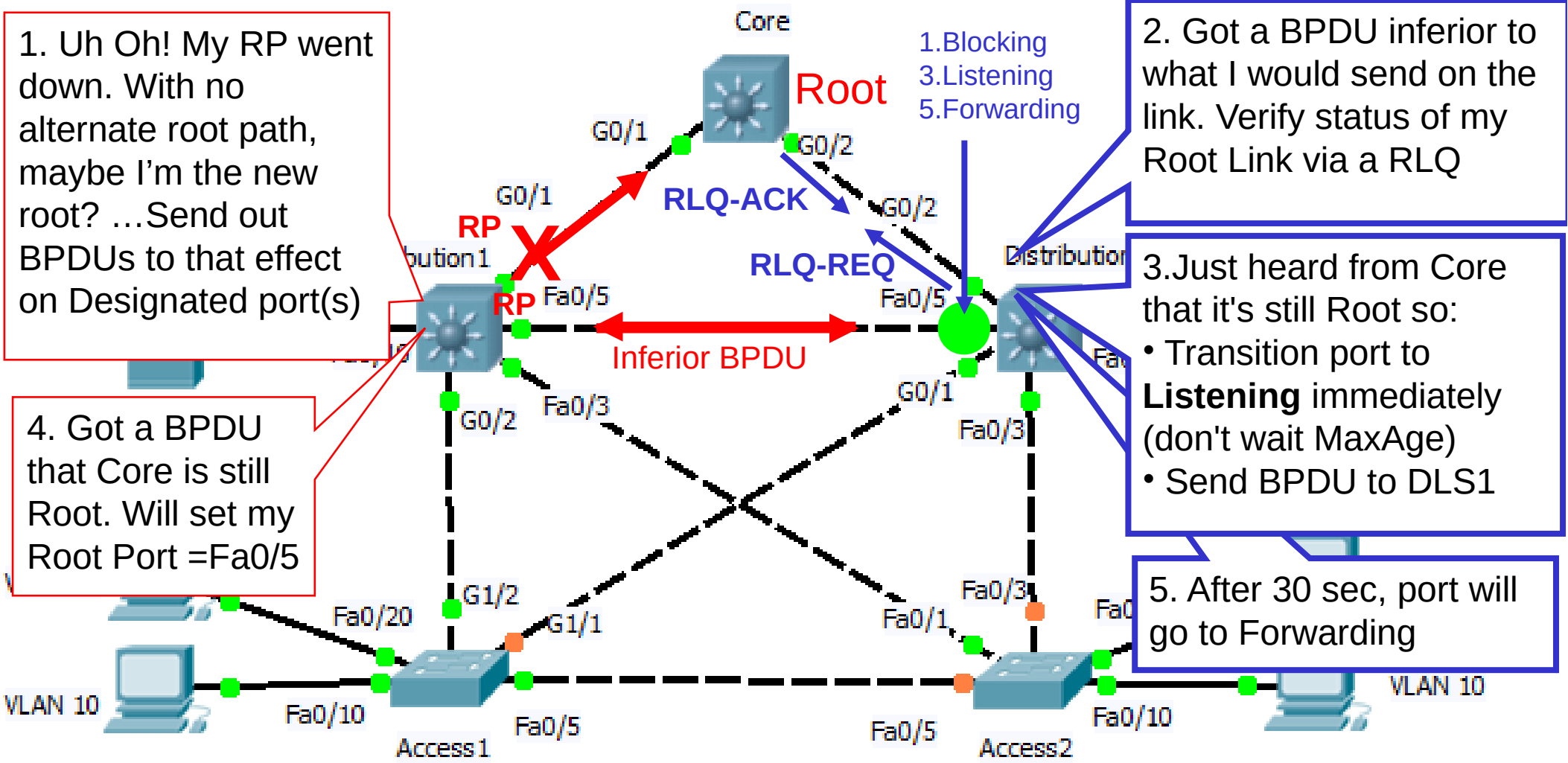
Same Switch

310P_126

My link to the Root has gone down. I have no alternate path to it. So, I'm the new root and send out my BPDUs on all ports.



- DLS1 is designated for the link to DLS2.
- Normally, it sends BPDUs on that link with Core shown as the Root Bridge.
- When G0/1 link goes down, with no other path to the root bridge, it declares itself as the root
- As a result, it formulates an Inferior BPDU with itself for both Root BID and Sender BID (having a root path cost of zero), sending it out all ports on which it would normally propagate Config BPDUs from the Root

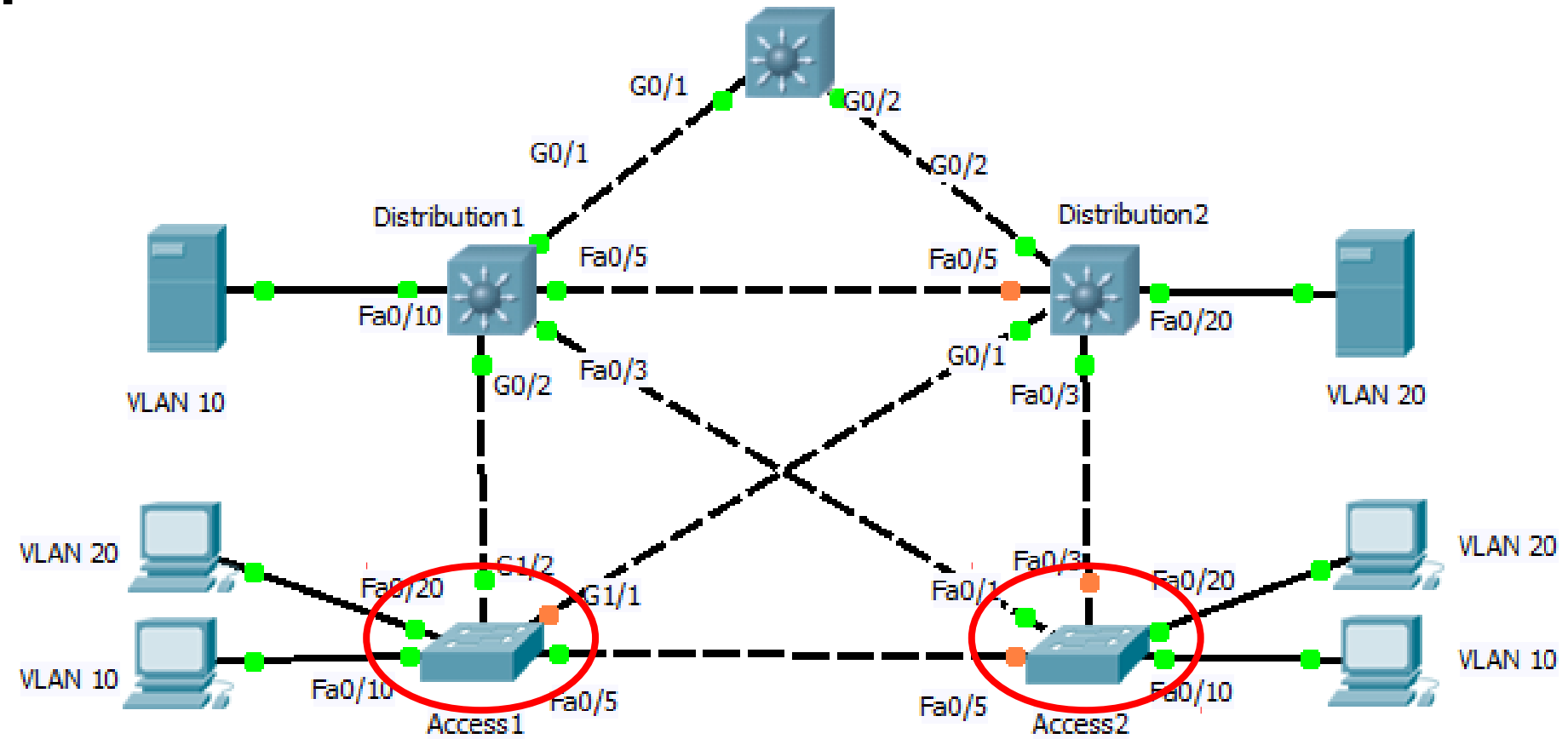


- BackboneFast saves time when a root port or blocked port receives an inferior BPDU from the Designated Bridge, signaling an indirect failure
- Inferior BPDUs occur when a Designated bridge loses connection to the root bridge
- Normally, switches wait Max Age before responding to an inferior BPDU
- With Backbonefast, alternate switches immediately verify path to Root

