Troubleshoot MTU on Catalyst 9000 Series Switches

Contents

Introduction

This document describes how to understand and troubleshoot Maximum Transmission Unit (MTU) on Catalyst 9000 series switches.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

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

- C9200
- C9300
- C9400
- C9500
- C9600

Note: You can configure the MTU size for all interfaces on a device at the same time with the global command **system mtu**. As of Cisco IOS® XE 17.1.1, Catalyst 9000 switches support Per-Port MTU. Per-Port MTU supports port level and port channel level MTU configuration. With Per-Port MTU you can set different MTU values for different interfaces as well as different port channel interfaces.

Note: Consult the appropriate configuration guide for the commands that are used in order to enable these features on other Cisco platforms.

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

MTU Summary Table

Total Fram Size = MTU + L2 Header

Port Type	**Default MTU - Bytes**	**Configured MTU - Bytes**	**L2 Header**	**Total Frame Size**
L2 Access	1500		18	1518
		9216	18	9234
L2 Trunk	1500		22	1522
		9216	22	9238
L3 Physical Port	1500		18	1518
		9216	18	9234
L3 SVI	1500		18	1518
		9216	18	9234

IP MTU on L3 port	1500	Range is supported.	18	Based on the ip mtu configured value.

MTU Q&A

What is MTU?

- MTU is the Maximum Transmit Unit a device can forward. In general, this Unit is the IP packet Length which includes the IP Header.
- L2 headers like, Dot1q tag, MacSec, SVL header and so on, are not accounted in this calculation.

What is L2 header and its length?

- A generic L2 header is 14 bytes + 4 bytes of CRC, and totals 18 bytes.
- A trunk adds 4 more bytes for the dot1q vlan tag, and totals 22 bytes.
- Similarly, MacSec adds its own header length on top of the typical L2 header length.
- SVL port adds its own header length on top of the typical L2 header length.
- So, Overall Packet on Wire is bumped on the wire.

What is the packet length handled by an interface?

- Catalyst 9000 switches handle packet sizes from 64 bytes to 9238 bytes.

What is Default MTU?

- The default MTU is the MTU the switch is set prior to any user configuration.
- The default MTU on any Catalyst 9000 switches is 1500 bytes.
- An Ethernet port forwards a 1500 byte Layer 3 packet + a Layer 2 header.

Does MTU check happen Ingress or Egress?

Egress: MTU is the Maximum Transmission Unit. It is an Egress check, the decision to fragment or transmit as is or drop is decided for egress.

- If the Port MTU is higher than the packet length to be routed out, Packet is sent as is.
- If the Packet is larger than the egress port MTU and if Egress port is
  - a Layer 3 port, packets is fragmented as per the MTU.
  - a Layer 2 port, Packets are dropped. (Fragmentation is done only at Layer 3)

Note: If a packet has the Do not Fragment (DNF) bit set in the IP header and Port MTU is less than the packet to be routed, Packet is dropped

---

Ingress: MTU check is also done for packets which arrive at an interface.

- If an interface receives a packet over its configured MTU, these packets are treated as oversized packets and dropped.

What are Jumbo Packets?

- On Catalyst 9000 switches, anything over 1500 bytes is a giant packet or a jumbo packet.
  - Example-1: If an interface MTU is configured to forward Jumbo frames size of 9216 bytes, it accepts or sends frames of 9216 bytes + Layer 2 headers.
  - Example-2: If an Interface MTU is configured to forward a Jumbo frame size of 5000 bytes, it

accepts or sends frames of 5000 bytes + Layer 2 headers.

Are Jumbo packets or Oversized packets considered error packets ?

- An interface drops received packets over configured MTU and reports packets as errors.
- If the interface is configured to carry a Jumbo MTU, and received packets are within this value, they are not counted as errors.

What is the Minimum Packet Size a port can handle?

- 64 bytes (L2 header, included) is the smallest valid packet size the switch accepts on Ingress.
- If a packet comes with less than 64 bytes on the wire, it is considered as a Runt and is dropped on Ingress.
- If a packet is supposed to transmit out and the packet is less than 64 bytes, the switch pads the packet to make it to a minimum of 64 bytes before transmission.

What happens when the System MTU is 9216 and SVL header adds an additional 64 bytes?

- Any header under the Layer 3 IP header is not accounted in MTU calculation.
- SVL link can transmit a packet size of 9216 + L2 Header + 64 bytes of SVL header.

What is IP MTU?

