

NETWORK TOOLS AND PROTOCOLS

Lab 13: Impact of MSS on Throughput

Document Version: 06-14-2019



Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput
Networks for Big Science Data Transfers"

Contents

Overvie	ew	3
Objecti	ives	3
Lab set	tings	3
	admap	
1 In	troduction to MSS	3
1.1	Maximum transmission unit (MTU)	3
1.2	Maximum segment size (MSS)	4
2 Lab topology		5
2.1	Starting hosts h1 and h2	6
2.2	Emulating 10 Gbps WAN with packet loss	7
3 Modifying maximum transmission unit (MTU)		10
3.1	Identifying interface's current MTU	10
3.2	Modifying MTU values on all interfaces	11
References		14

Overview

This lab introduces Maximum Transmission Unit (MTU), Maximum Segment Size (MSS), and their effect on network throughput in a high-bandwidth Wide Area Networks (WAN) with packet losses. Throughput measurements are conducted in this lab to compare the observed throughput while using a higher MSS against a normal MSS value.

Objectives

By the end of this lab, students should be able to:

- 1. Understand Maximum Transmission Unit (MTU).
- 2. Define Maximum Segment Size (MSS).
- 3. Identify interfaces' default MTU value.
- 4. Modify the MTU of an interface.
- 5. Understand the benefit of using a high MSS value in a WAN that incurs packet losses.
- 6. Emulate WAN properties in Mininet and achieve full throughput with high MSS.

Lab settings

The information in Table 1 provides the credentials of the machine containing Mininet.

Device	Account	Password
Client1	admin	password

Table 1. Credentials to access Client1 machine.

Lab roadmap

This lab is organized as follows:

- 1. Section 1: Introduction to MSS.
- 2. Section 2: Lab topology.
- 3. Section 3: Modifying maximum transmission Unit (MTU) and analyzing results.

1 Introduction to MSS

1.1 Maximum transmission unit (MTU)

The Maximum Transmission Unit (MTU) specifies the largest packet size (in bytes), including headers and data payload, that can be transmitted by the link-layer technology¹. Even though data rates have dramatically increased since Ethernet standardization, the MTU remains at 1500 bytes. A frame carrying more than 1500 bytes is referred to as a jumbo frame and can allow up to 9000 bytes.

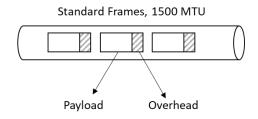


Figure 1. Standard Ethernet Frame's MTU

Figure 1 illustrates the standard Ethernet frame which has 1500 bytes MTU. Although most gigabit networks run with no impact while using the standard MTU, large numbers of frames increase CPU loads and overheads. In such cases jumbo frames can be used to mitigate excess overhead, as demonstrated in figure 2.

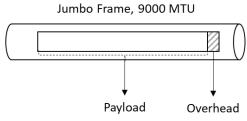


Figure 2. Jumbo Ethernet Frame's MTU

As shown in figure 2, jumbo frames impose lower overheads than normal frames (1500 MTU) by reducing the overall number of individual frames sent from source to destination. Not only does this reduce the number of headers needed to move the data, CPU load is also lessened due to a decrease in packet processing by routers and end devices.

1.2 Maximum segment size (MSS)

The Maximum Segment Size (MSS) is a parameter of the options field of the TCP header that specifies the largest amount of data, specified in bytes, that a computer or communications device can receive in a single TCP segment³. This value is specified in the TCP SYN packet during TCP's three-way handshake and is set permanently for the current session.

The MSS must be set to a value equal to the largest IP datagram (minus IP and TCP headers) that the host can handle in order to avoid fragmentation. Note that lowering the MSS will remove fragmentation, however it will impose larger overhead.

With highspeed networks, using half a dozen or so small probes to see how the network responds wastes a huge amount of bandwidth. Similarly, when packet loss is detected, the rate is decreased by a factor of two. TCP can only recover slowly from this rate reduction. The speed at which the recovery occurs is proportional to the MTU. Thus, it is recommended to use large frames.

In this lab, we show and compare the effect of jumbo frames versus standard frames in a WAN that incurs packet losses.

