Understand Cyclic Redundancy Check Errors on Nexus Switches

Contents

Introduction

This document describes Cyclic Redundancy Check (CRC) errors observed on interface counters and statistics of Cisco Nexus switches.

Prerequisites

Requirements

Cisco recommends that you understand the basics of Ethernet switching and the Cisco NX-OS Command Line Interface (CLI). For more information, refer to one of these applicable documents:

- Cisco Nexus 9000 NX-OS Fundamentals Configuration Guide, Release 10.2(x)
- Cisco Nexus 9000 Series NX-OS Fundamentals Configuration Guide, Release 9.3(x)
- Cisco Nexus 9000 Series NX-OS Fundamentals Configuration Guide, Release 9.2(x)
- Cisco Nexus 9000 Series NX-OS Fundamentals Configuration Guide, Release 7.x

- Troubleshooting Ethernet

Components Used

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

- Nexus 9000 series switches that starts from NX-OS software release 9.3(8)
- Nexus 3000 series switches that starts from NX-OS software release 9.3(8)

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, ensure that you understand the potential impact of any command.

Background Information

This document describes details about Cyclic Redundancy Check (CRC) errors observed on interface counters on Cisco Nexus series switches. This document describes what a CRC is, how it is used in the Frame Check Sequence (FCS) field of Ethernet frames, how CRC errors manifest on Nexus switches, and how CRC errors interact in Store-and-Forward switching. This article also describes Cut-Through switching scenarios, the most likely root causes of CRC errors, and how to troubleshoot and resolve CRC errors.

Applicable Hardware

The information in this document is applicable to all Cisco Nexus Series switches. Some of the information in this document can also be applicable to other Cisco routing and switching platforms, such as Cisco Catalyst routers and switches.

CRC Definition

A CRC is an error detection mechanism commonly used in computer and storage networks to identify data changed or corrupted during transmission. When a device connected to the network needs to transmit data, the device runs a computation algorithm based on cyclic codes against the data that results in a fixed-length number. This fixed-length number is called the CRC value, but colloquially, it is often called the CRC for short. This CRC value is appended to the data and transmitted through the network towards another device. This remote device runs the same cyclic code algorithm against the data and compares the value that results with the CRC appended to the data. If both values match, then the remote device assumes the data was transmitted across the network without corruption. If the values do not match, then the remote device assumes the data was corrupted in the transmission across the network. This corrupted data cannot be trusted and is discarded.

CRCs are used for error detection across multiple computer networking technologies, such as Ethernet (both wired and wireless variants), Token Ring, Asynchronous Transfer Mode (ATM), and Frame Relay. Ethernet frames have a 32-bit Frame Check Sequence (FCS) field at the end of the frame (immediately after the payload of the frame) where a 32-bit CRC value is inserted.

For example, consider a scenario where two hosts named Host-A and Host-B are directly connected to each other through their Network Interface Cards (NICs). Host-A needs to send the sentence “This is an example” to Host-B over the network. Host-A crafts an Ethernet frame destined to Host-B with a payload of “This is an example” and calculates that the CRC value of the frame is a hexadecimal value of 0xABCD. Host-A inserts the CRC value of 0xABCD into the FCS field of the Ethernet frame, then transmits the Ethernet frame out of Host-A NIC towards Host-B.

When Host-B receives this frame, it can calculate the CRC value of the frame with the use of the exact same algorithm as Host-A. Host-B calculates that the CRC value of the frame is a hexadecimal value of 0xABCD, which indicates to Host-B that the Ethernet frame was not corrupted while the frame was transmitted to Host-B.

CRC Error Definition

A CRC error occurs when a device (either a network device or a host connected to the network) receives an Ethernet frame with a CRC value in the FCS field of the frame that does not match the CRC value calculated by the device for the frame.

This concept is best demonstrated through an example. Consider a scenario where two hosts named Host-A and Host-B are directly connected to each other through their Network Interface Cards (NICs). Host-A needs to send the sentence “This is an example” to Host-B over the network. Host-A crafts an Ethernet frame destined to Host-B with a payload of “This is an example” and calculates that the CRC value of the frame is the hexadecimal value 0xABCD. Host-A inserts the CRC value of 0xABCD into the FCS field of the Ethernet frame, then transmits the Ethernet frame out of Host-A NIC towards Host-B.