- IP MTU is only applicable to IP packets. Other non-ip packet sizes are not accounted for with this command.
- IP MTU takes precedence over system MTU or per-port MTU for IP packets.
- IP MTU sets the maximum size an IP packet can be before it needs fragmentation.
- If physical or logical Layer 3 interface has an MTU of 1500 bytes with ip mtu of 1400 bytes, the fragmentation boundary is 1400 bytes regardless of system or per-port MTU setting.
- MTU is a value which needs to be matched with the peer router/switch. If peer device does not support the higher MTU value, use IP MTU or MTU to match both device capabilities.
- When IP MTU is configured, the device sizes the routing protocol packets to the configured ip MTU value. Some routing protocols rely on the matched MTU value to establish routing protocol neighborship.

Examples:

- Example 1: If an interface IP MTU is configured at 500 bytes with the interface MTU is default (no per-port mtu) and system MTU is 9000, the interface MTU is 9000 bytes, with IP fragmentation at 500 bytes.
- Example 2: A GRE tunnel is the egress interface, so the 24 bytes of GRE header needs to be accounted for in the packet size calculation (ip mtu 1476 + 24 bytes GRE header = 1500 total MTU).

What is the difference between System MTU and Per-Port MTU?

- System MTU is a global configuration, which sets the MTU of the whole device. This changes all the front panel physical ports and logical ports to the value set by the **system mtu** command.
- Per-port MTU allows setting an MTU value on a per interface basis, and this takes precedence over the system MTU configuration. Once the per-port setting is removed, the interface falls back to the system mtu.

Examples:

- Example 1: System MTU value is set to 9000, all the physical and logical ports MTU are set to 9000.
- Example 2: If an interface is configured with an MTU of 4000 and System MTU is 9000, the interface

then uses an MTU of 4000 while other ports use MTU 9000.

What is the impact of fragmentation due to MTU limitations?

- A device forwards an already fragmented packet normally in the data plane, but, if the device is responsible for the fragmentation or reassembly, there can be performance/resource problems that manifest.
- Fragmentation can have serious impact to the overall throughput and performance of applications and devices responsible for fragmentation handling.
- Fragmented packet handling in many platforms is done in software, and takes alot of CPU cycles to fragment or assemble fragmented packets.
- If your network experiences lots of fragmentation, ensure MTU is adjusted accordingly to match end to end packet flow without fragmentation.

What is PMTUD (Path MTU Discovery)?

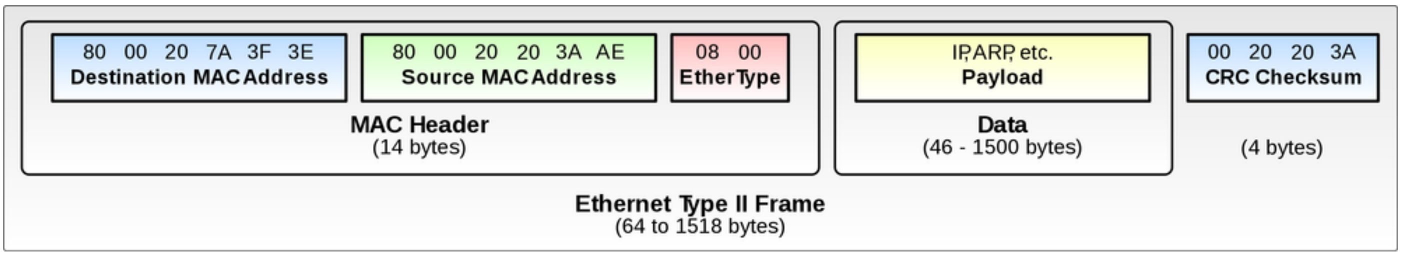
- TCP MSS as described earlier takes care of fragmentation at the two endpoints of a TCP connection, but it does not handle the case where there is a smaller MTU link in the middle between these two endpoints. PMTUD was developed in order to avoid fragmentation in the path between the endpoints. It is used to dynamically determine the lowest MTU along the path from a packet source to its destination.
- For more information on PMTUD, and how to troubleshoot, please consult [Resolve IPv4 Fragmentation, MTU, MSS, and PMTUD Issues with GRE and IPsec.]

