Source : https://www.spillthensxt.com/how-to-validate-mtu-in-an-nsx-t-environment/

Introduction:

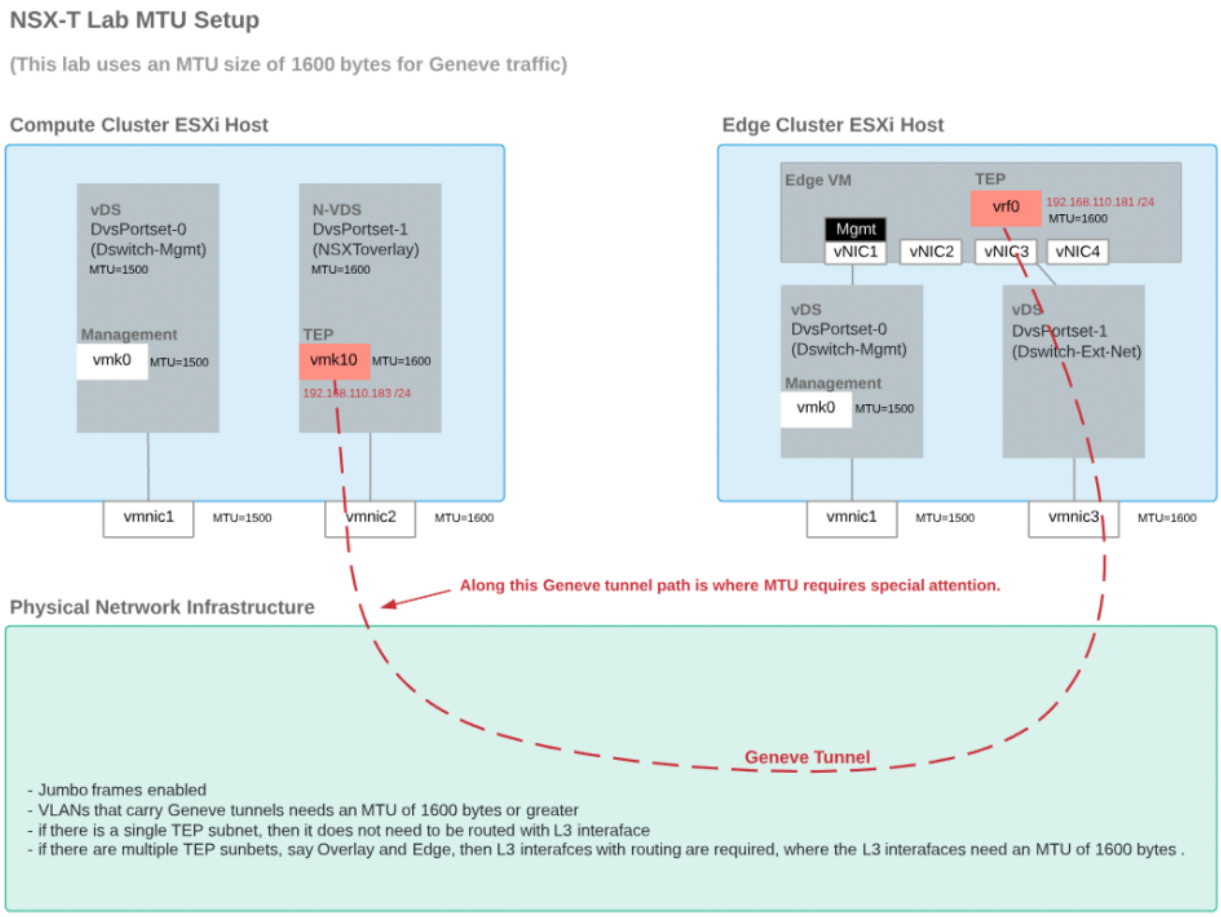
NSX-T leverages the Generic Network Virtualization Encapsulation (Geneve) protocol, a network virtualization tunneling protocol used to establish tunnels across transport nodes to carry overlay traffic. Transport nodes include VM and physical-based Edges, ESX hosts, and KVM Hypervisors, all of which require at least one Geneve Termination End Point (TEP). With encapsulation technologies, like Geneve, it is essential to increase the maximum transmission unit (MTU) supported both on transport nodes and the physical network underlay. This article looks at steps to validate MTU in an NSX-T Environment.

MTU Considerations in an NSX-T Environment:

Here are some concepts to keep in mind when considering MTU setup:

- **Ge**neve frames cannot be fragmented. The MTU size must be large enough to support the encapsulation overhead. We must provide an MTU size of 1600 or greater on any network that carries Geneve overlay traffic.
- Jumbo frames are Ethernet frames with more than 1500 bytes of payload.
- Depending on the payload size, a Geneve frame is often a Jumbo frame.
- The VMware Validated design uses an MTU size of 9000 bytes for Geneve traffic.
- To improve traffic throughput, you should strongly consider configuring the MTU size to at least 9000 bytes.
- When adjusting the MTU packet size, you must also configure the entire network path (VMkernel ports, virtual switches, physical switches, and routers) to support the same MTU packet size.
- If a device along the path does not support the required frame size and receives a frame larger than it's MTU, it will drop the frame.
- MTU has a meaning and is set at Layer 2 the VLAN level, and has a meaning at Layer 3 the interface level.
- All devices within a segment must have the same MTU.
- Some physical switches, such as D-Link, appear to require an MTU of 1700 bytes to support a virtual switch MTU of 1600 bytes.

