# Routerlab

**Summer semester 2018**

Worksheet 3
Group 08

Valentin Franck, 364066
Nikhil Singh, 387694

Pages: 23

Submission Date: May 28, 2018

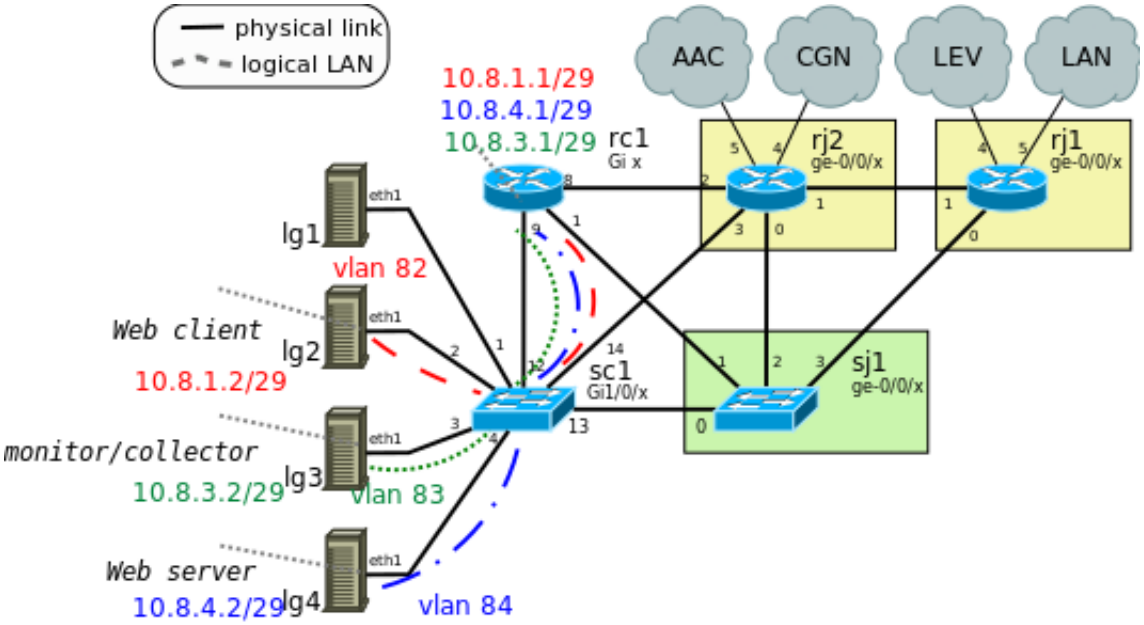


Figure 1: The IPv4 topology.

# Question 1

## 1a

See Figure 1.

## 1b

We also assigned IPv6 addresses and provide the traceroute from web client to web server:

```
root@group08-lg2:~# traceroute 10.8.4.2
traceroute to 10.8.4.2 (10.8.4.2), 30 hops max, 60 byte packets
 1  10.8.1.1 (10.8.1.1)  0.772 ms  0.938 ms  0.905 ms
 2  10.8.4.2 (10.8.4.2)  1.162 ms  1.133 ms  1.089 ms

root@group08-lg2:~# traceroute fc00:470:525b:f804::2
traceroute to fc00:470:525b:f804::2 (fc00:470:525b:f804::2), 30 hops
    max, 80 byte packets
 1  fc00:470:525b:f801::1 (fc00:470:525b:f801::1)  0.841 ms  0.784 ms
     0.770 ms
 2  fc00:470:525b:f804::2 (fc00:470:525b:f804::2)  1.289 ms  1.243 ms
     1.201 ms
```

The configuration:
SC1:

```
interface GigabitEthernet1/0/2
 description lg2
 switchport access vlan 82
 switchport mode access
!

interface GigabitEthernet1/0/3
 description lg3
```

```
 switchport access vlan 83
 switchport mode access
!

interface GigabitEthernet1/0/4
 description lg4
 switchport access vlan 84
 switchport mode access
!

interface GigabitEthernet1/0/12
 description lev-rc1
 switchport trunk allowed vlan 82-84
 switchport mode trunk
!
```

RC1:

```
interface GigabitEthernet9
 description lev-sc1
 no ip address
 duplex auto
 speed auto
!
interface GigabitEthernet9.82
 encapsulation dot1Q 82
 ip address 10.8.1.1 255.255.255.248
 ipv6 address FC00:470:525B:F801::1/64
 no cdp enable
!
interface GigabitEthernet9.83
 encapsulation dot1Q 83
 ip address 10.8.3.1 255.255.255.248
 ipv6 address FC00:470:525B:F803::1/64
 no cdp enable
!
interface GigabitEthernet9.84
 encapsulation dot1Q 84
 ip address 10.8.4.1 255.255.255.248
 ipv6 address FC00:470:525B:F804::1/64
 no cdp enable
!
```

LG2:

```
root@group08-lg2:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
     inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:20 brd ff:ff:ff:ff:ff:ff
     inet 172.16.0.209/20 brd 172.16.15.255 scope global eth0
        valid_lft forever preferred_lft forever
```

```
    inet6 fe80::216:3eff:feaf:820/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
   state UP group default qlen 1000
    link/ether 00:16:3e:af:08:21 brd ff:ff:ff:ff:ff:ff
    inet 10.8.1.2/29 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fc00:470:525b:f801:216:3eff:feaf:821/64 scope global
       mngtmpaddr dynamic
       valid_lft 2491291sec preferred_lft 504091sec
    inet6 fc00:470:525b:f801::2/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feaf:821/64 scope link
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
   default qlen 1000
    link/ether 00:16:3e:af:08:22 brd ff:ff:ff:ff:ff:ff

root@group08-lg2:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
    Iface
10.8.0.0        10.8.1.1        255.255.255.248 UG    0      0        0
    eth1
10.8.1.0        0.0.0.0         255.255.255.248 U     0      0        0
    eth1
10.8.2.0        10.8.1.1        255.255.255.248 UG    0      0        0
    eth1
10.8.3.0        10.8.1.1        255.255.255.248 UG    0      0        0
    eth1
10.8.4.0        10.8.1.1        255.255.255.248 UG    0      0        0
    eth1
172.16.0.0      0.0.0.0         255.255.240.0   U     0      0        0
    eth0
172.16.255.0    lion.routerlab  255.255.255.0   UG    0      0        0
    eth0
```

LG3

```
root@group08-lg3:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
   group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
   state UP group default qlen 1000
    link/ether 00:16:3e:af:08:30 brd ff:ff:ff:ff:ff:ff
    inet 172.16.0.210/20 brd 172.16.15.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feaf:830/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
   state UP group default qlen 1000
    link/ether 00:16:3e:af:08:31 brd ff:ff:ff:ff:ff:ff
```

```
    inet 10.8.3.2/29 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fc00:470:525b:f803::2/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fc00:470:525b:f803:216:3eff:feaf:831/64 scope global
       mngtmpaddr dynamic
       valid_lft 2591957sec preferred_lft 604757sec
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
   default qlen 1000
    link/ether 00:16:3e:af:08:32 brd ff:ff:ff:ff:ff:ff

root@group08-lg3:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
    Iface
10.8.1.0        10.8.3.1        255.255.255.248 UG    0      0        0
    eth1
10.8.2.0        10.8.3.1        255.255.255.248 UG    0      0        0
    eth1
10.8.3.0        0.0.0.0         255.255.255.248 U     0      0        0
    eth1
10.8.4.0        10.8.3.1        255.255.255.248 UG    0      0        0
    eth1
172.16.0.0      0.0.0.0         255.255.240.0   U     0      0        0
    eth0
172.16.255.0    lion.routerlab  255.255.255.0   UG    0      0        0
    eth0
```

