Advanced Networking 2020

Lab #2: COS and DSCP

Report

GROUP: 5

Authors:
Pavlos Lontorfos, plontorfos@os3.nl
Tom Carpaij tcarpaij@os3.nl
Sean Liao Sliao@os3.nl
Rutger Beltman rbeltman@os3.nl
University of Amsterdam

Task 0

Setup Server

```
Switch(config-if)#ip address 10.0.1.27 255.255.255.0
Switch(config-if)#aaa new-model
Switch(config)#no ip http server
Switch(config)#no ip http secure-server
Switch(config)#username tom password 0 123123
Switch(config)#username pavlos password 0 123123
Switch(config)#username sean password 0 123123
Switch(config)#username rutger password 0 123123
Switch(config)#enable password 0 password
copy running-config startup-config
```

User	Switch Port
Sean	1
Tom	2
Pavlos	3
Rutger	4

Task 1 : Data transfer without and with CoS configuration

1.1 In this task you are asked to perform data transfers between the nodes connected to the switch.

1.1.1 Use iperf to transfer data between a pair of nodes.

We changed the MTU size in order to achive better throughput.

```
sudo ip link set eno2 mtu 9000
```

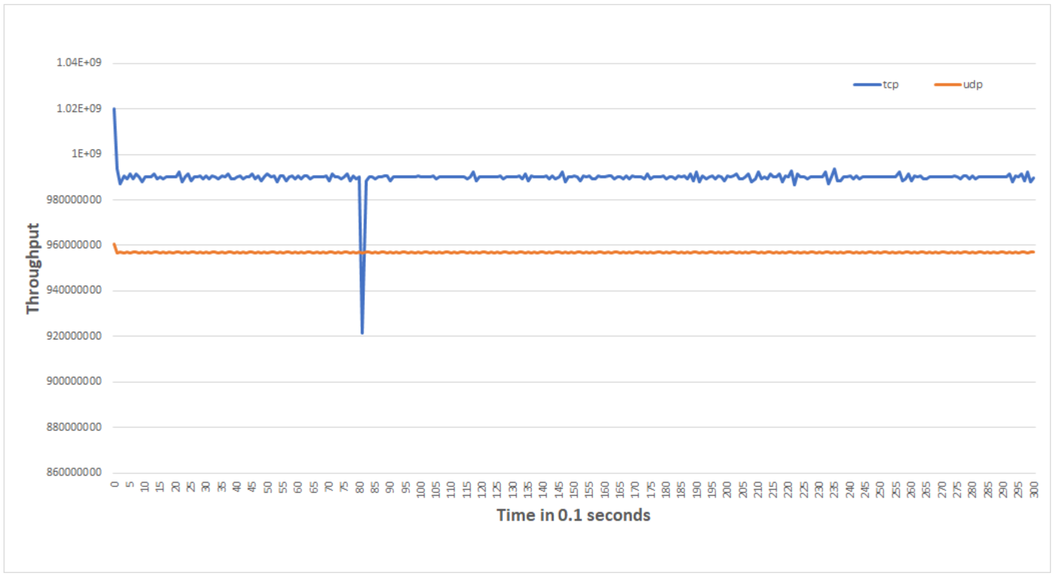


Figure 1: A plot of the achieved throughput results.Shows the results in comparison to each other.

1.1.2 Set up the network based on table.

We used iperf to communicate with Sean's server in port one, who is the receiver. We exchanged data for 30 seconds. The configuration of the switch was on **default** settings.
Our iperf commands were:

```
For UDP:
iperf —c 10.0.1.194 —p 8001 —u —P 10 —t 30 —b 1G
For TCP:
iperf —c 10.0.1.194 —p 8002 —P 10 —t 30
iperf —c 10.0.1.194 —p 8003 —P 10 —t 30
```

