

Troubleshooting STP on Catalyst Switches Running Cisco IOS System Software

Document ID: 28943

Introduction

Prerequisites

Requirements

Components Used

Conventions

Why STP Fails

Troubleshooting Forwarding Loops

Troubleshooting Excessive Topology Changes Causing Flooding

Troubleshooting Convergence Time Related Issues

STP Debugging Commands

Securing the Network Against Forwarding Loops

Related Information

Introduction

This document provides guidelines to use Cisco IOS® software to troubleshoot issues with Spanning–Tree Protocol (STP). There are specific commands which apply to the Catalyst 6500/6000 only; however, you can apply most of the principles to any Cisco Catalyst switch that runs Cisco IOS software.

Most STP troubleshooting revolves around three issues:

- forwarding loops
- excessive flooding due to a high rate of STP Topology Changes (TC)
- issues related to convergence time

Because bridging does not have any mechanism to track whether a certain packet is being forwarded multiple times (for example, an IP Time to Live [TTL] is used to discard traffic that is circulating too long in the network), only one path can exist between two devices in the same Layer 2 (L2) domain.

The purpose of STP is to block redundant ports based on an STP algorithm, to resolve redundant physical topology into a tree–like topology. A forwarding loop (such as an STP loop) occurs when no port in a redundant topology is blocked, and traffic is forwarded in circles indefinitely.

Once the forwarding loop starts, it will likely congest the lowest–bandwidth links along its path if all the links are of the same bandwidth, all links will likely be congested. This congestion will cause packet loss and will lead to a network down situation in the affected L2 domain.

With excessive flooding, the symptoms might not be as apparent. Some slow links might become congested by flooded traffic, and devices or users behind these congested links might experience slowness or total loss of connectivity.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Various Spanning Tree types and how to configure them. Refer to *Configuring STP and IEEE 802.1s MST* for more information.
- Various Spanning Tree features and how to configure them. Refer to *Configuring STP Features* for more information.

Components Used

The information in this document is based on these software and hardware versions:

- Catalyst 6500 with Supervisor 2 engine
- Cisco IOS Software Release 12.1(13)E

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

Refer to *Cisco Technical Tips Conventions* for more information on document conventions.

Why STP Fails

STP makes certain assumptions about its operational environment. These are the assumptions most relevant to this document:

- Each link between the two bridges is bidirectional. This means that, if A directly connects to B, then A will receive what B has sent and B will receive what A has sent, as long as the link is up between them.
- Each bridge that is running STP is able to regularly receive, process, and transmit STP Bridge Protocol Data Units (BPDUs), also known as STP packets.

While these assumptions appear logical and obvious, there are situations when they are not met. Most of these situations involve some sort of hardware issue; however, software defects may also lead to STP failures. Various hardware failures, misconfigurations, or miscabling cause the majority of STP failures, while software failures account for the minority. STP failures can also occur due to unnecessary additional connections that exist between the switches. VLANs go into a down state because of these additional connections. To resolve this problem, remove all of the unwanted connections between the switches.

When one of these assumptions is not met, one or more bridges might no longer receive or process the BPDUs. This means that the bridge (or bridges) will not be able to discover the network topology. Without knowledge of the correct topology, the switch can not block the loops. Therefore, the flooded traffic will circulate over the looped topology, consume all bandwidth, and bring down the network.

Examples of why the switches may not receive BPDUs include bad transceivers or Gigabit Interface Converters (GBICs), cabling issues, or hardware failures on the port, the linecard, or the Supervisor engine. One frequent reason for STP failures is a unidirectional link between the bridges. In such a condition, one bridge sends BPDUs, but the downstream bridge never receives them. STP processing can also be disrupted by an overloaded CPU (99 percent or more), because the switch is unable to process received BPDUs. BPDUs can be corrupted along the path from one bridge to the other, which also prevents proper STP behavior.

Aside from the forwarding loops, when no ports are blocked, there are situations when only certain packets are incorrectly forwarded through the blocking ports. In most cases, this is caused by software issues. Such behavior might cause slow-loops. This means some packets are looped, but the majority of the traffic is still flowing through the network, because the links are probably not congested.

The remaining sections in this document provide guidelines to troubleshoot the most common STP-related issues.

Troubleshooting Forwarding Loops

Forwarding loops vary greatly both in their origin (cause) and effect. Due to the wide variety of issues that can affect STP, this document can only provide general guidelines about how to troubleshoot forwarding loops.

Before you start to troubleshoot, you must obtain this information:

- An actual topology diagram that details all of the switches and bridges
- Their corresponding (interconnecting) port numbers
- STP configuration details, such as which switch is the root and backup root, which links have a non-default cost or priority, and the location of blocking ports

Generally, troubleshooting involves these steps (depending on the situation, some steps may not be necessary):

1. Identify the loop.

When a forwarding loop has developed in the network, these are the usual symptoms:

- ◆ Loss of connectivity to, from, and through affected network regions
- ◆ High CPU utilization on routers connected to affected segments or VLANs that can lead to various symptoms, such as routing protocol neighbor flapping or Hot Standby Router Protocol (HSRP) active router flapping
- ◆ High link utilization (often 100 percent)
- ◆ High switch backplane utilization (compared to the baseline utilization)
- ◆ Syslog messages that indicate packet looping in the network (for example HSRP duplicate IP address messages)
- ◆ Syslog messages that indicate constant address relearning or MAC address flapping messages
- ◆ An increasing number of output drops on many interfaces

Note: Any of these reasons alone may indicate different issues (or no issue at all). However, when many of these are observed at the same time, it is highly probable that a forwarding loop has developed in the network.

Note: The fastest way to verify this is to check the switch backplane traffic utilization:

```
cat# show catalyst6000 traffic-meter

traffic meter = 13% Never cleared
peak = 14% reached at 12:08:57 CET Fri Oct 4 2002
```

Note: The Catalyst 4000 with Cisco IOS software does not currently support this command.

