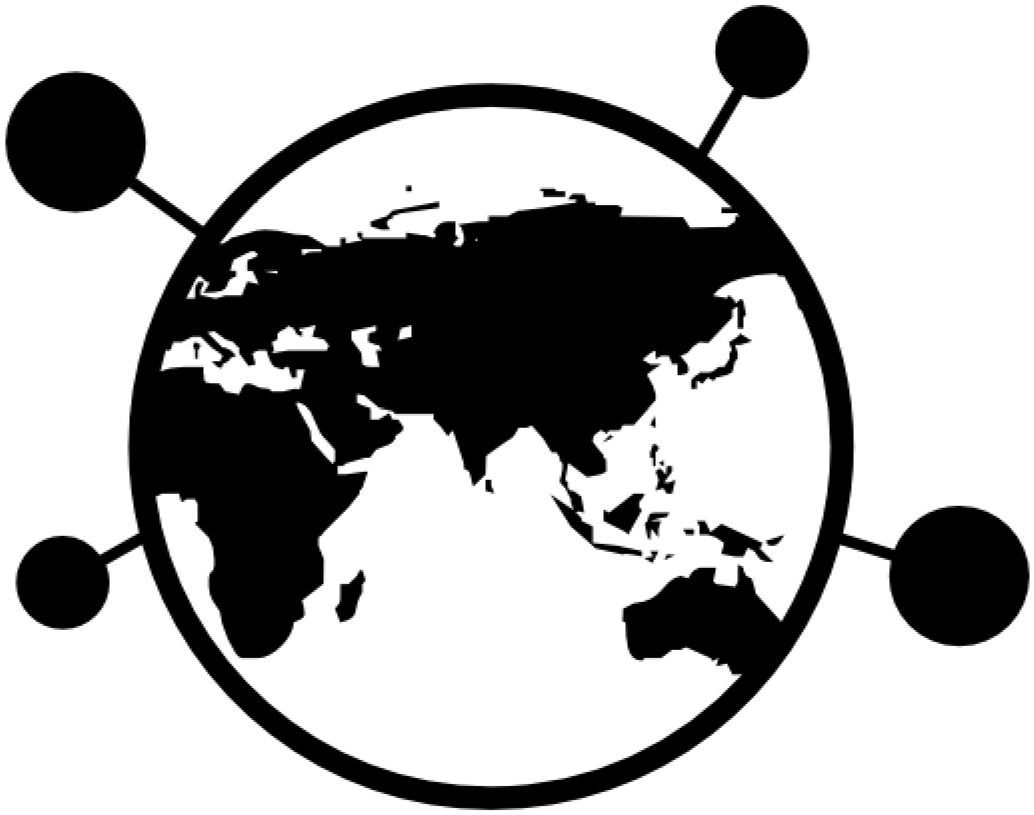
Network Management
Final Project

12/5/2022



SJSU Dept. of Computer Science

CS 158B・Paul Nguyen・Team 4

Anni Shao, Corinna Chang, Kazuya Nakashima, Swift Sheng

Table of Contents

1 Project Introduction

A group of four walking on the street of San Jose downtown. Boooooom! A UFO came out of nowhere and closed its distance at an astonishing speed, hovering over this group. In blinding tractor lights, those four people are taken from the group and disappeared into that mysterious flying saucer. Waking up from a coma, the group found out: Aliens are very curious about how to set up a small network and those four are the best of 158B.

The aliens required us to set up a POC for a small network which consisted of 12 devices and 3 different platforms. We were asked to provide the following features:

1. Able to discover the entire network.
2. Able to receive syslog from network devices.
3. Able to receive snmp messages from network devices.
4. Able to receive events when the interface or device is down.
5. Able to run snmp mibwalk.
6. Able to compile a new snmp mib when customers purchase a new device.
7. Able to make changes through snmp set operation.
8. Able to monitor device resources such as memory, bandwidth and cpu utilization.
9. Able to run automation scripts.
10. Able to set up an automatic task operation, dynamic topology.