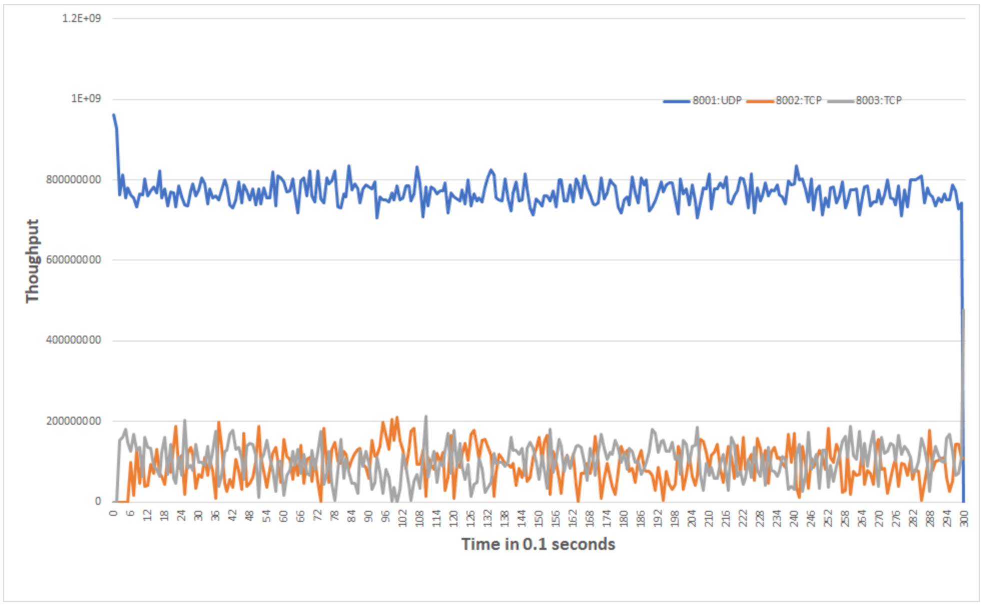


Figure 2: A plot of the achieved throughput results. Note that server 2 is using UDP and server 3 and 4 are using TCP.

We can see that UDP gets the whole available bandwidth. This is what we expected as UDP is a protocol that will keep sending packets in full speed (-b 1G), even if the receiver fails to keep up the pace. At the same time TCP keeps reducing its window due to congestion, and it ends up with a minimum amount of the bandwidth of the channel.

1.2 Now you have to overcome the interference by applying CoS. Try to improve the performance of the communication by starting from the degraded scenarios above.

We used the following commands to do CoS labeling on the iperf clients.

```
sudo ip link add link eno2 name eno2.2 type vlan id 2
sudo ip addr add 10.0.2.192/24 dev eno2.2
sudo ip link set dev eno2.2 up
sudo ip link set eno2.2 type vlan egress 0:7
```

The configuration used on the switch:

```
Using 2011 out of 524288 bytes
!
version 12.2
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Switch
!
boot-start-marker
boot-end-marker
```

```
!
enable password password
!
username tom privilege 15 password 0 123123
username pavlos privilege 15 password 0 123123
username sean privilege 15 password 0 123123
username rutger privilege 15 password 0 123123
!
!
aaa new-model
!
!
!
!
!
aaa session-id common
switch 2 provision ws-c3750g-24ps
system mtu routing 1500
!
!
!
mls qos
!
!
!
!
!
spanning-tree mode pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
!
!
class-map match-all class-UDP
 match access-group name traffic-UDP
!
!
policy-map policy-UDP
 class class-UDP
  set dscp cs5
!
!
!
!
interface GigabitEthernet2/0/1
!
interface GigabitEthernet2/0/2
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport priority extend trust
 mls qos trust cos
```

```
!
interface GigabitEthernet2/0/3
 mls qos cos 5
!
interface GigabitEthernet2/0/4
 mls qos cos 6
!
interface GigabitEthernet2/0/5
!
.
.
.
!
interface Vlan1
 ip address 10.0.1.193 255.255.255.0
!
ip classless
no ip http server
no ip http secure-server
!
!
!
!
!
line con 0
line vty 5 15
!
end
```

