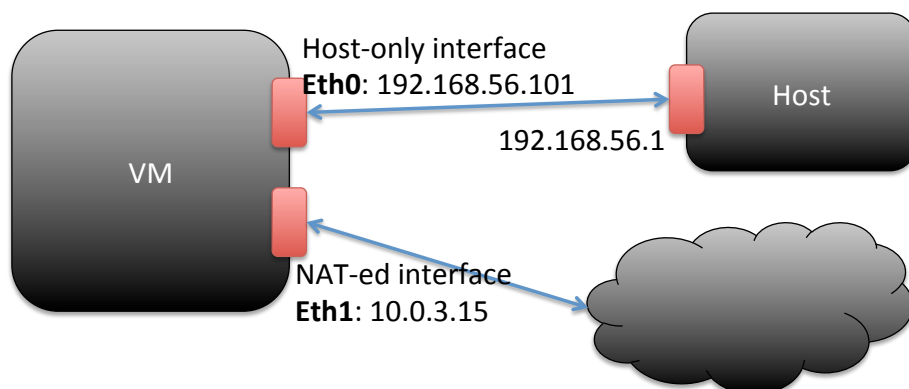


Hands-On VI

Transport Control Protocol

All practice within this hands-on are performed within our latest VM image (ubuntu **16.04**)
infoN-vm.ova



PART I

- Connection setup and release
- Connection reset
- Using wget

Observe Connection Setup and Release

1. Prepare wireshark and start capture data on the Interface that connect to the Internet (NAT-ed interface)
2. Run 'wget' to download just one small enough file. Ex:

```
$ wget https://iplab.naist.jp/class/infoN/2017/  
materials/2017-01-class-overview.pdf
```

SETUP

No.	Time	Source	Destination	Protocol	Length	Info
11	5.012836000	10.0.3.15	163.221.11.43	TCP	74	47915 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460
14	5.014944000	163.221.11.43	10.0.3.15	TCP	60	47915 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
15	5.014989000	10.0.3.15	163.221.11.43	TCP	54	47915 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
16	5.015155000	10.0.3.15	163.221.11.43	HTTP	219	GET /class/infoW/2015/materials/2015-01-class-over
17	5.015321000	163.221.11.43	10.0.3.15	TCP	60	47915 [ACK] Seq=1 Ack=166 Win=65535 Len=0
18	5.017271000	163.221.11.43	10.0.3.15	TCP	782	[TCP segment of a reconstructed flow]

▶ Frame 14: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ Ethernet II, Src: RealtekU 12:35:02 (52:54:00:12:35:02), Dst: CadmusCo 8d:de:b1 (08:00:27:8d:de:b1)
 ▶ Internet Protocol Version 4, Src: 163.221.11.43 (163.221.11.43), Dst: 10.0.3.15 (10.0.3.15)
 ▼ Transmission Control Protocol, Src Port: http (80), Dst Port: 47915 (47915), Seq: 0, Ack: 1, Len: 0
 Source port: http (80)
 Destination port: 47915 (47915)
 [Stream index: 0]
 Sequence number: 0 (relative sequence number)
 Acknowledgment number: 1 (relative ack number)

RELEASE

No.	Time	Source	Destination	Protocol	Length	Info
157	5.032108000	163.221.11.43	10.0.3.15	HTTP	3217	HTTP/1.1 200 OK (application/pdf)
158	5.032115000	10.0.3.15	163.221.11.43	TCP	54	47915 > http [ACK] Seq=166 Ack=198272 Win=65535 Le
159	5.032452000	10.0.3.15	163.221.11.43	TCP	54	47915 > http [FIN, ACK] Seq=166 Ack=198272 Win=655
160	5.032532000	163.221.11.43	10.0.3.15	TCP	60	47915 [ACK] Seq=198272 Ack=167 Win=65535 Le
161	5.033941000	163.221.11.43	10.0.3.15	TCP	60	47915 [FIN, ACK] Seq=198272 Ack=167 Win=655
162	5.033957000	10.0.3.15	163.221.11.43	TCP	54	47915 > http [ACK] Seq=167 Ack=198273 Win=42168 Le