LG4

```
root@group08-lg4:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
   group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
   state UP group default qlen 1000
    link/ether 00:16:3e:af:08:40 brd ff:ff:ff:ff:ff:ff
    inet 172.16.0.211/20 brd 172.16.15.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feaf:840/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
   state UP group default qlen 1000
    link/ether 00:16:3e:af:08:41 brd ff:ff:ff:ff:ff:ff
    inet 10.8.4.2/29 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fc00:470:525b:f804::2/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feaf:841/64 scope link
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
   default qlen 1000
    link/ether 00:16:3e:af:08:42 brd ff:ff:ff:ff:ff:ff
```

| TCP Bandwidth | MSS(Client) | MSS(Server) | TCP window size(Client) | TCP window size(Server) |
|---|---|---|---|---|
| 22.1 Gbit/s | 65483 bytes | 21888 bytes | 2.50 MByte | 85.3 KByte |
| 22.0 Gbit/s | 65483 bytes | 21888 bytes | 2.50 MByte | 85.3 KByte |
| 21.8 Gbit/s | 65483 bytes | 21888 bytes | 2.50 MByte | 85.3 KByte |
| 22.1 Gbit/s | 65483 bytes | 21888 bytes | 2.50 MByte | 85.3 KByte |
| 22.5 Gbit/s | 65483 bytes | 21888 bytes | 2.50 MByte | 85.3 KByte |

Table 1: Iperf server and client running on the same machine.

```
root@group08-lg4:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
    Iface
10.8.1.0        10.8.4.1        255.255.255.248 UG    0      0        0
    eth1
10.8.2.0        10.8.4.1        255.255.255.248 UG    0      0        0
    eth1
10.8.3.0        10.8.4.1        255.255.255.248 UG    0      0        0
    eth1
10.8.4.0        0.0.0.0         255.255.255.248 U     0      0        0
    eth1
172.16.0.0      0.0.0.0         255.255.240.0   U     0      0        0
    eth0
172.16.255.0    lion.routerlab  255.255.255.0   UG    0      0        0
    eth0
```

# Question 2

## 2a

We used screen to launch the iperf server and have it run in background while launching the client. Here is the output of the server and the client, used to create Table 1.

```
root@group08-lg2:~# iperf -s -m
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 57121
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  25.8 GBytes  22.1 Gbits/sec
[  4] MSS size 21888 bytes (MTU 21928 bytes, unknown interface)
[  5] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 57122
[  5]  0.0-10.0 sec  25.6 GBytes  22.0 Gbits/sec
[  5] MSS size 21888 bytes (MTU 21928 bytes, unknown interface)
[  4] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 57123
[  4]  0.0-10.0 sec  25.3 GBytes  21.8 Gbits/sec
[  4] MSS size 21888 bytes (MTU 21928 bytes, unknown interface)
[  5] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 57124
[  5]  0.0-10.0 sec  25.8 GBytes  22.1 Gbits/sec
[  5] MSS size 21888 bytes (MTU 21928 bytes, unknown interface)
[  4] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 57125
[  4]  0.0-10.0 sec  26.1 GBytes  22.4 Gbits/sec
[  4] MSS size 21888 bytes (MTU 21928 bytes, unknown interface)
```

```
root@group08-lg2:~# iperf -c localhost -m
------------------------------------------------------------
Client connecting to localhost, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 57121 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  25.8 GBytes  22.1 Gbits/sec
[  3] MSS size 65483 bytes (MTU 65523 bytes, unknown interface)
root@group08-lg2:~# iperf -c localhost -m
------------------------------------------------------------
Client connecting to localhost, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 57122 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  25.6 GBytes  22.0 Gbits/sec
[  3] MSS size 65483 bytes (MTU 65523 bytes, unknown interface)
root@group08-lg2:~# iperf -c localhost -m
------------------------------------------------------------
Client connecting to localhost, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 57123 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  25.3 GBytes  21.8 Gbits/sec
[  3] MSS size 65483 bytes (MTU 65523 bytes, unknown interface)
root@group08-lg2:~# iperf -c localhost -m
------------------------------------------------------------
Client connecting to localhost, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 57124 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  25.8 GBytes  22.1 Gbits/sec
[  3] MSS size 65483 bytes (MTU 65523 bytes, unknown interface)
root@group08-lg2:~# iperf -c localhost -m
------------------------------------------------------------
Client connecting to localhost, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 57125 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  26.1 GBytes  22.5 Gbits/sec
[  3] MSS size 65483 bytes (MTU 65523 bytes, unknown interface)
```

### 2b

Here is the output of the server and the client, used to create Table 2.

```
root@group08-lg4:~# iperf -s -m
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.8.4.2 port 5001 connected with 10.8.1.2 port 44988
```

| TCP Bandwidth (C.) | Bandwidth (S.) | MSS | TCP window size (C.) | window size (S.) |
|---|---|---|---|---|
| 706 Mbits/sec | 705 Mbits/sec | 1448 bytes | 85.0 KByte | 85.3 KByte |
| 642 Mbits/sec | 642 Mbits/sec | 1448 bytes | 85.0 KByte | 85.3 KByte |
| 896 Mbits/sec | 894 Mbits/sec | 1448 bytes | 85.0 KByte | 85.3 KByte |
| 676 Mbits/sec | 676 Mbits/sec | 1448 bytes | 85.0 KByte | 85.3 KByte |
| 778 Mbits/sec | 776 Mbits/sec | 1448 bytes | 85.0 KByte | 85.3 KByte |
| 626 Mbits/sec | 625 Mbits/sec | 1448 bytes | 85.0 KByte | 85.3 KByte |
| 633 Mbits/sec | 632 Mbits/sec | 1448 bytes | 85.0 KByte | 85.3 KByte |

Table 2: Iperf server and client running on LG2 and LG4 respectively.

```
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec   842 MBytes   705 Mbits/sec
[  4] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
[  5] local 10.8.4.2 port 5001 connected with 10.8.1.2 port 44989
[  5]  0.0-10.0 sec   766 MBytes   642 Mbits/sec
[  5] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
[  4] local 10.8.4.2 port 5001 connected with 10.8.1.2 port 44990
[  4]  0.0-10.0 sec  1.04 GBytes   894 Mbits/sec
[  4] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
[  5] local 10.8.4.2 port 5001 connected with 10.8.1.2 port 44991
[  5]  0.0-10.0 sec   806 MBytes   676 Mbits/sec
[  5] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
[  4] local 10.8.4.2 port 5001 connected with 10.8.1.2 port 44992
[  4]  0.0-10.0 sec   928 MBytes   776 Mbits/sec
[  4] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
[  5] local 10.8.4.2 port 5001 connected with 10.8.1.2 port 44993
[  5]  0.0-10.0 sec   746 MBytes   625 Mbits/sec
[  5] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
[  4] local 10.8.4.2 port 5001 connected with 10.8.1.2 port 44994
[  4]  0.0-10.0 sec   754 MBytes   632 Mbits/sec
[  4] MSS size 1448 bytes (MTU 1500 bytes, ethernet)

root@group08-lg2:~# iperf -c 10.8.4.2 -m
------------------------------------------------------------
Client connecting to 10.8.4.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.8.1.2 port 44988 connected with 10.8.4.2 port 5001

[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   842 MBytes   706 Mbits/sec
[  3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
root@group08-lg2:~#
root@group08-lg2:~# iperf -c 10.8.4.2 -m
------------------------------------------------------------
Client connecting to 10.8.4.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.8.1.2 port 44989 connected with 10.8.4.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   766 MBytes   642 Mbits/sec
[  3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
root@group08-lg2:~# iperf -c 10.8.4.2 -m
------------------------------------------------------------
Client connecting to 10.8.4.2, TCP port 5001
```

```
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.8.1.2 port 44990 connected with 10.8.4.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.04 GBytes   896 Mbits/sec
[  3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
root@group08-lg2:~# iperf -c 10.8.4.2 -m
------------------------------------------------------------
Client connecting to 10.8.4.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.8.1.2 port 44991 connected with 10.8.4.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   806 MBytes   676 Mbits/sec
[  3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
root@group08-lg2:~# iperf -c 10.8.4.2 -m
------------------------------------------------------------
Client connecting to 10.8.4.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.8.1.2 port 44992 connected with 10.8.4.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   928 MBytes   778 Mbits/sec
[  3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
root@group08-lg2:~# iperf -c 10.8.4.2 -m
------------------------------------------------------------
Client connecting to 10.8.4.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.8.1.2 port 44993 connected with 10.8.4.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   746 MBytes   626 Mbits/sec
[  3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
root@group08-lg2:~# iperf -c 10.8.4.2 -m
------------------------------------------------------------
Client connecting to 10.8.4.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.8.1.2 port 44994 connected with 10.8.4.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   754 MBytes   633 Mbits/sec
[  3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
```