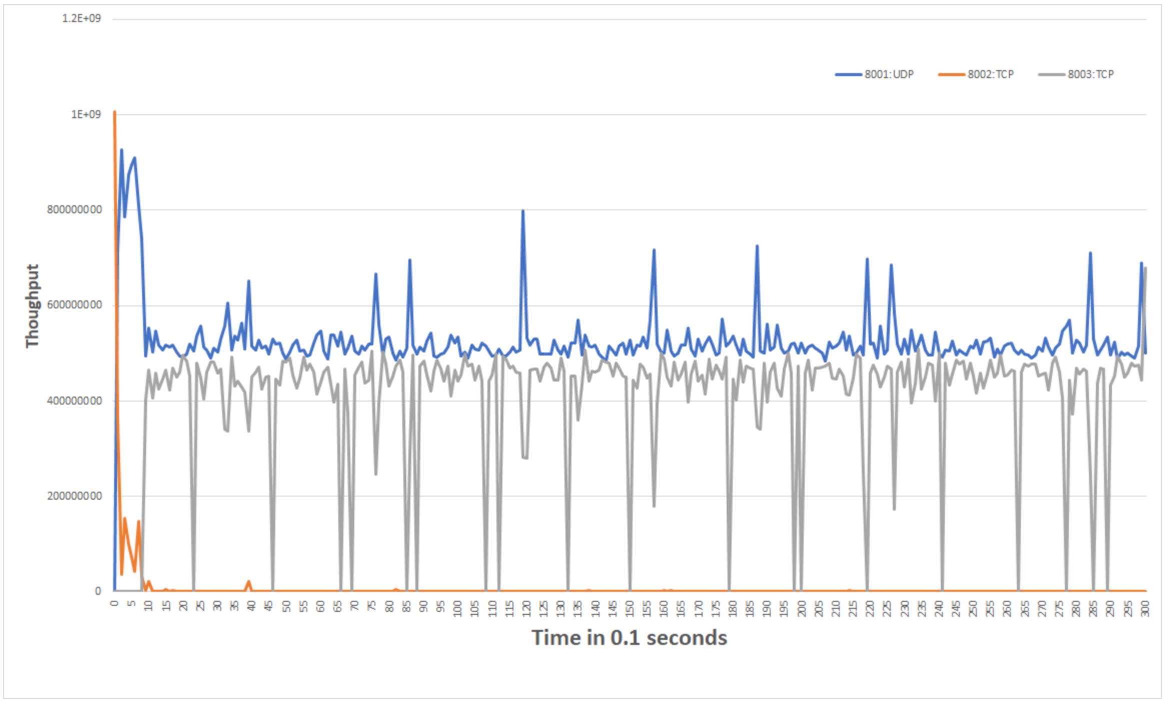


Figure 3: A plot of the achieved throughput results. Note that server 2 is using UDP and server 3 and 4 are using TCP.

We can observe UDP has the highest throughput and both TCP streams are lower. The higher throughput of the two TCP streams is only slightly less favored than the UDP stream. From figure 6 we learn that the 6 and 7 CoS values are sent to the same queue. Though, the steam from server 4(port 8003), has a higher cos value. As a result it gets higher priority and it ends up getting the whole bandwidth. Since the high throughput TCP stream is in different queue from the UDP, these two streams compete for throughput and get roughly the same bandwidth.

1.3 You will now have to understand traffic Shaping and Sharing and show the difference between them.

The configuration used on the switch:

1.3.2 For testing SRR Shaping, implement the settings in table

We used the following configuration (only changes are shown):

```
interface GigabitEthernet2/0/1
 srr-queue bandwidth shape 8 0 0 2
 mls qos trust cos
!
interface GigabitEthernet2/0/2
 switchport priority extend trust
 mls qos cos 7
 mls qos trust cos
```

```
!
interface GigabitEthernet2/0/3
 mls qos cos 5
 mls qos trust cos
!
interface GigabitEthernet2/0/4
!
interface GigabitEthernet2/0/5
 mls qos cos 7
 mls qos trust cos
!
!
interface Vlan1
 ip address 10.0.1.193 255.255.255.0
!
interface Vlan2
 ip address 10.0.2.193 255.255.255.0
!
```

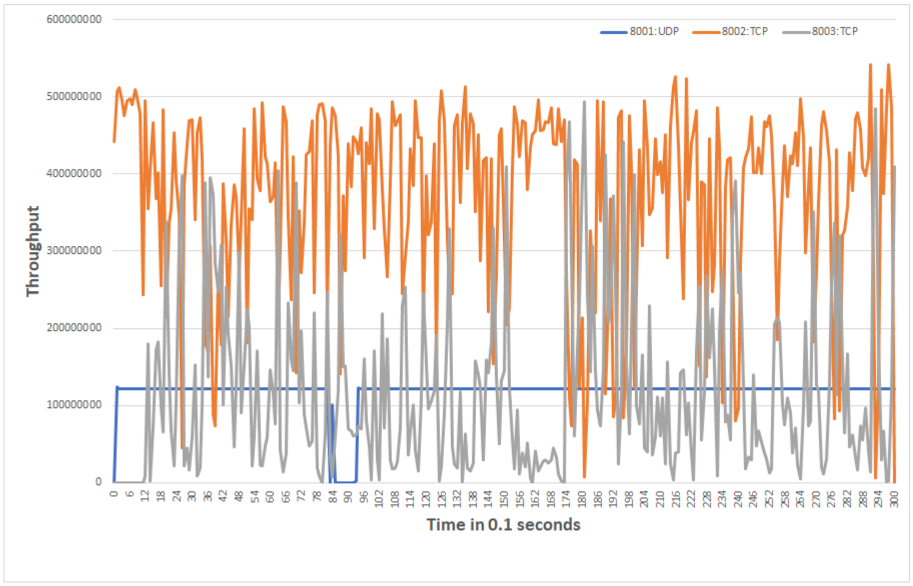


Figure 4: A plot of the achieved throughput results. Note that server 2 is using UDP and server 3 and 4 are using TCP.