▶ Frame 161: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ Ethernet II, Src: RealtekU 12:35:02 (52:54:00:12:35:02), Dst: CadmusCo 8d:de:b1 (08:00:27:8d:de:b1)
 ▶ Internet Protocol Version 4, Src: 163.221.11.43 (163.221.11.43), Dst: 10.0.3.15 (10.0.3.15)
 ▼ Transmission Control Protocol, Src Port: http (80), Dst Port: 47915 (47915), Seq: 198272, Ack: 167, Len: 0
 Source port: http (80)
 Destination port: 47915 (47915)
 [Stream index: 0]
 Sequence number: 198272 (relative sequence number)
 Acknowledgment number: 167 (relative ack number)

Simulate Connection Reset: Terminating Program

1. Prepare wireshark and capture data on the Interface that connect to the Internet (NAT-ed interface)
2. Run 'wget' to download one big enough file, so we have time to cut the download process in the middle. Ex:

```
$ wget http://download.virtualbox.org/virtualbox/5.1.22/VirtualBox-5.1.22-115126-OSX.dmg
```

3. Don't wait until it finish, in the middle of download process, press Ctrl+C

RESET by Terminating Program

o.	Time	Source	Destination	Protocol	Length	Info
852	8.531722000	184.25.56.42	10.0.3.15	TCP	60	http > 58646 [ACK] Seq=1680773 Ack=164 Win=65535 L
853	8.651418000	184.25.56.42	10.0.3.15	TCP	1428	[TCP segment of a reassembled PDU]
854	8.651451000	10.0.3.15	184.25.56.42	TCP	54	58646 > http [RST] Seq=164 Win=0 Len=0
855	8.651496000	184.25.56.42	10.0.3.15	TCP	1428	[TCP segment of a reassembled PDU]
856	8.651501000	10.0.3.15	184.25.56.42	TCP	54	58646 > http [RST] Seq=164 Win=0 Len=0
857	8.651582000	184.25.56.42	10.0.3.15	TCP	60	http > 58646 [RST, ACK] Seq=4100791295 Ack=164 Win

.... .. = Urgent: Not set
.... .. = Acknowledgment: Not set
.... .. = Push: Not set
▼.... .. = Reset: Set
▼[Expert Info (Chat/Sequence): Connection reset (RST)]
[Message: Connection reset (RST)]
[Severity level: Chat]
[Group: Sequence]
.... .. = Syn: Not set
.... .. = Fin: Not set

Simulate Connection Reset: Connecting Nonexistent Port

1. Prepare wireshark and capture data on the Interface that connect to the Internet (NAT-ed interface)
2. Telnet a remote server at its nonexistent port. Ex:

```
$ telnet sh.naist.jp 80
```

RESET by Terminating Program

No.	Time	Source	Destination	Protocol	Length	Info
8	5.002548000	10.0.3.15	163.221.8.12	DNS	71	Standard query 0x45f8 A sh.naist.jp
9	5.004778000	163.221.8.11	10.0.3.15	DNS	239	Standard query response 0x45f8 A 163.221.10.10
10	5.004886000	10.0.3.15	163.221.8.11	DNS	71	Standard query 0x7dc7 AAAA sh.naist.jp
11	5.005079000	163.221.8.12	10.0.3.15	DNS	195	Standard query response 0x45f8 A 163.221.10.10
12	5.005092000	10.0.3.15	163.221.8.12	ICMP	223	Destination unreachable (Port unreachable)
13	5.006317000	163.221.8.11	10.0.3.15	DNS	251	Standard query response 0x7dc7 AAAA 2001:200:16a:
14	5.006520000	10.0.3.15	163.221.10.10	TCP	74	45389 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460
15	5.007994000	163.221.10.10	10.0.3.15	TCP	60	http > 45389 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Transmission Control Protocol, Src Port: http (80), Dst Port: 45389 (45389), Seq: 1, Ack: 1, Len: 0

Source port: http (80)
Destination port: 45389 (45389)
[Stream index: 0]
Sequence number: 1 (relative sequence number)
Acknowledgment number: 1 (relative ack number)
Header length: 20 bytes

►Flags: 0x014 (RST, ACK)
Window size value: 0
[Calculated window size: 0]
[Window size scaling factor: -1 (unknown)]
►Checksum: 0x4f70 [validation disabled]
►[SEQ/ACK analysis]

ANY
QUESTIONS
?