However, damage on the physical media connecting Host-A to Host-B corrupts the contents of the frame such that the sentence within the frame changes to “This was an example” instead of the desired payload of “This is an example”.

When Host-B receives this frame, it can calculate the CRC value of the frame and include the corrupted payload in the calculation. Host-B calculates that the CRC value of the frame is a hexadecimal value of 0xDEAD, which is different from the 0xABCD CRC value within the FCS field of the Ethernet frame. This difference in CRC values tells Host-B that the Ethernet frame was corrupted while the frame was transmitted to Host-B. As a result, Host-B cannot trust the contents of this Ethernet frame, so it can drop it. Host-B can usually increment some sort of error counter on its Network Interface Card (NIC) as well, such as the “input errors”, “CRC errors”, or “RX errors” counters.

Common Symptoms of CRC Errors

CRC errors typically manifest themselves in one of two ways:

1. Incrementing or non-zero error counters on interfaces of network-connected devices.
2. Packet/Frame loss for traffic that traverses the network due to network-connected devices that drop corrupted frames.

These errors manifest themselves in slightly different ways contingent on the device you work with at the time. These sub-sections go into detail for each type of device.

Received Errors on Windows Hosts

CRC errors on Windows hosts typically manifest as a non-zero **Received Errors** counter displayed in the output of the **netstat -e** command from the Command Prompt. An example of a non-zero Received Errors counter from the Command Prompt of a Windows host is here:

<#root>

>

netstat -e

Interface Statistics

Received

```
           Sent
Bytes                    1116139893      3374201234
Unicast packets           101276400        49751195
Non-unicast packets               0               0
Discards                          0               0

Errors                        47294

               0
Unknown protocols                 0
```

The NIC and its respective driver must support accounting of CRC errors received by the NIC in order for the number of Received Errors reported by the **netstat -e** command to be accurate. Most modern NICs and their respective drivers support accurate accounting of CRC errors received by the NIC.

RX Errors on Linux Hosts

CRC errors on Linux hosts typically manifest as a non-zero “RX errors” counter displayed in the output of the **ifconfig** command. An example of a non-zero RX errors counter from a Linux host is here:

<#root>

$

ifconfig eth0

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.0.2.10  netmask 255.255.255.128  broadcast 192.0.2.255
        inet6 fe80::10  prefixlen 64  scopeid 0x20<link>
        ether 08:62:66:be:48:9b  txqueuelen 1000  (Ethernet)
        RX packets 591511682  bytes 214790684016 (200.0 GiB)

RX errors 478920

  dropped 0  overruns 0  frame 0
        TX packets 85495109  bytes 288004112030 (268.2 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

CRC errors on Linux hosts can also manifest as a non-zero “RX errors” counter displayed in the output of **ip -s link show** command. An example of a non-zero RX errors counter from a Linux host is here:

<#root>

$

ip -s link show eth0

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1
    link/ether 08:62:66:84:8f:6d brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    32246366102 444908978 478920       647     0       419445867
    TX: bytes  packets  errors  dropped carrier collsns
    3352693923 30185715 0       0       0       0
    altname enp11s0
```

The NIC and its respective driver must support accounting of CRC errors received by the NIC in order for the number of RX Errors reported by the **ifconfig** or **ip -s link show** commands to be accurate. Most modern NICs and their respective drivers support accurate accounting of CRC errors received by the NIC.

CRC Errors on Network Devices

Network devices operate in one of two forwarding modes:

- Store-and-Forward forwarding mode
- Cut-Through forwarding mode

The way a network device handles a received CRC error differs contingent on its forwarding modes. The subsections here describe the specific behavior for each forwarding mode.

Input Errors on Store-and-Forward Network Devices

When a network device operating in a Store-and-Forward forwarding mode receives a frame, the network device can buffer the entire frame (“Store”) before you validate the CRC value of the frame, make a forwarding decision on the frame, and transmit the frame out of an interface (“Forward”). Therefore, when a network device operating in a Store-and-Forward forwarding mode receives a corrupted frame with an incorrect CRC value on a specific interface, it can drop the frame and increment the “Input Errors” counter on the interface.

In other words, corrupt Ethernet frames are not forwarded by network devices operating in a Store-and-Forward forwarding mode; they are dropped on ingress.

Cisco Nexus 7000 and 7700 Series switches operate in a Store-and-Forward forwarding mode. An example of a non-zero Input Errors counter and a non-zero CRC/FCS counter from a Nexus 7000 or 7700 Series switch is here:

```
<#root>

switch#

show interface


<snip>
Ethernet1/1 is up
  RX
    241052345 unicast packets  5236252 multicast packets  5 broadcast packets
    245794858 input packets  17901276787 bytes
    0 jumbo packets  0 storm suppression packets
    0 runts  0 giants  579204 CRC/FCS  0 no buffer
    579204 input error  0 short frame  0 overrun   0 underrun  0 ignored
    0 watchdog  0 bad etype drop  0 bad proto drop  0 if down drop
    0 input with dribble  0 input discard
    0 Rx pause
```

CRC errors can also manifest themselves as a non-zero “FCS-Err” counter in the output of **show interface counters** errors. The "Rcv-Err" counter in the output of this command can also have a non-zero value, which is the sum of all input errors (CRC or otherwise) received by the interface. An example of this is shown here:

<#root>

```
switch#
```

show interface counters errors

```
<snip>
--------------------------------------------------------------------------------
Port          Align-Err    FCS-Err   Xmit-Err    Rcv-Err  UnderSize OutDiscards
--------------------------------------------------------------------------------
Eth1/1                0     579204          0     579204          0            0
```

Input and Output Errors on Cut-Through Network Devices

When a network device operating in a Cut-Through forwarding mode starts to receive a frame, the network device can make a forwarding decision on the frame header and begin to transmit the frame out of an interface as soon as it receives enough of the frame to make a valid forwarding decision. As frame and packet headers are at the beginning of the frame, this forwarding decision is usually made before the payload of the frame is received.

The FCS field of an Ethernet frame is at the end of the frame, immediately after the frame's payload. Therefore, a network device operating in a Cut-Through forwarding mode can already have started to transmit the frame out of another interface by the time it can calculate the CRC of the frame. If the CRC calculated by the network device for the frame does not match the CRC value present in the FCS field, that means the network device forwarded a corrupted frame into the network. When this happens, the network device can increment two counters:

1. The "Input Errors" counter on the interface where the corrupted frame was originally received.
2. The "Output Errors" counter on all interfaces where the corrupted frame was transmitted. For unicast traffic, this can typically be a single interface – however, for broadcast, multicast, or unknown unicast traffic, this could be one or more interfaces.

An example of this is shown here, where the output of the **show interface** command indicates multiple corrupted frames were received on Ethernet1/1 of the network device and transmitted out of Ethernet1/2 due to the Cut-Through forwarding mode of the network device:

<#root>

```
switch#
```

show interface

```
<snip>
Ethernet1/1 is up
  RX
    46739903 unicast packets  29596632 multicast packets  0 broadcast packets
    76336535 input packets  6743810714 bytes
    15 jumbo packets  0 storm suppression bytes
    0 runts  0 giants  47294 CRC  0 no buffer
    47294 input error  0 short frame  0 overrun   0 underrun  0 ignored
    0 watchdog  0 bad etype drop  0 bad proto drop  0 if down drop
    0 input with dribble  0 input discard
    0 Rx pause

Ethernet1/2 is up
```

```
  TX
    46091721 unicast packets  2852390 multicast packets  102619 broadcast packets
    49046730 output packets  3859955290 bytes
    50230 jumbo packets
    47294 output error  0 collision  0 deferred  0 late collision
    0 lost carrier  0 no carrier  0 babble  0 output discard
    0 Tx pause
```

CRC errors can also manifest themselves as a non-zero “FCS-Err” counter on the ingress interface and non-zero "Xmit-Err" counters on egress interfaces in the output of **show interface counters** errors. The "Rcv-Err" counter on the ingress interface in the output of this command can also have a non-zero value, which is the sum of all input errors (CRC or otherwise) received by the interface. An example of this is shown here:

<#root>

switch#

show interface counters errors

<snip>