In this step we are testing the SRR shaping. We observe that the maximum bandwidth that UDP can reach now is 1/8 of the available bandwidth. Indeed,UDP tranfers steadily with about 125Mb/s which is the 1/8 of a 1Gb/s connection. At the same time the TCP connections are limited to 1/2 of the total bandwith. Once again, we observe that the 2 TCP streams are competing for a combined transfer rate of 500Mb/s

```
Switch#show mls qos interface GigabitEthernet 2/0/1 queueing
GigabitEthernet2/0/1
Egress Priority Queue : disabled
```

```
Shaped queue weights (absolute) :  8 0 0 2
Shared queue weights  :  25 25 25 25
The port bandwidth limit : 100  (Operational Bandwidth:100.0)
The port is mapped to qset : 1


Switch#show mls qos interface GigabitEthernet 2/0/1 statistics

 queue:    threshold1   threshold2   threshold3
-----------------------------------------------
 queue 0:           0           0           0
 queue 1:     1220129           0         226
 queue 2:           0           0           0
 queue 3:      200596           0           0

  output queues dropped:
 queue:    threshold1   threshold2   threshold3
-----------------------------------------------
 queue 0:           0           0           0
 queue 1:     1139279           0           0
 queue 2:           0           0           0
 queue 3:       22203           0           0
```

1.3.3 for testing SRR Sharing, implement the settings in the table

We used the following configuration (only changes are shown):

```
interface GigabitEthernet2/0/1
 srr-queue bandwidth share 15 1 1 70
 mls qos trust cos
!
interface GigabitEthernet2/0/2
 switchport priority extend trust
 mls qos cos 7
 mls qos trust cos
!
interface GigabitEthernet2/0/3
 mls qos cos 5
 mls qos trust cos
!
interface GigabitEthernet2/0/4
!
interface GigabitEthernet2/0/5
 mls qos cos 7
 mls qos trust cos
!
!
interface Vlan1
 ip address 10.0.1.193 255.255.255.0
```

```
!
interface Vlan2
 ip address 10.0.2.193 255.255.255.0
!
```

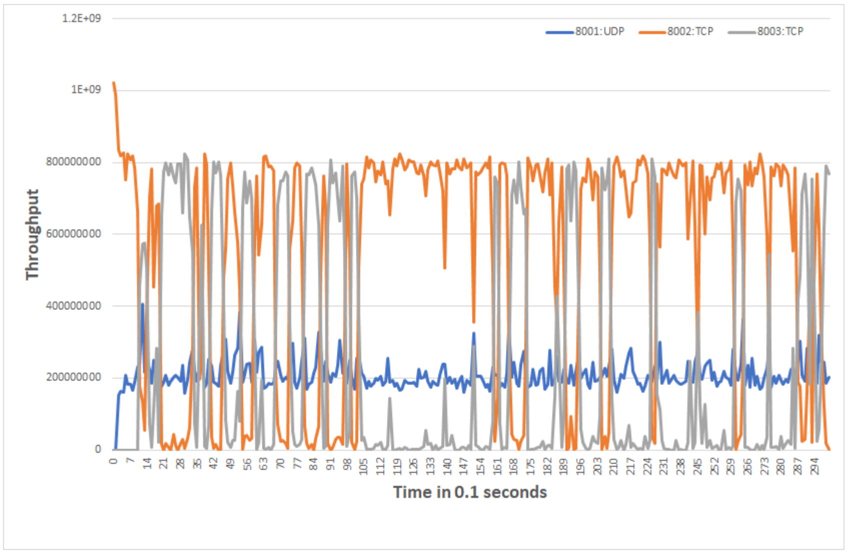


Figure 5: A plot of the achieved throughput results. Note that server 2 is using UDP and server 3 and 4 are using TCP.