If the current traffic level is well above normal or if the baseline level is not known, check whether the peak level has been achieved recently and whether it is close to the current traffic level. For example, if the peak traffic level is 15 percent and it was reached just two minutes ago and current traffic level is 14 percent, then that would mean that the switch is working under an unusually high

load.

If the traffic load is at a normal level, then that probably means that there is either no loop or that this device is not involved in the loop. However, it still could be involved in a slow loop.

2. Discover the topology (scope) of the loop.

Once it has been established that the reason for the network outage is a forwarding loop, the highest priority is to stop the loop and restore the network operation. In order to stop the loop, you must know which ports are involved in the loop: look at the ports with the highest link utilization (packets per second). The **show interface** Cisco IOS software command displays the utilization for each interface.

In order to display only the utilization information and the interface name (for a quick analysis), you might use Cisco IOS software regular expression output filtering. Issue the **show interface | include line|\sec** command to display only the packet per second statistics and the interface name:

```
cat# show interface | include line|\sec

GigabitEthernet2/1 is up, line protocol is down
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
GigabitEthernet2/2 is up, line protocol is down
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
GigabitEthernet2/3 is up, line protocol is up
  5 minute input rate 99765230 bits/sec, 24912 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
GigabitEthernet2/4 is up, line protocol is up
  5 minute input rate 1000 bits/sec, 27 packets/sec
  5 minute output rate 101002134 bits/sec, 25043 packets/sec
GigabitEthernet2/5 is administratively down, line protocol is down
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
GigabitEthernet2/6 is administratively down, line protocol is down
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
GigabitEthernet2/7 is up, line protocol is down
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
GigabitEthernet2/8 is up, line protocol is up
  5 minute input rate 2000 bits/sec, 41 packets/sec
  5 minute output rate 99552940 bits/sec, 24892 packets/sec
```

Pay particular attention to the interfaces with the highest link utilization. In this example, these are interfaces g2/3, g2/4, and g2/8; they are probably the ports that are involved in the loop.

3. Break the loop.

To break the loop, you must shut down or disconnect the involved ports. It is very important to not only stop the loop but to also find and fix the root cause of the loop. It is relatively easier to break the loop.

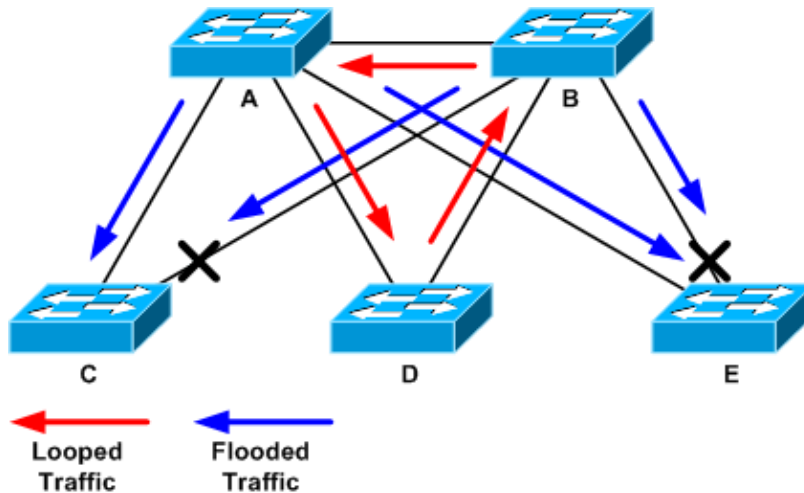
Note: In order to help subsequent cause analysis, you need not shut down or disconnect all ports at once; instead, shut them down one at a time. It is generally better to shut down ports at the aggregation point affected by the loop, such as a distribution or core switch. If you shut down all of the ports at once and enable or reconnect them one-by-one, it might not work; the loop will be stopped and might not start immediately after the offending port is reconnected. Therefore, it would be difficult to correlate failure to any particular port.

Note: It is recommended that you collect information before you reboot the switches to break the loop. Otherwise, subsequent root cause analysis will be very difficult.

After you disable or disconnect each port, you must check whether the switch backplane utilization is back to a normal level.

Note: Keep in mind that, usually, some ports are not sustaining the loop but, rather, are flooding the traffic arriving with the loop. When you shut down such flooding ports, you will only reduce backplane utilization a small amount, but you will not stop the loop.

In the next example topology, the loop is between switches A, B, and D. Therefore, links AB, AD, and BD are sustaining. If you shut down any of these links, you will stop the loop. Links AC, AE, BC, and BE are just flooding traffic arriving with the loop.



After the sustaining port is shut down, backplane utilization will go down to a normal value. It is very important to note which ports shutdown brought backplane utilization (and other ports utilization) to a normal level.

At this point, the loop will be stopped and the network operation should improve; however, because the original cause of the loop was probably not fixed, there might still be some issues outstanding.

4. Find and fix the cause of the loop.

Once the loop has been stopped, you need to determine the reason why the loop began. This is often the most difficult part of the process, because the reasons can vary. It is also difficult to formalize an exact procedure which works in every case. However, these are some general guidelines:

- ◆ Investigate the topology diagram, to find a redundant path. This includes the sustaining port found in the previous step that comes back to the same switch (the path packets were taking during the loop). In the previous example topology, this path is AD–DB–BA.
- ◆ For every switch on the redundant path, check for these issues:
 - a. Does the switch know the correct STP root?