BackboneFast – In Words

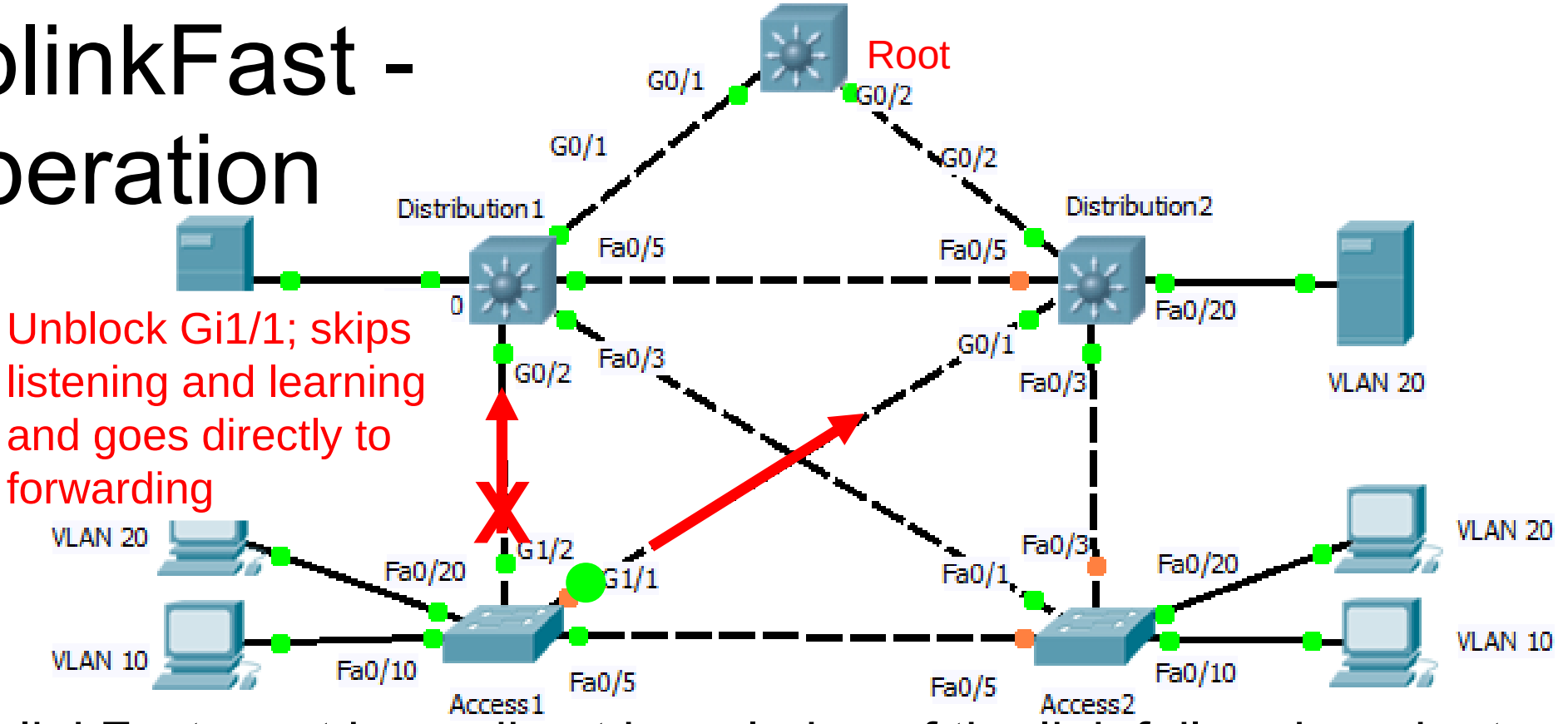
- This reduces the re-convergence time when a device receives an inferior BPDU from a neighbour on a blocked/root port. This means:
  - the root port on that neighbour, the designated bridge, has failed (so it is attempting to become the root), OR
  - the designated bridge has reconfigured with an inferior root path cost (i.e. the root path cost through this device is better)
- The device first confirms root reachability by sending a Root Link Query Request (RLQ-REQ) BPDU out the root port, targeting the root BID.
  - other devices with the same root BID forwards it root-wards (any device having a different root BID will return RLQ-NAK)
  - if this reaches the targeted root, it returns a RLQ-ACK BPDU
  - once the ACK is received, the Blocked port moves to immediately to Listening and Learning to become the designated port for the segment
- Backbone Fast must be configured on ***all*** switches in the domain.
- RLQ: SNAP frame, Cisco OUI 0x00000c,
  Request msg Protocol-ID = 0x0108; ACK msg Protocol-ID = 0x0109

UplinkFast – Towards Distribution



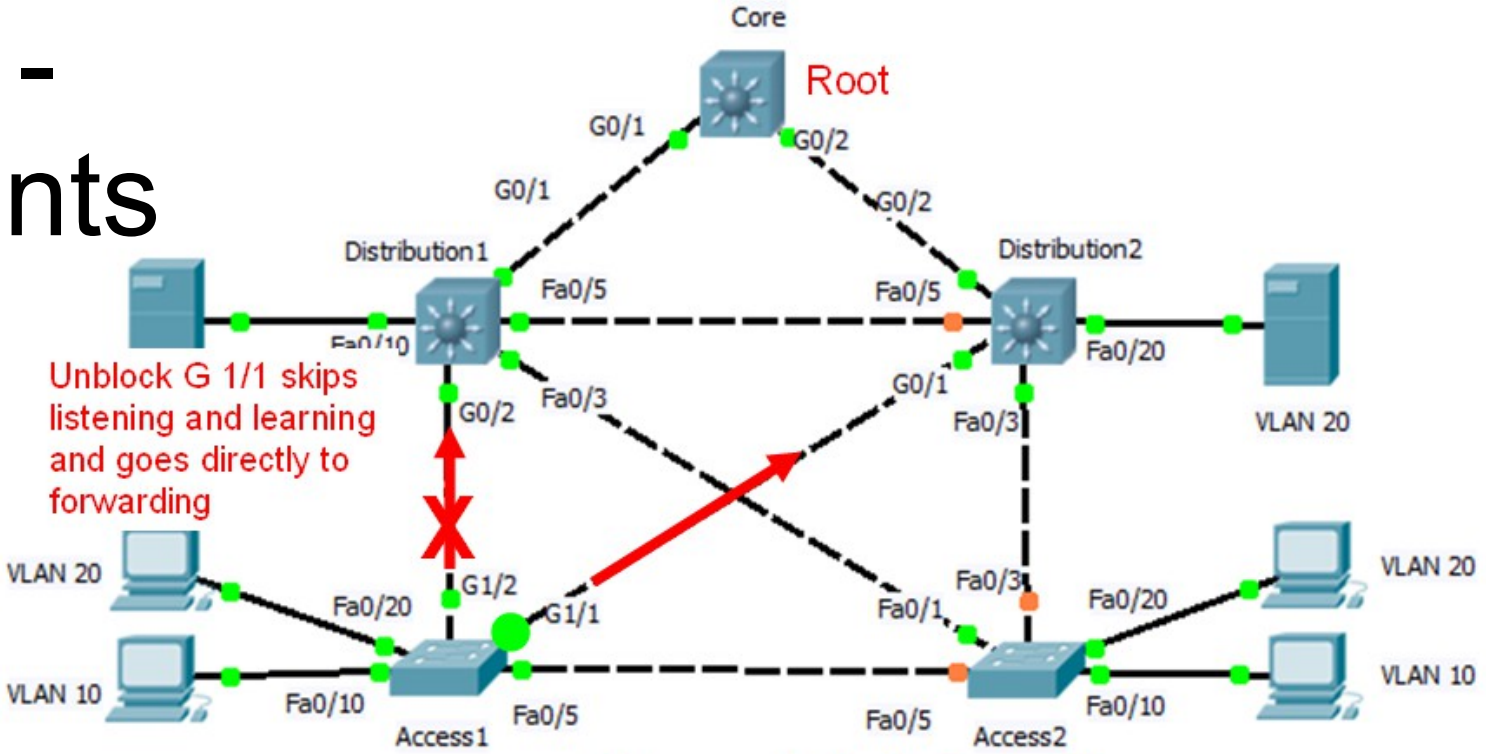
- Uplinkfast allows access layer switches having redundant links to multiple distribution switches, the ability to converge quickly when its root path fails.
  - **Only** for “Leaf nodes” (end nodes) of the spanning tree.
  - **Not** for use within backbone or distribution switches