PART II

- Observe packet loss
- Observe packet corruption
- Observe packet re-ordering
- Observe Nagle Algorithm
- Using scp
- Using iperf

iperf

- Install iperf3

```
$ sudo apt-get install iperf3
```

- iperf is a client-server application. For this practice we have set-up 5 server:

- 163.221.52.226:50001-50005

- *a server can only serve one connection at a time, that means if you can't connect to one server instance, please try the other

- Please also capture iperf's data using wireshark (start capturing on your NAT-ed interface before execute iperf)

iperf parameters

Parameters we use:

- **-c** : we will work as client
- **-l 10** : set the length of buffer. To better see how Nagle algorithm work, we should sending small packets (< 40 byte)
- **-N** : switch to disable TCP delay (Nagle algorithm)
- **-p [50001|50002|50003|50004|50005]** : port number. Use this to change between iperf's server instance

iperf with Nagle's

```
in1@in1-VirtualBox:~$ iperf3 -c 163.221.52.177 -l 10
Connecting to host 163.221.52.177, port 5201
[ 4] local 10.0.3.15 port 52526 connected to 163.221.52.177 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr   Cwnd
[ 4]  0.00-1.00   sec    2.46 MBytes  20.7 Mbits/sec    0    14.3 KBytes
[ 4]  1.00-2.00   sec    2.23 MBytes  18.7 Mbits/sec    0    14.3 KBytes
[ 4]  2.00-3.00   sec    2.19 MBytes  18.4 Mbits/sec    0    14.3 KBytes
[ 4]  3.00-4.00   sec    2.27 MBytes  19.1 Mbits/sec    0    14.3 KBytes
[ 4]  4.00-5.00   sec    2.23 MBytes  18.7 Mbits/sec    0    14.3 KBytes
[ 4]  5.00-6.00   sec    2.19 MBytes  18.4 Mbits/sec    0    14.3 KBytes
[ 4]  6.00-7.00   sec    2.29 MBytes  19.2 Mbits/sec    0    14.3 KBytes
[ 4]  7.00-8.00   sec    2.20 MBytes  18.5 Mbits/sec    0    14.3 KBytes
[ 4]  8.00-9.00   sec    2.22 MBytes  18.6 Mbits/sec    0    14.3 KBytes
[ 4]  9.00-10.00  sec    2.28 MBytes  19.1 Mbits/sec    0    17.1 KBytes
- - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[ 4]  0.00-10.00  sec   22.6 MBytes  18.9 Mbits/sec    0             sender
[ 4]  0.00-10.00  sec   22.2 MBytes  18.6 Mbits/sec                  receiver
```