All switches in an L2 network should agree on a common STP root. It is a clear symptom of problems when bridges consistently display a different ID for the STP root in a particular VLAN or STP instance. Issue the **show spanning-tree vlan *vlan-id*** command to display the root bridge ID for a given VLAN:

```
cat# show spanning-tree vlan 333

MST03
Spanning tree enabled protocol mstp
Root ID    Priority    32771
           Address    0050.14bb.6000
           Cost      20000
```

```

Port          136 (GigabitEthernet3/8)
Hello Time    2 sec  Max Age 20 sec  Forward Delay 15 sec

Bridge ID Priority 32771 (priority 32768 sys-id-ext 3)
Address       00d0.003f.8800
Hello Time    2 sec  Max Age 20 sec  Forward Delay 15 sec

Interface      Role Sts Cost      Prio.Nbr Status
-----
Gi3/8          Root FWD 20000     128.136 P2p
Po1            Desg FWD 20000     128.833 P2p

```

The VLAN number can be found from the port, because ports involved in the loop were established in previous steps. If the ports in question are trunks, often all VLANs on the trunk are involved. If this is not the case (for example, if it appears that the loop has happened on a single VLAN) then you can try to issue the **show interfaces | include L2|line|broadcast** command (only on Supervisor 2 and later engines on Catalyst 6500/6000 series switches, because Supervisor 1 does not provide per-VLAN switching statistics). Look at VLAN interfaces only. The VLAN with the highest amount of switched packets will most often be the one where the loop occurred:

```

cat# show int | include L2|line|broadcast

Vlan1 is up, line protocol is up
  L2 Switched: ucast: 653704527 pkt, 124614363025 bytes - mcast:
    23036247 pkt, 1748707536 bytes
  Received 23201637 broadcasts, 0 runts, 0 giants, 0 throttles
Vlan10 is up, line protocol is up
  L2 Switched: ucast: 2510912 pkt, 137067402 bytes - mcast:
    41608705 pkt, 1931758378 bytes
  Received 1321246 broadcasts, 0 runts, 0 giants, 0 throttles
Vlan11 is up, line protocol is up
  L2 Switched: ucast: 73125 pkt, 2242976 bytes - mcast:
    3191097 pkt, 173652249 bytes
  Received 1440503 broadcasts, 0 runts, 0 giants, 0 throttles
Vlan100 is up, line protocol is up
  L2 Switched: ucast: 458110 pkt, 21858256 bytes - mcast:
    64534391 pkt, 2977052824 bytes
  Received 1176671 broadcasts, 0 runts, 0 giants, 0 throttles
Vlan101 is up, line protocol is up
  L2 Switched: ucast: 70649 pkt, 2124024 bytes - mcast:
    2175964 pkt, 108413700 bytes
  Received 1104890 broadcasts, 0 runts, 0 giants, 0 throttles

```

In this example, VLAN 1 accounts for the highest number of broadcasts and L2-switched traffic.

b. Is the root port identified correctly?

The root port should have the lowest cost to the root bridge (sometimes one path is shorter in terms of hops but longer in terms of cost, as low-speed ports have higher costs).

To determine which port is considered the root for a given VLAN, issue the **show spanning-tree vlan *vlan*** command:

```

cat# show spanning-tree vlan 333

MST03
Spanning tree enabled protocol mstp
Root ID    Priority    32771
Address    0050.14bb.6000
Cost       20000
Port       136 (GigabitEthernet3/8)

```

```

Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

Bridge ID Priority 32771 (priority 32768 sys-id-ext 3)
Address 00d0.003f.8800
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

Interface Role Sts Cost Prio.Nbr Status
-----
Gi3/8 Root FWD 20000 128.136 P2p
Po1 Desg FWD 20000 128.833 P2p

```

c. Are BPDUs received regularly on the root port and on ports which are supposed to be blocking?

BPDUs are sent by the root bridge at every hello interval (two seconds by default). Non-root bridges receive, process, modify, and propagate the BPDUs that are received from the root.

Issue the **show spanning-tree interface *interface* detail** command to see if the BPDUs are being received:

```

cat# show spanning-tree interface g3/2 detail

Port 130 (GigabitEthernet3/2) of MST00 is backup blocking
Port path cost 20000, Port priority 128, Port Identifier 128.130.
Designated root has priority 0, address 0007.4f1c.e847
Designated bridge has priority 32768, address 00d0.003f.8800
Designated port id is 128.129, designated path cost 2000019
Timers: message age 4, forward delay 0, hold 0
Number of transitions to forwarding state: 0
Link type is point-to-point by default, Internal
Loop guard is enabled by default on the port
BPDU: sent 3, received 53

```

```

cat# show spanning-tree interface g3/2 detail

Port 130 (GigabitEthernet3/2) of MST00 is backup blocking
Port path cost 20000, Port priority 128, Port Identifier 128.130.
Designated root has priority 0, address 0007.4f1c.e847
Designated bridge has priority 32768, address 00d0.003f.8800
Designated port id is 128.129, designated path cost 2000019
Timers: message age 5, forward delay 0, hold 0
Number of transitions to forwarding state: 0
Link type is point-to-point by default, Internal
Loop guard is enabled by default on the port
BPDU: sent 3, received 54

```

Note: One BPDU has been received between the two outputs of the command (the counter went from 53 to 54).

The counters shown are actually counters maintained by the STP process itself. This means that, if the receive counters incremented, not only was BPDU received by a physical port but it was also received by the STP process.

If the received BPDU counter is not incrementing on the port that is supposed to be the root alternate or backup port, then check whether the port is receiving any multicasts at all (BPDUs are sent as multicast). Issue the **show interface *interface* counters** command:

```

cat# show interface g3/2 counters

Port InOctets InUcastPkts InMcastPkts InBcastPkts
Gi3/2 14873036 2 89387 0

```

```

Port          OutOctets  OutUcastPkts  OutMcastPkts  OutBcastPkts
Gi3/2         114365997      83776         732086         19

cat# show interface g3/2 counters

Port          InOctets    InUcastPkts  InMcastPkts  InBcastPkts
Gi3/2         14873677      2             89391         0

Port          OutOctets  OutUcastPkts  OutMcastPkts  OutBcastPkts
Gi3/2         114366106      83776         732087         19

```

(A brief description for STP port roles can be found in the Brief Summary of STP port roles section of Spanning–Tree Protocol Enhancements using Loop Guard and BPDU Skew Detection Features.)