2 Proposed Solution

2.1. Tools used to simulate network devices

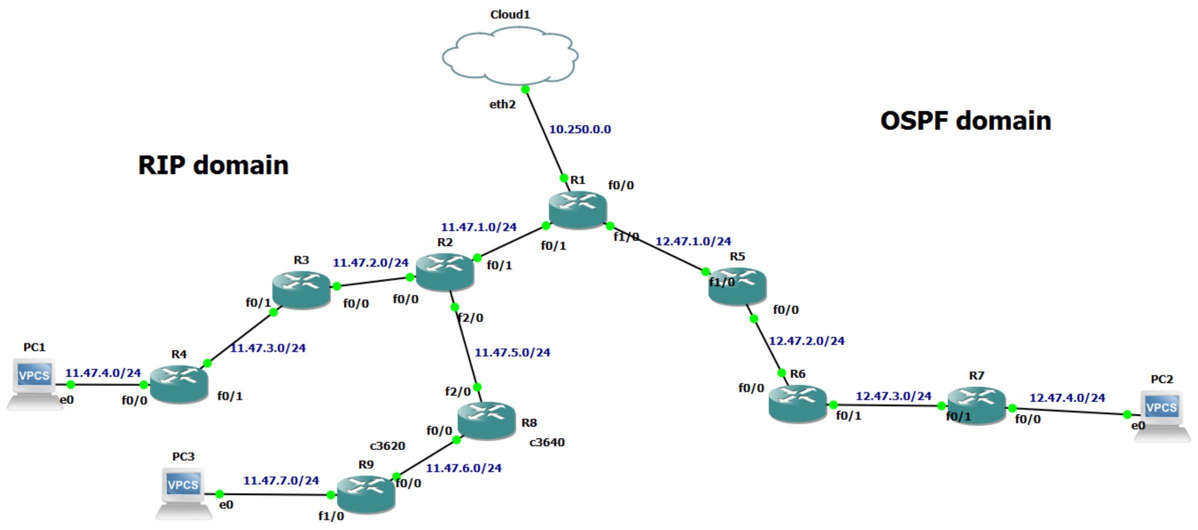
- GNS3 & VirtualBox
- MG-Soft
- PRTG

Graphical Network Simulator-3 (GNS3) is a network software used to simulate complex networks. It allows both virtual and real devices to be implemented into the network and be able to communicate with each other. In addition, a GNS3 server (GNS3 VM) is also installed and is runned using VirtualBox. The server is for running any virtual devices necessary for emulating the network. By using both GNS3 and GNS3 VM, we are able to simulate a network that includes both virtual devices (running through VirtualBox) and the localhost.

MG-SOFT MIB Browser is network monitoring software that is used for monitoring and managing any SNMP device on the network. This MIB Browser uses SNMPv1, SNMPv2, SNMPv3 protocols over UDP and TCP in IPv4 and IPv6 network, for our project it will be used to monitor IPv4 network. Also, MG-SOFT MIB Browser allows SNMP Get, GetNext, GetBulk and SNMP Set operations. This SNMP Set operation will be used to make changes in the network.

Paessler Router Traffic Grapher (PRTG) is a network monitoring software that is used to monitor and collect statistics from system conditions. In our project, PRTG is used to monitor our network hosts, such as routers, PCs, and the localhost. PRTG is able to monitor a wide range of system conditions, such as uptime, system health, and CPU load. We placed an emphasis on the syslog receiver and SNMP trap receiver sensors to monitor any changes made to the hosts. In addition, PRTG is able to automatically notify users when any anomaly is detected on the network.

2.2. Topology



Our network consists of a total of 12 devices and 3 different platforms that use routers c3620, c3640, and c3725. R1, R2, R3, and R4 use c3620 routers; R5, R6, and R7 use c3725 routers; and R8 and R9 use c3640 routers. PC1 is connected to the network through R4, while PC2 is connected through R7. Lastly, PC3 is connected through R9. The localhost is connected to this network through R1's connection with Cloud1.

2.3. Protocols used

- RIP protocol: R1, R2, R3, R4, R8, R9
- OSPF protocol: R1, R5, R6, R7

The network uses two different protocols: Routing Information Protocol (RIP) and Open Shortest Path First (OSPF). OSPF is an interior gateway protocol that is used to distribute IP routing information through a single AS. RIP is a dynamic routing protocol that counts hops as the metric to determine the best path, with the maximum hop being 15. On the other hand, OSPF is more intelligent and usually implemented by larger organizations, as it consumes more computation power and memory than RIP.