```
--------------------------------------------------------------------------------
Port          Align-Err
```

FCS-Err

Xmit-Err

Rcv-Err

```
  UnderSize OutDiscards
--------------------------------------------------------------------------------
```

Eth1/1

0

47294

0

47294

0 0

Eth1/2

0 0

47294

0 0 0

The network device can also modify the CRC value in the frame’s FCS field in a specific manner that signifies to upstream network devices that this frame is corrupt. This behavior is known as “stomping” the CRC. The precise manner in which the CRC is modified varies from one platform to another, but generally,

it inverts the current CRC value present in the frame’s FCS field. Here is an example of this:

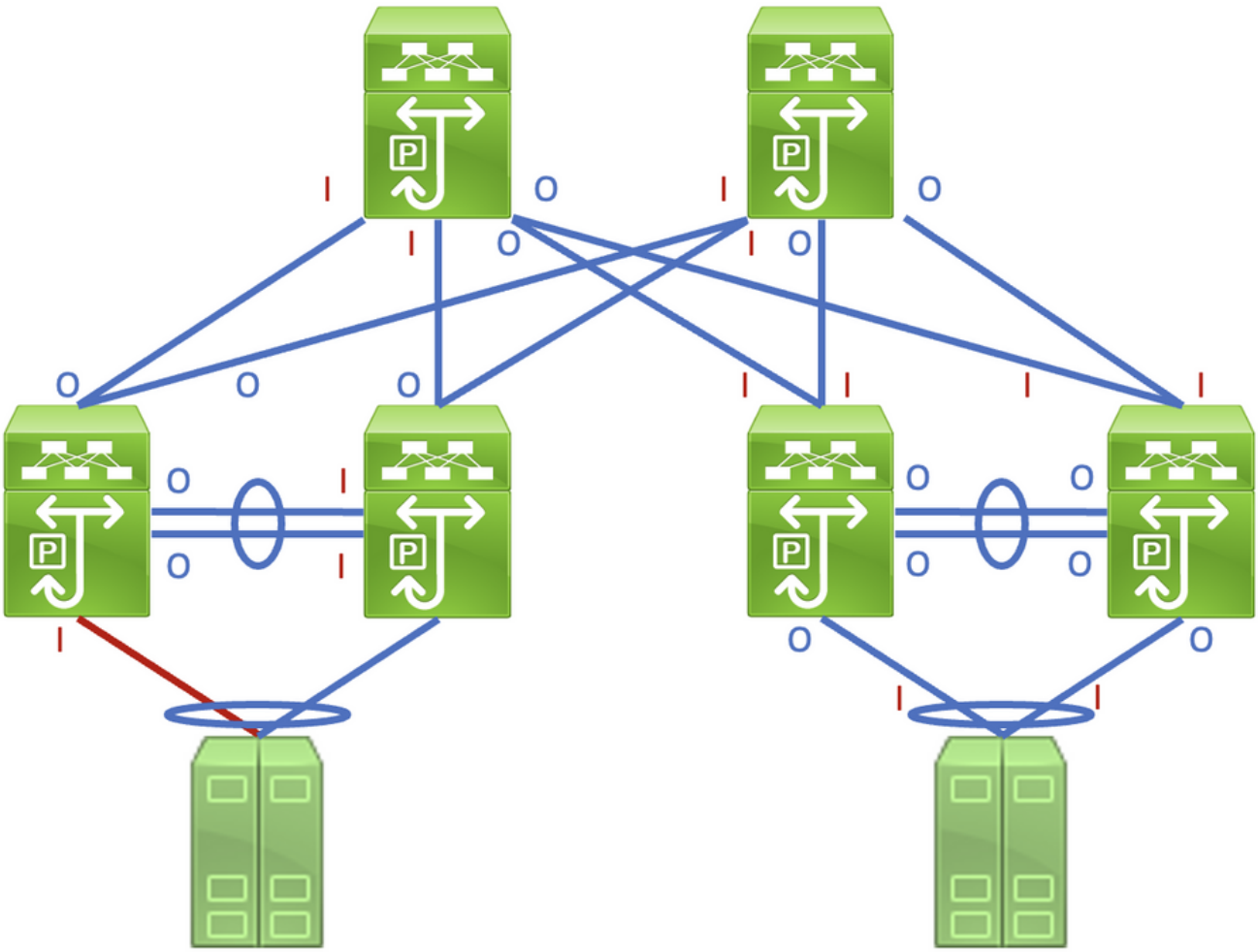
```
Original CRC: 0xABCD (1010101111001101)
Stomped CRC:  0x5432 (0101010000110010)
```

As a result of this behavior, network devices operating in a Cut-Through forwarding mode can propagate a corrupt frame throughout a network. If a network consists of multiple network devices operating in a Cut-Through forwarding mode, a single corrupt frame can cause input error and output error counters to increment on multiple network devices within your network.

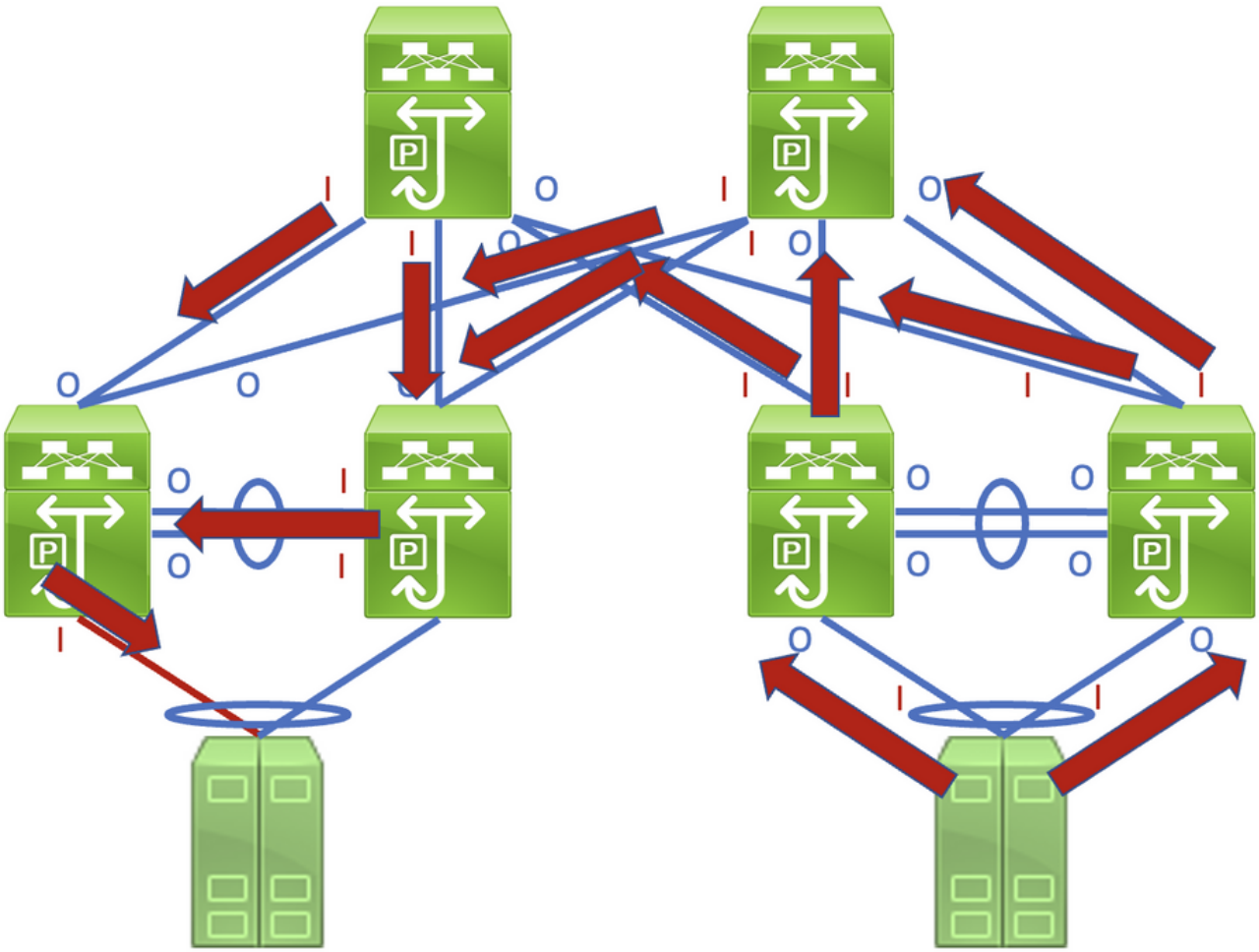