If no BPDUs are received, check whether the port is not counting errors. Issue the **show interface interface counters errors** command:

```

cat# show interface g4/3 counters errors

Port      Align-Err  FCS-Err  Xmit-Err  Rcv-Err  UnderSize  OutDiscards
Gi4/3     0          0        0         0         0          0

Port      Single-Col  Multi-Col  Late-Col  Excess-Col  Carri-Sen  Runts  Giants
Gi4/3     0          0         0         0          0         0      0

```

It is possible that the BPDUs are received by the physical port but still do not reach the STP process. If the commands used in the previous two examples show that some multicasts are received, and errors are not incrementing, then check whether the BPDUs are being dropped at the STP process level. Issue the **remote command switch test spanning–tree process–stats** command on the Catalyst 6500:

```

cat# remote command switch test spanning-tree process-stats

-----TX STATS-----
transmission rate/sec      = 2
paks transmitted           = 5011226
paks transmitted (opt)     = 0
opt chunk alloc failures   = 0
max opt chunk allocated    = 0

-----RX STATS-----
receive rate/sec         = 1
paks received at stp isr   = 3947627
paks queued at stp isr    = 3947627
paks dropped at stp isr = 0
drop rate/sec           = 0
paks dequeued at stp proc  = 3947627
paks waiting in queue      = 0
queue depth                = 7(max) 12288(total)

-----PROCESSING STATS-----
queue wait time (in ms)    = 0(avg) 540(max)
processing time (in ms)    = 0(avg) 4(max)
proc switch count          = 100
add vlan ports             = 20
time since last clearing   = 2087269 sec

```

The command used in this example displays STP process statistics. It is important to verify that the drop counters are not increasing and that received packets are increasing.

If received packets are not increasing but the physical port is receiving multicasts, verify that the packets are being received by the switch in–band interface (the interface of the CPU).

Issue the **remote command switch show ibc | i rx_input** command on the Catalyst 6500/6000:

```
cat# remote command switch show ibc | i rx_input
rx_inputs=5626468, rx_cumbytes=859971138

cat# remote command switch show ibc | i rx_input
rx_inputs=5626471, rx_cumbytes=859971539
```

This example shows that, between the outputs, the in-band port has received 23 packets.

Note: These 23 packets are not only BPDUs; this is a global counter for all packets received by the in-band port.

If there is no indication that BPDUs are being dropped on the local switch or port, you must move to the switch on the other side of the link and verify whether that switch is sending BPDUs.

d. Are BPDUs sent regularly on non-root, designated ports?

If, according to the port role, the port is sending BPDUs but the neighbor is not receiving them check whether BPDUs are actually being sent. Issue the **show spanning-tree interface interface detail** command:

```
cat# show spanning-tree interface g3/1 detail

Port 129 (GigabitEthernet3/1) of MST00 is designated forwarding
  Port path cost 20000, Port priority 128, Port Identifier 128.129.
  Designated root has priority 0, address 0007.4f1c.e847
  Designated bridge has priority 32768, address 00d0.003f.8800
  Designated port id is 128.129, designated path cost 2000019
  Timers: message age 0, forward delay 0, hold 0
  Number of transitions to forwarding state: 0
  Link type is point-to-point by default, Internal
  Loop guard is enabled by default on the port
  BPDUs: sent 1774, received 1
```

```
cat# show spanning-tree interface g3/1 detail

Port 129 (GigabitEthernet3/1) of MST00 is designated forwarding
  Port path cost 20000, Port priority 128, Port Identifier 128.129.
  Designated root has priority 0, address 0007.4f1c.e847
  Designated bridge has priority 32768, address 00d0.003f.8800
  Designated port id is 128.129, designated path cost 2000019
  Timers: message age 0, forward delay 0, hold 0
  Number of transitions to forwarding state: 0
  Link type is point-to-point by default, Internal
  Loop guard is enabled by default on the port
  BPDUs: sent 1776, received 1
```

In this example, two BPDUs have been sent out between the outputs.

Note: The STP process maintains the **BPDUs: sent** counter. This means that the counter indicates that the BPDUs have been sent toward the physical port, to be eventually sent out. Check whether the port counters are increasing for transmitted multicast packets. Issue the **show interface interface counters** command. This can help determine if BPDUs are going out or not:

```
cat# show interface g3/1 counters
```

Port	InOctets	InUcastPkts	InMcastPkts	InBcastPkts
Gi3/1	127985312	83776	812319	19

Port	OutOctets	OutUcastPkts	OutMcastPkts	OutBcastPkts
Gi3/1	131825915	3442	872342	386

cat# **show interface g3/1 counters**

Port	InOctets	InUcastPkts	InMcastPkts	InBcastPkts
Gi3/1	127985312	83776	812319	19

Port	OutOctets	OutUcastPkts	OutMcastPkts	OutBcastPkts
Gi3/1	131826447	3442	872346	386

With all of these steps, the idea is to find the switch or link where BPDUs are not received, sent, or processed.

It is possible, however unlikely, that the STP has calculated the correct state for the port, but due to a control plane issue, it was unable to set this state on the forwarding hardware. A loop can be created, if the supposed blocking port is not blocked at the hardware level. If you suspect such an issue in your network, contact Cisco Technical Support for further assistance.

5. Restore the redundancy.

Once the device or link that is causing the loop has been found, this device must be isolated from the network, or actions must be taken to resolve the issue (such as replace the fibre or GBIC). The redundant links, disconnected in Step 3, must be restored.