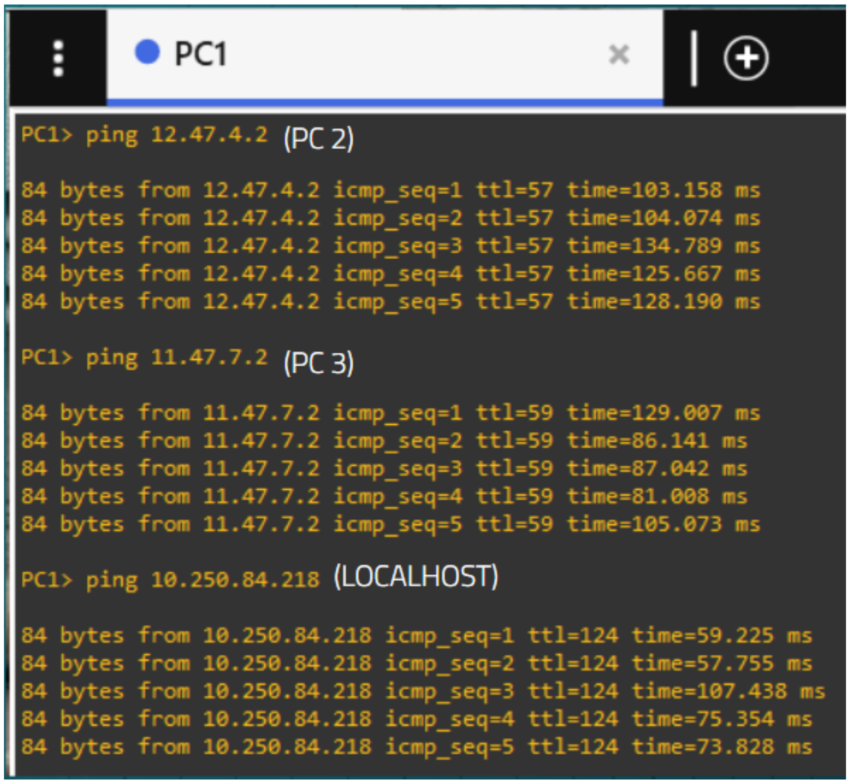
2.4. How connectivity is tested

Connectivity is tested by pinging from each device to all other devices on the network, which includes all PCs on the topology and the localhost. This process is further explained in section 3.1 of this report.

3 Validation of all requested features

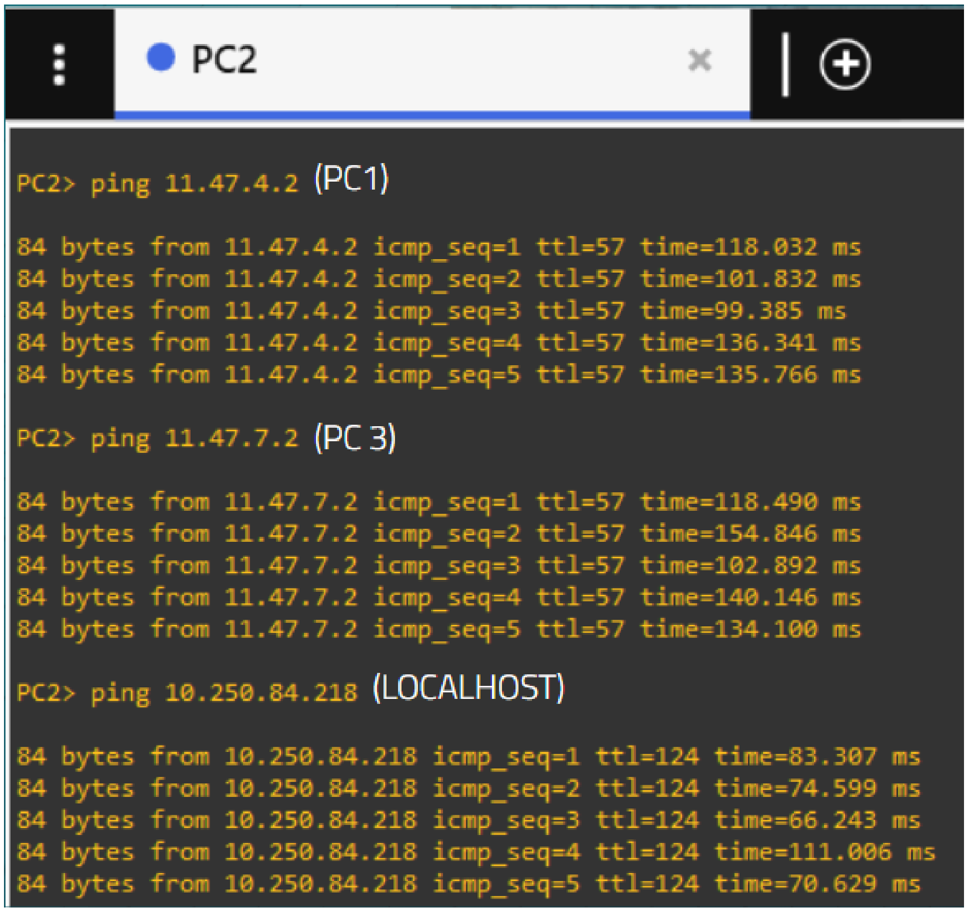
3.1. Able to discover the entire network