IPv6 MTU

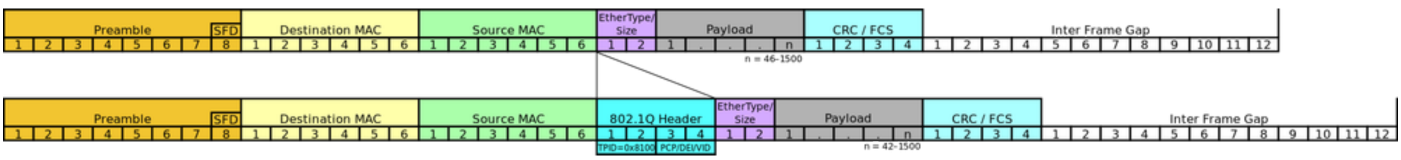
- IPv6 MTU operates in the same way as IP MTU.
- To configure, use ipv6 mtu instead of ip mtu under the interface configuration.
- The minimum size for IPv6 MTU is 1280, versus IPv4 is 832 bytes.
- IPv6 PMTUD works similarly to IPv4. For more details, see [IP Routing Configuration Guide, Cisco IOS® XE Amsterdam 17.3.x (Catalyst 9500 Switches)]

Ethernet Frames

Standard Ethernet Frame, with no Dot1Q, or other tags.



Dot1Q Ethernet Frame



Configure and Verify MTU

Configure MTU

This configuration can be done globally, or at the per-port level with Cisco IOS® XE 17.1.1 or higher, Check your hardware supports this configuration.

- Once the port-specific configuration is removed, the port uses the global system MTU setting.

```
<#root>

### Global System MTU set to 1800 bytes ###



9500H(config)#

system mtu ?


 <1500-9216> MTU size in bytes

   <-- Size range that is configurable



9500H(config)#

system mtu 1800     <-- Set global to 1800 bytes


Global Ethernet MTU is set to 1800 bytes

.
Note: this is the Ethernet payload size, not the total
Ethernet frame size, which includes the Ethernet
header/trailer and possibly other tags, such as ISL or
802.1q tags.

<-- CLI provides information about what is counted as MTU



### Per-Port MTU set to 9216 bytes ###

9500H(config)#

int TwentyFiveGigE1/0/1


9500H(config-if)#

mtu 9126        <-- Interface specific MTU configuration
```

Verify MTU

This section describes how to verify both the software and hardware settings for MTU.

- Verify the Software configured MTU and the Hardware MTU.
- Traffic loss can occur if hardware does not match the configured MTU in software.

Software MTU Verification

<#root>

9500H#show system mtu

Global Ethernet MTU is

1800 bytes

.