UplinkFast - Operation



- UplinkFast must have direct knowledge of the link failure in order to move a blocked port into a forwarding state
- In addition to its Root Port, must have 1+ other potential root ports
- If its Root Port fails, the next-lowest cost path is unblocked and used without delay (almost)
- This switchover occurs within 1 second; TCN isn't necessary or sent

UplinkFast - Requirements



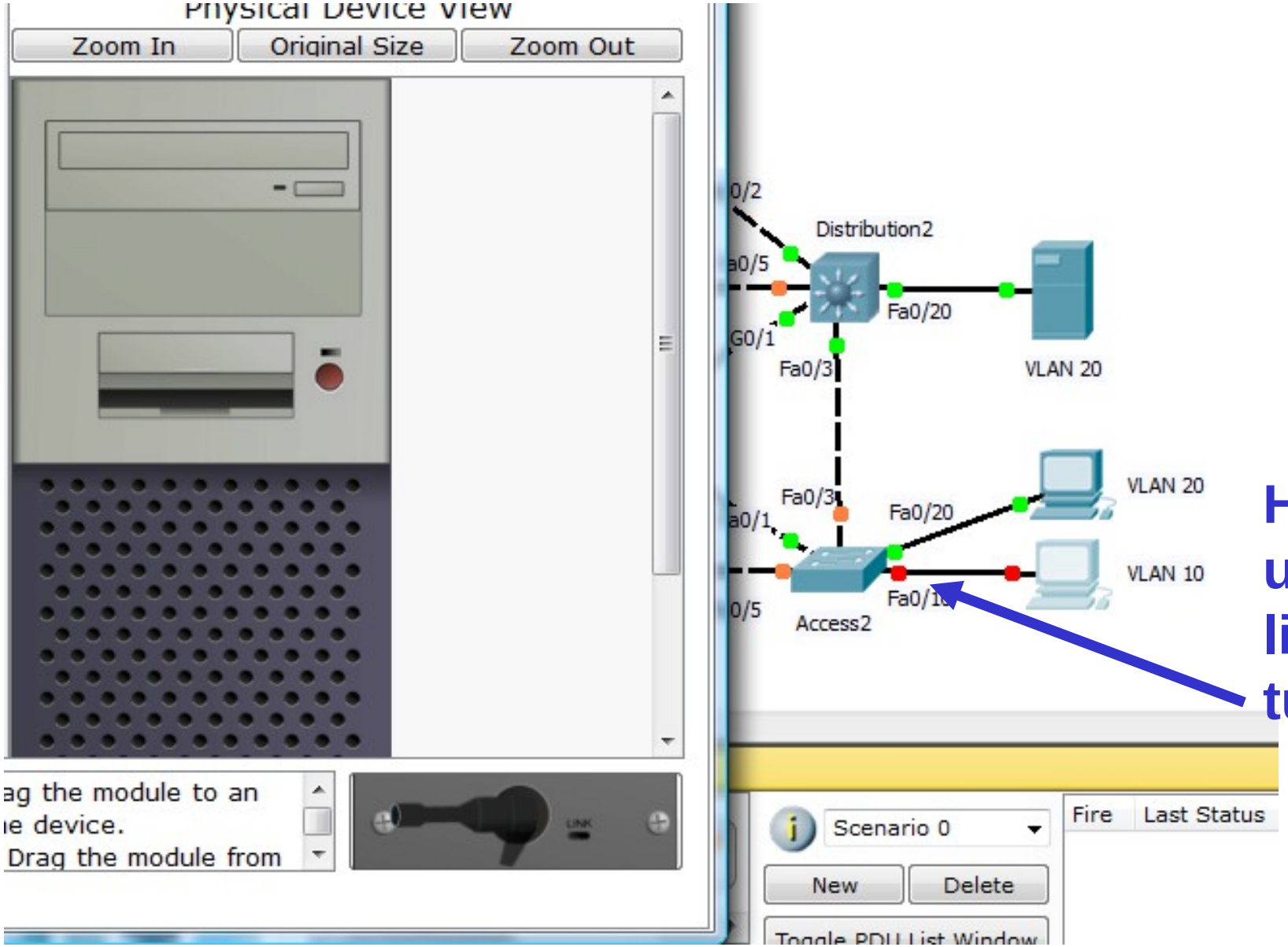

```
ALS1(config)#spanning-tree uplinkfast
```

- Is enabled for the entire switch and all VLANs
  - Not available or supported on a per-VLAN basis
- Keeps track of all possible paths to the Root Bridge.
  - So, it's NOT allowed to be the Root Bridge
  - Switch priority raised to 49,152 (0xC000) to make it unlikely to be elected the Root Bridge.