NSX-T Lab Topology:



Capturing a Geneve frame using nsxcli:

Let's take a look at a Geneve encapsulated frame in the lab.

Rutger Blom has provided some excellent examples of traffic captures along the NSX-T data path, which we will use to capture a Guest VM SSH session. As you can see in the NSX Lap Topology diagram, lab vmnic2 is the ESXi host physical nic that will carry the Gevene tunnel:

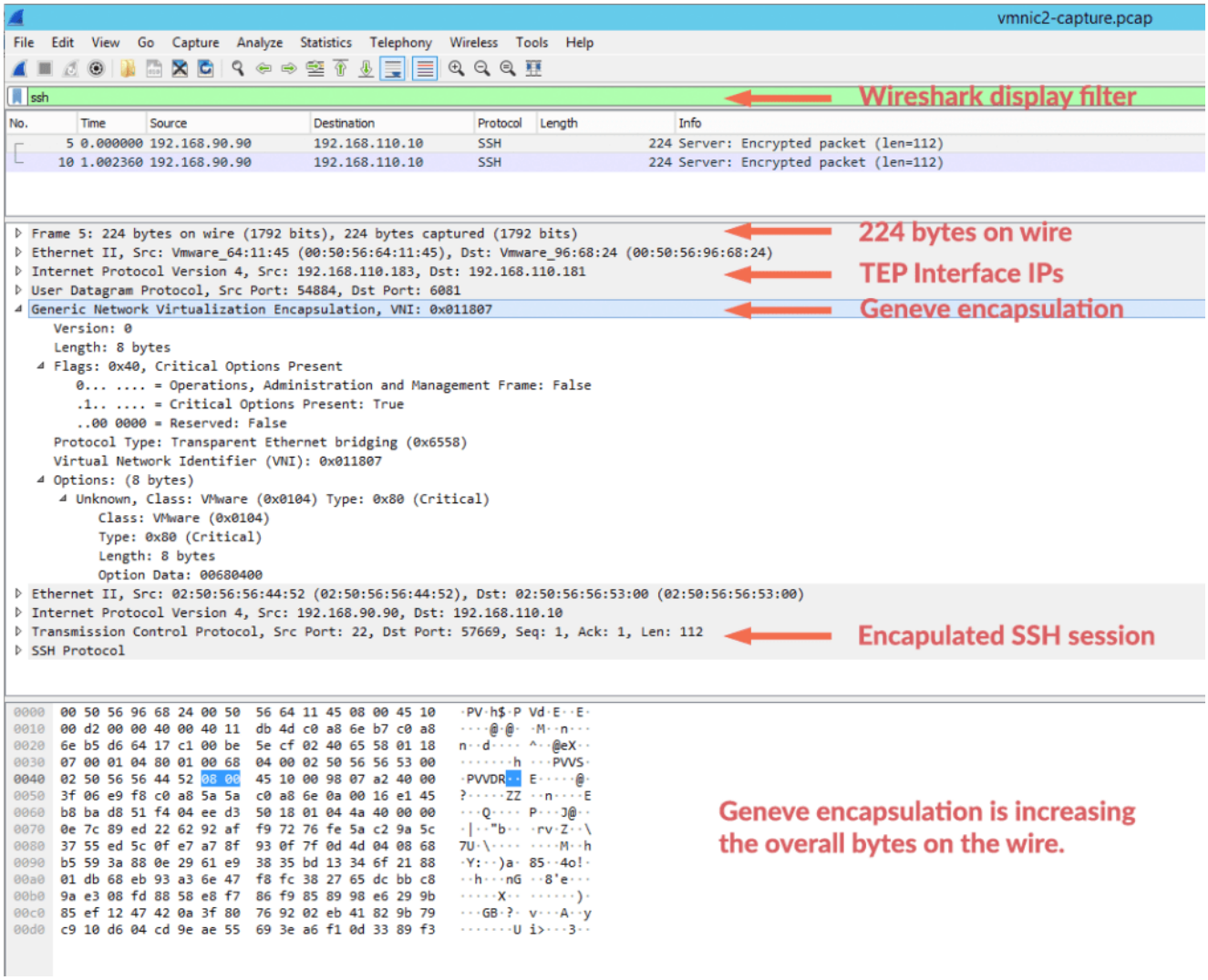
```
[root@esxcna01-s1:/tmp] nsxcli -c start capture interface vmnic2 direction
output file vmnic2-
 capture.pcap count 10
Capture 10 packets to file initiated,
 enter Ctrl-C to terminate before all packets captured
```

Examining the Geneve frame with Wireshark:

In this capture note that:

- esxcna01-s1 is the ESXi host where Guest VM with IP address 192.168.90.90 resides
- 192.168.110.10 represents an interface in the non-virtualized environment

- the traffic is captured at a point in the network where the traffic from virtual to physical is Geneve encapsulated
- An SSH session is established between SSH server 192.168.90.90 and SHS client 192.168.110.10
- 192.168.110.183 is the TEP interface on ESXi host esxcna01-s1
- 192.168.110.181 is the TEP interface on Edge



Regarding the Geneve frame:

- the frame is 224 bytes on the wire
- this is an IP datagram, where the IP Source and Destination are the ESXi host and Edge TEP interfaces on the 192.168.110.0/24 subnet
- this is a UDP segment, with a UDP destination port of 6081. (IANA has assigned port 6081 as the fixed well-known destination port for Geneve.)
- Wireshark is conveniently able to decode the Geneve header with a Virtual Network Identifier (VNI) of 0x011807
- The Inner Ethernet Header is an Ethernet II frame, with the TCP based SSH session

Most importantly for this discussion, **the Geneve encapsulation is increasing the overall bytes on the wire**.

Steps to Validate the MTU in an NSX-T Environment:

OK, so here is the section you've been waiting for, validating MTU in an NSX-T Environment. Let's break the process down into steps.

Step 1: Confirm the MTU is setup consistently across Host Transport Nodes:

The goal is to verify the N-VDS, TEP kernel interface, and physical NIC all have the same jumbo frame MTU size.

```
- On an ESXi Host Transport Node, determine the N-VDS name:

[root@esxcna01-s1:~] nsxdp-cli vswitch instance list
 DvsPortset-1 (NSXToverlay)       a9 f7 33 2f 8d 44 4e 1d-a0 de 40 a6 c5 ae
f1 7e .   <--- the N-VDS name is NSXToverlay
 Total Ports:1536 Available:1517
   Client                         PortID          DVPortID
MAC                 Uplink
   Management                     67108865
00:00:00:00:00:00   n/a
   vmnic2                         67108866        uplink1
00:00:00:00:00:00
   Shadow of vmnic2               67108867
00:50:56:58:f5:37   n/a
   vmk10                          67108868        10
00:50:56:64:11:45   vmnic2
   vmk50                          67108869        083a0efc-e69b-4dd0-9db0-
39de2d24c295 00:50:56:6d:06:78   void
   vdr-vdrPort                    67108870        vdrPort
02:50:56:56:44:52   vmnic2
   VM7.eth0                       67108871        1435b3c2-e944-4173-91e5-
51d9d21fef3c 00:50:56:96:a9:45   vmnic2

- determine the MTU on the N-VDS named NSXToverlay:

 [root@esxcna01-s1:~] nsxdp-cli vswitch mtu get -dvs NSXToverlay
 1600
<--- the N-VDS MTU is 1600 bytes


- notice above that the vmk10, the TEP interface is using Uplink vmnic2

- verify that the N-VDS physical NIC(s) are using this same MTU, in this
csae 1600:

[root@esxcna01-s1:~] esxcfg-nics -l
 Name    PCI          Driver      Link Speed      Duplex MAC Address
MTU    Description
 vmnic0  0000:02:00.0 e1000       Up   1000Mbps   Full   00:50:56:01:44:05
1500   Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
 vmnic1  0000:02:01.0 e1000       Up   1000Mbps   Full   00:50:56:01:10:b9
1500   Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
 vmnic2  0000:02:02.0 e1000       Up   1000Mbps   Full   00:50:56:01:10:bb
1600   Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)   <--
- this MTU looks good
 vmnic3  0000:02:03.0 e1000       Up   1000Mbps   Full   00:50:56:01:10:bc
1500   Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
```

```
 vmnic4  0000:02:04.0 e1000       Down 0Mbps      Half   00:50:56:01:10:c1
1500   Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
 vmnic5  0000:02:05.0 e1000       Down 0Mbps      Half   00:50:56:01:10:c2
1500   Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)

- verify that the vmk10 kernel interface is also using the same MTU, in
this case 1600:

[root@esxcna01-s1:~] esxcfg-vmknic -l
 Interface  Port Group/DVPort/Opaque Network        IP Family IP Address
Netmask         Broadcast       MAC Address       MTU     TSO MSS   Enabled
Type                NetStack
 vmk0       623                                     IPv4
192.168.110.81                          255.255.255.0   192.168.110.255
00:50:56:01:44:05 1500    65535     true    STATIC
defaultTcpipStack
 vmk0       623                                     IPv6
fe80::250:56ff:fe01:4405                64
00:50:56:01:44:05 1500    65535     true    STATIC, PREFERRED
defaultTcpipStack
 vmk1       624                                     IPv4      10.10.20.81
255.255.255.0   10.10.20.255    00:50:56:6a:5e:cf 1500    65535     true
STATIC              defaultTcpipStack
 vmk1       624                                     IPv6
fe80::250:56ff:fe6a:5ecf                64
00:50:56:6a:5e:cf 1500    65535     true    STATIC, PREFERRED
defaultTcpipStack
 vmk10      10                                      IPv4
192.168.110.183                         255.255.255.0   192.168.110.255
00:50:56:64:11:45 1600    65535     true    STATIC              vxlan   <--
- this MTU looks good
 vmk10      10                                      IPv6
fe80::250:56ff:fe64:1145                64
00:50:56:64:11:45 1600    65535     true    STATIC, PREFERRED   vxlan
 vmk50      083a0efc-e69b-4dd0-9db0-39de2d24c295    IPv4      169.254.1.1
255.255.0.0     169.254.255.255 00:50:56:6d:06:78 1500    65535     true
STATIC              hyperbus
 vmk50      083a0efc-e69b-4dd0-9db0-39de2d24c295    IPv6
fe80::250:56ff:fe6d:678                 64
00:50:56:6d:06:78 1500    65535     true    STATIC, PREFERRED   hyperbus
 [root@esxcna01-s1:~]
```