iperf without Nagle's

```
in1@in1-VirtualBox:~$ iperf3 -c 163.221.52.177 -l 10 -N
Connecting to host 163.221.52.177, port 5201
[ 4] local 10.0.3.15 port 52528 connected to 163.221.52.177 port 5201
[ ID] Interval            Transfer    Bandwidth    Retr  Cwnd
[ 4] 0.00-1.00 sec      186 KBytes  1.52 Mbits/sec    0   17.1 KBytes
[ 4] 1.00-2.00 sec      178 KBytes  1.46 Mbits/sec    0   28.5 KBytes
[ 4] 2.00-3.00 sec      175 KBytes  1.43 Mbits/sec    0   28.5 KBytes
[ 4] 3.00-4.00 sec      181 KBytes  1.48 Mbits/sec    0   29.9 KBytes
[ 4] 4.00-5.00 sec      170 KBytes  1.39 Mbits/sec    0   29.9 KBytes
[ 4] 5.00-6.00 sec      170 KBytes  1.39 Mbits/sec    0   29.9 KBytes
[ 4] 6.00-7.00 sec      177 KBytes  1.45 Mbits/sec    0   29.9 KBytes
[ 4] 7.00-8.00 sec      170 KBytes  1.40 Mbits/sec    0   29.9 KBytes
[ 4] 8.00-9.00 sec      174 KBytes  1.42 Mbits/sec    0   29.9 KBytes
[ 4] 9.00-10.00 sec     172 KBytes  1.41 Mbits/sec    0   29.9 KBytes
- - - - -
[ ID] Interval            Transfer    Bandwidth    Retr
[ 4] 0.00-10.00 sec     1.71 MBytes  1.44 Mbits/sec    0
[ 4] 0.00-10.00 sec     1.71 MBytes  1.44 Mbits/sec    0
sender
receiver
```

What is the difference?
What is that means?
How is that happened?



Wireshark: iperf with Nagle's

No.	Time	Source	Destination	Protocol	Length	Info
31233	2.535182	163.221.52.177	10.0.3.15	TCP	60	targus-getdata1 > 52534 [ACK] Seq=1 Ack=3468188 Wi
31234	2.535188	10.0.3.15	163.221.52.177	TCP	484	52534 > targus-getdata1 [PSH, ACK] Seq=3468188 Ack
31235	2.535291	163.221.52.177	10.0.3.15	TCP	60	targus-getdata1 > 52534 [ACK] Seq=1 Ack=3468618 Wi
31236	2.535297	10.0.3.15	163.221.52.177	TCP	324	52534 > targus-getdata1 [PSH, ACK] Seq=3468618 Ack
31237	2.535407	163.221.52.177	10.0.3.15	TCP	60	targus-getdata1 > 52534 [ACK] Seq=1 Ack=3468888 Wi
31238	2.535438	10.0.3.15	163.221.52.177	TCP	384	52534 > targus-getdata1 [PSH, ACK] Seq=3468888 Ack
31239	2.535555	163.221.52.177	10.0.3.15	TCP	60	targus-getdata1 > 52534 [ACK] Seq=1 Ack=3469218 Wi
31240	2.535583	10.0.3.15	163.221.52.177	TCP	284	52534 > targus-getdata1 [PSH, ACK] Seq=3469218 Ack

► Frame 31232: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)
► Ethernet II, Src: CadmusCo_8d:de:b1 (08:00:27:8d:de:b1), Dst: RealtekU 12:35:02 (52:54:00:12:35:02)
► Internet Protocol Version 4, Src: 10.0.3.15 (10.0.3.15), Dst: 163.221.52.177 (163.221.52.177)
▼ Transmission Control Protocol, Src Port: 52534 (52534), Dst Port: targus-getdata1 (5201), Seq: 3468028, Ack: 1, Len: 160
Source port: 52534 (52534)
Destination port: targus-getdata1 (5201)
[Stream index: 1]
Sequence number: 3468028 (relative sequence number)
[Next sequence number: 3468188 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Header length: 20 bytes
► Flags: 0x018 (PSH, ACK)
Window size value: 29200

Statistic in Protocol Hierarchy with Nagle's

nagle_l10.pcap [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.3.15	163.221.52.177	TCP	74	52533 > targus-getdata1 [SYN] Seq=0 Win=29200 Len=

Wireshark: Protocol Hierarchy Statistics

Display filter: none

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
▼ Frame	100.00 %	123132	100.00 %	21011982	10.182	0	0	0.000
▼ Ethernet	100.00 %	123132	100.00 %	21011982	10.182	0	0	0.000
▼ Internet Protocol Version 4	100.00 %	123132	100.00 %	21011982	10.182	0	0	0.000
▼ Transmission Control Protocol	100.00 %	123132	100.00 %	21011982	10.182	61573	3694336	1.790
Data	49.99 %	61558	82.42 %	17317586	8.392	61558	17317586	8.392
Malformed Packet	0.00 %	1	0.00 %	60	0.000	1	60	0.000