To ensure that the entire network is discoverable, we tested out the connectivity of each device by pinging to all other devices on the network. As seen in the images below, PC1, PC2, PC3, and the localhost are all able to ping each other. This way we know that the entire network is connected and discoverable.



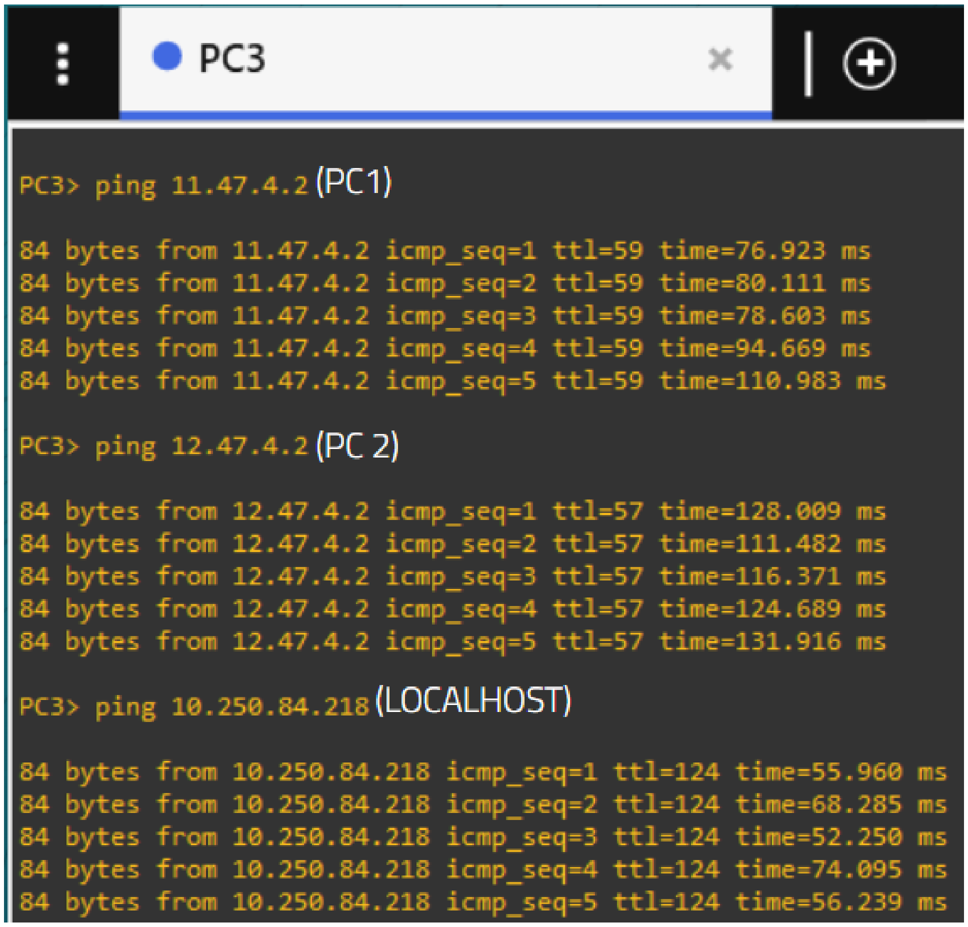
```
PC1> ping 12.47.4.2 (PC 2)

84 bytes from 12.47.4.2 icmp_seq=1 ttl=57 time=103.158 ms
84 bytes from 12.47.4.2 icmp_seq=2 ttl=57 time=104.074 ms
84 bytes from 12.47.4.2 icmp_seq=3 ttl=57 time=134.789 ms
84 bytes from 12.47.4.2 icmp_seq=4 ttl=57 time=125.667 ms
84 bytes from 12.47.4.2 icmp_seq=5 ttl=57 time=128.190 ms

PC1> ping 11.47.7.2 (PC 3)

84 bytes from 11.47.7.2 icmp_seq=1 ttl=59 time=129.007 ms
84 bytes from 11.47.7.2 icmp_seq=2 ttl=59 time=86.141 ms
84 bytes from 11.47.7.2 icmp_seq=3 ttl=59 time=87.042 ms
84 bytes from 11.47.7.2 icmp_seq=4 ttl=59 time=81.008 ms
84 bytes from 11.47.7.2 icmp_seq=5 ttl=59 time=105.073 ms

PC1> ping 10.250.84.218 (LOCALHOST)

84 bytes from 10.250.84.218 icmp_seq=1 ttl=124 time=59.225 ms
84 bytes from 10.250.84.218 icmp_seq=2 ttl=124 time=57.755 ms
84 bytes from 10.250.84.218 icmp_seq=3 ttl=124 time=107.438 ms
84 bytes from 10.250.84.218 icmp_seq=4 ttl=124 time=75.354 ms
84 bytes from 10.250.84.218 icmp_seq=5 ttl=124 time=73.828 ms
```