This confirms that the N-VDS, TEP kernel interface(s), and physical NIC(s) all have the same jumbo frame MTU, in this case 1600 bytes.

Step 2: Confirm the MTU is setup consistently across Edge Transport Nodes:

```
nsxtedge01> get logical-routers
 Logical Router
 UUID                                   VRF    LR-ID  Name
Type                        Ports
 736a80e3-23f6-5a2d-81d6-bbefb2786666   0      0
TUNNEL                      3
 3ef116ea-7adc-48bb-bc89-89fd16502087   1      6146   DR-lab-tier-0
DISTRIBUTED_ROUTER_TIER0    5
 34823c67-1efd-49b6-b495-29dec792f377   2      14337  SR-lab-tier-1-tenant-
2           SERVICE_ROUTER_TIER1        5
 7be9fece-e558-4949-b1a2-eaffa26fe0c5   3      8194   SR-lab-tier-0
SERVICE_ROUTER_TIER0        6
```

```
 c1763624-cfe9-44d2-96e3-c2413107a22e   4      11266  DR-lab-tier-1-tenant-
2            DISTRIBUTED_ROUTER_TIER1    4
 9d278256-3211-425f-afbe-0011be89876b   5      12289  DR-lab-tier-1-tenant-
1            DISTRIBUTED_ROUTER_TIER1    5
 2938a6d8-c129-4f7e-8356-ce696d07738e   6      13313  SR-lab-tier-1-tenant-
1            SERVICE_ROUTER_TIER1        5

- in  this case vrf 0 is the Edge Geneve Tunnel interface:

nsxtedge01> vrf 0
nsxtedge01(vrf)> get int
 Logical Router
 UUID                                   VRF    LR-ID  Name
Type
 736a80e3-23f6-5a2d-81d6-bbefb2786666   0      0
TUNNEL
 Interfaces
     Interface     : 9fd3c667-32db-5921-aaad-7a88c80b5e9f
     Ifuid         : 258
     Mode          : blackhole
 Interface     : f322c6ca-4298-568b-81c7-a006ba6e6c88 Ifuid         : 257
Mode          : cpu Interface     : 72dd0b68-e71e-5b53-b801-f7c246f3fdc9
Ifuid         : 327 Name          : Mode          : lif IP/Mask       :
192.168.110.180/24 MAC           : 00:50:56:96:8e:e8 LS port       :
d0955bdb-7b4d-5a88-ba92-ac4eb212ff00 Urpf-mode     : PORT_CHECK Admin
: up Op_state      : up MTU           : 1600          <--- this MTU looks
good, and comes from the Uplink profile applied to the Edge

- confirm the Edge fastpath interfaces MTU

nsxtedge01> get int | find Interface|MTU
 Interface: bond0
   MTU: 1500
 Interface: eth0
   MTU: 1500
 Interface: fp-eth0
   MTU: 1600         <--- this Edge fastpath interfaces MTU looks good
 Interface: fp-eth1
   MTU: 1600         <--- this Edge fastpath interfaces MTU looks good
 Interface: fp-eth2
   MTU: 1500
```

This confirms that the Edge TEP kernel interface(s), and Edge fastpath interfaces all have the same jumbo frame MTU, in this case 1600 bytes.

Step 3: Validate the MTU on any vDS that may be being used by an NSX-T Edge to carry Geneve traffic.

```
- Notice that this Edge Cluster ESXi host, hosts NSX-T-Edge01:

[root@esx03-s1:~] net-stats -l
 PortNum          Type SubType SwitchName       MACAddress
ClientName
 50331650            4       0 DvsPortset-0     00:50:56:01:48:99  vmnic0
 50331652            4       0 DvsPortset-0     00:50:56:01:48:9a  vmnic1
 50331654            3       0 DvsPortset-0     00:50:56:01:48:99  vmk0
 50331655            3       0 DvsPortset-0     00:50:56:6e:5b:51  vmk1
 50331656            3       0 DvsPortset-0     00:50:56:68:22:2d  vmk2
 50331658            5       9 DvsPortset-0     00:50:56:96:88:43  NSX-T-
Edge01-2.4.1.0.0-13716575.eth0
```

```
 50331659            5       9 DvsPortset-0     00:50:56:96:8e:e8  NSX-T-
Edge01-2.4.1.0.0-13716575.eth1
 67108866            4       0 DvsPortset-1     00:50:56:01:48:9b  vmnic2
 67108868            4       0 DvsPortset-1     00:50:56:01:48:9c  vmnic3
 67108870            5       9 DvsPortset-1     00:50:56:96:6f:58  NSX-T-
Edge01-2.4.1.0.0-13716575.eth2
 67108871            5       9 DvsPortset-1     00:50:56:96:6e:d8  NSX-T-
Edge01-2.4.1.0.0-13716575.eth3
```