2 Lab topology

Let's get started with creating a simple Mininet topology using Miniedit. The topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

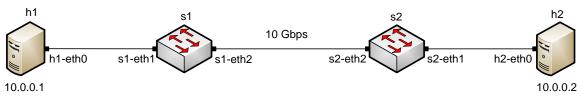


Figure 3. Lab topology.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type password.

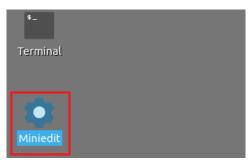


Figure 4. Miniedit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *Open* to load the lab's topology. Locate the *Lab 13.mn* topology file and click on *Open*.

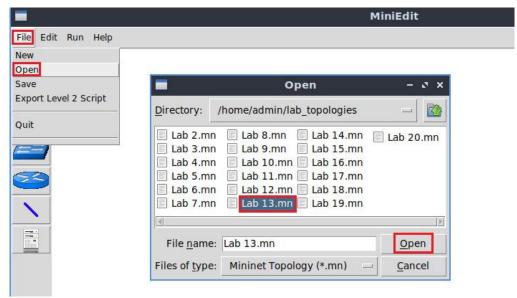


Figure 5. Miniedit's Open dialog.

Step 3. Before starting the measurements between host h1 and host h2, the network must be started. Click on the *Run* button located at the bottom left of Miniedit's window to start the emulation.



Figure 6. Running the emulation.

The above topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

2.1 Starting hosts h1 and h2

Step 1. Hold the right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands on that host.

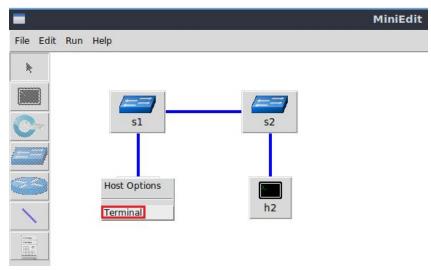


Figure 7. Opening a terminal on host h1.

Step 2. Apply the same steps on host h2 and open its *Terminal*.

Step 3. Test connectivity between the end-hosts using the ping command. On host h1, type the command ping 10.0.0.2. This command tests the connectivity between host h1 and host h2. To stop the test, press Ctrl+c. The figure below shows a successful connectivity test.

```
"Host: h1" - x x

root@admin-pc:~# ping 10.0.0.2

PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.33 ms

64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.056 ms

64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.048 ms

64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.042 ms

64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.043 ms

64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.044 ms

^C

--- 10.0.0.2 ping statistics ---

6 packets transmitted, 6 received, 0% packet loss, time 91ms

rtt min/avg/max/mdev = 0.042/0.260/1.327/0.477 ms

root@admin-pc:~#
```

Figure 8. Connectivity test using ping command.

2.2 Emulating 10 Gbps WAN with packet loss

This section emulates a WAN with packet loss. We will first set the bandwidth between host 1 and host h2 to 10 Gbps. Then, we will emulate a 1% packet loss and measure the throughput.

Step 1. Launch a Linux terminal by holding the Ctrl+Alt+T keys or by clicking on the Linux terminal icon.



Figure 9. Shortcut to open a Linux terminal.

The Linux terminal is a program that opens a window and permits you to interact with a command-line interface (CLI). A CLI is a program that takes commands from the keyboard and sends them to the operating system to perform.

Step 2. In the terminal, type the command below. When prompted for a password, type password and hit *Enter*. This command introduces 1% packet loss on switch S1's s1-eth2 interface.



Figure 10. Adding 1% packet loss to switch S1's s1-eth2 interface.

Step 3. Modify the bandwidth of the link connecting the switch S1 and switch S2: on the same terminal, type the command below. This command sets the bandwidth to 10 Gbps on switch S1's s1-eth2 interface. The $\boxed{\texttt{tbf}}$ parameters are the following:

rate: 10gbitburst: 5,000,000limit: 15,000,000

sudo tc qdisc add dev s1-eth2 parent 1: handle 2: tbf rate 10gbit burst 5000000 limit 15000000

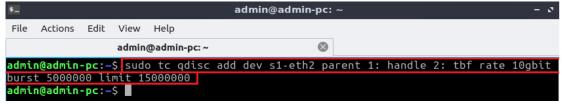


Figure 11. Limiting the bandwidth to 10 Gbps on switch S1's s1-eth2 interface.