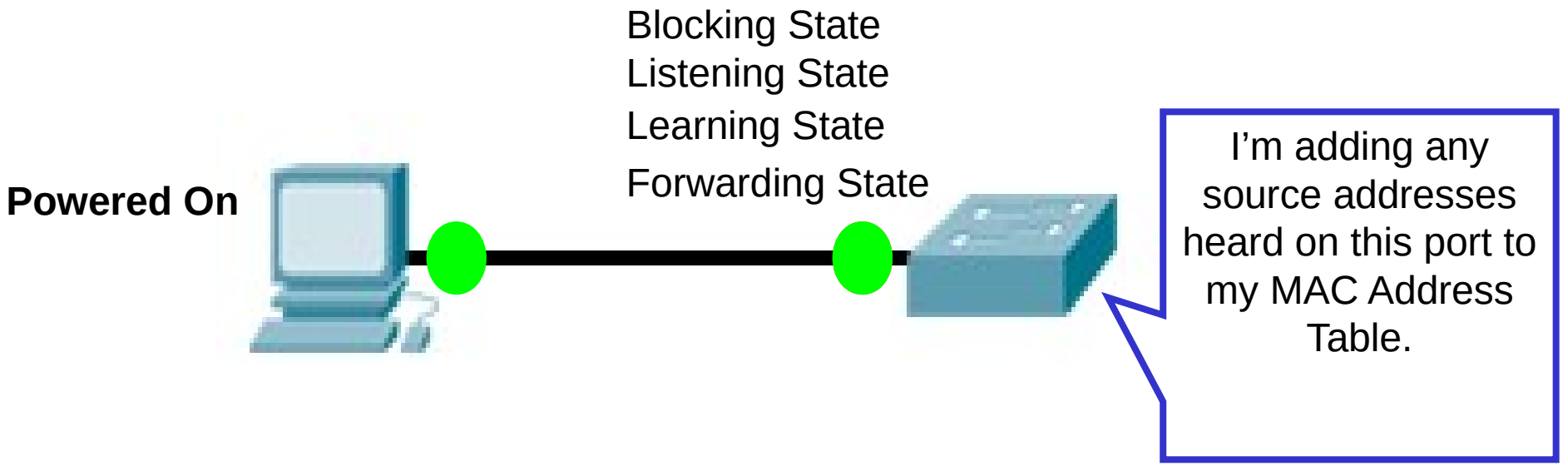
UplinkFast – In Words

- This optimization reduces the recovery time when a root port (or link) fails on a non-root and non-transit device (i.e. an Access Layer switch), by:
  - Guarding against this device becoming the Root (through increasing its bridge priority and port costs)
  - Tracking alternate ports where root BPDUs are received
- When a root port failure occurs, instead of sourcing a TCN, the device:
  - Clears its CAM of all addresses reached through the failed port
  - Transitions the least cost standby root port immediately from Blocking into the Forwarding state
  - Out of this new root port, sends a dummy multicast frame (DA=01.00.0C.CD.CD.CD) for each learned MAC address, with it as the source MAC (to update the CAM of all other switches)

PortFast – Towards Access



PortFast for Startup-delay Problem

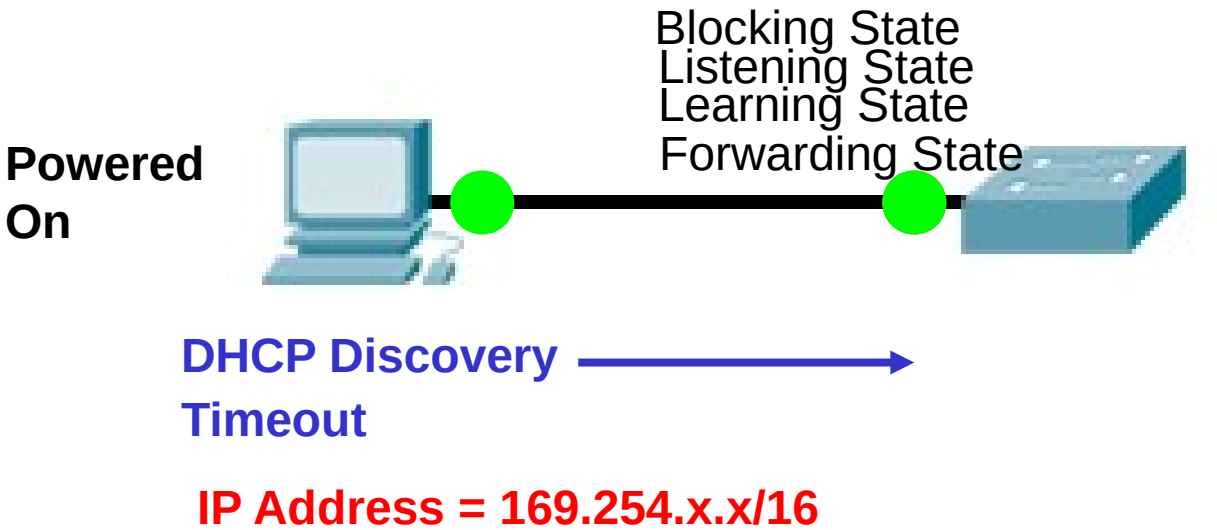


- Host powered on.
- Port goes from blocking state directly to listening state (15 sec)
  - Determines where switch fits into spanning tree topology.
- After 15 seconds port moves to learning state (15 sec).
  - Switch learns MAC addresses on this port.
- After 15 seconds port moves to forwarding state (30 sec total).

PortFast – Eliminate Startup-delay

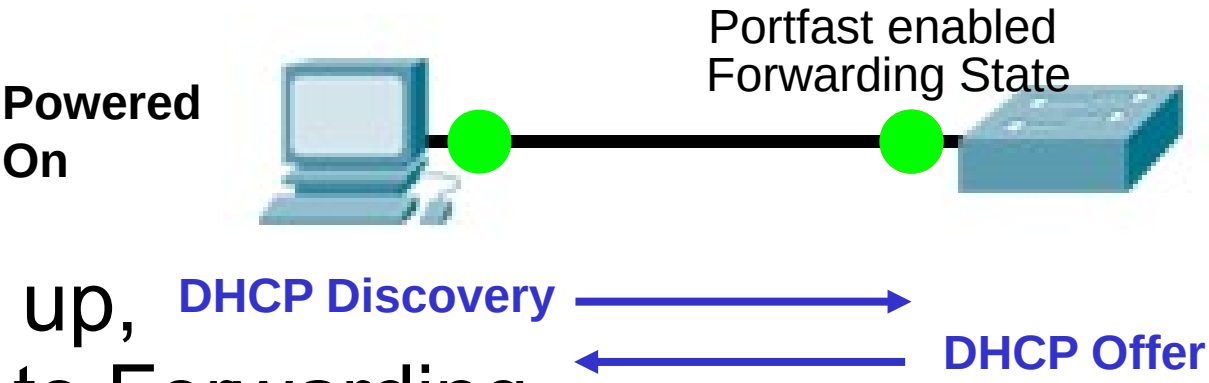
Without PortFast:

- Host sends DHCP Discovery
- Host never gets IP address info



With PortFast:

- When a port comes up, it goes immediately to Forwarding
- PortFast prevents DHCP timeouts



PortFast – No TCN

- Portfast also suppresses TCNs to enhance stability and efficiency:
- Through normal bootup/shutdown of a user PC, the link goes up or down causing the switch to source a TCN
- Individually, there's no significant impact. But given enough hosts, the topology could be subjected to a large number of nuisance TCNs and unnecessary flushing of MAC address tables
- This leads to excessive flooding of "unknown" unicast frames
- With Portfast, TCNs are blocked (ie. NOT sent) due to state changes on the port

PortFast - Configuration

- **Warning**: Only configure PortFast on ports with hosts! Temporary bridging loops can occur if hubs or switches are connected to interfaces with PortFast enabled
- Note, however, that the port still participates in STP (!) (Unless "BPDU filter" is enabled; see later)
- If a PortFast port receives a BPDU, there are four possible responses, depending on any additional configuration

Per-port config:

```
ALS1(config)#interface range fa 0/10 - 24
ALS1(config-if-range)#switchport mode access
ALS1(config-if-range)# spanning-tree portfast
```

Global (entire switch) config:

```
ALS1(config)# spanning-tree portfast default
```

PortFast – Extra Commands

- There's a macro to auto-config access ports:

```
ALS1(config-if)# switchport host
switchport mode will be set to access
spanning-tree portfast will be enabled
channel group will be disabled
ALS1(config-if)#
```

- Verify portfast:

```
ALS1#show spanning-tree int fa0/6 portfast
VLAN0010            enabled
ALS1#
```

Key points for xxx-*Fast* Features

- Where must / may each feature be configured? In each case, is it *must* or *may* ?
- What time is saved (ie. Max Age, or Fwd Delay(s))?
- Is TCN still sent, is it omitted, or is it blocked?
- How is it configured (globally or on specific ports)? What is the (single!) command per feature?
- Make a handy chart with each feature vs all the characteristics (plus a short description)

Performance: Guarding PortFast

- In a valid configuration, PortFast-configured interfaces should never receive BPDUs.
- Reception of a BPDU by a PortFast-configured interface signals a misconfiguration or the connection of an unauthorized device.
- Four levels of response exist for BPDU violations:
  1. Port *completely* ignores BPDUs (BPDU Filter, i/f)
  2. Normal but always skip Listening/Learning (default)
  3. Port loses portFast status (BPDU Filter, global)
  4. Port gets error-disabled (BPDU Guard)
- Eliminating BPDUs (BPDU Filter, i/f) should be used **only** when absolutely required!