[root@esx03-s1:~] **esxtop**, then type n for network:

- Notice that NSX-T-Edge01 has NICs on DvsPortset-0 and DvsPortset-1:

```
9:51:00pm up 18 days  9:06, 613 worlds, 1 VMs, 4 vCPUs; CPU load average:
0.15, 0.14, 0.14
 PORT-ID USED-BY                           TEAM-PNIC DNAME
PKTTX/s  MbTX/s   PSZTX    PKTRX/s  MbRX/s   PSZRX %DRPTX %DRPRX
   33554433 Management                                n/a vSwitch0
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
   50331649 Management                                n/a DvsPortset-0
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
   50331650 vmnic0                                      - DvsPortset-0
0.00    0.00    0.00    1672.55   11.34  889.00   0.00   0.00
   50331651 Shadow of vmnic0                          n/a DvsPortset-0
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
   50331652 vmnic1                                      - DvsPortset-0
26.51    0.18  900.00    1634.98   11.16  894.00   0.00   0.00
   50331653 Shadow of vmnic1                          n/a DvsPortset-0
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
   50331654 vmk0                                   vmnic1 DvsPortset-0
3.43    0.01  290.00       5.72    0.00   60.00   0.00   0.00
   50331655 vmk1                                   vmnic1 DvsPortset-0
15.45    0.17 1424.00      23.84    0.02  117.00   0.00   0.00
   50331656 vmk2                                   vmnic1 DvsPortset-0
0.00    0.00    0.00       5.34    0.00   60.00   0.00   0.00
   50331657 vdr-vdrPort                            vmnic1 DvsPortset-0
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
   50331658 671805:NSX-T-Edge01-2.4.1.0.0-     vmnic1 DvsPortset-0
1.53    0.00   60.00       6.87    0.00   61.00   0.00   0.00
   50331659 671805:NSX-T-Edge01-2.4.1.0.0-     vmnic1 DvsPortset-0
6.10    0.01  126.00      12.21    0.01   93.00   0.00   0.00
   67108865 Management                                n/a DvsPortset-1
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
   67108866 vmnic2                                      - DvsPortset-1
0.00    0.00    0.00       1.34    0.00   99.00   0.00   0.00
   67108867 Shadow of vmnic2                          n/a DvsPortset-1
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
   67108868 vmnic3                                      - DvsPortset-1
0.38    0.00   74.00       0.95    0.00  109.00   0.00   0.00
   67108869 Shadow of vmnic3                          n/a DvsPortset-1
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
   67108870 671805:NSX-T-Edge01-2.4.1.0.0-     vmnic3 DvsPortset-1
0.38    0.00   74.00       0.57    0.00  138.00   0.00   0.00
   67108871 671805:NSX-T-Edge01-2.4.1.0.0-     vmnic3 DvsPortset-1
0.00    0.00    0.00       0.00    0.00    0.00   0.00   0.00
```

Verify that Edge interfaces with Geneve traffic over a vDS have an MTU that matches with the system-wide MTU, in this case 1600:

```
 [root@esx03-s1:~] esxcli network vswitch dvs vmware list
```

```
 DSwitch-Mgmt
    Name: DSwitch-Mgmt
    VDS ID: b1 75 16 50 72 84 37 a8-3f 1b 18 e4 55 d0 97 0d
    Class: etherswitch
    Num Ports: 2452
    Used Ports: 11
    Configured Ports: 512
    MTU: 1500                  <--- this vDS MTU is fine, since it's for
non-Geneve traffic
    CDP Status: both
    Beacon Timeout: -1
    Uplinks: vmnic1, vmnic0
    VMware Branded: true
    DVPort:
          Client: vmnic0
          DVPortgroup ID: dvportgroup-16
          In Use: true
          Port ID: 156
      Client: vmnic1      DVPortgroup ID: dvportgroup-16      In Use: true
Port ID: 157      Client: vmk0      DVPortgroup ID: dvportgroup-17      In
Use: true      Port ID: 4      Client: vmk1      DVPortgroup ID:
dvportgroup-17      In Use: true      Port ID: 6      Client: vmk2
DVPortgroup ID: dvportgroup-108      In Use: true      Port ID: 65
Client: NSX-T-Edge01-2.4.1.0.0-13716575.eth0      DVPortgroup ID:
dvportgroup-18      In Use: true      Port ID: 24      Client: NSX-T-
Edge01-2.4.1.0.0-13716575.eth1      DVPortgroup ID: dvportgroup-18      In
Use: true      Port ID: 29
 DSwitch-Ext-Net
    Name: DSwitch-Ext-Net
    VDS ID: 0c 94 16 50 fa 3a 1d fa-76 4e 8a d0 17 11 03 c5
    Class: etherswitch
    Num Ports: 2452
    Used Ports: 7
    Configured Ports: 512
    MTU: 1600                  <--- this vDS MTU looks good
    CDP Status: both
    Beacon Timeout: -1
    Uplinks: vmnic3, vmnic2
    VMware Branded: true
    DVPort:
          Client: vmnic2
          DVPortgroup ID: dvportgroup-85
          In Use: true
          Port ID: 18
      Client: vmnic3      DVPortgroup ID: dvportgroup-85      In Use: true
Port ID: 19      Client: NSX-T-Edge01-2.4.1.0.0-13716575.eth2
DVPortgroup ID: dvportgroup-86      In Use: true      Port ID: 1
Client: NSX-T-Edge01-2.4.1.0.0-13716575.eth3      DVPortgroup ID:
dvportgroup-1273      In Use: true      Port ID: 420
```