It is important to do as little manipulation as possible to the device or link that is causing the loop, because many conditions that lead to a loop can be very transient, intermittent, and unstable. This means that, if the condition is cleared during or after troubleshooting, it may take a while before such a condition occurs again. It is possible that the condition may not occur again at all. Every effort should be made to preserve the condition, so that it can be further investigated by Cisco Technical Support. It is important that you collect information about the condition before you reset the switches. If a condition is gone, it is often impossible to determine the root cause of the loop. To find the device or link that triggers the loop is a major achievement, but you need to ensure that another failure of the same kind does not cause the loop again. For more information, refer to the Securing the Network Against Forwarding Loops section of this document.

Troubleshooting Excessive Topology Changes Causing Flooding

The role of the TC mechanism is to correct L2 forwarding tables after the forwarding topology has changed. This is necessary to avoid a connectivity outage because, after a TC, some MAC addresses previously accessible through particular ports might become accessible through different ports. TC shortens the forwarding table aging time on all switches in the VLAN where the TC occurs; thus, if the address is not relearned, it will age-out and flooding will occur to ensure packets reach the destination MAC address.

TC is triggered by the change of a port's STP state to or from the STP forwarding state. After TC, even if the particular destination MAC address has aged-out, flooding should not continue for long. The address will be relearned by the first packet that comes from the host whose MAC address has been aged-out. The issue might arise when TCs are occurring repeatedly, with short intervals. The switches will constantly be fast-aging their forwarding tables, so flooding will be nearly constant.

Note: With Rapid STP or Multiple STP (IEEE 802.1w and IEEE 802.1s), TC is triggered by a change of the

ports state to forwarding, as well as the role change from designated to root. With Rapid STP, the L2 forwarding table is immediately flushed, as opposed to 802.1d, which shortens the aging time. The immediate flushing of the forwarding table restores connectivity faster, but will cause more flooding.

TC should be a rare event in a well-configured network. When a link on a switch port goes up or down, there is eventually a TC, once the STP state of the port is changing to or from forwarding. When the port is flapping, this would cause repetitive TCs and flooding.

Ports with the STP portfast feature enabled will not cause TCs when going to or from the forwarding state. The configuration of portfast on all end-device ports (such as printers, PCs, and servers) should limit TCs to a low amount and is highly recommended. For more information on TCs, refer to Understanding Spanning-Tree Protocol Topology Changes.

If there are repetitive TCs on the network, you must identify the source of these TCs and take action to reduce them, to bring the flooding to a minimum.

With 802.1d, STP information about a TC event is propagated among the bridges through a TC Notification (TCN), which is a special type of BPDU. If you follow the ports that are receiving TCN BPDUs, you can find the device that is originating TCs.

Establish if Flooding is Caused by STP TCs

Normally, you can determine that there is flooding from slow performance, packet drops on links that are not supposed to be congested, and the packet analyzer showing multiple unicast packets to the same destination that is not on the local segment.

For more information on unicast flooding, refer to Unicast Flooding in Switched Campus Networks.

On a Catalyst 6500/6000 that runs Cisco IOS software, you can check the forwarding engine counter (only on the Supervisor 2 engine) to estimate the amount of flooding. Issue the **remote command switch show earl statistics | i MISS_DA|ST_FR** command:

```
cat# remote command switch show earl statistics | i MISS_DA|ST_FR
      ST_MISS_DA      =      18      530308834
      ST_FRMS         =      97      969084354

cat# remote command switch show earl statistics | i MISS_DA|ST_FR
      ST_MISS_DA      =       4      530308838
      ST_FRMS         =     23      969084377
```

In this example, the first column shows the change since the last time this command was executed, and the second column shows the cumulative value since the last reboot. The first line shows the amount of flooded frames, and the second line shows the amount of processed frames. If the two values are close together, or the first value is increasing at a high rate, it might be that the switch is flooding traffic. However, this can only be used in conjunction with other ways to verify flooding, as the counters are not granular. There is one counter per switch, not per port or VLAN. It is normal to see some flooding packets, as the switch will always flood if the destination MAC address is not in the forwarding table. This will be the case when the switch receives a packet with a destination address that has not yet been learned.

Track Down the Source of the TCs

If the VLAN number is known for the VLAN where excessive flooding is occurring, check the STP counters to see if the number of TCs is high or incrementing regularly. Issue the **show spanning-tree vlan vlan-id detail** command (in this example, VLAN 1 is used):

```
cat# show spanning-tree vlan 1 detail
```

```
VLAN0001 is executing the ieee compatible Spanning Tree protocol
Bridge Identifier has priority 32768, sysid 1, address 0007.0e8f.04c0
Configured hello time 2, max age 20, forward delay 15
Current root has priority 0, address 0007.4f1c.e847
Root port is 65 (GigabitEthernet2/1), cost of root path is 119
Topology change flag not set, detected flag not set
Number of topology changes 1 last change occurred 00:00:35 ago
from GigabitEthernet1/1
Times: hold 1, topology change 35, notification 2
hello 2, max age 20, forward delay 15
Timers: hello 0, topology change 0, notification 0, aging 300
```

If the VLAN number is not known, you can use the packet analyzer or check the TC counters for all VLANs.

Take Steps to Prevent Excessive TCs

You can monitor the number of topology changes counter to see if it is increasing regularly. Then, move to the bridge that is connected to the port that is shown, to receive the last TC (in the previous example, port GigabitEthernet1/1) and see from where the TC came for that bridge. This process must be repeated until the end-station port without STP portfast enabled is found, or until the flapping link is found that needs to be fixed. The entire procedure needs to be repeated if TCs are still coming from other sources. If the link belongs to an end-host, you should configure the portfast feature to prevent the generation of TCs.

Note: In the Cisco IOS software STP implementation, the counter for TCs will only increment if a TCN BPDU is received by a port in a VLAN. If a normal configuration BPDU with a set TC flag is received then the TC counter is not incremented. This means that, if you suspect a TC to be the reason for flooding, it is best to start tracking down the sources for the TC from the STP root bridge in that VLAN. It will have the most accurate information regarding the amount and source of the TCs.