### 2c

Yes we observe the differences in the values from Q2(a) and (b). The parameters that have changed are TCP Bandwidth, MSS and TCP Window size. TCP Bandwidth in Q2(a) was the same in both client and server side which is way higher than Q2(b) where TCP Bandwidth is minutely less in some cases on the server side than the client side. We observed different MSS on client and server side in Q2(a). On the client side it is approximately three times more than the MSS of server which is also more than 10 times the MSS in Q2(b). TCP Window size on the server is by default same on both Q2(a) and Q2(b). However, TCP window size of the client side in Q2(a) is 2.50 MByte which is quite larger than Q2(b) which is 85.0 KByte.

The reason for these changes is that in the latter case the data actually has to be transmitted over the network, causing the typical delays (transmission, propagatation, processing and queuing delay). In the first case it was just sent to the loopback interface, which is a virtual interface

emulated by the kernel and therefore does not have the physical limitations of a real network.

### 2d

Nagle's Algorithm is used to reduce the overhead in TCP by merging various smaller packets into one larger packet. This algorithm might not be suitable for interactive real time applications, e.g. multiplayer games or application layer protocols like Telnet, which often only expects a single byte (one key stroke). If this was delayed by Nagle's algorithm, it would affect the user experience and really slow down the protocol. So if one wants to simulate such environments, where it makes sense to immediately send even small packets, one needs to disable Nagle's algorithm.

Bidirectional measurements make most sense, when data really is sent in both directions. The current client-server architecture of many distributed systems results in very asymmetric traffic. So the client usually sends very little data and the server a lot of data (consider multimedia streaming). There might even be scenarios, in which data is only sent in one direction (using a connectionless protocol like UDP. An example might be a network with a lot of real time sensor data, that is sent via UDP (fire and forget). In such a network it does not makes sense to perform bidirectional bandwidth tests.

The output from the bidirectional measurement of the client:

```
root@group08-lg2:~# iperf -c 10.8.4.2 -d
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------

------------------------------------------------------------
Client connecting to 10.8.4.2, TCP port 5001
TCP window size:  264 KByte (default)
------------------------------------------------------------
[  3] local 10.8.1.2 port 44913 connected with 10.8.4.2 port 5001
[  5] local 10.8.1.2 port 5001 connected with 10.8.4.2 port 34324
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   325 MBytes   272 Mbits/sec
[  5]  0.0-10.1 sec   328 MBytes   274 Mbits/sec
```

The output from the server:

```
root@group08-lg4:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.8.4.2 port 5001 connected with 10.8.1.2 port 44913
------------------------------------------------------------
Client connecting to 10.8.1.2, TCP port 5001
TCP window size:  162 KByte (default)
------------------------------------------------------------
[  6] local 10.8.4.2 port 34324 connected with 10.8.1.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.0 sec   328 MBytes   275 Mbits/sec
[  4]  0.0-10.1 sec   325 MBytes   271 Mbits/sec
```

## Question 3

### 3a

As per the Harpoon paper published by Joel Sommers, Hyungsuk Kim and Paul Barford tools like iperf lack the richness and diversity of packets streams observed in the live internet. Harpoon

targets traffic generation at the IP flow level. It gives router designers and network operators insight of system demeanor under real operating conditions.

### 3b

Harpoon combines seven distributional models: file size, inter-file request time, session duration, inter-session request time, IP range, user ON time and number of active users. These model work at different levels in the hierarchy: user, session and file level. These parameters are drawn from different distributions.

- file size - the size of the file that is transferred
- inter-file request time - the time between file consecutive file transfer requests.
- IP spatial distribution - the space from which IP addresses of destination and source are drawn for a session
- inter-session start times - the time interval that separates the start of consecutive sessions
- session duration - the time interval during which file transfer requests between source and destination take place
- user ON time - the time a user is active (drawn from a distribution)
- active users - the number of active users, which may vary over time as it does in the Internet

In order to simulate UDP traffic harpoon simply transfers datagram of a fixed size at a constant bit rate, while TCP file transfers are controlled by protocol dynamics (run at end hosts).

### 3c

We explained these parameters in Question 3b: inter-connection time is the same as inter-file request time and active sessions corresponds to the number of active users. There have been some changes between different versions of harpoon, which explain the differences in architecture and names. In the terminology used in our config files harpoon only consists of a two level architecture.

### 3d

```
root@group08-lg4:/usr/local/harpoon/run_harpoon.sh -f /root/profiles/
    web-server.xml -v 10 -w 300

root@group08-lg4:/usr/local/harpoon# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
    State
tcp        0      0 0.0.0.0:8180            0.0.0.0:*
    LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*
    LISTEN
tcp        0      0 0.0.0.0:10000           0.0.0.0:*
    LISTEN
tcp6       0      0 :::22                   :::*
    LISTEN
udp        0      0 0.0.0.0:68              0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         DGRAM                    7752     /run/systemd
    /notify
```

```
unix  2      [ ]         DGRAM                    7754     /run/systemd
    /cgroups-agent
unix  2      [ ACC ]     STREAM     LISTENING     7758     /run/systemd
    /private
unix  2      [ ]         DGRAM                    7765     /run/systemd
    /journal/syslog
unix  4      [ ]         DGRAM                    7776     /run/systemd
    /journal/dev-log
unix  2      [ ACC ]     STREAM     LISTENING     7779     /run/systemd
    /fsck.progress
unix  2      [ ACC ]     SEQPACKET  LISTENING     8042     /run/udev/
    control
unix  2      [ ACC ]     STREAM     LISTENING     7789     /run/systemd
    /journal/stdout
unix  5      [ ]         DGRAM                    7791     /run/systemd
    /journal/socket
unix  3      [ ]         DGRAM                    8878
unix  2      [ ]         DGRAM                    240184
unix  3      [ ]         STREAM     CONNECTED     9344
unix  2      [ ]         DGRAM                    8593
unix  3      [ ]         STREAM     CONNECTED     11029    /run/systemd
    /journal/stdout
unix  3      [ ]         DGRAM                    7757
unix  3      [ ]         DGRAM                    8681
unix  3      [ ]         STREAM     CONNECTED     8492
unix  2      [ ]         DGRAM                    137863
unix  2      [ ]         DGRAM                    8860
unix  3      [ ]         STREAM     CONNECTED     11028
unix  2      [ ]         DGRAM                    9471
unix  3      [ ]         DGRAM                    8879
unix  2      [ ]         DGRAM                    8047
unix  3      [ ]         DGRAM                    8680
unix  3      [ ]         DGRAM                    7756
unix  2      [ ]         DGRAM                    9436
unix  3      [ ]         STREAM     CONNECTED     8677
unix  3      [ ]         DGRAM                    8880
unix  3      [ ]         STREAM     CONNECTED     9345     /run/systemd
    /journal/stdout
unix  3      [ ]         STREAM     CONNECTED     8678     /run/systemd
    /journal/stdout
unix  3      [ ]         STREAM     CONNECTED     8494     /run/systemd
    /journal/stdout
unix  3      [ ]         DGRAM                    8881
```

```
root@group08-lg2:/usr/local/harpoon/run_harpoon.sh -f /root/profiles/
    web-client.xml -v 10 -w 300 -c
```