Step 4: Collect IP addressing for all TEP interfaces:

In my lab I have the following:

```
[root@esxcna01-s1:~] esxcfg-vmknic -l | grep vxlan
vmk10      10                                      IPv4
192.168.110.183                         255.255.255.0   192.168.110.255
00:50:56:64:11:45 1600    65535     true    STATIC              vxlan
```

```
 vmk10      10                                      IPv6
fe80::250:56ff:fe64:1145                64
00:50:56:64:11:45 1600    65535     true    STATIC, PREFERRED   vxlan


[root@esxcna02-s1:~] esxcfg-vmknic -l | grep vxlan
 vmk10      10                                      IPv4
192.168.110.182                         255.255.255.0   192.168.110.255
00:50:56:6d:66:f4 1600    65535     true    STATIC              vxlan
 vmk10      10                                      IPv6
fe80::250:56ff:fe6d:66f4                64
00:50:56:6d:66:f4 1600    65535     true    STATIC, PREFERRED   vxlan

nsxtedge01> vrf 0
nsxtedge01(vrf)> get int
 Logical Router
 UUID                                   VRF    LR-ID  Name
Type
 736a80e3-23f6-5a2d-81d6-bbefb2786666   0      0
TUNNEL
 Interfaces
     Interface     : 9fd3c667-32db-5921-aaad-7a88c80b5e9f
     Ifuid         : 258
     Mode          : blackhole
 Interface     : f322c6ca-4298-568b-81c7-a006ba6e6c88 Ifuid         : 257
Mode          : cpu Interface     : 72dd0b68-e71e-5b53-b801-f7c246f3fdc9
Ifuid         : 327 Name          : Mode          : lif IP/Mask       :
192.168.110.180/24 MAC           : 00:50:56:96:8e:e8 LS port       :
d0955bdb-7b4d-5a88-ba92-ac4eb212ff00 Urpf-mode     : PORT_CHECK Admin
: up Op_state      : up MTU           : 1600

nsxtedge02> vrf 0
 nsxtedge02(vrf)> get int
 Logical Router
 UUID                                   VRF    LR-ID  Name
Type
 736a80e3-23f6-5a2d-81d6-bbefb2786666   0      0
TUNNEL
 Interfaces
     Interface     : 9fd3c667-32db-5921-aaad-7a88c80b5e9f
     Ifuid         : 258
     Mode          : blackhole
 Interface     : 4378f73b-e0f8-5743-9cd6-5d06710f29d4 Ifuid         : 334
Name          : Mode          : lif IP/Mask       : 192.168.110.181/24 MAC
: 00:50:56:96:68:24 LS port       : ee60c9da-aaa5-500b-b241-47395a99f089
Urpf-mode     : PORT_CHECK Admin         : up Op_state      : up MTU
: 1600 Interface     : f322c6ca-4298-568b-81c7-a006ba6e6c88 Ifuid         :
257 Mode          : cpu
```

Step 5: Use vmkping to test between all TEP interfaces, over the VXLAN network stack, initiated from a Compute ESXi Host:

From Step 4, the following TEP IPs have been collected: (Note that it's possible to have multiple TEP interfaces on Hosts and edges.)

- Compute Host1 TEP = 192.168.110.183
- Compute Host2 TEP = 192.168.110.182
- Edge 1 TEP = 192.168.110.180

- Edge 2 TEP = 192.168.110.181

In this scenario, the System-wide MTU is 1600 bytes, with the don't fragment bit set, test with a size of 1572. The number of ICMP data bytes to be sent is 1572, with an 8 byte ICMP header, this will produce in1600 byte frame.