In this step we are testing SSR Sharing. We can see that UDP is using up to 15% of the bandwidth (actually due to our configuration it uses 15/87 which is a bit more) which results to about 200Mb/s. Similarly for TCP it uses up to 70/87 of the total bandwidth which results in about 800 Mb/s

```
Switch#show mls qos interface GigabitEthernet 2/0/1 queueing
GigabitEthernet2/0/1
Egress Priority Queue : disabled
Shaped queue weights (absolute) :  0 0 0 0
Shared queue weights  :  15 1 1 70
The port bandwidth limit : 100  (Operational Bandwidth:100.0)
The port is mapped to qset : 1

Switch#show mls qos interface GigabitEthernet 2/0/1 statistics

  output queues enqueued:
 queue:    threshold1   threshold2   threshold3
-----------------------------------------------
 queue 0:          0            0            0
 queue 1:     549852            0           99
 queue 2:          0            0            0
 queue 3:     321750            0            0

  output queues dropped:
 queue:    threshold1   threshold2   threshold3
```

```
-----------------------------------------------
 queue 0:           0           0           0
 queue 1:     1887349           0           0
 queue 2:           0           0           0
 queue 3:       26436           0           0
```

Task 2

2.1 Explain the difference between CoS and DSCP, why these are relevant?

CoS operates on the VLAN level, while DSCP operates on the IP level, so they run on different layers (layer 2 and layer 3 , respectively). Another difference is that CoS doesn't not actually modify the IP packet, but only the Ethernet frame. DSCP modifies the DS field in the IP header. Though the modification of the headers is not really relevant in this case, the layers of operation are relevant. The switch assigned to us is layer 2 switch, though it can do DiffServ (DSCP).
DSCP uses 8 bit field. CoS uses a 4 bit field. The field size is only really relevant in larger network setups.

2.2

We chose to use the DSCP option available on *iperf*, as for these tests we can define our own DSCP values. We configure the switch to trust the DSCP values of each client. The option in *iperf* is given by the *-S* flag.

```
 iperf -S 0x20 -c 10.0.1.194 -p 8001 -u -P 10 -t 30 -b 1G
 iperf -S 0xA0 -c 10.0.1.194 -p 8002 -P 10 -t 30
```

2.3

We can find the queues in the Cisco manual:

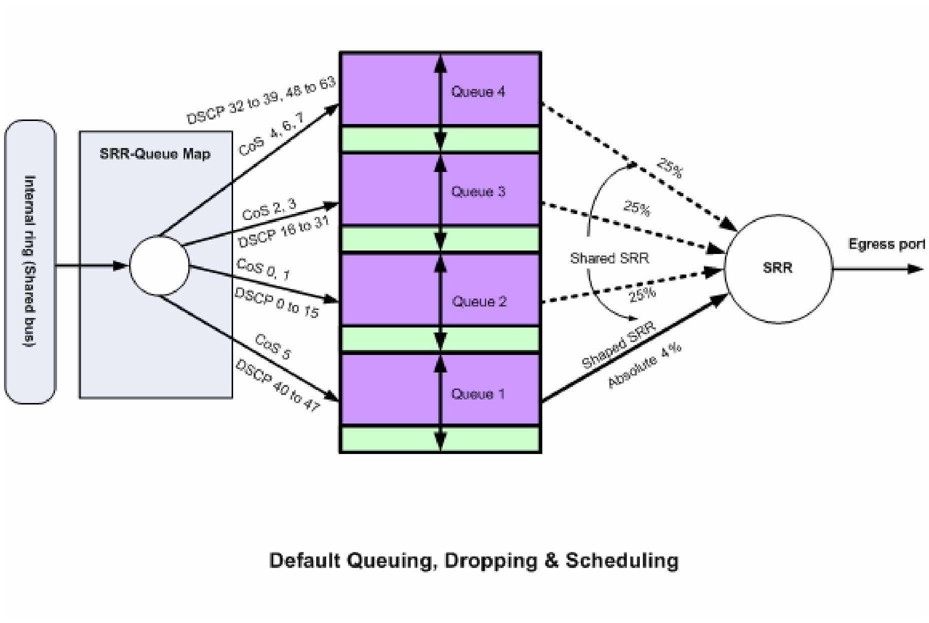
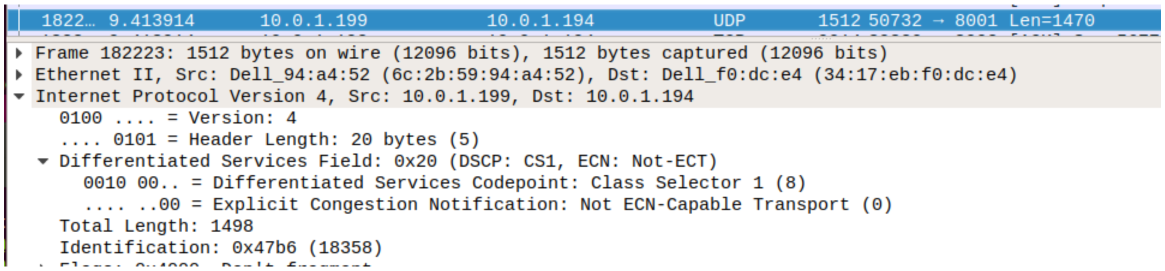