Help Close

Proportion of data within one frame

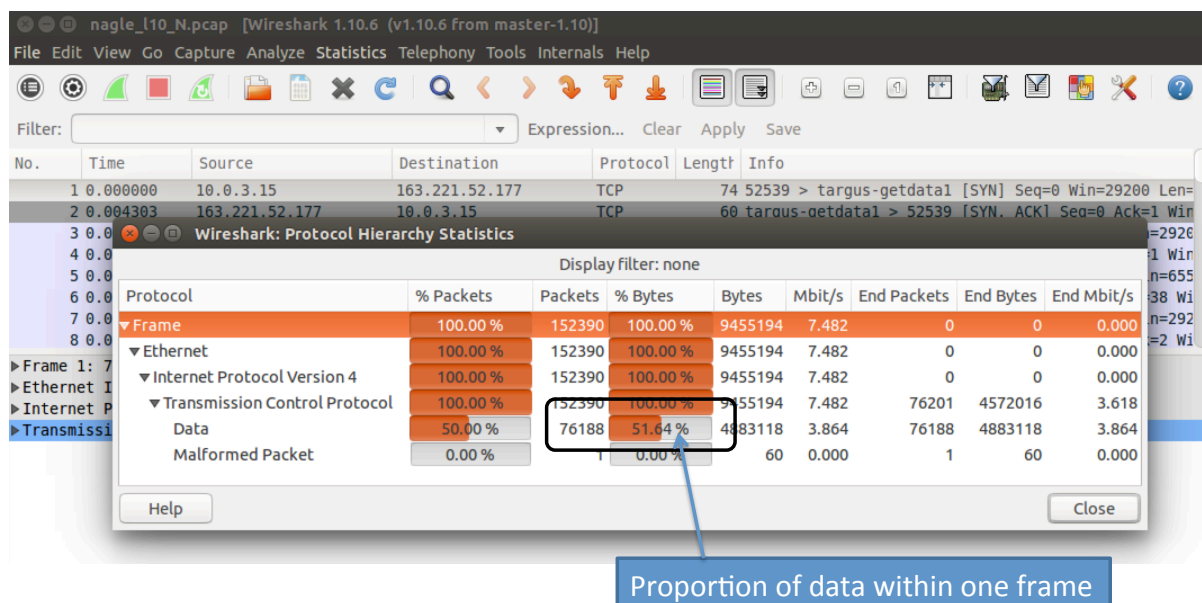
Wireshark: iperf without Nagle's

No.	Time	Source	Destination	Protocol	Length	Info
151415	9.878322000	10.0.3.15	163.221.52.177	TCP	64	52536 > targus-getdata1 [PSH, ACK] Seq=764038 Ack=
151416	9.878350000	163.221.52.177	10.0.3.15	TCP	60	targus-getdata1 > 52536 [ACK] Seq=1 Ack=764038 Win
151417	9.878351000	163.221.52.177	10.0.3.15	TCP	60	targus-getdata1 > 52536 [ACK] Seq=1 Ack=764048 Win
151418	9.878368000	10.0.3.15	163.221.52.177	TCP	64	52536 > targus-getdata1 [PSH, ACK] Seq=764048 Ack=
151419	9.878404000	10.0.3.15	163.221.52.177	TCP	64	52536 > targus-getdata1 [PSH, ACK] Seq=764058 Ack=
151420	9.878449000	163.221.52.177	10.0.3.15	TCP	60	targus-getdata1 > 52536 [ACK] Seq=1 Ack=764058 Win
151421	9.878451000	163.221.52.177	10.0.3.15	TCP	60	targus-getdata1 > 52536 [ACK] Seq=1 Ack=764068 Win
151422	9.878467000	10.0.3.15	163.221.52.177	TCP	64	52536 > targus-getdata1 [PSH, ACK] Seq=764068 Ack=