If the System-wide MTU is 9000 bytes, test with an ICMP payload size of 8972. With the system-wide MTU of 9000 bytes in the environment, it will need to work with an ICMP payload size of 8972. With the ICMP data bytes to be sent is 9872 and an 8 byte ICMP header, this will produce a 9000 byte frame.

```
Here are the command options for vmkping:

[root@esxcna01-s1:~] vmkping
 vmkping [args] [host]
    args:
       -4               use IPv4 (default)
       -6               use IPv6
       -c         set packet count
       -d               set DF bit (IPv4) or disable fragmentation (IPv6)
       -D               vmkernel TCP stack debug mode
       -i     set interval (secs)
       -I    outgoing interface - for IPv6 scope or IPv4
                        bypasses routing lookup
       -N     set IP*_NEXTHOP - bypasses routing lookup
                        for IPv4, -I option is required
       -s          set the number of ICMP data bytes to be sent.
                        The default is 56, which translates to a 64 byte
                        ICMP frame when added to the 8 byte ICMP header.
                        (Note: these sizes does not include the IP header).
       -t           set IPv4 Time To Live or IPv6 Hop Limit
       -v               verbose
       -W      set timeout to wait if no responses are
                        received (secs)
   -X               XML output format for esxcli framework.   -S
The network stack instance name. If unspecified                    the
default netstack instance is used.
 NOTE: In vmkernel TCP debug mode, vmkping traverses
          VSI and pings various configured addresses.
Since the N-VDS MTU is 1600 bytes in the lab, test with an ICMP payload of
1572:

[root@esxcna01-s1:~] vmkping ++netstack=vxlan 192.168.110.180 -d -s 1572  -
I vmk10
 PING 192.168.110.180 (192.168.110.180): 1572 data bytes
 1580 bytes from 192.168.110.180: icmp_seq=0 ttl=64 time=128.921 ms
 1580 bytes from 192.168.110.180: icmp_seq=1 ttl=64 time=1.616 ms
 1580 bytes from 192.168.110.180: icmp_seq=2 ttl=64 time=1.383 ms
 --- 192.168.110.180 ping statistics ---
 3 packets transmitted, 3 packets received, 0% packet loss
 round-trip min/avg/max = 1.383/43.973/128.921 ms

 [root@esxcna01-s1:~] vmkping ++netstack=vxlan 192.168.110.181 -d -s 1572
-I vmk10
 PING 192.168.110.181 (192.168.110.181): 1572 data bytes
 1580 bytes from 192.168.110.181: icmp_seq=0 ttl=64 time=57.811 ms
 1580 bytes from 192.168.110.181: icmp_seq=1 ttl=64 time=1.328 ms
 1580 bytes from 192.168.110.181: icmp_seq=2 ttl=64 time=1.377 ms
 --- 192.168.110.181 ping statistics ---
```

```
 3 packets transmitted, 3 packets received, 0% packet loss
 round-trip min/avg/max = 1.328/20.172/57.811 ms

 [root@esxcna01-s1:~] vmkping ++netstack=vxlan 192.168.110.182 -d -s 1572
-I vmk10
 PING 192.168.110.182 (192.168.110.182): 1572 data bytes
 1580 bytes from 192.168.110.182: icmp_seq=0 ttl=64 time=2.282 ms
 1580 bytes from 192.168.110.182: icmp_seq=1 ttl=64 time=1.024 ms
 1580 bytes from 192.168.110.182: icmp_seq=2 ttl=64 time=1.138 ms
 --- 192.168.110.182 ping statistics ---
 3 packets transmitted, 3 packets received, 0% packet loss
 round-trip min/avg/max = 1.024/1.481/2.282 ms

 [root@esxcna01-s1:~] vmkping ++netstack=vxlan 192.168.110.183 -d -s 1572
-I vmk10
 PING 192.168.110.183 (192.168.110.183): 1572 data bytes
 1580 bytes from 192.168.110.183: icmp_seq=0 ttl=64 time=0.123 ms
 1580 bytes from 192.168.110.183: icmp_seq=1 ttl=64 time=0.115 ms
 1580 bytes from 192.168.110.183: icmp_seq=2 ttl=64 time=0.108 ms
 --- 192.168.110.183 ping statistics ---
 3 packets transmitted, 3 packets received, 0% packet loss
 round-trip min/avg/max = 0.108/0.115/0.123 ms
```

Warning: If you find that the vmkping drops from an ESXi host to an NSX-T Edge when the payload is 2200 bytes, reference the following Knowledge Base article: https://kb.vmware.com/s/article/70878, where an NSX Edge Node and associated T0/T1 SRs cannot generate ICMP responses when the ICMP request is larger than 2020 bytes.