Trace and Isolate CRC Errors

The first step in order to identify and resolve the root cause of CRC errors is to isolate the source of the CRC errors to a specific link between two devices within your network. One device connected to this link can have an interface output errors counter with a value of zero or is not incrementing, while the other device connected to this link can have a non-zero or incrementing interface input errors counter. This suggests that traffic egresses the interface of one device intact is corrupted at the time of the transmission to the remote device and is counted as an input error by the ingress interface of the other device on the link.

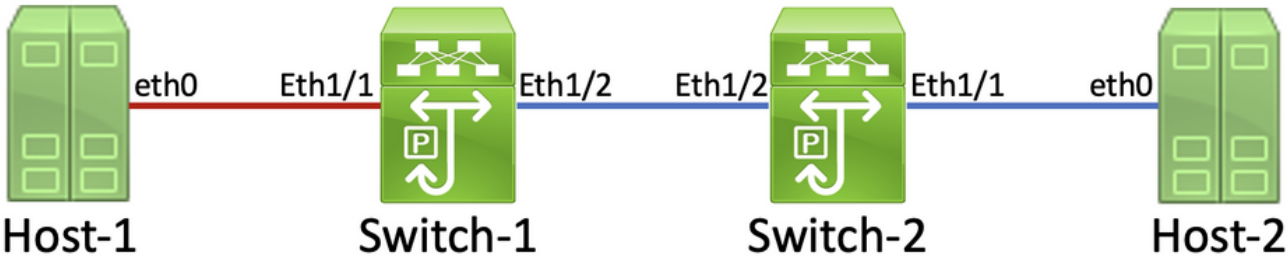
To identify this link in a network that consists of network devices operating in a Store-and-Forward forwarding mode is a straightforward task. However, if you identify this link in a network that consists of network devices operating in a Cut-Through forwarding mode is more difficult, as many network devices can have non-zero input and output error counters. An example of this phenomenon can be seen in the topology here, where the link highlighted in red is damaged such that traffic traverses the link is corrupted. Interfaces labeled with a red "I" indicate interfaces that could have non-zero input errors, while interfaces labeled with a blue "O" indicate interfaces that could have non-zero output errors.



This document describes Cyclic Redundancy Check (CRC) errors observed on interface counters and statistics of Cisco Nexus switches.



A detailed process to trace and identify a damaged link is best demonstrated through an example. Consider the topology here:



In this topology, interface Ethernet1/1 of a Nexus switch named Switch-1 is connected to a host named Host-1 through Host-1's Network Interface Card (NIC) eth0. Interface Ethernet1/2 of Switch-1 is connected to a second Nexus switch, named Switch-2, through Switch-2's interface Ethernet1/2. Interface Ethernet1/1 of Switch-2 is connected to a host named Host-2 through Host-2 NIC eth0.

The link between Host-1 and Switch-1 through Switch-1's Ethernet1/1 interface is damaged, and causes traffic that traverses the link to be intermittently corrupted. However, whether the link is damaged is unknown at this point. You must trace the path the corrupted frames leave in the network through non-zero or incremented input and output error counters to locate the damaged link in this network.

In this example, Host-2 NIC reports that it is receiving CRC errors.