As we can see in the screenshot in Figure 2 generated on the harpoon client 22130 kbps of incoming traffic and 237 kbp of outgoing traffic is generated. Of course these numbers vary slightly with the server sending about a hundred times more traffic than the client.

### 3e

```
root@group08-lg2:/usr/local/harpoon# iptraf-ng
```

We show the detailed interface statistics in Figure 2.

```
lev-sc1#show interfaces Gi1/0/12 | include bits
  30 second input rate 38167000 bits/sec, 3918 packets/sec
```

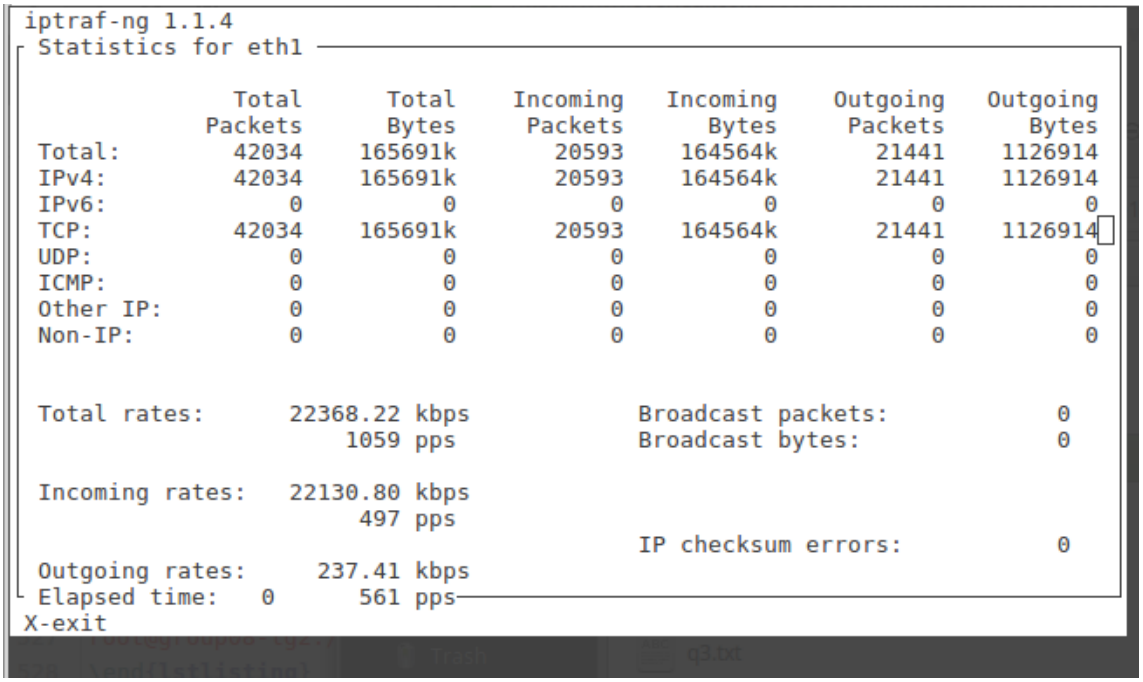


Figure 2: The iptraf output showing detailed interface statistics.

```
  30 second output rate 38147000 bits/sec, 3917 packets/sec
lev-sc1#show interfaces Gi1/0/2 | include bits
  30 second input rate 317000 bits/sec, 562 packets/sec
  30 second output rate 31834000 bits/sec, 2723 packets/sec
lev-sc1#show interfaces Gi1/0/3 | include bits
  30 second input rate 0 bits/sec, 0 packets/sec
  30 second output rate 11000 bits/sec, 1 packets/sec
lev-sc1#show interfaces Gi1/0/4 | include bits
  30 second input rate 26264000 bits/sec, 2266 packets/sec
  30 second output rate 302000 bits/sec, 537 packets/sec

lev-rc1#show interfaces Gi9 | include bits
  30 second input rate 19688000 bits/sec, 2167 packets/sec
  30 second output rate 19702000 bits/sec, 2168 packets/sec
```

### 3f

We simply changed the number of active sessions trying different values and using the python tool to figure out different values for different target traffic rates:

```
root@group08-lg2:~# python harpoon-master/selfconf/harpoon_reconf.py -d
    -c profiles/web-client.xml -s profiles/web-server.xml -i 300 -r
    2000000
target volume:  150000000.0
interval duration:  300
client conf file:  profiles/web-client.xml
server conf file:  profiles/web-server.xml
targetbytes 150000000.0 simbytes 206930175 median 2 mean 1 stdev
    0.519809599952 max 3 flows 2146
number of sessions should be 1 to achieve volume of  150000000 bytes
    (2000000.0 bits/sec)
```

We used the following numbers of sessions. We left the remaining parameters untouched. Our

| Number of active sessions | Total (kbps) | Incoming (kbps) | Outgoing (kbps) |
|---|---|---|---|
| 1 | 2057 | 2039 | 18 |
| 100 | 175426 | 173849 | 1577 |
| 500 | 44618 | 43887 | 731 |
| 1000 | 121360 | 120138 | 1222 |
| 5000 | 85882 | 84901 | 980 |

Table 3: Different numbers of active sessions and the reported data rates (by iptraf-ng at the client).

data rates are shown in Table 3.

# Question 4

## 4a

```
lev-rc1(config-if)#ip address 100.100.100.100 255.255.255.255
lev-rc1(config)#interface Loopback 0
lev-rc1(config-if)#no shutdown
lev-rc1(config)#ip flow-export source Loopback0
lev-rc1(config)#ip flow-export destination 10.8.3.2 7777
lev-rc1(config)#interface Gi9.82
lev-rc1(config-subif)#ip flow ingress

lev-rc1#sh ip flow export
Flow export v1 is enabled for main cache
  Export source and destination details :
  VRF ID : Default
    Source(1)       100.100.100.100 (Loopback0)
    Destination(1)  10.8.3.2 (7777)
  Version 1 flow records
  118876 flows exported in 4962 udp datagrams
  0 flows failed due to lack of export packet
  0 export packets were sent up to process level
  0 export packets were dropped due to no fib
  0 export packets were dropped due to adjacency issues
  0 export packets were dropped due to fragmentation failures
  0 export packets were dropped due to encapsulation fixup failures
```