Destination port: targus-getdata1 (5201)
[Stream index: 1]
Sequence number: 764048 (relative sequence number)
[Next sequence number: 764058 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Header length: 20 bytes
►Flags: 0x018 (PSH, ACK)
Window size value: 29200
[Calculated window size: 29200]
[Window size scaling factor: -2 (no window scaling used)]
►Checksum: 0xe5c1 [validation disabled]
►[SEQ/ACK analysis]
►Data (10 bytes)

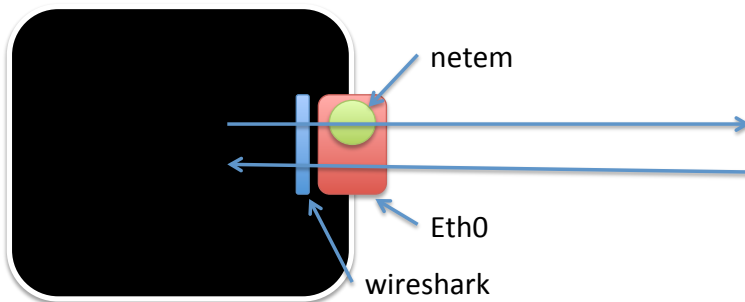
You can also observe that **iperf data without Nagle** are always encapsulated within its own packet, while some **iperf data with Nagle** are concatenated with other data

Statistic in Protocol Hierarchy without Nagle's

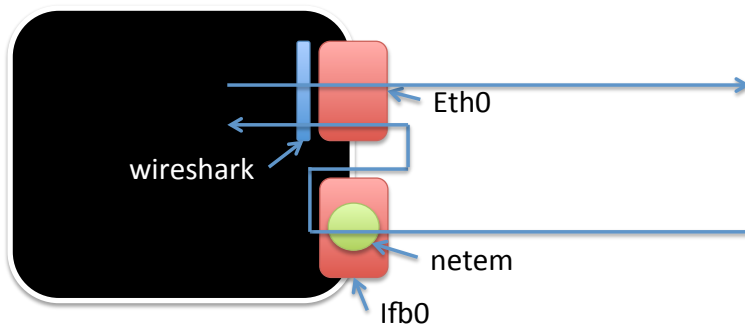


ANY
QUESTIONS
?

Applying Netem for Egress Data



Applying Netem for Ingress Data



Applying netem for ingress data

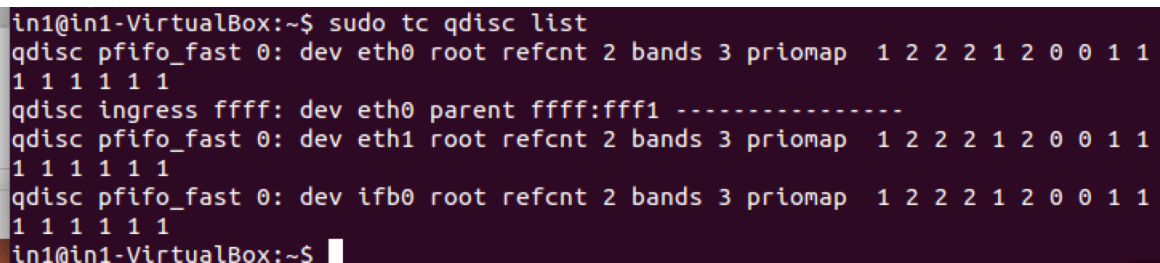
1. Add an Intermediate Functional Block pseudo-device (ifb)

```
$ sudo modprobe ifb  
$ sudo ip link set dev ifb0 up
```

2. Redirect incoming to eth0 through ifb0

```
$ sudo tc qdisc add dev eth0 ingress  
  
$ sudo tc filter add dev eth0 parent ffff: protocol  
ip u32 match u32 0 0 flowid 1:1 action mirrored egress  
redirect dev ifb0
```