PortFast – Default Behaviour

- **N.B.** Exact behaviour may depend on make & model of switch
- Consistent behaviour is that the port:
  - *state always* starts immediately as Forwarding, skipping the Listening and Learning states
  - *role* always starts at Designated (ie. this port must carry traffic from attached segment)
- *If* BPDUs received, port role may change between Designated and Root, or even Blocked, according to the normal election process while still skipping Listening & Learning states
- **Don't be fooled** by our weekly labs:
  - we put almost zero traffic through the network, ...
  - so BPDUs can always get through, ...
  - and any loops are eliminated immediately,
  - without causing a broadcast storm or network crash

  But **do not expect this** to be the case in production networks!

Two ideas here!
- another switch connected!
- what happens due to BPDUs?
- Also does *Not* mean no BPDUs are sent!

Portfast+BPDU Filtering: Characteristics

- BPDU filtering always stops a switch from sending BPDUs on PortFast-enabled interfaces, so unnecessary BPDUs aren't sent to hosts
- If enabled *globally* (entire switch), BPDU filtering:
  - affects all operational PortFast ports on switches that do not have BPDU filtering configured on the individual ports.
  - upon startup, the port transmits ten BPDUs to "probe" the port
  - if a BPDU is received at *any* time, the port loses its PortFast status, BPDU filtering is disabled and STP operates normally, sending or receiving BPDUs on the port, as it would with any other STP port
- If enabled on an *interface*, BPDU filtering:
  - ignores any/all received BPDUs
  - stays permanently as Desg/Fwd
  - takes precedence over global BPDU filtering

BPDU Filtering - Config+Verification (1)

- BPDU filtering **globally**:

```
ALS1(config)#spanning-tree portfast bpdufilter default
```

```
ALS1# show spanning-tree summary
Switch is in pvst mode
Root bridge for: none
Extended system ID                  is enabled
Portfast Default                    is disabled
PortFast BPDU Guard Default         is disabled
Portfast BPDU Filter Default        is disabled
Loopguard Default                   is disabled
EtherChannel misconfig guard        is enabled
UplinkFast                          is disabled
BackboneFast                        is disabled
[... output omitted ... ]
```

BPDU Filtering - Config+Verification (2)

- BPDU filtering on a **port**:

```
ALS1(config-if)# spanning-tree bpdufilter enable
```

```
ALS1# show spanning-tree interface fastEthernet 4/4 detail

Port 196 (FastEthernet4/4) of VLAN0010 is forwarding
Port path cost 19, Port priority 160, Port Identifier 160.196
Designated root has priority 32768, address 00d0.00b8.140a
Designated bridge has priority 32768, address 00d0.00b8.140a
Designated port id is 160.196, designated path cost 0
Timers:message age 0, forward delay 0, hold 0
Number of transitions to forwarding state:1
The port is in portfast mode by portfast trunk configuration
Link type is point-to-point by default
Bpdu filter is enabled
BPDU:sent 0, received 0
```

PortFast+BPDU Guard: Characteristics

- BPDU Guard always puts the port into **errdisable** state upon receipt of a BPDU as a preventive step to avoid potential bridging loops
- To exit **errdisable** state, either:
  - the admin should fix the topology problem and then must manually re-enable the interface (shut/no shut), or
  - a time-out interval has been enabled, after which the port is automatically re-enabled
- If BPDUs are again received, the port will of course be set to **errdisable**
- BPDU Filtering *per interface* takes precedence:
  - BPDU's are discarded before BPDU Guard sees them

BPDU Guard – Config+Verification (1)

- BPDU guard globally:

```
ALS1(config)# spanning-tree portfast bpduguard default
```

```
ALS1# show spanning-tree summary
Switch is in pvst mode
Root bridge for: none
Extended system ID                  is enabled
Portfast Default                    is enabled
PortFast BPDU Guard Default         is enabled
[... output omitted ... ]
```

Show triggered: **ALS1# show interfaces status err-disabled**

*Mar 1 00:51:50.680: %SPANTREE-2-BLOCK_BPDUGUARD: **Received BPDU** on port Fa0/7 with BPDU Guard enabled. Disabling port

*Mar 1 00:51:50.680: %PM-4-ERR_DISABLE: bpduguard **error detected** on Fa0/7, putting Fa0/7 in **err-disable state**

BPDU Guard – Config+Verification (2)

- BPDU guard on a port:

ALS1(config-if)# **spanning-tree bpduguard enable**

```
Switch# show spanning-tree interface fastEthernet 4/4 detail

Port 196 (FastEthernet4/4) of VLAN0010 is forwarding
Port path cost 19, Port priority 160, Port Identifier 160.196
Designated root has priority 32768, address 00d0.00b8.140a
Designated bridge has priority 32768, address 00d0.00b8.140a
Designated port id is 160.196, designated path cost 0
Timers:message age 0, forward delay 0, hold 0
Number of transitions to forwarding state:1
The port is in portfast mode by portfast trunk configuration
Link type is point-to-point by default
Bpdu guard is enabled
BPDU: sent 1377, received 0
```

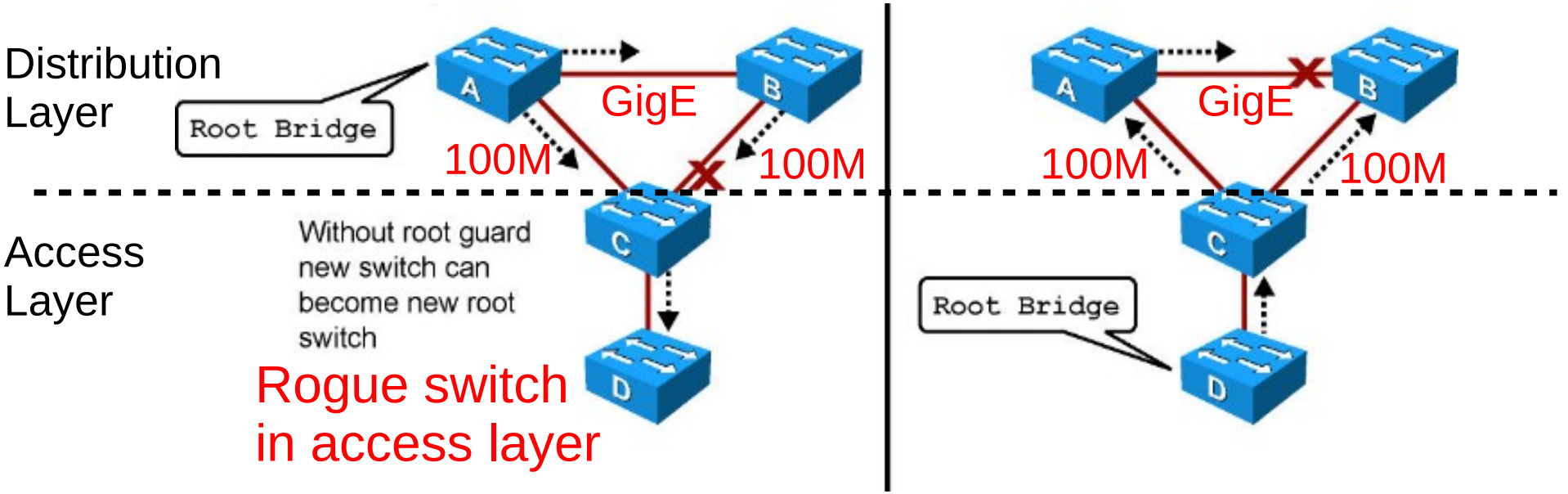
Show triggered: **ALS1# show interfaces status err-disabled**