Troubleshooting Convergence Time Related Issues

There are situations when the actual operation of STP does not match the expected behavior. These are the two most frequent issues:

- STP convergence or reconvergence takes longer than expected.
- The resulting topology is different than expected.

Most often, these are the reasons for this behavior:

- A mismatch between the real and documented topology
- Misconfiguration, such as an inconsistent configuration of STP timers, exceeding STP diameter, or portfast misconfiguration
- Overloaded switch CPU during convergence or reconvergence
- Software defect

As mentioned earlier, this document can only provide general guidelines for troubleshooting, due to the wide variety of issues that could affect STP.

To understand why the convergence takes longer than expected, look at the sequence of STP events to find out what was happening and in what order. Because the STP implementation in Cisco IOS software does not have special logging (except for specific events, such as port inconsistencies), you can use Cisco IOS software STP debugging capabilities to understand what is happening.

For STP, with a Catalyst 6500/6000 that runs Cisco IOS software, processing is done on the Switch Processor (SP) (or Supervisor), so the debugs need to be enabled on the SP. For Cisco IOS software bridge groups, processing is done on the Route Processor (RP), so the debugs need to be enabled on the RP (MSFC).

STP Debugging Commands

Many STP **debug** commands are intended for development engineering use. They do not provide any output that is meaningful to someone without detailed knowledge of the STP implementation in Cisco IOS software. Some debugs can provide output which is instantly readable, such as port state changes, role changes, events such as TCs, and a dump of received and transmitted BPDUs. This section does not provide a complete description of all of the debugs, but rather briefly introduces the most frequently used ones.

Note: When you use **debug** commands, enable the minimum necessary debugs. If real-time debugs are not needed, record the output to the log rather than print it to the console. Excessive debugs can overload the CPU and disrupt switch operation. To direct debug output to the log instead of to the console or to Telnet sessions, issue the **logging console informational** and **no logging monitor** commands in global configuration mode.

To see the general events log, issue the **debug spanning-tree event** command for Per VLAN Spanning-Tree (PVST) and Rapid-PVST. This is the first debug that gives a general idea of what is happening with STP.

In Multiple Spanning-Tree (MST) mode, it does not work to issue the **debug spanning-tree event** command. Therefore, issue the **debug spanning-tree mstp roles** command to see the port role changes.

To see the port STP state changes, issue the **debug spanning-tree switch state** command together with the **debug pm vp** command:

```
cat-sp# debug spanning-tree switch state

Spanning Tree Port state changes debugging is on

cat-sp# debug pm vp

Virtual port events debugging is on
Nov 19 14:03:37: SP:      pm_vp 3/1(333): during state forwarding, got event 4(remove)
Nov 19 14:03:37: SP: @@@ pm_vp 3/1(333):
    forwarding -> notforwarding

Port 3/1 (was forwarding) goes down in vlan 333

Nov 19 14:03:37: SP: *** vp_fwdchange: single: notfwd: 3/1(333)
Nov 19 14:03:37: SP: @@@ pm_vp 3/1(333): notforwarding -> present
Nov 19 14:03:37: SP: *** vp_linkchange: single: down: 3/1(333)
Nov 19 14:03:37: SP: @@@ pm_vp 3/1(333): present -> not_present
Nov 19 14:03:37: SP: *** vp_statechange: single: remove: 3/1(333)
Nov 19 14:03:37: SP:      pm_vp 3/2(333): during state notforwarding,
    got event 4(remove)
Nov 19 14:03:37: SP: @@@ pm_vp 3/2(333): notforwarding -> present
Nov 19 14:03:37: SP: *** vp_linkchange: single: down: 3/2(333)

Port 3/2 (was not forwarding) in vlan 333 goes down

Nov 19 14:03:37: SP: @@@ pm_vp 3/2(333): present -> not_present
Nov 19 14:03:37: SP: *** vp_statechange: single: remove: 3/2(333)

Nov 19 14:03:53: SP:      pm_vp 3/1(333): during state not_present,
    got event 0(add)
Nov 19 14:03:53: SP: @@@ pm_vp 3/1(333): not_present -> present
Nov 19 14:03:53: SP: *** vp_statechange: single: added: 3/1(333)
Nov 19 14:03:53: SP:      pm_vp 3/1(333): during state present,
    got event 8(linkup)
```

```

Nov 19 14:03:53: SP: @@@ pm_vp 3/1(333): present ->
notforwarding
Nov 19 14:03:53: SP: STP SW: Gi3/1 new blocking req for 0 vlans
Nov 19 14:03:53: SP: *** vp_linkchange: single: up: 3/1(333)

Port 3/1 link goes up and blocking in vlan 333

Nov 19 14:03:53: SP: pm_vp 3/2(333): during state not_present,
got event 0(add)
Nov 19 14:03:53: SP: @@@ pm_vp 3/2(333): not_present -> present
Nov 19 14:03:53: SP: *** vp_statechange: single: added: 3/2(333)
Nov 19 14:03:53: SP: pm_vp 3/2(333): during state present,
got event 8(linkup)
Nov 19 14:03:53: SP: @@@ pm_vp 3/2(333): present ->
notforwarding
Nov 19 14:03:53: SP: STP SW: Gi3/2 new blocking req for 0 vlans
Nov 19 14:03:53: SP: *** vp_linkchange: single: up: 3/2(333)

Port 3/2 goes up and blocking in vlan 333

Nov 19 14:04:08: SP: STP SW: Gi3/1 new learning req for 1 vlans
Nov 19 14:04:23: SP: STP SW: Gi3/1 new forwarding req for 0 vlans
Nov 19 14:04:23: SP: STP SW: Gi3/1 new forwarding req for 1 vlans
Nov 19 14:04:23: SP: pm_vp 3/1(333): during state notforwarding,
got event 14(forward_notnotify)
Nov 19 14:04:23: SP: @@@ pm_vp 3/1(333): notforwarding ->
forwarding
Nov 19 14:04:23: SP: *** vp_list_fwdchange: forward: 3/1(333)

Port 3/1 goes via learning to forwarding in vlan 333