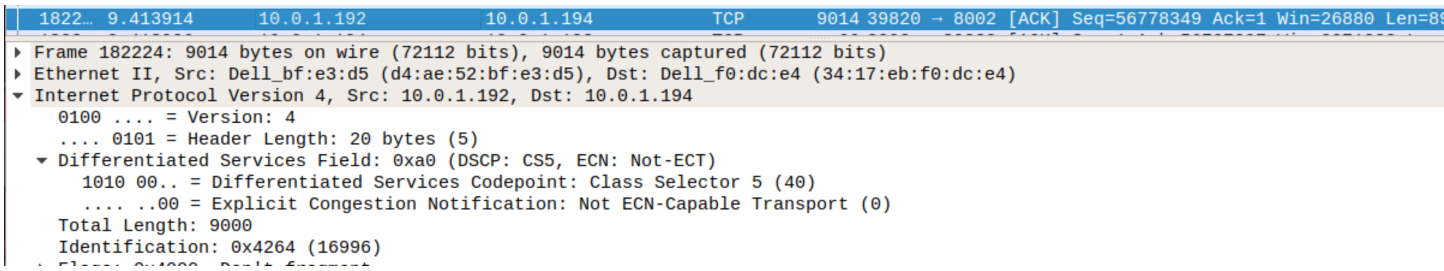
Figure 6: Image from the manual which shows the queues on the Cisco switch

2.4



```
1822…  9.413914      10.0.1.199          10.0.1.194          UDP      1512 50732 → 8001 Len=1470
▸ Frame 182223: 1512 bytes on wire (12096 bits), 1512 bytes captured (12096 bits)
▸ Ethernet II, Src: Dell_94:a4:52 (6c:2b:59:94:a4:52), Dst: Dell_f0:dc:e4 (34:17:eb:f0:dc:e4)
▾ Internet Protocol Version 4, Src: 10.0.1.199, Dst: 10.0.1.194
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
      0010 00.. = Differentiated Services Codepoint: Class Selector 1 (8)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 1498
    Identification: 0x47b6 (18358)
```

Figure 7: The DSCP labeled UDP packet



```
1822…  9.413914      10.0.1.192          10.0.1.194          TCP      9014 39820 → 8002 [ACK] Seq=56778349 Ack=1 Win=26880 Len=8
▸ Frame 182224: 9014 bytes on wire (72112 bits), 9014 bytes captured (72112 bits)
▸ Ethernet II, Src: Dell_bf:e3:d5 (d4:ae:52:bf:e3:d5), Dst: Dell_f0:dc:e4 (34:17:eb:f0:dc:e4)
▾ Internet Protocol Version 4, Src: 10.0.1.192, Dst: 10.0.1.194
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0xa0 (DSCP: CS5, ECN: Not-ECT)
      1010 00.. = Differentiated Services Codepoint: Class Selector 5 (40)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 9000
    Identification: 0x4264 (16996)
```

Figure 8: The DSCP labeled TCP packet

2.5

```
Switch#show mls qos interface gigabitEthernet 2/0/1 statistics
.
.
  output queues enqueued:
 queue:    threshold1   threshold2   threshold3
-----------------------------------------------
 queue 0:      336959           0           0
 queue 1:      604719           0          52
 queue 2:           0           0           0
 queue 3:           1           0           0

  output queues dropped:
 queue:    threshold1   threshold2   threshold3
-----------------------------------------------
 queue 0:       26164           0           0
 queue 1:     1891516           0           0
 queue 2:           0           0           0
 queue 3:           0           0           0
```