Other Enhancements

- There are four other enhancements to supplement and/or supplant STP:
  - **Root guard** which protects against hackers & accidental (stupidity?) connection of extra switch
  - **Loop guard** which is (Control Plane) protection against loops from uni-directional links
  - **UDLD** which is (L2 - Data Plane) protection against loops from uni-directional links
  - **Flex-links** which provide automatic 50 msec fail-over; they are mutually exclusive with STP
- We'll look at each of these in the coming section

Root Guard – Basics

- Root Guard is on host ports of access-layer switches (and possibly one layer higher up). It prevents a leaf-node switch from becoming root
- Lower-layer switches (eg. access) shouldn't be root switches because:
  – lower bandwidth;
  – less powerful/capable switches
  – sub-optimal topology
- The scenario below shows a topology before and after connecting a switch (either accidentally or maliciously). Note the speed of the links and consider the path that traffic (left->right) follows before and after



Root Guard - Operation

- The recommended best-practice is to enable root guard on all access ports so that a Root bridge can never be established through these ports
- Configured ports are "locked" to Designated role
- In the example, Switch C should block the port connecting to Switch D when it receives a superior BPDU, by transitioning the port to **root-inconsistent** state. No traffic passes through a port in this state.
- When Switch D stops sending superior BPDUs, the port automatically unblocks, and proceeds through regular STP transitions of Listening, Learning, and eventually to the Forwarding state. Recovery is always fully automatic with no administrative intervention required.

Root Guard – Config & Verification

- Root Guard is configured per-interface

```
ALS1(config)# interface FastEthernet 0/1
ALS1(config-if)# spanning-tree guard root
ALS1(config)# end

%SPANTREE-2-ROOTGUARDBLOCK: Port 1/1 tried to become non-
designated in VLAN 77. Moved to root-inconsistent state.
```

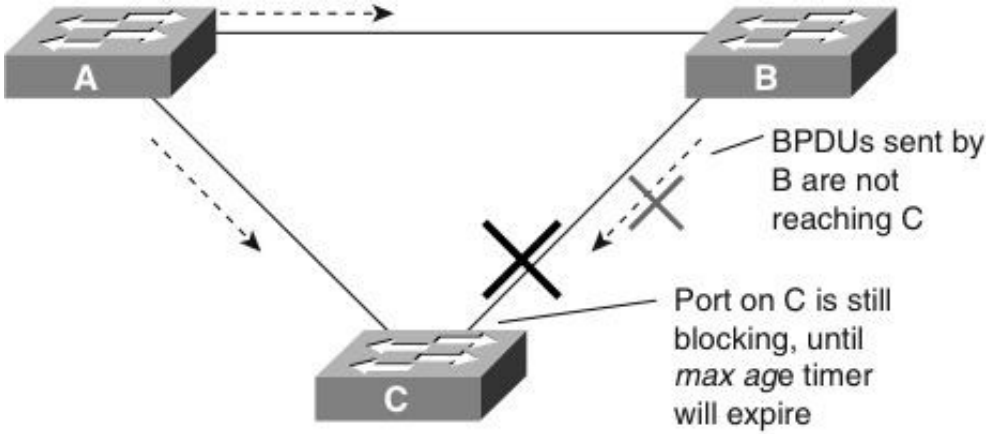
Show triggered: **ALS1# show spanning-tree inconsistentports**

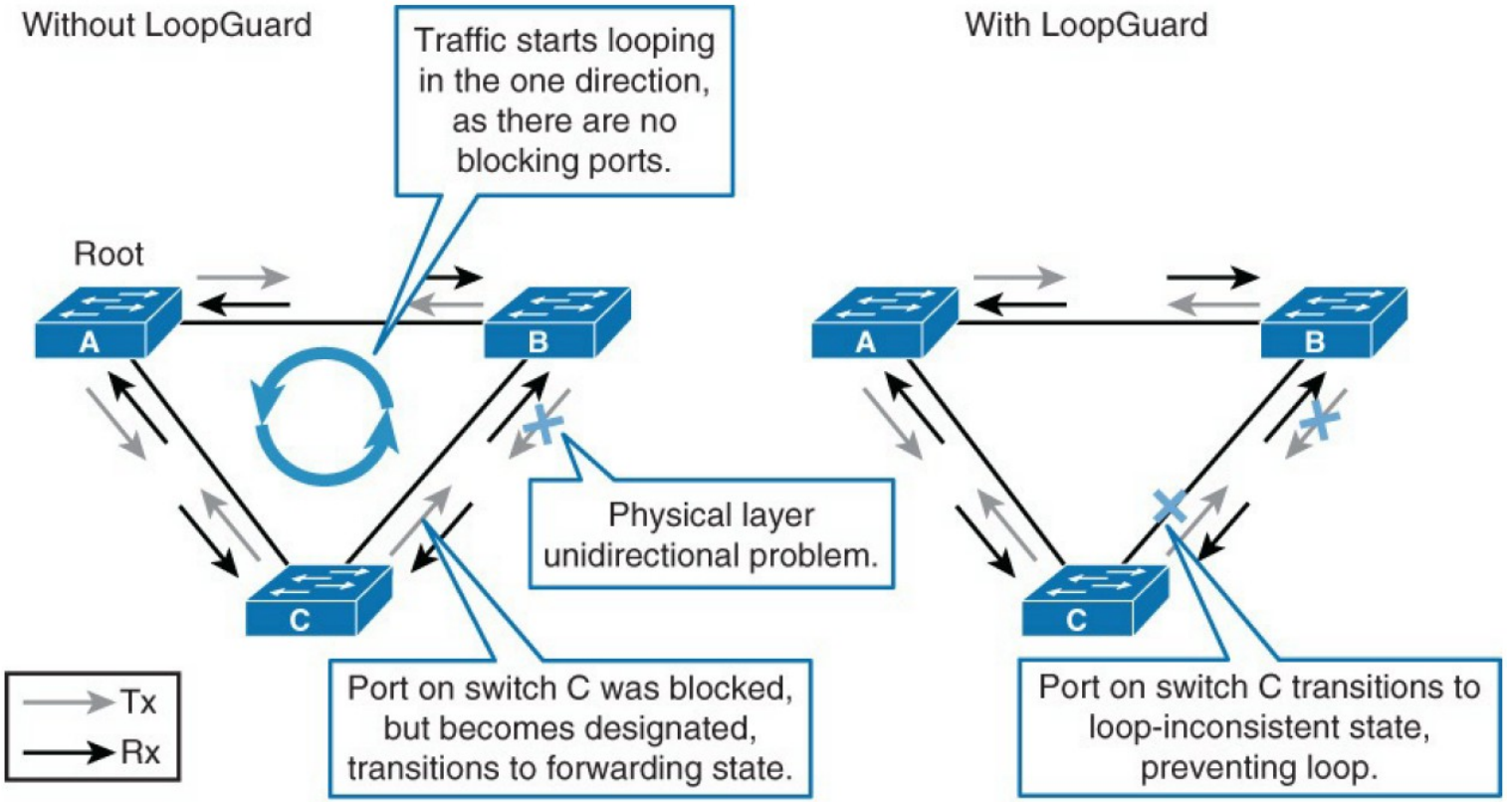
```
Name         Interface              Inconsistency
--------     ---------------        ----------------------
VLAN0001     FastEthernet3/1        Port Type Inconsistent
VLAN0001     FastEthernet3/2        Port Type Inconsistent
VLAN1002     FastEthernet3/1        Port Type Inconsistent
VLAN1002     FastEthernet3/2        Port Type Inconsistent
Number of inconsistent ports (segments) in the system: 4
```