Step 4. The user can now verify the rate limit configuration by using the liperf3 tool to measure throughput. To launch iPerf3 in server mode, run the command liperf3 -s in host h2's terminal:

```
iperf3 -s
```

```
"Host: h2" - 🗈 X
root@admin-pc:~# iperf3 -s
Server listening on 5201
```

Figure 12. Host h2 running iPerf3 as server.

Step 5. Now to launch iPerf3 in client mode again by running the command iperf3 - c 10.0.0.2 in host h1's terminal:

```
iperf3 -c 10.0.0.2
```

```
"Host: h1"
root@admin-pc:~# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 15] local 10.0.0.1 port 36164 connected to 10.0.0.2 port 5201
[ ID] Interval
                         Transfer
                                     Bitrate
                                                     Retr Cwnd
 15]
       0.00-1.00 sec 1.03 GBytes 8.87 Gbits/sec
                                                     7118
                                                              156 KBytes
 15]
        1.00-2.00
                         846 MBytes
                                     7.10 Gbits/sec
                                                     5787
                                                             60.8 KBytes
                   sec
 15]
       2.00-3.00
                   sec
                         894 MBytes
                                     7.50 Gbits/sec
                                                      6209
                                                             663 KBytes
 15]
       3.00-4.00
                   sec
                         872 MBytes
                                     7.32 Gbits/sec
                                                      6605
                                                              154 KBytes
 15]
       4.00-5.00
                   sec
                         890 MBytes
                                     7.47 Gbits/sec
                                                     6095
                                                             167 KBytes
 15]
       5.00-6.00
                   sec
                         900 MBytes
                                     7.55 Gbits/sec
                                                     6631
                                                             87.7 KBytes
                                     9.46 Gbits/sec
 15]
       6.00-7.00
                   sec 1.10 GBytes
                                                     7751
                                                             112 KBytes
       7.00-8.00
 15]
                   sec 1.07 GBytes
                                     9.22 Gbits/sec 8085
                                                             82.0 KBytes
 15]
       8.00-9.00
                   sec
                         880 MBytes
                                     7.38 Gbits/sec
                                                      7307
                                                             144 KBytes
[ 15]
       9.00-10.00
                   sec
                          956 MBytes
                                     8.02 Gbits/sec
                                                     6602
                                                             59.4 KBytes
 IDl Interval
                         Transfer
                                     Bitrate
                                                      Retr
                                                     68190
 15]
       0.00-10.00 sec 9.30 GBytes
                                     7.99 Gbits/sec
                                                                        sender
       0.00-10.04 sec 9.29 GBytes 7.95 Gbits/sec
 15]
                                                                      receiver
iperf Done.
root@admin-pc:~#
```

Figure 13. iPerf3 throughput test.

Note the measured throughput now is approximately 7.99 Gbps, which is different than the value assigned in the total rule (10 Gbps). In the next section, the test is repeated but using a higher MSS.

Step 6. In order to stop the server, press Ctrl+c in host h2's terminal. The user can see the throughput results in the server side too. The summarized data on the server is similar to that of the client side's and must be interpreted in the same way.

3 Modifying maximum transmission unit (MTU)

As explained previously, jumbo frames offer throughput improvements in networks incurring packet losses. In this section, the user will change the MTU of a network interface in Linux.

3.1 Identifying interface's current MTU

Step 1. To identify the MTU of a network interface of a device, the <u>lifconfig</u> is used. On host h1's terminal, type in the following command:

ifconfig

```
"Host: h1"
root@admin-pc:~# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
       inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
       inet6 fe80::d011:f8ff:fe04:lab2 prefixlen 64 scopeid 0x20<link>
       ether d2:11:f8:04:1a:b2 txqueuelen 1000 (Ethernet)
       RX packets 224882 bytes 14942450 (14.9 MB)
       RX errors 0 dropped 0 overruns 0 frame 0
       TX packets 294732 bytes 10095401018 (10.0 GB)
       TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
       inet 127.0.0.1 netmask 255.0.0.0
       inet6 ::1 prefixlen 128 scopeid 0x10<host>
       loop txqueuelen 1000 (Local Loopback)
       RX packets 0 bytes 0 (0.0 B)
       RX errors 0 dropped 0 overruns 0 frame 0
       TX packets 0 bytes 0 (0.0 B)
       TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
oot@admin-pc:~#
```

Figure 14. Identifying interface's MTU.