## 4b

Note, that we are listening to all flows. We could restrict it to flow coming from 100.100.100.100, though.

```
root@group08-lg3:~# tail /etc/flow-tools/flow-capture.conf
# Store flows in /var/flow/mysecondrouter. Rotate files every
# 5 minutes.
# -w /var/flow/mysecondrouter -n 275 0/10.3.2.6/3002

# Example 3:
# Same as above, but only listen at address 10.3.2.5, and store
# files under 'YYYY/YYYY-MM/YYYY-MM-DD' directories.
# -w /var/flow/mysecondrouter -n 275 -N 3 10.3.2.5/10.3.2.6/3002

-w /tmp/flows 0/0/7777
```

```
root@group08-lg3:~# /etc/init.d/flow-capture start

root@group08-lg3:~# flow-cat < /tmp/flows/2018/2018-05/2018-05-27/ft-
    v01.2018-05-27.105055+0000 | flow-report

# ---- ----- ----- Report Information ---- ---- ----
# build-version:          flow-tools 0.68
# name:                   default
# type:                   summary-detail
# options:                +header,+xheader,+totals
# fields:                 +other
# records:                0
# first-flow:             1527416860 Sun May 27 10:27:40 2018
# last-flow:              1527417338 Sun May 27 10:35:38 2018
# now:                    1527418823 Sun May 27 11:00:23 2018
#
# mode:                   normal
# capture hostname:       group08-lg3
# capture start:          Sun May 27 10:50:55 2018
# capture end:            Sun May 27 11:00:00 2018
# capture period:         545 seconds
# compress:               on
# byte order:             little
# stream version:         3
# export version:         1
# lost flows:             0
# corrupt packets:        0
# sequencer resets:       0
# capture flows:          20040
#
#  ['/usr/bin/flow-rptfmt', '-f', 'ascii']
Ignores:                  0
Total Flows:              20040
Total Octets:             19564567
Total Packets:            371887
Total Duration (ms):      166928
Real Time:                1527417338
Average Flow Time:        8.000000
Average Packets/Second:   52.000000
Average Flows/Second:     976.000000
Average Packets/Flow:     18.000000
Flows/Second:             0.115350
Flows/Second (real):      0.000013

Average IP packet size distribution:

   1-32   64   96  128  160  192  224  256  288  320  352  384  416
      448  480
  .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000
      .000 .000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
  .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

Packets per flow distribution:
```

```
     1     2     4     8    12    16    20    24    28    32    36    40    44
          48    52
  .000  .000  .000  .287  .407  .117  .060  .031  .020  .013  .008  .007  .006
       .005  .005

    60   100   200   300   400   500   600   700   800   900  >900
  .007  .015  .007  .002  .001  .000  .000  .000  .000  .000  .001

Octets per flow distribution:

     32    64   128   256   512  1280  2048  2816  3584  4352  5120  5888  6656
          7424  8192
  .000  .000  .000  .000  .411  .491  .047  .018  .011  .006  .004  .002  .001
       .001  .001

  8960  9728 10496 11264 12032 12800 13568 14336 15104 15872 >15872
  .001  .000  .000  .000  .000  .000  .000  .000  .000  .000  .002

Flow Time Distribution (ms):

    10    50   100   200   500  1000  2000  3000  4000  5000  6000  7000  8000
         9000 10000
  .892  .098  .005  .001  .001  .001  .001  .000  .000  .000  .000  .000  .000
       .000  .000

 12000 14000 16000 18000 20000 22000 24000 26000 28000 30000 >30000
  .000  .000  .000  .000  .000  .000  .000  .000  .000  .000  .000
```

In the flow-report output we can see some information regarding the flow such as when it was captured, how many flows, packets, octets etc. were captured, which version was used and so on. Most importantly we can see how many, packets were captured, what was the packet size, how many bytes per flow were captured and how long lasted every flow. We can also see the distribution of these values apart from the absolute numbers and the averages.

### 4c

In Figure 3, 4 and 5 we show the graphs for the standard configuration.

In Figure 6, 7 and 8 the graphs for the maximum throughput are shown. Unfortunately there seems to be a delay of about 30 minutes until the flows captured at the router actually are sent to the monitor. Therefore and due to our ending reservation we may not have captured a complete round of 5 minutes of maximum throughput. We provide our graphs anyway with what we have got.

```
root@group08-lg3:/tmp/flows/2018/2018-05/2018-05-28# flow-cat < tmp-v01
    .2018-05-28.174622+0000 | flow-report
# ---- ----- ----- Report Information ---- ---- ---
# build-version:          flow-tools 0.68
# name:                   default
# type:                   summary-detail
# options:                +header,+xheader,+totals
# fields:                 +other
# records:                0
# first-flow:             1527528186 Mon May 28 17:23:06 2018
# last-flow:              1527528598 Mon May 28 17:29:58 2018
# now:                    1527530005 Mon May 28 17:53:25 2018
#
# mode:                   streaming
```

```
# compress:                 off
# byte order:               little
# stream version:           3
# export version:           1
#
#  ['/usr/bin/flow-rptfmt', '-f', 'ascii']
Ignores:                    0
Total Flows:                71899
Total Octets:               66190891
Total Packets:              1240568
Total Duration (ms):        12457552
Real Time:                  1527528598
Average Flow Time:          173.000000
Average Packets/Second:     53.000000
Average Flows/Second:       920.000000
Average Packets/Flow:       17.000000
Flows/Second:               5.239928
Flows/Second (real):        0.000047

Average IP packet size distribution:

   1-32   64   96  128  160  192  224  256  288  320  352  384  416
      448  480
  .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000
      .000 .000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
  .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

Packets per flow distribution:

      1    2    4    8   12   16   20   24   28   32   36   40   44
         48   52
  .000 .000 .000 .450 .278 .094 .052 .032 .020 .013 .010 .007 .006
      .004 .004

     60  100  200  300  400  500  600  700  800  900 >900
   .006 .012 .007 .002 .001 .001 .000 .000 .000 .000 .001

Octets per flow distribution:

     32   64  128  256  512 1280 2048 2816 3584 4352 5120 5888 6656
        7424 8192
  .000 .000 .000 .000 .565 .337 .050 .017 .009 .005 .003 .002 .001
      .001 .001

  8960 9728 10496 11264 12032 12800 13568 14336 15104 15872 >15872
  .001 .001  .001  .000  .000  .000  .000  .000  .000  .000  .003

Flow Time Distribution (ms):

    10    50  100  200  500 1000 2000 3000 4000 5000 6000 7000 8000
       9000 10000
  .209 .384 .006 .138 .208 .029 .020 .003 .002 .001 .000 .001 .000
      .000 .000
```