List all queue discipline after adding ifb interface



```
in1@in1-VirtualBox:~$ sudo tc qdisc list  
qdisc pfifo_fast 0: dev eth0 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1  
1 1 1 1 1 1  
qdisc ingress ffff: dev eth0 parent ffff:fff1 -----  
qdisc pfifo_fast 0: dev eth1 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1  
1 1 1 1 1 1  
qdisc pfifo_fast 0: dev ifb0 root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1  
1 1 1 1 1 1  
in1@in1-VirtualBox:~$
```

Packet Loss

1. Emulate packet loss on your ifb interface, run this command:

```
$ sudo tc qdisc add dev ifb0 root netem loss 10%
```

```
$ sudo apt-get install openssh-server
```

2. Prepare wireshark and capture data on your host-only interface (eth0)

3. Run 'scp' on your host machine to send a file to your VM. Ex:

```
$ scp capture1.pcap in1@192.168.56.101:~/
```

Change 'capture1.pcap' with any file you would like to use

LOSS PACKET (CHARACTERIZE BY DUPLICATE ACKNOWLEDGMENT)

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
3297	5.457380	192.168.56.1	192.168.56.101	TCP	78	[TCP Dup ACK 3294#1] 51325 > ssh [ACK] Seq=4137370 A
3299	5.458084	192.168.56.1	192.168.56.101	TCP	78	[TCP Dup ACK 3294#2] 51325 > ssh [ACK] Seq=4137370 A
3301	5.458801	192.168.56.1	192.168.56.101	TCP	78	[TCP Dup ACK 3294#3] 51325 > ssh [ACK] Seq=4137370 A
3720	5.474493	192.168.56.1	192.168.56.101	TCP	78	[TCP Dup ACK 3718#1] 51325 > ssh [ACK] Seq=4687906 A
3722	5.474991	192.168.56.1	192.168.56.101	TCP	78	[TCP Dup ACK 3718#2] 51325 > ssh [ACK] Seq=4687906 A
3746	5.485895	192.168.56.1	192.168.56.101	TCP	78	[TCP Dup ACK 3744#1] 51325 > ssh [ACK] Seq=4687906 A
3758	5.896983	192.168.56.1	192.168.56.101	TCP	66	[TCP Dup ACK 3756#1] 51325 > ssh [ACK] Seq=4688047 A

[Calculated window size: 1510/2]
[Window size scaling factor: 16]
▶Checksum: 0xd5ec [validation disabled]
▶Options: (24 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, No-Operation (NOP), No-Operation (NOP), SACK
▼[SEQ/ACK analysis]
▼[TCP Analysis Flags]
[This is a TCP duplicate ack]
[Duplicate ACK #: 1]
▼[Duplicate to the ACK in frame: 3294]
▼[Expert Info (Note/Sequence): Duplicate ACK (#1)]
[Message: Duplicate ACK (#1)]
[Severity level: Note]
[Group: Sequence]

Packet Re-Ordering

1. Restore default qdisc on your ifb interface:

```
$ sudo tc qdisc del dev ifb0 root
```
2. Emulate packet re-ordering on your ifb interface, run this command:

```
$ sudo tc qdisc add dev ifb0 root netem delay 10ms reorder 25% 50%
```
3. Prepare wireshark and capture data on your host-only interface (eth0)
4. Run 'scp' on your host machine to send a file to your VM. Ex:

```
$ scp capture1.pcap in1@192.168.56.101:~/
Change 'capture1.pcap' with any file you would like to use
```