Fig. 3.1.1 Pinging from PC1 to PC2, PC3, and localhost.



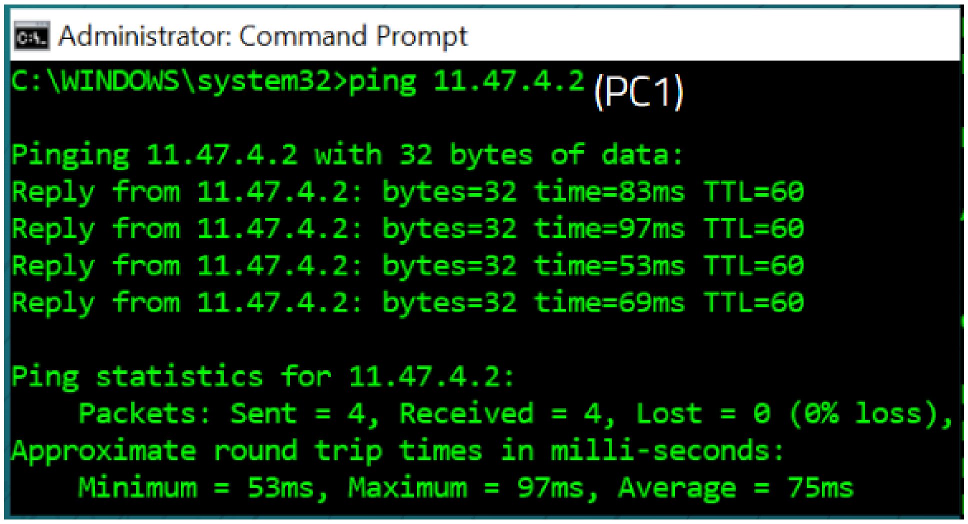
```
PC2> ping 11.47.4.2 (PC1)

84 bytes from 11.47.4.2 icmp_seq=1 ttl=57 time=118.032 ms
84 bytes from 11.47.4.2 icmp_seq=2 ttl=57 time=101.832 ms
84 bytes from 11.47.4.2 icmp_seq=3 ttl=57 time=99.385 ms
84 bytes from 11.47.4.2 icmp_seq=4 ttl=57 time=136.341 ms
84 bytes from 11.47.4.2 icmp_seq=5 ttl=57 time=135.766 ms

PC2> ping 11.47.7.2 (PC 3)

84 bytes from 11.47.7.2 icmp_seq=1 ttl=57 time=118.490 ms
84 bytes from 11.47.7.2 icmp_seq=2 ttl=57 time=154.846 ms
84 bytes from 11.47.7.2 icmp_seq=3 ttl=57 time=102.892 ms
84 bytes from 11.47.7.2 icmp_seq=4 ttl=57 time=140.146 ms
84 bytes from 11.47.7.2 icmp_seq=5 ttl=57 time=134.100 ms

PC2> ping 10.250.84.218 (LOCALHOST)

84 bytes from 10.250.84.218 icmp_seq=1 ttl=124 time=83.307 ms
84 bytes from 10.250.84.218 icmp_seq=2 ttl=124 time=74.599 ms
84 bytes from 10.250.84.218 icmp_seq=3 ttl=124 time=66.243 ms
84 bytes from 10.250.84.218 icmp_seq=4 ttl=124 time=111.006 ms
84 bytes from 10.250.84.218 icmp_seq=5 ttl=124 time=70.629 ms
```

Fig. 3.1.2 Pinging from PC2 to PC1, PC3, and localhost.