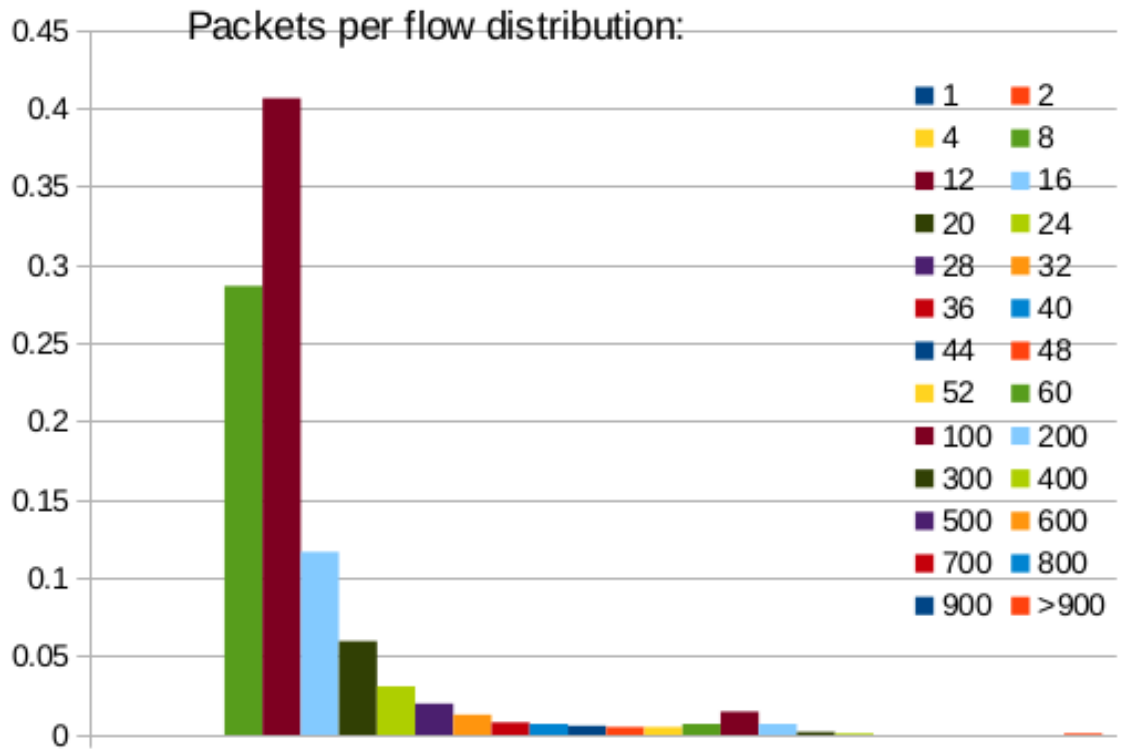


Figure 3: The packets per flow distribution with standard configuration.

| 12000 | 14000 | 16000 | 18000 | 20000 | 22000 | 24000 | 26000 | 28000 | 30000 | >30000 |
|---|---|---|---|---|---|---|---|---|---|---|
| .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 | .000 |

## Question 5

### 5a

```
lev-rc1(config)#snmp-server community OUR_SECRET RO
```

### 5b

```
root@group08-lg3:~# service snmpd start
root@group08-lg3:~# ps -eaf | grep snmpd
Debian-+ 23773     1  0 08:32 ?        00:00:00 /usr/sbin/snmpd -Lsd -
    Lf /dev/null -u Debian-snmp -g Debian-snmp -I -smux mteTrigger
    mteTriggerConf -f
root     23834 20676  0 08:36 hvc0     00:00:00 grep snmpd

root@group08-lg3:~# snmpget -v1 -Of -c OUR_SECRET 10.8.3.1 iso.org.dod.
    internet.mgmt.mib-2.system.sysUpTime.0
.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.sysUpTimeInstance =
    Timeticks: (103560) 0:17:15.60
```

### 5c

UDP is used, because it behaves far better in networks, where a lot of packets are lost (UDP even begins to perform better at about 5%+ packet loss). In case of packet loss, the data is lost and cannot be recovered, because the sender does not know about the loss and does not keep a copy of

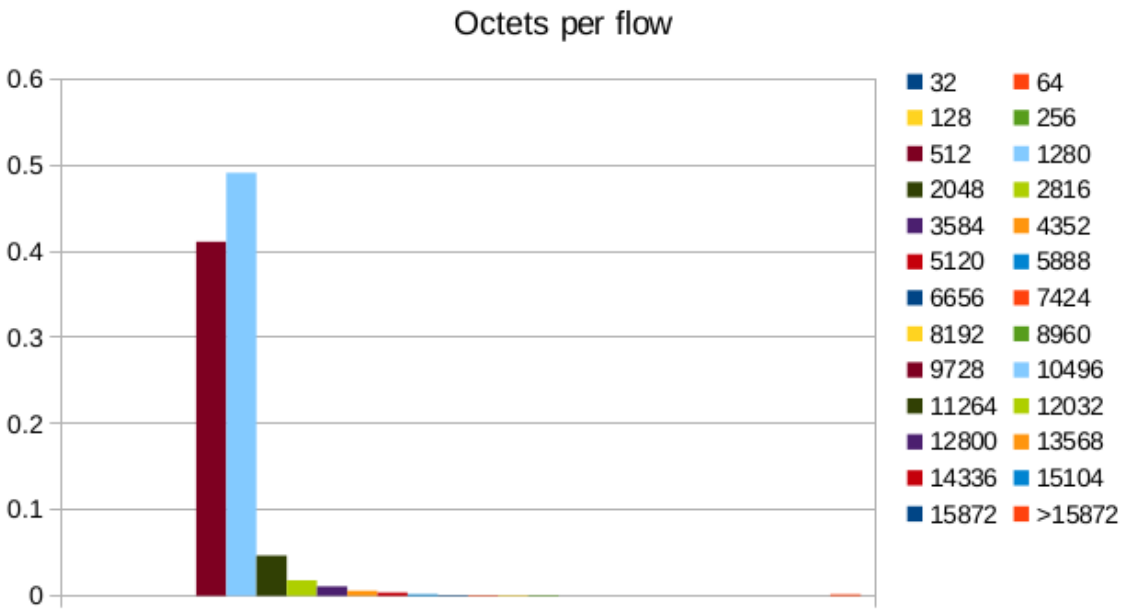


Figure 4: The octets per flow distribution with standard configuration.

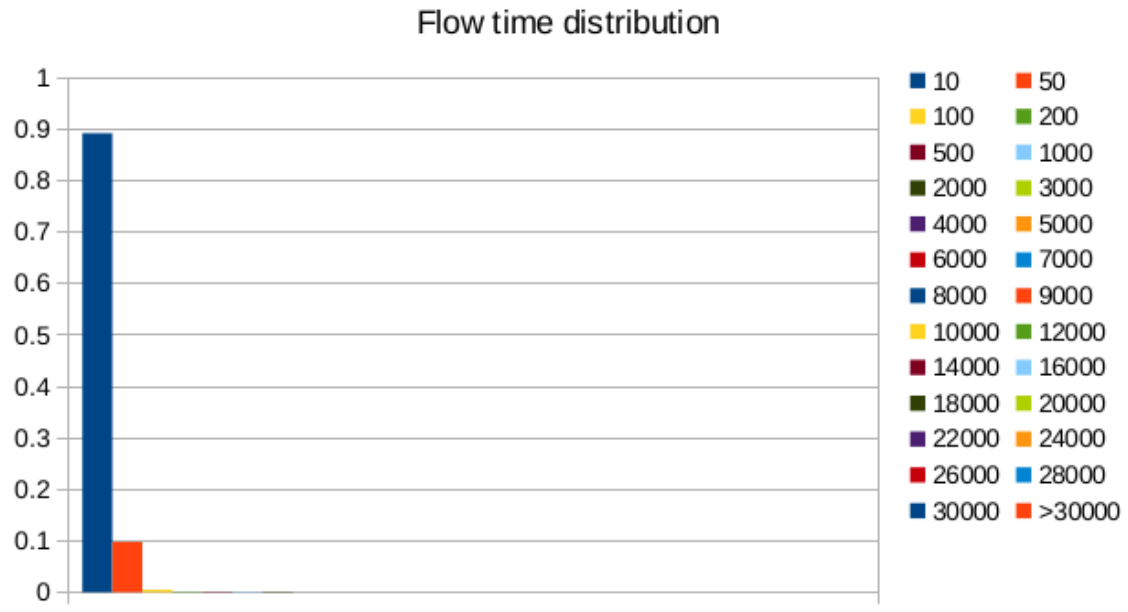


Figure 5: The flow time distribution with standard configuration.