```
Host-2$
```

ip -s link show eth0

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1
    link/ether 00:50:56:84:8f:6d brd ff:ff:ff:ff:ff:ff
```

RX:

```
 bytes  packets
```

errors

```
dropped overrun mcast
    32246366102 444908978
```

478920

```
       647     0       419445867
    TX: bytes  packets  errors  dropped carrier collsns
    3352693923 30185715 0       0       0       0
    altname enp11s0
```

You know that Host-2 NIC connects to Switch-2 via interface Ethernet1/1. You can confirm that interface Ethernet1/1 has a non-zero output errors counter with the **show interface** command.

<#root>

```
Switch-2#
```

show interface

```
<snip>
Ethernet1/1 is up
admin state is up, Dedicated Interface
    RX
      30184570 unicast packets  872 multicast packets  273 broadcast packets
      30185715 input packets  3352693923 bytes
      0 jumbo packets  0 storm suppression bytes
      0 runts  0 giants 0 CRC  0 no buffer
      0 input error  0 short frame  0 overrun   0 underrun  0 ignored
      0 watchdog  0 bad etype drop  0 bad proto drop  0 if down drop
      0 input with dribble  0 input discard
      0 Rx pause
    TX
      444907944 unicast packets  932 multicast packets  102 broadcast packets
      444908978 output packets  32246366102 bytes
      0 jumbo packets
      478920 output error  0 collision  0 deferred  0 late collision
      0 lost carrier  0 no carrier  0 babble  0 output discard
      0 Tx pause
```

Since the output errors counter of interface Ethernet1/1 is non-zero, there is most likely another interface of Switch-2 that has a non-zero input errors counter. You can use the **show interface counters errors non-zero** command in order to identify if any interfaces of Switch-2 have a non-zero input errors counter.

<#root>

```
Switch-2#
show interface counters errors non-zero
<snip>
--------------------------------------------------------------------------------
Port          Align-Err
FCS-Err


Xmit-Err


Rcv-Err

  UnderSize OutDiscards
--------------------------------------------------------------------------------
Eth1/1                 0          0
478920

          0          0           0
Eth1/2                 0
478920

          0
478920

          0           0

--------------------------------------------------------------------------------
Port         Single-Col  Multi-Col   Late-Col  Exces-Col  Carri-Sen       Runts
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Port          Giants SQETest-Err Deferred-Tx IntMacTx-Er IntMacRx-Er Symbol-Err
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Port         InDiscards
--------------------------------------------------------------------------------
```

You can see that Ethernet1/2 of Switch-2 has a non-zero input errors counter. This suggests that Switch-2 is receiving corrupted traffic on this interface. You can confirm which device is connected to Ethernet1/2 of Switch-2 through the Cisco Discovery Protocol (CDP) or Link Local Discovery Protocol (LLDP) features. An example of this is shown here with the **show cdp neighbors** command.

<#root>

```
Switch-2#
show cdp neighbors
<snip>
    Capability Codes: R - Router, T - Trans-Bridge, B - Source-Route-Bridge
    S - Switch, H - Host, I - IGMP, r - Repeater,
    V - VoIP-Phone, D - Remotely-Managed-Device,
    s - Supports-STP-Dispute
```

```
Device-ID          Local Intrfce  Hldtme Capability  Platform      Port ID
Switch-1
(FDO12345678)
Eth1/2
         125    R S I s   N9K-C9318OYC-
Eth1/2
```

You now know that Switch-2 is receiving corrupted traffic on its Ethernet1/2 interface from Switch-1's Ethernet1/2 interface, but you do not yet know whether the link between Switch-1's Ethernet1/2 and Switch-2's Ethernet1/2 is damaged and causes the corruption, or if Switch-1 is a cut-through switch forwarding corrupted traffic it is receiving. You must log into Switch-1 to verify this.

You can confirm Switch-1's Ethernet1/2 interface has a non-zero output errors counter with the **show interfaces** command.

```
<#root>
Switch-1#
show interface
<snip>
Ethernet1/2 is up
admin state is up, Dedicated Interface
    RX
      30581666 unicast packets  178 multicast packets  931 broadcast packets
      30582775 input packets  3352693923 bytes
      0 jumbo packets  0 storm suppression bytes
      0 runts  0 giants 0 CRC  0 no buffer
      0 input error  0 short frame  0 overrun   0 underrun  0 ignored
      0 watchdog  0 bad etype drop  0 bad proto drop  0 if down drop
      0 input with dribble  0 input discard
      0 Rx pause
TX
      454301132 unicast packets  734 multicast packets  72 broadcast packets
      454301938 output packets  32246366102 bytes
      0 jumbo packets
478920 output error
  0 collision  0 deferred  0 late collision
      0 lost carrier  0 no carrier  0 babble  0 output discard
      0 Tx pause
```