<-- Global level MTU

9500H#

show interfaces mtu

Port Name MTU
Twe1/0/1

9216 <-- Per-Port MTU override

Twe1/0/2

1800 <-- No per-port MTU uses global MTU

<...snip...>

9500H#

show interfaces TwentyFiveGigE 1/0/1 | inc MTU
MTU 9216

bytes, BW 1000000 Kbit/sec, DLY 10 usec,

9500H#

show interfaces TwentyFiveGigE 1/0/2 | inc MTU
MTU 1800 bytes,

BW 25000000 Kbit/sec, DLY 10 usec,

Hardware MTU Verification

<#root>

9500H#

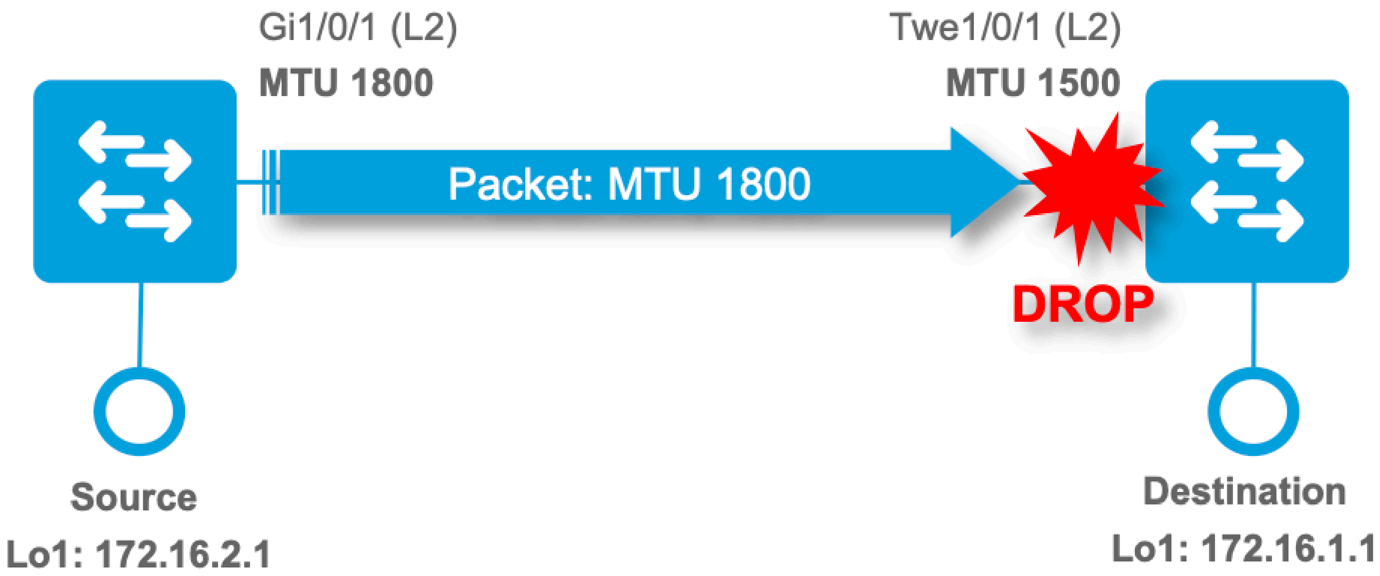
show platform software fed active ifm mappings

```
Interface

IF_ID

    Inst Asic Core Port SubPort Mac Cntx LPN GPN Type Active
TwentyFiveGigE1/0/1

 0x8

        1    0    1    20      0       16   4     1   101 NIF   Y

<-- Retrieve the IF_ID for use in the next command


TwentyFiveGigE1/0/2

0x9

        1    0    1    21      0       17   5     2   102 NIF   Y

9500H#

show platform software fed active ifm if-id 0x8 | inc MTU


                Jumbo MTU ............

[9216]  <-- Hardware matches software configuration


9500H#

show platform software fed active ifm if-id 0x9 | in MTU

                Jumbo MTU ............

[1800]  <-- Hardware matches software configuration
```

Note: **Show platform software fed <active|standby>** can vary. Certain platforms require **show platform hardware fed switch <active|standby|sw_num>**.

Troubleshoot MTU

Topology



Ingress Packet Drops (Lower Ingress MTU)

If either of these counters increment, it usually it means that the received packets have arrived over the configured MTU.

- Giants counter in **show interface** command.
- ValidOverSize counter in **show controller** command.

<#root>

```
9500H#

show int twentyFiveGigE 1/0/3 | i MTU
MTU 1500 bytes,

BW 100000 Kbit/sec, DLY 100 usec,
   0 runts,

0 giants

, 0 throttles

<-- No giants counted



9500H#

show controllers ethernet-controller twentyFiveGigE 1/0/3 | i ValidOverSize


            0 Deferred frames

0 ValidOverSize frames <-- No giants counted


### 5 pings from neighbor device with MTU 1800 to ingress port MTU 1500 ###


9500H#
```

```
show int twentyFiveGigE 1/0/3 | i MTU|giant

MTU 1500 bytes, BW 100000 Kbit/sec, DLY 100 usec,
   0 runts,

5 giants

, 0 throttles

<-- 5 giants counted



9500H#

show controllers ethernet-controller twentyFiveGigE 1/0/3 | i ValidOverSize


            0 Deferred frames

5 ValidOverSize frames <-- 5 giants counted
```

Details about the **show controllers ethernet-controller** command.

- If packets arrive over the configured MTU and fail the CRC check, they are counted as InvalidOverSize.
- If packets arrive within the configured MTU and fail the CRC check, they are counted as FcsErr.

<#root>

```
9500H#

show controllers ethernet-controller twentyFiveGigE 1/0/3 | i Fcs|InvalidOver


            0 Good (>1 coll) frames

0 InvalidOverSize frames <-- MTU too large and bad CRC


            0 Gold frames dropped

 0 FcsErr frames          <-- MTU within limits with bad CRC
```

Configure and Verify IP MTU

Configure IP MTU

This section describes how to configure ip MTU on a tunnel interface.

- IP MTU can be configured to influence the size of IP packets generated by the local system (such as routing protocol updates), or can be used to set a size which at fragmentation is to occur.