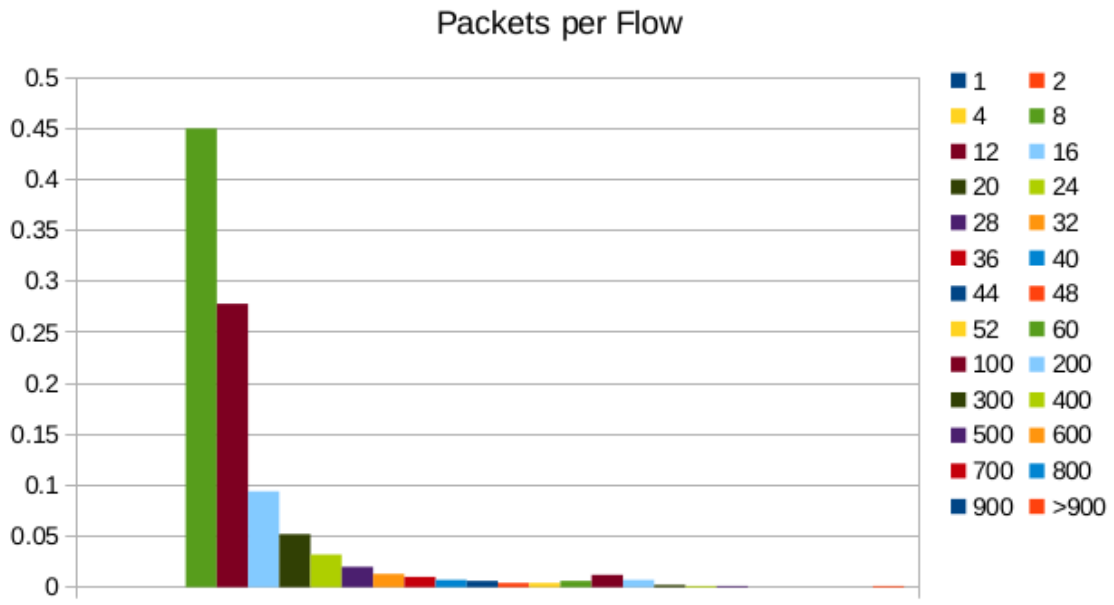


Figure 6: The packets per flow distribution with maximum throughput.

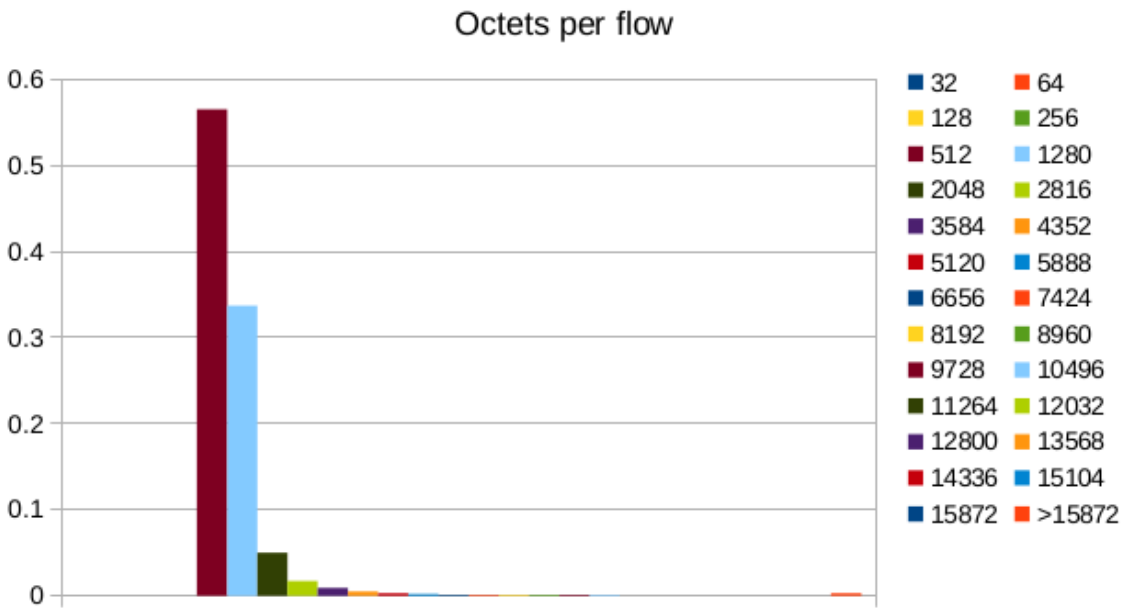


Figure 7: The octets per flow distribution with maximum throughput.

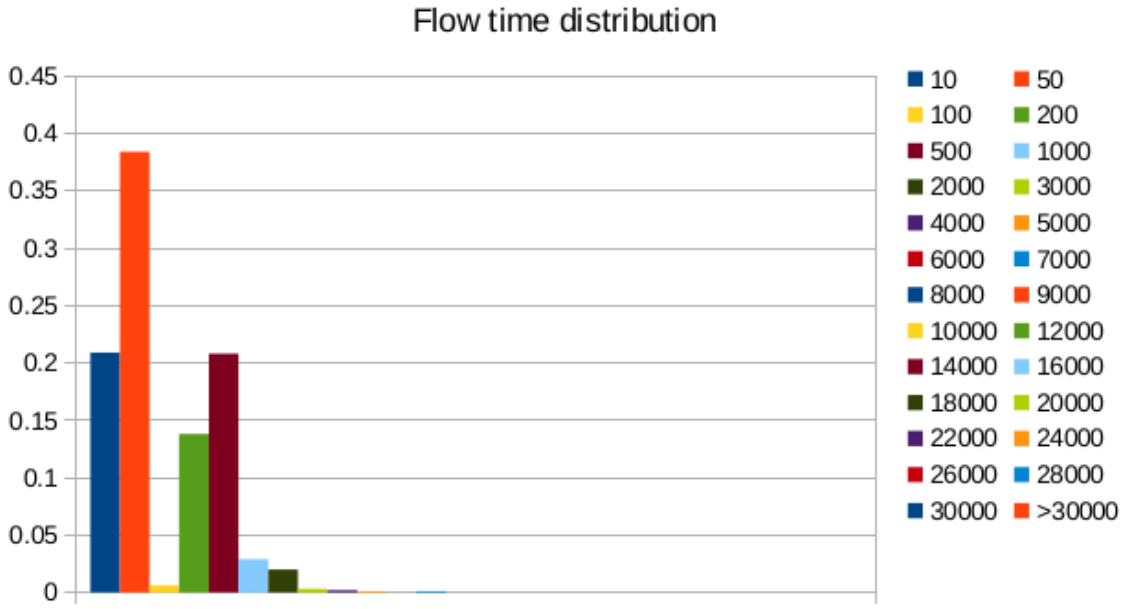


Figure 8: The flow time distribution with maximum throughput.

the packet. However, another request can be sent by the manager to get the current information (which might have changed in the mean time). In SNMP there also is the option to use SNMP Informs, which are acknowledged at application layer and therefore the most reliable choice if such acknowledgement is required.
If there is no agent running on the router, the sender will not receive an answer and it will appear as if the packet was lost. But for the sender it is impossible to tell the difference between packet loss, no running agent on the router and an incorrect community string.
The SNMP community string serves the purpose of authenticating the device requesting from or setting information on the agent. If it is incorrect the manager will not retrieve a notification that authentication failed but wait for a timeout:

```
root@group08-lg3:~# snmpget -v1 -Of -c WRONG_SECRET 10.8.3.1 iso.org.
    dod.internet.mgmt.mib-2.system.sysUpTime.0
Timeout: No Response from 10.8.3.1.
```

Unfortunately, at least in version 1 all community strings are sent in cleartext, which makes SNMP an insecure protocol, especially if devices are not configured to be in read-only mode (as we did).