As shown in Figure 14, the interface *h1-eth0* has an MTU of 1500 bytes. The same steps can be performed on host h2's interface.

Step 2. In order to identify the MTU on the switches' interfaces, launch the Client's terminal located on the Desktop, and type in the following command:

ifconfig

```
admin@admin-pc: ~
File Actions Edit View Help
                admin@admin-pc: ~
                                               0
admin@admin-pc:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
       inet6 fe80::66c8:4b01:27cb:b43c prefixlen 64 scopeid 0x20<link>
       ether 00:50:56:ae:fb:5c txqueuelen 1000 (Ethernet)
       RX packets 0 bytes 0 (0.0 B)
       RX errors 0 dropped 0 overruns 0 frame 0 TX packets 1343 bytes 222919 (222.9 KB)
       TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
       device interrupt 19 base 0x2000
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
       inet 127.0.0.1 netmask 255.0.0.0
       inet6 :: 1 prefixlen 128 scopeid 0x10<host>
       loop txqueuelen 1000 (Local Loopback)
       RX packets 36167 bytes 2372795 (2.3 MB)
       RX errors 0 dropped 0 overruns 0 frame 0 TX packets 36167 bytes 2372795 (2.3 MB)
       TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
       ether 6a:b2:69:87:90:c9 txqueuelen 1000 (Ethernet)
       RX packets 1180301 bytes 54368305697 (54.3 GB)
       RX errors 0 dropped 0 overruns 0 frame 0 TX packets 1044079 bytes 69065295 (69.0 MB)
       TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> | mtu 1500
       inet6 fe80::34d1:60ff:fe13:2c23 prefixlen 64 scopeid 0x20<link>
       ether 36:d1:60:13:2c:23 txqueuelen 1000 (Ethernet)
       RX packets 1044043 bytes 69061330 (69.0 MB)
       RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1176682 bytes 54235115264 (54.2 GB)
       TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
inet6 fe80::bc27:86ff:fe2d:5d89 prefixlen 64 scopeid 0x20<link>
       ether be:27:86:2d:5d:89 txqueuelen 1000 (Ethernet)
       RX packets 1044008 bytes 69057435 (69.0 MB)
       RX errors 0 dropped 0 overruns 0 frame 0
       TX packets 1176718 bytes 54235119229 (54.2 GB)
       TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
s2-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> | mtu 1500 |
        inet6 fe80::108c:80ff:fef8:cbc prefixlen 64 scopeid 0x20<link>
       ether 12:8c:80:f8:0c:bc txqueuelen 1000 (Ethernet)
       RX packets 1176682 bytes 54235115264 (54.2 GB)
       RX errors 0 dropped 0 overruns 0 frame 0
       TX packets 1044043 bytes 69061330 (69.0 MB)
       TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
admin@admin-pc:~$
```

Figure 15. Identifying switches' interfaces' MTU.

Each switch in the topology has two interfaces: switch S1 has s1-eth1 and s1-eth2, switch S2 interfaces are s2-eth1 and s2-eth2. The MTU value on all interfaces are 1500 bytes.

3.2 Modifying MTU values on all interfaces

To modify the MTU of a network interface use the following command:

ifconfig <iface> mtu <bytes>

Step 1. To change the MTU to 9000 bytes, on host h1's terminal, type in the following command:

"Host: h1" - X X

root@admin-pc:~# ifconfig h1-eth0 mtu 9000
root@admin-pc:~#

Figure 17. Changing host h1's interface MTU.