<#root>

```
C9300(config)#
```

interface tunnel 1

```
C9300(config-if)#
```

ip mtu 1400

interface Tunnel1

```
 ip address 10.11.11.2 255.255.255.252
```

ip mtu 1400   <-- IP MTU command sets this line at 1400

```
 ip ospf 1 area 0
 tunnel source Loopback0
 tunnel destination 192.168.1.1
```

Verify IP MTU

Software IP MTU Verification

<#root>

```
C9300#
```

sh ip interface tunnel 1 <-- Show the IP level configuration of the interface

```
Tunnel1 is up, line protocol is up
  Internet address is 10.11.11.2/30
  Broadcast address is 255.255.255.255
  Address determined by setup command
```

MTU is 1400 bytes <-- max size of IP packet before fragmentation occurs

Hardware IP MTU Verification

<#root>

```
C9300#sh platform software fed switch active ifm interfaces tunnel
Interface
```

IF_ID

```
              State
----------------------------------------------------------------------
Tunnel1
```

0x00000050

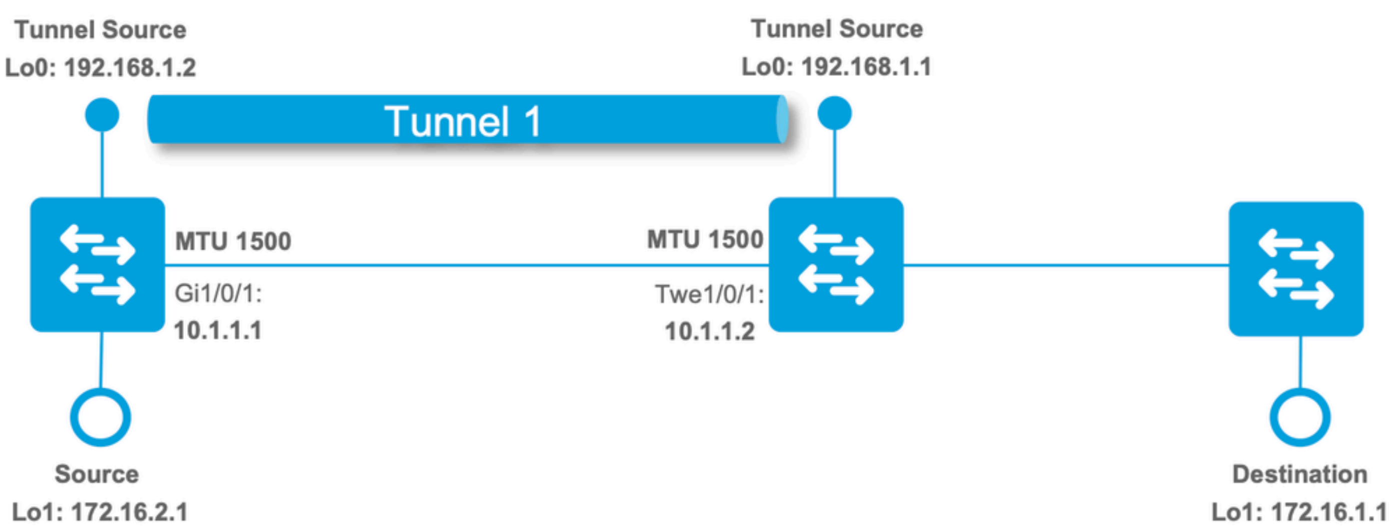
```
          READY
```

<-- Retrieve the IF_ID for use in the next command

```
C9300#sh platform software fed switch active ifm if-id 0x00000050
Interface IF_ID

         : 0x0000000000000050

<-- The interface ID (IF_ID)



Interface Name           : Tunnel1


Interface Block Pointer : 0x7fe98cc2d118
Interface Block State   : READY
Interface State         : Enabled
Interface Status        : ADD, UPD
Interface Ref-Cnt       : 4

Interface Type          : TUNNEL



<...snip...>


       Tunnel Sub-mode: 0 [none]
       Hw Support : Yes
       Tunnel Vrf : 0


 IPv4 MTU : 1400                       <-- Hardware matches software configuration
<...snip...>
```

Troubleshoot IP MTU

Topology



IP Fragmentation

When packets are sent through a Tunnel interface, fragmentation can happen in two ways noted in these examples.

Standard IP Fragmentation

Fragmentation of the original packet to reduce MTU before tunnel encapsulation.

- Only the ingress device is responsible for this fragmentation action, with fragments to be reassembled at the actual endpoint rather than the tunnel endpoint.
- This kind of packet fragmentation is not as resource intensive to accomplish.