## 5d

We use the following command to capture the packets:

```
root@group08-lg3:~# tshark -i eth1 -w capture.pcap
Running as user "root" and group "root". This could be dangerous.
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:44: dofile has been disabled
    due to running Wireshark as superuser. See https://wiki.wireshark.
    org/CaptureSetup/CapturePrivileges for help in running Wireshark
    as an unprivileged user.
Capturing on 'eth1'
[2702983.326642] device eth1 entered promiscuous mode
[...]
```

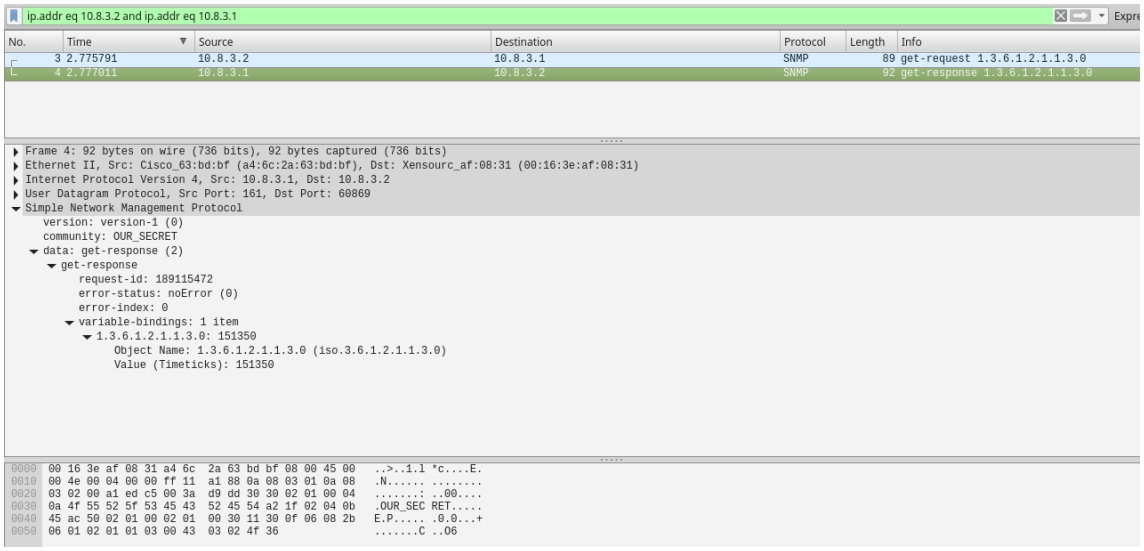


Figure 9: The SNMPget response.

Then we create an snmpget request as shown above and capture it. We analyze everything in Wireshark using a filter to get rid of non-SNMP packets (see Figure 9). Obviously we see the usual protocol stack with UDP at the transport layer. On top of it we find SNMP. We can see the community secret and the version in plaintext. Wireshark shows the message type and the data payload. The main difference between request and response is the value, which is displayed in the response. It is the number of timeticks. On the command line SNMP displays this value both as the number of time ticks and as a human readable time, along with the OID where it was found. In the packet the OID is a numeric value, while it is a human readable string in the cli. It is also possible to provide the numerical value, when requesting the OID.

## 5e

We set the MIBs on rc1:

```
lev-rc1(config)#snmp-server location sweethome
lev-rc1(config)#snmp-server contact ourcontact
```

Then we do the snmpwalk and capture it with tshark as above. The full output is provided in an extra file in the attachments as it is more than 800 lines:

```
root@group08-lg3:~# snmpwalk -v1 -Of -c OUR_SECRET 10.8.3.1 iso.org.dod
    .internet.mgmt.mib-2.system
[...]
.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.sysUpTimeInstance =
    Timeticks: (414864) 1:09:08.64
.iso.org.dod.internet.mgmt.mib-2.system.sysContact.0 = STRING:
    ourcontact
.iso.org.dod.internet.mgmt.mib-2.system.sysName.0 = STRING: lev-rc1
.iso.org.dod.internet.mgmt.mib-2.system.sysLocation.0 = STRING:
    sweethome
.iso.org.dod.internet.mgmt.mib-2.system.sysServices.0 = INTEGER: 78
[...]
```

snmpwalk basically saves users time by always automatically issuing the get-next-request, as it is shown by Wireshark (see attached capture file and Figure 10). Therefore it is used to retrieve a complete subtree of information with just one call instead of calling each unit of information with an individual get request. It will stop when the information requested is no longer part of the requested OID. If it is performed on the system OID it usually retrieves the complete tree of

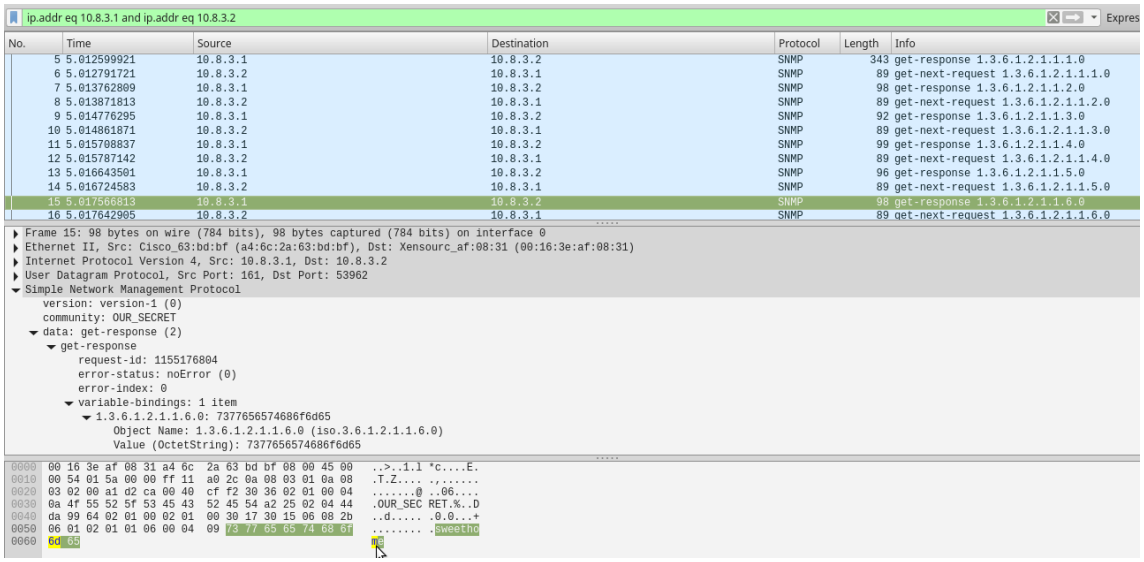


Figure 10: The SNMPwalk capture in Wireshark.

information stored on the machine in the system MIB group. AS we can see in the capture each snmp-next-request is respondede with the same kind of snmp-response as in 5d.

# Included files

capture_question_05d.pcap, capture_question_05e.pcap, snmpwalk.txt, q03-config-rc1.txt, q03-config-sc1.txt, q04-config-rc1.txt