```

To understand why STP behaves in a certain way, it is often useful to see the BPDUs that are received and sent by the switch:

```

cat-sp# debug spanning-tree bpdu receive

Spanning Tree BPDU Received debugging is on
Nov 6 11:44:27: SP: STP: VLAN1 rx BPDU: config protocol = ieee,
packet from GigabitEthernet2/1 , linktype IEEE_SPANNING ,
enc type 2, encsize 17
Nov 6 11:44:27: SP: STP: enc 01 80 C2 00 00 00 00 06 52 5F 0E 50 00 26 42 03
Nov 6 11:44:27: SP: STP: Data 0000000000000000074F1CE8470000001380480006525F0E4
080100100140002000F00
Nov 6 11:44:27: SP: STP: VLAN1 Gi2/1:0000 00 00 00 000000074F1CE847 00000013
80480006525F0E40 8010 0100 1400 0200 0F00

```

This debug works for PVST, Rapid-PVST, and MST modes; but it does not decode the contents of the BPDUs. However, you can use it to ensure that BPDUs are received.

To see the contents of the BPDU, issue the **debug spanning-tree switch rx decode** command together with the **debug spanning-tree switch rx process** command for PVST and Rapid-PVST. Issue the **debug spanning-tree mstp bpdu-rx** command to see the contents of the BPDU for MST:

```

cat-sp# debug spanning-tree switch rx decode

Spanning Tree Switch Shim decode received packets debugging is on

cat-sp# debug spanning-tree switch rx process

Spanning Tree Switch Shim process receive bpdu debugging is on

Nov 6 12:23:20: SP: STP SW: PROC RX: 0180.c200.0000<-0006.525f.0e50 type/len 0026
Nov 6 12:23:20: SP: encap SAP linktype ieee-st vlan 1 len 52 on vl Gi2/1
Nov 6 12:23:20: SP: 42 42 03 SPAN

```

```

Nov 6 12:23:20: SP:      CFG P:0000 V:00 T:00 F:00 R:0000 0007.4f1c.e847 00000013
Nov 6 12:23:20: SP:      B:8048 0006.525f.0e40 80.10 A:0100 M:1400 H:0200 F:0F00

Nov 6 12:23:22: SP: STP SW: PROC RX: 0180.c200.0000<-0006.525f.0e50 type/len 0026
Nov 6 12:23:22: SP:      encap SAP linktype ieee-st vlan 1 len 52 on vl Gi2/1
Nov 6 12:23:22: SP:      42 42 03 SPAN
Nov 6 12:23:22: SP:      CFG P:0000 V:00 T:00 F:00 R:0000 0007.4f1c.e847 00000013
Nov 6 12:23:22: SP:      B:8048 0006.525f.0e40 80.10 A:0100 M:1400 H:0200 F:0F00

```

For the MST mode, you can enable detailed BPDU decode with this **debug** command:

```

cat-sp# debug spanning-tree mstp bpdu-rx

Multiple Spanning Tree Received BPDUs debugging is on
Nov 19 14:37:43: SP: MST:BPDU DUMP [rcvd_bpdu Gi3/2 Repeated]
Nov 19 14:37:43: SP: MST:   Proto:0 Version:3 Type:2 Role: DesgFlags[   F   ]
Nov 19 14:37:43: SP: MST:   Port_id:32897 cost:2000019
Nov 19 14:37:43: SP: MST:   root_id   :0007.4f1c.e847 Prio:0
Nov 19 14:37:43: SP: MST:   br_id    :00d0.003f.8800 Prio:32768
Nov 19 14:37:43: SP: MST:   age:2 max_age:20 hello:2 fwdelay:15
Nov 19 14:37:43: SP: MST:   V3_len:90 PathCost:30000 region:STATIC rev:1
Nov 19 14:37:43: SP: MST:   ist_m_id :0005.74
Nov 19 14:37:43: SP: MST:BPDU DUMP [rcvd_bpdu Gi3/2 Repeated]
Nov 19 14:37:43: SP: MST:   Proto:0 Version:3 Type:2 Role: DesgFlags[   F   ]
Nov 19 14:37:43: SP: MST:   Port_id:32897 cost:2000019
Nov 19 14:37:43: SP: MST:   root_id   :0007.4f1c.e847 Prio:0
Nov 19 14:37:43: SP: MST:   br_id    :00d0.003f.8800 Prio:32768
Nov 19 14:37:43: SP: MST:   age:2 max_age:20 hello:2 fwdelay:15
Nov 19 14:37:43: SP: MST:   V3_len:90 PathCost:30000 region:STATIC rev:1
Nov 19 14:37:43: SP: MST:   ist_m_id :0005.7428.1440 Prio:32768 Hops:18
    Num Mrec: 1
Nov 19 14:37:43: SP: MST: stci=3  Flags[   F   ] Hop:19 Role:Desg [Repeated]
Nov 19 14:37:43: SP: MST:          br_id:00d0.003f.8800 Prio:32771 Port_id:32897
    Cost:2000028.1440 Prio:32768 Hops:18 Num Mrec: 1
Nov 19 14:37:43: SP: MST: stci=3  Flags[   F   ] Hop:19 Role:Desg [Repeated]
Nov 19 14:37:43: SP: MST:          br_id:00d0.003f.8800 Prio:32771 Port_id:32897
    Cost:20000

```

Note: For Cisco IOS Software Release 12.1.13E and later, conditional debugs for STP are supported. This means that you can debug BPDUs that are received or transmitted on a per-port or per-VLAN basis.

Issue the **debug condition vlan** *vlan_num* or **debug condition interface** *interface* commands, to limit the scope of the debug output to per-interface or per-VLAN.