Step 6: Use ping to test between all TEP interfaces, over the VXLAN network stack, initiated from an NSX-T Edge:

If you bumped into 2200 byte limitation referenced in Step 5, then initiate ping testing from the Edge vrf 0 TEP interface, where there is no such limitation.

```
nsxtedge01> ping 192.168.110.181 repeat 2 size 1572 dfbit ENABLE vrf 0
 PING 192.168.110.181 (192.168.110.181): 1572 data bytes
 --- 192.168.110.181 ping statistics ---
 2 packets transmitted, 0 packets received, 100.0% packet loss
 1580 bytes from 192.168.110.181: icmp_seq=0 ttl=64 time=4.363 ms
 1580 bytes from 192.168.110.181: icmp_seq=1 ttl=64 time=3.742 ms
 --- 192.168.110.181 ping statistics ---
 2 packets transmitted, 2 packets received, 0.0% packet loss
 round-trip min/avg/max/stddev = 3.742/4.053/4.363/0.311 ms


 nsxtedge01> ping 192.168.110.182 repeat 2 size 1572 dfbit ENABLE vrf 0
 PING 192.168.110.182 (192.168.110.182): 1572 data bytes
 1580 bytes from 192.168.110.182: icmp_seq=0 ttl=64 time=2.402 ms
 1580 bytes from 192.168.110.182: icmp_seq=1 ttl=64 time=2.589 ms
 --- 192.168.110.182 ping statistics ---
 2 packets transmitted, 2 packets received, 0.0% packet loss
 round-trip min/avg/max/stddev = 2.402/2.495/2.589/0.094 ms

 nsxtedge01> ping 192.168.110.183 repeat 2 size 1572 dfbit ENABLE vrf 0
 PING 192.168.110.183 (192.168.110.183): 1572 data bytes
 1580 bytes from 192.168.110.183: icmp_seq=0 ttl=64 time=2.829 ms
 1580 bytes from 192.168.110.183: icmp_seq=1 ttl=64 time=2.975 ms
```

```
--- 192.168.110.183 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 2.829/2.902/2.975/0.073 ms
```

Step 7: Use ping to test between an NSX-T Overlay backed Guest VM and physical devices outside the NSX-T environment.

 # ping -s 1472 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 1472(**1500**) bytes of data. <— notice that the ping payload of 1472 bytes results in a 1500 byte frame
76 bytes from 8.8.8.8: icmp_seq=1 ttl=37 (truncated)
76 bytes from 8.8.8.8: icmp_seq=2 ttl=37 (truncated)
76 bytes from 8.8.8.8: icmp_seq=3 ttl=37 (truncated)
76 bytes from 8.8.8.8: icmp_seq=4 ttl=37 (truncated)
76 bytes from 8.8.8.8: icmp_seq=5 ttl=37 (truncated)
76 bytes from 8.8.8.8: icmp_seq=6 ttl=37 (truncated)
76 bytes from 8.8.8.8: icmp_seq=7 ttl=37 (truncated)

Here are some sample Problem Descriptions for an incorrectly configured MTU somewhere in the environment:

MTU configuration errors can lead to problem descriptions such as the following:

- From an NSX-T Overlay backed Guest VM, I can ping all Internet sites just fine. However, I can only load some websites using a web browser.
- From an NSX-T Overlay backed Guest VM, the web browser can't get the web server's SSL certificate.
- From an NSX-T Overlay backed Guest VM, the mail client can read mail headers, but not the email contents.
- Pings with large payloads work within the NSX-T environment, but not destined to physical devices outside the NSX-T environment.
- The Edge Transport Node status is degraded, and Tunnel Status appears down on some transport nodes.

Using the NSX-T Policy API to set system-wide MTU:

```
 Keep in mind that you can update the global confiuguration MTU as follows:

- Determine if the system-wide MTU is set:

GET https://Error! Hyperlink reference not
valid./policy/api/v1/infra/global-config

For example:
GET https://nsxtmgr.core.hypervizor.com/policy/api/v1/infra/global-config

{
     "resource_type": "GlobalConfig",
     "id": "global-config",
     "display_name": "default",
     "path": "/infra/global-config",
     "relative_path": "global-config",
     "marked_for_delete": false,
```

```
      "_create_user": "system",
      "_create_time": 1561055613843,
      "_last_modified_user": "system",
      "_last_modified_time": 1561055613843,
      "_system_owned": true,
      "_protection": "NOT_PROTECTED",
      "_revision": 0
 }

- Set the system-wide MTU to 1600 bytes:

 PATCH Error! Hyperlink reference not valid.

For example:
GET https://nsxtmgr.core.hypervizor.com/policy/api/v1/infra/global-config
 {
   "display_name": "global-config",
   "path": "/infra/global-config",
   "relative_path": "global-config",
   "mtu": 1600,
   "_revision": 0
 }

- Verify the system-wide MTU is set:

GET https://nsxtmgr.core.hypervizor.com/policy/api/v1/infra/global-config

{
      "mtu": 1600,                          <--- the system-wide MTU has been
set to 1600 bytes
      "resource_type": "GlobalConfig",
      "id": "global-config",
      "display_name": "global-config",
      "path": "/infra/global-config",
      "relative_path": "global-config",
      "marked_for_delete": false,
      "_create_user": "system",
      "_create_time": 1561055613843,
      "_last_modified_user": "admin",
      "_last_modified_time": 1571586984942,
      "_system_owned": true,
      "_protection": "NOT_PROTECTED",
      "_revision": 1
 }
```

This should help with validating MTU in an NSX-T environment. For some tips on how to set the revision number correctly on an API call, reference this article: https://www.spillthensxt.com/nsx-t-disable-dfw/