Task 3

3.1

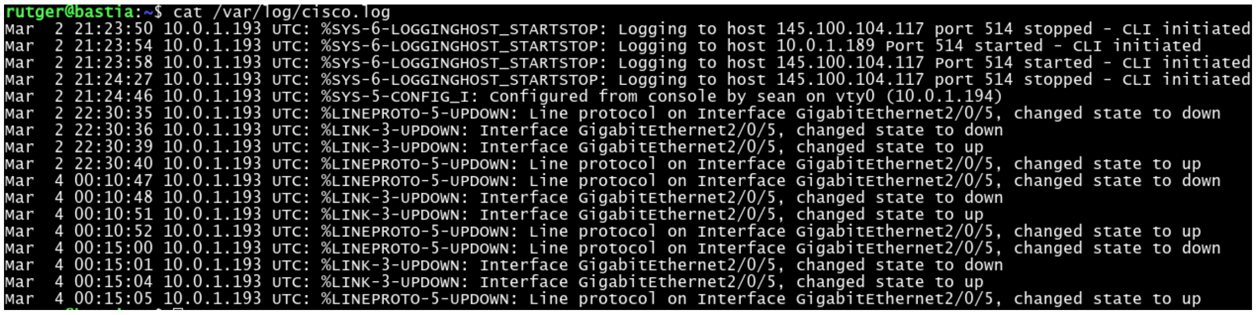
On the switch, set remote logging

```
logging 10.0.1.189

---

Switch#show logging
Syslog logging: enabled (0 messages dropped, 0 messages rate-limited, 0 flushes, 0 overruns, xml disabl

No Active Message Discriminator.



No Inactive Message Discriminator.


    Console logging: level debugging, 147 messages logged, xml disabled,
                     filtering disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging:  level debugging, 147 messages logged, xml disabled,
                     filtering disabled
    Exception Logging: size (4096 bytes)
    Count and timestamp logging messages: disabled
    File logging: disabled
    Persistent logging: disabled

No active filter modules.

    Trap logging: level informational, 151 message lines logged
        Logging to 10.0.1.189  (udp port 514,  audit disabled,
              authentication disabled, encryption disabled, link up),
              9 message lines logged,
              0 message lines rate-limited,
              0 message lines dropped-by-MD,
              xml disabled, sequence number disabled
              filtering disabled
```

On the server, set syslog-ng to log from udp

```
source net {
    udp(port(514));
};
destination cisco { file("/var/log/cisco.log"); };
log { source(net); destination(cisco); };
```

Finally to show that the Bastia server is able to retrieve the syslogs the following output is provided



```
rutger@bastia:~$ cat /var/log/cisco.log
Mar  2 21:23:50 10.0.1.193 UTC: %SYS-6-LOGGINGHOST_STARTSTOP: Logging to host 145.100.104.117 port 514 stopped - CLI initiated
Mar  2 21:23:54 10.0.1.193 UTC: %SYS-6-LOGGINGHOST_STARTSTOP: Logging to host 10.0.1.189 Port 514 started - CLI initiated
Mar  2 21:23:58 10.0.1.193 UTC: %SYS-6-LOGGINGHOST_STARTSTOP: Logging to host 145.100.104.117 Port 514 started - CLI initiated
Mar  2 21:24:27 10.0.1.193 UTC: %SYS-6-LOGGINGHOST_STARTSTOP: Logging to host 145.100.104.117 port 514 stopped - CLI initiated
Mar  2 21:24:46 10.0.1.193 UTC: %SYS-5-CONFIG_I: Configured from console by sean on vty0 (10.0.1.194)
Mar  2 22:30:35 10.0.1.193 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet2/0/5, changed state to down
Mar  2 22:30:36 10.0.1.193 UTC: %LINK-3-UPDOWN: Interface GigabitEthernet2/0/5, changed state to down
Mar  2 22:30:39 10.0.1.193 UTC: %LINK-3-UPDOWN: Interface GigabitEthernet2/0/5, changed state to up
Mar  2 22:30:40 10.0.1.193 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet2/0/5, changed state to up
Mar  4 00:10:47 10.0.1.193 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet2/0/5, changed state to down
Mar  4 00:10:48 10.0.1.193 UTC: %LINK-3-UPDOWN: Interface GigabitEthernet2/0/5, changed state to down
Mar  4 00:10:51 10.0.1.193 UTC: %LINK-3-UPDOWN: Interface GigabitEthernet2/0/5, changed state to up
Mar  4 00:10:52 10.0.1.193 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet2/0/5, changed state to up
Mar  4 00:15:00 10.0.1.193 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet2/0/5, changed state to down
Mar  4 00:15:01 10.0.1.193 UTC: %LINK-3-UPDOWN: Interface GigabitEthernet2/0/5, changed state to down
Mar  4 00:15:04 10.0.1.193 UTC: %LINK-3-UPDOWN: Interface GigabitEthernet2/0/5, changed state to up
Mar  4 00:15:05 10.0.1.193 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet2/0/5, changed state to up
```