You can see that Ethernet1/2 of Switch-1 has a non-zero output errors counter. This suggests that the link between Switch-1's Ethernet1/2 and Switch-2's Ethernet1/2 is not damaged; instead, Switch-1 is a cut-

through switch forwarding corrupted traffic it receives on some other interface. As previously demonstrated with Switch-2, you can use the show interface counters errors non-zero command in order to identify if any interfaces of Switch-1 have a non-zero input errors counter.

```
<#root>

Switch-1#

show interface counters errors non-zero

<snip>
--------------------------------------------------------------------------------
Port          Align-Err

FCS-Err


Xmit-Err


Rcv-Err

  UnderSize OutDiscards
--------------------------------------------------------------------------------
Eth1/1                0

478920

          0

478920

          0           0
Eth1/2                0          0

478920

          0          0           0

--------------------------------------------------------------------------------
Port         Single-Col  Multi-Col   Late-Col  Exces-Col  Carri-Sen       Runts
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Port          Giants SQETest-Err Deferred-Tx IntMacTx-Er IntMacRx-Er Symbol-Err
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Port         InDiscards
--------------------------------------------------------------------------------
```

You can see that Ethernet1/1 of Switch-1 has a non-zero input errors counter. This suggests that Switch-1 is receiving corrupted traffic on this interface. You know that this interface connects to Host-1's eth0 NIC. You can review Host-1's eth0 NIC interface statistics to confirm whether Host-1 sends corrupted frames out of this interface.

```
<#root>

Host-1$
```

```
ip -s link show eth0


2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1
    link/ether 00:50:56:84:8f:6d brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    73146816142 423112898 0       0     0       437368817


TX:

 bytes  packets

errors

  dropped carrier collsns
    3312398924 37942624

0

       0       0       0
    altname enp11s0
```

The eth0 NIC statistics of Host-1 suggest the host does not transmit corrupted traffic. This suggests that the link between Host-1's eth0 and Switch-1's Ethernet1/1 is damaged and is the source of this traffic corruption. You need to troubleshoot this link to identify the faulty component that causes this corruption and replace it.

Root Causes of CRC Errors

The most common root cause of CRC errors is a damaged or malfunctioning component of a physical link between two devices. Examples include:

- Failing or damaged physical medium (copper or fiber) or Direct Attach Cables (DACs).
- Failing or damaged transceivers/optics.
- Failing or damaged patch panel ports.
- Faulty network device hardware (this includes specific ports, line card Application-Specific Integrated Circuits [ASICs], Media Access Controls [MACs], fabric modules, and so on).

- Malfunctioning network interface card inserted in a host.

It is also possible for one or more misconfigured devices to inadvertently causes CRC errors within a network. One example of this is a Maximum Transmission Unit (MTU) configuration mismatch between two or more devices within the network that causes large packets to be incorrectly truncated. When you identify and resolve this configuration issue it can correct CRC errors within a network as well.

Resolve CRC Errors

You can identify the specific malfunctioning component through a process of elimination:

1. Replace the physical medium (either copper or fiber) or DAC with a known-good physical medium of the same type.
2. Replace the transceiver inserted in one device interface with a known-good transceiver of the same model. If this does not resolve the CRC errors, replace the transceiver inserted in the other device interface with a known-good transceiver of the same model.
3. If any patch panels are used as part of the damaged link, move the link to a known-good port on the patch panel. Alternatively, eliminate the patch panel as a potential root cause by connecting the link

without the patch panel if possible.
4. Move the damaged link to a different, known-good port on each device. You can need to test multiple different ports to isolate a MAC, ASIC, or line card failure.
5. If the damaged link involves a host, move the link to a different NIC on the host. Alternatively, connect the damaged link to a known-good host to isolate a failure of the host NIC.

If the malfunctioning component is a Cisco product (such as a Cisco network device or transceiver) that is covered by an active support contract, you can open a support case with Cisco TAC, include your problem details, and have the malfunctioning component replaced through a Return Material Authorization (RMA).

Related Information

- Nexus 9000 Cloud Scale ASIC CRC Identification & Tracing Procedure
- Cisco Technical Support & Downloads