Securing the Network Against Forwarding Loops

To handle STP's inability to deal correctly with certain failures, Cisco has developed a number of features and enhancements to protect the networks against forwarding loops.

Troubleshooting STP helps to isolate and possibly find the cause for a particular failure, while the implementation of these enhancements is the only way to secure the network against forwarding loops.

These are methods to protect your network against forwarding loops:

1. Enable Unidirectional Link Detection (UDLD) on all switch-to-switch links. For more information on UDLD, refer to Understanding and Configuring the Unidirectional Link Detection Protocol Feature.
2. Enable Loop Guard on all switches. For more information on Loop Guard, refer to Spanning-Tree Protocol Enhancements using Loop Guard and BPDU Skew Detection Features.

When enabled, UDLD and Loop Guard eliminate the majority of the possible causes for forwarding loops. Rather than create a forwarding loop, the offending link (or all links dependent on the failing hardware) is shut down or blocked.

Note: While these two features appear somewhat redundant, each has its unique capabilities. Therefore, use both features at the same time to provide the highest level of protection. For a detailed comparison of UDLD and Loop Guard, refer to Loop Guard vs. Unidirectional Link Detection.

There are different opinions about whether you have to use aggressive or normal UDLD. It should be noted that aggressive UDLD will not provide more protection against loops compared to normal mode UDLD. Aggressive UDLD detects port–stuck scenarios (when the link is up, but there are no associated traffic blackholes). The downside of this added functionality is that aggressive UDLD can potentially disable links when no consistent failure is present. Often people confuse the modification of the UDLD `hello` interval with the aggressive UDLD feature. This is incorrect. Timers can be modified in both UDLD modes.

Note: In rare cases, aggressive UDLD can shut down all of the uplink ports, which essentially isolates the switch from the rest of the network. For example, this could happen when both upstream switches are experiencing very high CPU utilization and aggressive mode UDLD is used. Therefore, it is recommended that you configure `errordisable-timeouts`, if the switch does not have out–of–band management in place.

3. Enable portfast on all end–station ports.

You must enable portfast to limit the amount of TCs and subsequent flooding, which can affect the performance of the network. Only use this command with ports that connect to end stations. Otherwise, an accidental topology loop can cause a data–packet loop and disrupt the switch and network operation.



Caution: Exercise caution when you use the **no spanning–tree portfast** command. This

command only removes any port specific portfast commands. This command implicitly enables portfast if you define the **spanning–tree portfast default** command in global configuration mode and if the port is not a trunk port. If you do not configure portfast globally, the **no spanning–tree portfast** command is equivalent to the **spanning–tree portfast disable** command.

4. Set EtherChannels to `desirable` mode on both sides (where supported) and `non-silent` option.

`Desirable` mode will enable Port Aggregation Protocol (PAgP) to ensure runtime consistency between the channelling peers. This gives an additional degree of protection against loops, especially during channel reconfigurations (such as when links join or leave the channel, and link failure detection). There is a built–in Channel Misconfiguration Guard, which is enabled by default and which prevents forwarding loops due to channel misconfiguration or other conditions. For more information on this feature, refer to Understanding EtherChannel Inconsistency Detection.

5. Do not disable auto–negotiation (if supported) on switch–to–switch links.

Auto–negotiation mechanisms can convey remote fault information, which is the quickest way to detect failure at the remote side. Should failure be detected at the remote side, the local side brings down the link even if the link is still receiving pulses. Compared to high–level detection mechanisms such as UDLD, auto–negotiation is very fast (within microseconds) but lacks the end–to–end coverage of UDLD (such as the entire datapath: CPU forwarding logic port1 port2 forwarding logic CPU versus port1 port2). Aggressive UDLD mode provides similar functionality to that of auto–negotiation with regards to failure detection. When negotiation is supported on both sides of the link, there is no need to enable aggressive mode UDLD.

6. Use caution when you tune the STP timers.

STP timers are dependent on each other and on the network topology. STP may not work correctly with arbitrary modifications made to the timers. For more information about STP timers, refer to [Understanding and Tuning Spanning Tree Protocol Timers](#).

7. If denial of service attacks are possible, secure the network STP perimeter with Root Guard.

Root Guard and BPDU Guard allow you to secure STP against influence from the outside. If such an attack is a possibility, Root Guard and BPDU Guard must be used to protect the network. For more information on Root Guard and BPDU Guard, refer to these documents:

- ◆ [Spanning-Tree Protocol Root Guard Enhancement](#)
- ◆ [Spanning Tree Portfast BPDU Guard Enhancement](#)

8. Enable BPDU Guard on portfast-enabled ports, to prevent STP from being affected by unauthorized network devices (such as hubs, switches, and bridging routers) that are connected to the ports.

If Root Guard is properly configured, it will already prevent the STP from being influenced from the outside. If BPDU Guard is enabled, it will shut down the ports that are receiving any BPDUs (not only superior BPDUs). This can be useful if such incidents need to be investigated, because BPDU Guard produces the syslog message and shuts down the port. It should be noted that short-living loops are not prevented by Root or BPDU Guards, if two portfast-enabled ports are connected directly or through the hub.

9. Avoid user traffic on the management VLAN. The management VLAN is contained to a building block, not the entire network.

The switch management interface receives broadcast packets on the management VLAN. Should excessive broadcasts occur (such as a broadcast storm or a malfunctioning application), the switch CPU might become overloaded, which could possibly distort STP operation.

10. A predictable (hardcoded) STP root and backup STP root placement.

The STP root and backup STP root must be configured so that convergence, in the case of failures, occurs in a predictable way and builds optimal topology in every scenario. Do not leave the STP priority at the default value, to prevent unpredictable root switch selection.

Related Information

- [LAN Product Support](#)
- [LAN Switching Technology Support](#)
- [Technical Support & Documentation – Cisco Systems](#)

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Sep 01, 2005

Document ID: 28943