OUT-OF-ORDER PACKET

Filter: tcp.analysis.out_of_order

No.	Time	Source	Destination	Protocol	Length	Info
1800	6.036641000	192.168.56.101	192.168.56.1	SSHv2	102	[TCP Out-Of-Order] Encrypted response packet len=3
1848	6.036942000	192.168.56.101	192.168.56.1	SSHv2	102	[TCP Out-Of-Order] Encrypted response packet len=3
1850	6.037585000	192.168.56.101	192.168.56.1	SSHv2	102	[TCP Out-Of-Order] Encrypted response packet len=3
2365	6.061395000	192.168.56.101	192.168.56.1	SSHv2	102	[TCP Out-Of-Order] Encrypted response packet len=3
2524	6.061989000	192.168.56.101	192.168.56.1	SSHv2	102	[TCP Out-Of-Order] Encrypted response packet len=3
2595	6.063106000	192.168.56.101	192.168.56.1	SSHv2	102	[TCP Out-Of-Order] Encrypted response packet len=3
3091	6.077230000	192.168.56.101	192.168.56.1	SSHv2	102	[TCP Out-Of-Order] Encrypted response packet len=3
3139	6.077770000	192.168.56.101	192.168.56.1	SSHv2	102	[TCP Out-Of-Order] Encrypted response packet len=3

[Calculated window size: 1252992]

[Window size scaling factor: 128]

▶Checksum: 0xf201 [validation disabled]

▶Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps

▼[SEQ/ACK analysis]

[Bytes in flight: 36]

▼[TCP Analysis Flags]

▼[This frame is a (suspected) out-of-order segment]

▼[Expert Info (Warn/Sequence): Out-Of-Order segment]

Packet Corruption

1. Make sure to enable checksum validation for TCP in your wireshark

edit > preferences > protocols > TCP -> enable 'Validate the TCP checksum if possible'

2. Disable your interface's checksum offloading feature. Failed to do this and your wireshark analysis will be flooded by 'false' checksum error.

```
$ sudo ethtool -K enp0s3 rx off tx off sg off gro  
off gso off
```

Packet Corruption

3. Restore default qdisc:

```
$ sudo tc qdisc del dev ifb0 root
```

4. Emulate packet corruption on your network interface, run this command:

```
$ sudo tc qdisc add dev enp0s3 root netem corrupt  
10%
```

Replace **enp0s3** with your host-only interface

5. Prepare wireshark and capture data on your host-only interface
6. Run 'scp' on your host machine to send a file to your VM. Ex:

```
$ scp capture1.pcap in1@192.168.56.101:~/  
Change 'capture1.pcap' with any file you would like to use
```

CORRUPT

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	0a:00:27:00:00:00	Broadcast	ARP	60	Who has 192.168.56.101? Tell 192.168.56.1
2	0.000019000	CadmusCo ab:06:dc	0a:00:27:00:00:00	ARP	42	192.168.56.101 is at 08:00:27:ab:06:dc
3	0.000094000	192.168.56.1	192.168.56.101	TCP	78	51314 > ssh [SYN] Seq=0 Win=65535 Len=0 MSS=1460 W
4	0.000111000	192.168.56.101	192.168.56.1	TCP	74	ssh > 51314 [SYN, ACK] Seq=0 Ack=1 Win=28960 [TCP
5	0.998917000	192.168.56.101	192.168.56.1	TCP	74	[TCP Retransmission] ssh > 51314 [SYN, ACK] Seq=0
6	0.999166000	192.168.56.1	192.168.56.101	TCP	66	51314 > ssh [ACK] Seq=1 Ack=1 Win=8235 Len=0 TSval
7	0.999422000	192.168.56.1	192.168.56.101	SSHv2	87	Encrypted request packet len=21
8	0.999436000	192.168.56.101	192.168.56.1	TCP	66	ssh > 51314 [ACK] Seq=1 Ack=22 Win=227 Len=0 TSval
Transmission Control Protocol, Src Port: ssh (22), Dst Port: 51314 (51314), Seq: 0, Ack: 1, Len: 0						
Source port: ssh (22)						
Destination port: 51314 (51314)						
[Stream index: 0]						
Sequence number: 0 (relative sequence number)						
Acknowledgment number: 1 (relative ack number)						
Header length: 40 bytes						
Flags: 0x012 (SYN, ACK)						
Window size value: 28960						
[Calculated window size: 28960]						
Checksum: 0x0ec4 [incorrect, should be 0xfec3 (maybe caused by "TCP checksum offload"?)]						
Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps						

ANY
QUESTIONS
?