<#root>

```
### Tunnel Source Device: Tunnel IP MTU 1400 | Interface MTU 1500 ###


C9300#

ping 172.16.1.1 source Loopback 1 size 1500 repeat 10 <-- ping with size over IP MTU 1400


Type escape sequence to abort.
Sending 100, 1500-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
Packet sent with a source address of 172.16.2.1
!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/1/1 ms



### Tunnel Destination Device: Ingress Capture Twe1/0/1 ###


9500H#

show monitor capture 1


Status Information for Capture 1



Target Type:


Interface: TwentyFiveGigE1/0/1, Direction: IN  <-- Ingress Physical interface

9500H#sh monitor capture 1 buffer br | inc IPv4|ICMP

    9  22.285433   172.16.2.1 b^F^R 172.16.1.1

IPv4 1434 Fragmented IP protocol (proto=ICMP 1, off=0, ID=6c03)

   10  22.285526   172.16.2.1 b^F^R 172.16.1.1 ICMP 162 Echo (ping) request id=0x0004, seq=0/0, ttl=255


   11  22.286295   172.16.2.1 b^F^R 172.16.1.1

IPv4 1434 Fragmented IP protocol (proto=ICMP 1, off=0, ID=6c04)

   12  22.286378   172.16.2.1 b^F^R 172.16.1.1 ICMP 162 Echo (ping) request id=0x0004, seq=1/256, ttl=2

<-- Fragmentation occurs on the Inner ICMP packet

(proto=ICMP 1)
```

```
<-- Fragments are not reassembled until they reach the actual endpoint device 172.16.1.1
```

Post Tunnel Encapsulation Fragmentation

Fragmentation of the actual tunnel packet to reduce MTU once encapsulation has occurred, but the device detects MTU is too large.

- In this case, the tunnel destination is the device responsible for fragment reassembly, rather then the true destination endpoint.
- This case happens when there is a configuration issue. The device is set for a higher IP MTU than the actual port or system MTU can handle after tunnel headers are applied.
- In this case, the tunnel source must fragment the tunnel itself, and the tunnel destination must reassemble the tunnel headers in order to send the packets to the next hop or destination.
- This kind of header fragmentation can add significant processing overhead; it depends on the rate of the flows that must be handled.
- Depending on the platform, code, and traffic rate, you can also see packet loss and drops in CoPP Class Forus traffic.

<#root>

```
### Tunnel Source Device: Tunnel IP MTU 1500 | Interface MTU 1500 ###

C9300(config-if)#

ip mtu 1500


%Warning: IP MTU value set 1500 is greater than the current transport value 1476, fragmentation may occu
<-- Device warns the user that this can cause fragmentation (this is a configuration issue)


### Tunnel Destination Device: Ingress Capture Twe1/0/1 ###

9500H#

show monitor capture 1


Status Information for Capture 1
Target Type:


Interface: TwentyFiveGigE1/0/1, Direction: IN <-- Ingress Physical interface


9500H

#sh monitor capture 1 buffer br | i IPv4|ICMP

    1   0.000000

192.168.1.2 b^F^R 192.168.1.1



IPv4 1514 Fragmented IP protocol (proto=Generic Routing Encapsulation 47

, off=0, ID=4501)
    2   0.000042   172.16.2.1 b^F^R 172.16.1.1   ICMP 60 Echo (ping) request id=0x0005, seq=0/0, ttl=25
    3   2.000598
```

```
192.168.1.2 b^F^R 192.168.1.1


IPv4 1514 Fragmented IP protocol (proto=Generic Routing Encapsulation 47

, off=0, ID=4502)
    4   2.000642   172.16.2.1 b^F^R 172.16.1.1   ICMP 60 Echo (ping) request id=0x0005, seq=1/256, ttl=


<-- Fragmentation has occurred on the outer GRE header(proto=Generic Routing Encapsulation 47)
<-- Fragments must be reassembled at the Tunnel endpoint, in this case the 9500
```

Cisco Bug IDs

Cisco bug ID CSCvr84911 System MTU not respected after reload.

Cisco bug ID CSCvq30464CAT9400: MTU config not applied to inactive ports which become active.

Cisco bug ID CSCvh04282 Cat9300 non-default system MTU config value is not respected after reload.

Related Information

- Technical Support & Documentation - Cisco Systems
- Interface and Hardware Components Configuration Guide, Cisco IOS® XE Amsterdam 17.3.x (Catalyst 9500 Switches)
- Interface and Hardware Components Configuration Guide, Cisco IOS® XE Amsterdam 17.3.x (Catalyst 9600 Switches)
- Resolve IPv4 Fragmentation, MTU, MSS, and PMTUD Issues with GRE and IPsec