Step 2. To change the MTU to 9000 bytes, on host h2's terminal, type in the following command:

ifconfig h2-eth0 mtu 9000

"Host: h2" - x x

root@admin-pc:~# ifconfig h2-eth0 mtu 9000
root@admin-pc:~#

Figure 18. Changing host h2's interface MTU.

Step 3. Similarly, the MTU values of switch S1 and switch S2's interfaces must be changed to 9000 bytes. In order to modify the MTU values, type the following command on the Client's terminal. When prompted for a password, type password and hit *Enter*.

sudo ifconfig s1-eth1 mtu 9000

sudo ifconfig s1-eth2 mtu 9000

sudo ifconfig s2-eth1 mtu 9000

sudo ifconfig s2-eth2 mtu 9000

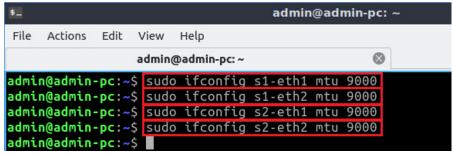


Figure 19. Changing MTU values on the switches.

Step 4. The user can now verify the effect of modifying the MTU values on the switches and the effect of MSS by using the liperf3 tool to measure throughput. To launch iPerf3 in server mode, run the command liperf3 -s in host h2's terminal:

```
iperf3 -s
```

```
"Host: h2" - 🗴 ×
root@admin-pc:~# iperf3 -s

Server listening on 5201
```

Figure 20. Host h2 running iPerf3 as server.

```
iperf3 -c 10.0.0.2 -M 9000
```

```
"Host: h1"
Connecting to host 10.0.0.2, port 5201
 15] local 10.0.0.1 port 36182 connected to 10.0.0.2 port 5201
                                  Bitrate
 ID] Interval
                                                 Retr Cwnd
                      Transfer
 151
      0.00-1.00 sec 1.17 GBytes 10.0 Gbits/sec 1399
                                                        2.03 MBytes
 151
       1.00-2.00 sec 1.16 GBytes 9.93 Gbits/sec
                                                1432
                                                         271 KBytes
                                                 1168
      2.00-3.00 sec
                       930 MBytes
 15]
                                  7.80 Gbits/sec
                                                         323 KBytes
 15]
       3.00-4.00
                      1.15 GBytes
                                  9.92 Gbits/sec
                                                 1341
                                                         201 KBytes
                  sec
 15]
      4.00-5.00 sec 1.16 GBytes
                                  9.93 Gbits/sec
                                                 1276
                                                         332 KBytes
 15]
      5.00-6.00 sec 1.14 GBytes
                                  9.81 Gbits/sec
                                                 1369
                                                         411 KBytes
      6.00-7.00 sec
 15]
                      935 MBytes
                                  7.84 Gbits/sec 1307
                                                         297 KBytes
 15]
       7.00-8.00 sec 1.16 GBytes
                                  9.93 Gbits/sec 1389
                                                         122 KBytes
 15]
       8.00-9.00
                  sec 1.16 GBytes
                                  9.93 Gbits/sec 1385
                                                         306 KBytes
       9.00-10.00 sec
                      1.14 GBytes
                                  9.83 Gbits/sec 1468
                                                         288 KBytes
 ID] Interval
                      Transfer
                                  Bitrate
                                                 Retr
 15]
      0.00-10.00 sec 11.1 GBytes 9.49 Gbits/sec 13534
                                                                  sender
 15]
      0.00-10.04 sec 11.0 GBytes 9.45 Gbits/sec
                                                                receiver
iperf Done.
root@admin-pc:~#
```

Figure 21. iPerf3 throughput test with a 9000 MSS value.

Notice the measured throughput now is approximately 10 Gbps, which is similar to the value assigned in the total rule (10 Gbps).

Step 6. In order to stop the server, press Ctrl+c in host h2's terminal. The user can see the throughput results in the server side too. The summarized data on the server is similar to that of the client side's and must be interpreted in the same way.

This concludes Lab 13. Stop the emulation and then exit out of MiniEdit.

References

1. Huh, Eui-Nam, and Hyunseung Choo, "Performance enhancement of TCP in high-speed networks," *Information Sciences* 178, no. 2 (2008), 352-362