Unidirectional Links – Basics

- Uni-directional links are created in multiple different ways:
  - cabling/connector failure, e.g. on a fibre link where one fibre in the full-duplex pair has been damaged
  - hardware failure (e.g. transceiver failure)
- Uni-directional links that can transmit but not receive are the type that cause loops which STP cannot fix: ports that should be blocked fail to receive BPDUs and transition to Designated as a result.
- These loops can bring down the network, with the greatest risk being trunk links (since they carry the most traffic)
- Two features can guard against this type of problem
  - Loop Guard: protection implemented by Control Plane
  - Uni-Directional Link Detection (UDLD) in the Data Plane

Loop Guard - Illustrations





Loop Guard - Characteristics

- Since Loop Guard *requires* receiving BPDUs for proper operation:
  - it should *only* be set on Root or Alternate/Blocking ports
  - it should *not* be set on ports that can become designated
  - … for all possible combinations of active topologies!
- If a non-designated port stops receiving BPDUs, the port is placed into **loop-inconsistent** blocking state
- If a port in the loop-inconsistent receives a BPDU, the port transitions through STP states according to the received BPDU. As a result, recovery is automatic, and no manual intervention is necessary

Loop Guard – Per VLAN

- Loop Guard is implemented in the Control Plane so even though Loop Guard is configured on a per-port basis, on a trunk port the feature blocks inconsistent ports on a per-VLAN basis
  - for example, if BPDUs are not received for only one particular VLAN, the switch blocks only that VLAN (that is, the port for only that VLAN, moves to the loop-inconsistent STP state).
- For LAG interfaces, the channel status goes to the loop-inconsistent state for all the ports belonging to the channel group for the particular VLAN not receiving BPDUs

Loop Guard – Config & Verification (1)

- Loop guard globally:

```
Sw(config)# spanning-tree loopguard default
```

- When enabled globally, the switch enables Loop Guard only on ports considered to be point-to-point (full-duplex links)
- Blocking and unblocking events generate log messages:

SPANTREE-2-LOOPGUARDBLOCK: No BPDUs were received on port 4/4 in vlan 3. Moved to **loop-inconsistent** state

SPANTREE-2-LOOPGUARDUNBLOCK: port 4/4 restored in vlan 3

Loop Guard – Illustration (2)

… So if Loop Guard can be set globally, how do you avoid it being applied to ports that are already designated??

"If BPDUs are not received on a nondesignated port ... that port is moved into the STP loop-inconsistent blocking state"
"The Loop Guard works on nondesignated ports and does not allow the port to become designated through the expiration of maximum age."

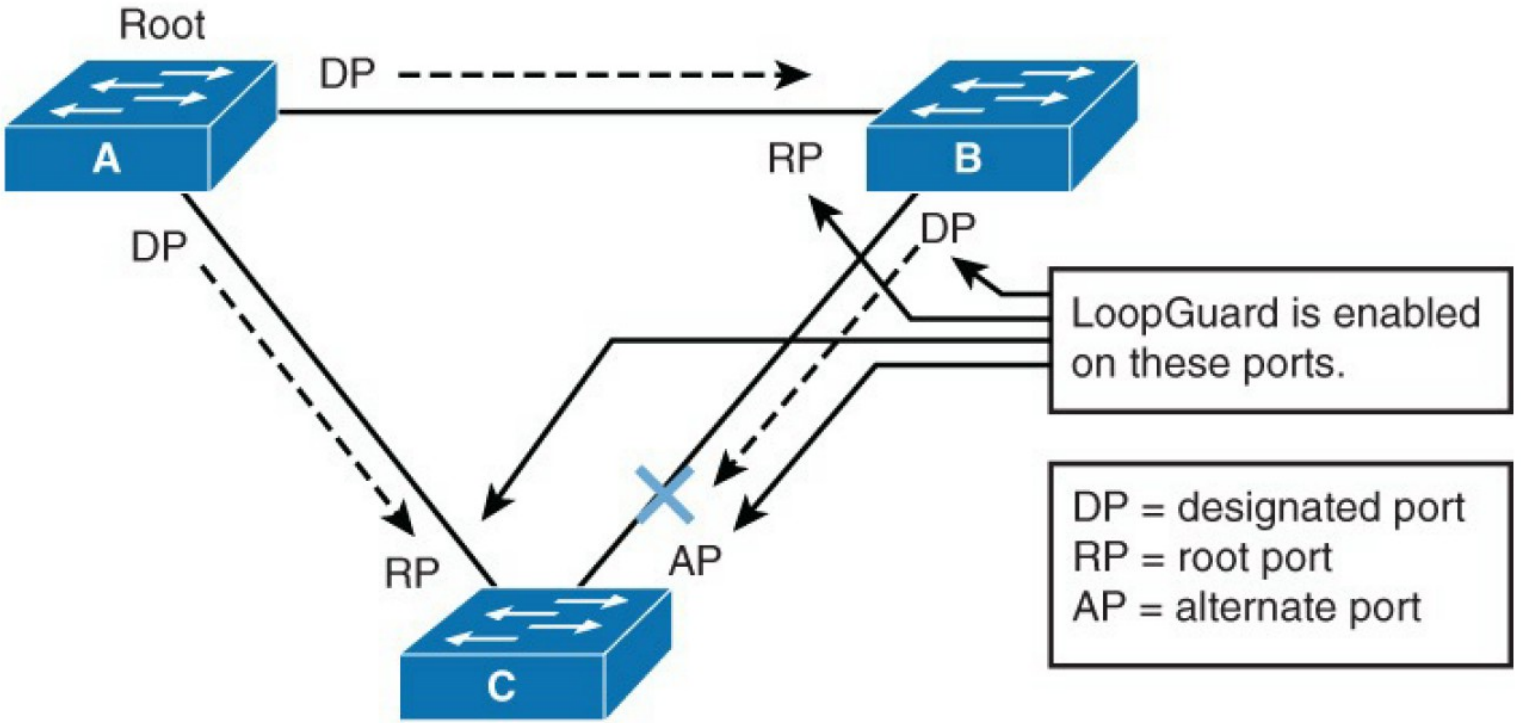


Fig 4-23 from FLG

Loop Guard – Config & Verification (2)

- Loop guard on a port:

```
Sw(config-if)# spanning-tree guard loop
```

```
Switch# show spanning-tree interface fastEthernet 4/4 detail

Port 196 (FastEthernet4/4) of VLAN0003 is forwarding
Port path cost 19, Port priority 160, Port Identifier 160.196
[... output omitted …]
Number of transitions to forwarding state: 1
Link type is point-to-point by default
Loop guard is enabled on the port
BPDU: sent 1, received 4501
```

Show triggered: **ALS1# show spanning-tree inconsistentports**

Show triggered: **ALS1# show spanning-tree interface fa0/1**

```
[... output omitted ...]
VLAN0002         Desg BKN*19     28.3    P2p *LOOP_Inc
[... output omitted ...]
```

UDLD – Basics

- Loop Guard has an important limitation: it *requires* receiving BPDUs for proper operation so can *only* be set on particular ports (ie. Root or Alt/Blk);
  ** UDLD overcomes this limitation **
- Uni-Directional Link Detection is it's own totally separate, independent L2 protocol (other vendors have equivalents: e.g. BFD on Nokia routers)
  http://packetlife.net/blog/2011/mar/7/udld/ https://en.wikipedia.org/wiki/UDLD
- UDLD is helpful to all protocols using an interface, but is especially useful as a complement to STP
- UDLD timers can be set so that it's "fast enough" to prevent STP loops, or (much) faster to speed up convergence of any/all other protocols using an i/f.