Figure 9: Currrent lines that are contained in the syslog sent from the cisco server to bastia

3.2

On the switch, set

```
snmp-server community read_only_community RO

---

Switch#show snmp community

Community name: read_only_community
Community Index: read_only_community
Community SecurityName: read_only_community
storage-type: nonvolatile active
```

3.3

on servers, install and enable snmpd

```
arccy@nevers snmpwalk -v 1 -c read_only_community localhost | head
SNMPv2-MIB::sysDescr.0 = STRING: Linux nevers 5.5.3-arch1-1 #1 SMP PREEMPT Tue, 11 Feb 2020 15:35:41
+0000 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1036177) 2:52:41.77
SNMPv2-MIB::sysContact.0 = STRING: root@localhost
SNMPv2-MIB::sysName.0 = STRING: nevers
SNMPv2-MIB::sysLocation.0 = STRING: Unknown
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (1) 0:00:00.01
SNMPv2-MIB::sysORID.1 = OID: TUNNEL-MIB::tunnelMIB
SNMPv2-MIB::sysORID.2 = OID: SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.3 = OID: SNMP-MPD-MIB::snmpMPDCompliance
```

3.4

Cacti runs on `http://145.100.104.104/cacti` The credentials for the guest accounts are:
Username: guest
Password:Vyy5nKLf4MFWJ5Gm!

Alternatively (since cacti is buggy) a SNMP/Prometheus/Grafana stack is available at `http://nevers.studlab.os3.nl:3000` with credentials:
Username: guest
Password: supersecureguestpassword

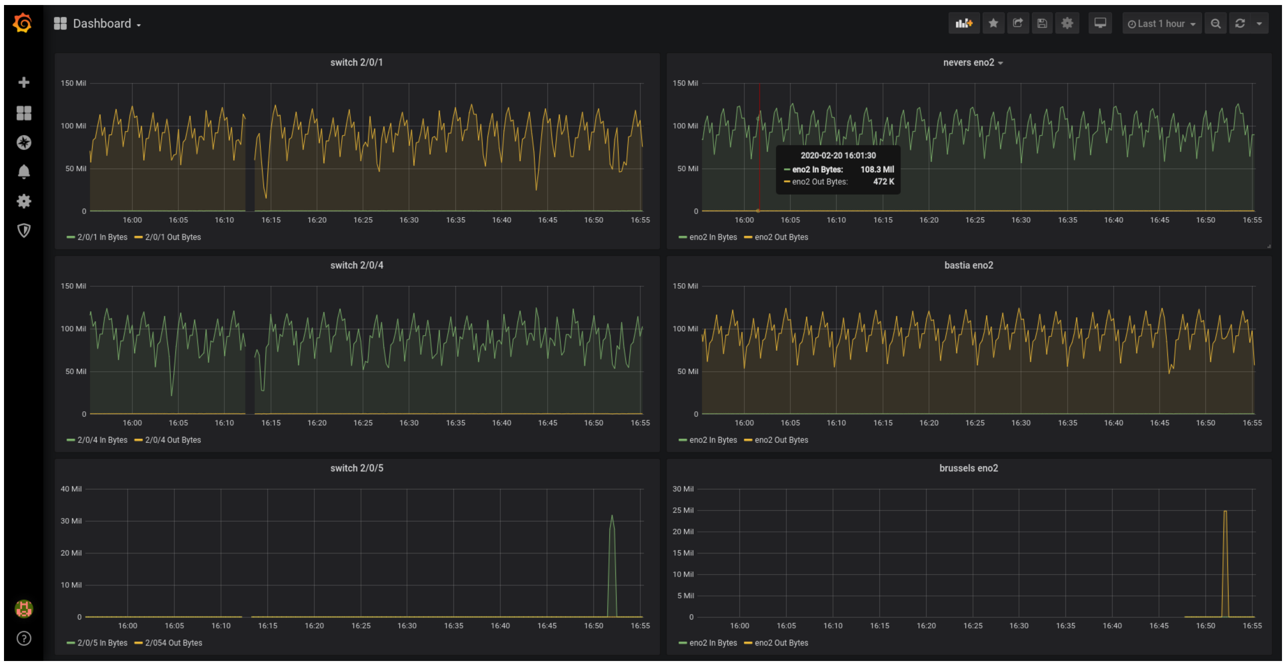


Figure 10: Grafana Dashboard