UDLD – Characteristics

Two modes of operation (less / more useful!)

- Normal mode: if UDLD messages stop arriving (~3 x msg interval), the UDLD port state changes to **undetermined** (… and nothing else happens!)
- Aggressive mode: if UDLD messages stop, UDLD tries to re-establish the connection by sending 8 rapid-fire messages to the neighbor. If still no response, the port is put in **err-disable** state (very visible!)
- Aggressive mode UDLD detects a greater variety of link problems and is more relevant to twisted-pair (copper) links

http://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/10591-77.html
http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3560/software/release/12-1_19_ea1/configuration/guide/3560scg/swudld.html

"The need for UDLD has been eliminated when operating over 10GbE, as per 802.3ae/D3.2"

UDLD – Configuration

- UDLD is disabled by default
- Global config: affects fiber-optic interfaces only

```
Sw(config)# udld enable                ! Normal mode
Sw(config)# udld aggressive            ! Aggressive
```

- Per port config: affects twisted-pair and fiber ports

```
Sw(config-if)# udld port               ! Normal mode
Sw(config-if)# udld port aggressive    ! Aggressive
```

  As always, per-port setting overrides the global

- Err-disabled ports are reset:
  – by admin port bounce (shut / no shut)
  – by **`udld reset`** command
  – automatically, if time-out interval has been enabled

UDLD – Verification

Show triggered: `Sw# show interface status err-disabled`

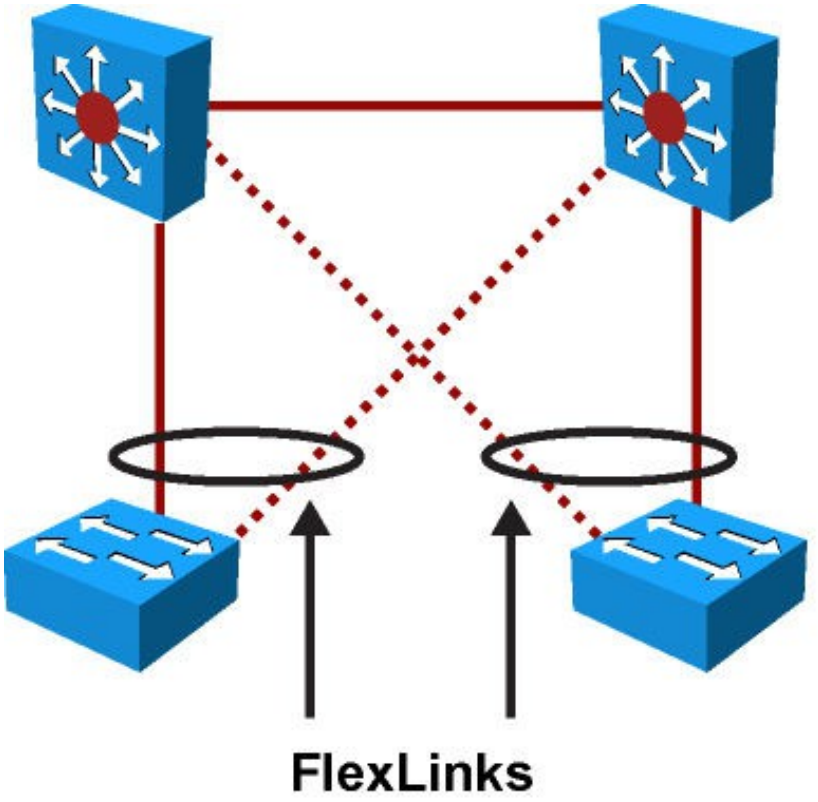
```
Sw# show udld gi5/1
Interface Gi5/1
---
Port enable administrative config setting: Enabled / in aggressive mode
Port enable operational state: Enabled / in aggressive mode
Current bidirectional state: Bidirectional
Current operational state: Advertisement - Single neighbor detected
Message interval: 15
Time out interval: 5
Entry 1
---
Expiration time: 38      ! Time remaining until UDLD is triggered
Device ID: 1
Current neighbor state: Bidirectional
Device name: FOX06310RW1
Port ID: Gi1/1
Neighbor echo 1 device: FOX0627A001
Neighbor echo 1 port: Gi5/1
Message interval: 15
Time out interval: 5
CDP Device name: SwitchB
```

Loop Guard vs UDLD Aggressive mode

	Loop Guard	**Aggressive Mode UDLD**
Configuration	Per port	Per port
Action granularity	Per VLAN	Per port
Auto-recovery	Yes	Yes, with err-disable timeout feature
Protection against STP failures caused by unidirectional links	Yes, when enabled on all root ports and alternate ports in redundant topology	Yes, when enabled on all links in redundant topology
Protection against STP failures caused by problem in software in designated bridge not sending BPDUs	Yes	No
Protection against mis-wiring	No	Yes

Flex-Links – Basics

- Flex Links are a Layer 2 redundancy solution; they are mutually exclusive to STP (one or the other per port!)
- Flex Links are based on *pairs* of L2 ports, 1 active + 1 standby, originating on a common switch; ports may be interfaces or LAGs



- Flex Links provide sub-50 msec re-convergence time, regardless of the number of VLANs or MAC addresses configured on switch uplink ports
- Flex Links can coexist with STP configured on other switches, but those switches are unaware of the Flex Links

Flex Links - Characteristics

- Only for **Layer 2** ports & LAGs, not VLANs or Layer 3 ports
- Flex Links are like a "married couple": exclusively 1-to-1
- A Layer 2 port (**active** link) is assigned another Layer 2 port as the Flex Link (**backup** link).
- When the active link is forwarding, the backup link is in standby mode. Only one port is forwarding at a time.
- If the **active** link goes down, the **backup** link starts forwarding. Traffic automatically and immediately reverts to the **active** link as soon as it comes back up; the **backup** link goes to standby
- STP is automatically disabled on Flex Link ports.
  Flex Link ports do not participate in STP, even if VLANs present on the port are configured for STP. The human admin is responsible for ensuring that no loops exist!!

Flex Links – Config & Verification

- Per port configuration:

```
Sw(config-if)# switchport backup {interface}
```

```
Switch(config)# interface fa0/1
Switch(config-if)# switchport backup interface fa0/2
Switch(config-if)# end
Switch# show interface switchport backup

Switch Backup Interface Pairs:
Active Interface Backup Interface State
---------------- ---------------- --------------------
FastEthernet0/1  FastEthernet0/2  Active Up/Backup Stdby
```

Reminder

- *LOTS* of details do not appear in these slides
- You are responsible for *reading* the textbook to gain the knowledge (memorization) and understanding (apply the knowledge)