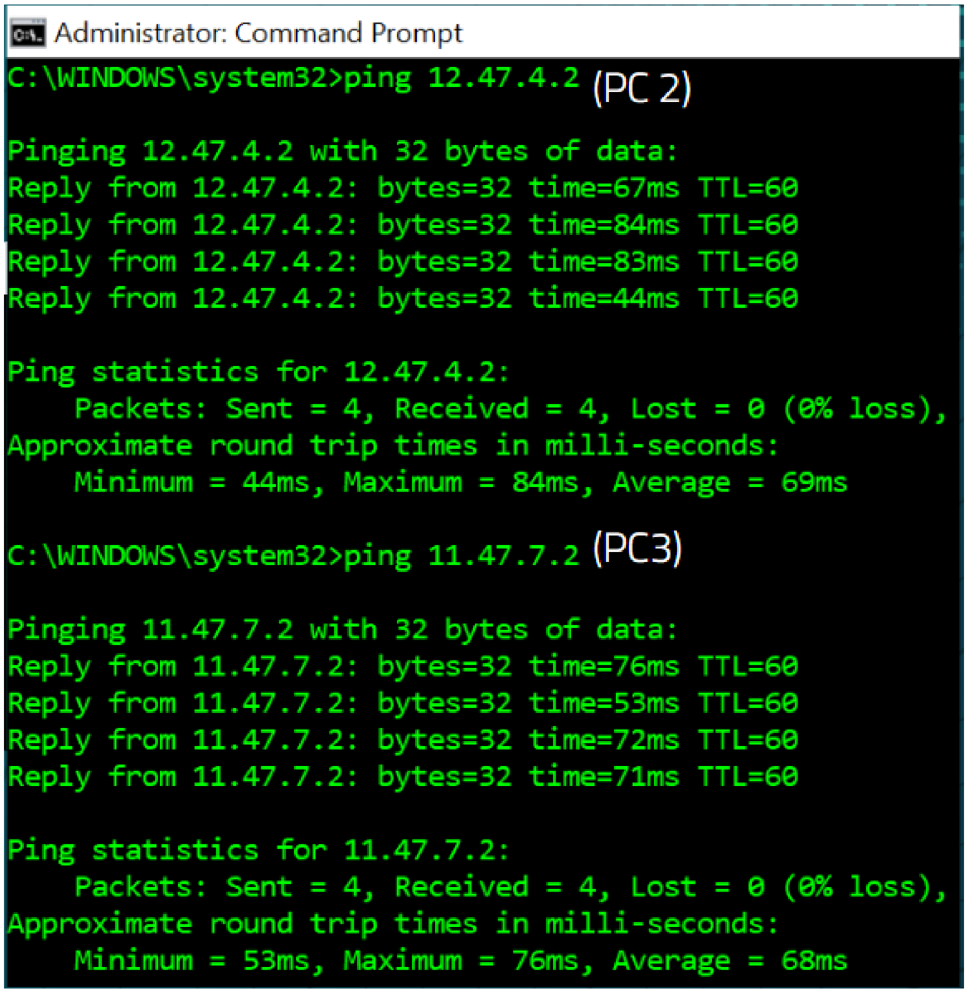
```
PC3> ping 11.47.4.2 (PC1)

84 bytes from 11.47.4.2 icmp_seq=1 ttl=59 time=76.923 ms
84 bytes from 11.47.4.2 icmp_seq=2 ttl=59 time=80.111 ms
84 bytes from 11.47.4.2 icmp_seq=3 ttl=59 time=78.603 ms
84 bytes from 11.47.4.2 icmp_seq=4 ttl=59 time=94.669 ms
84 bytes from 11.47.4.2 icmp_seq=5 ttl=59 time=110.983 ms

PC3> ping 12.47.4.2 (PC 2)

84 bytes from 12.47.4.2 icmp_seq=1 ttl=57 time=128.009 ms
84 bytes from 12.47.4.2 icmp_seq=2 ttl=57 time=111.482 ms
84 bytes from 12.47.4.2 icmp_seq=3 ttl=57 time=116.371 ms
84 bytes from 12.47.4.2 icmp_seq=4 ttl=57 time=124.689 ms
84 bytes from 12.47.4.2 icmp_seq=5 ttl=57 time=131.916 ms

PC3> ping 10.250.84.218 (LOCALHOST)

84 bytes from 10.250.84.218 icmp_seq=1 ttl=124 time=55.960 ms
84 bytes from 10.250.84.218 icmp_seq=2 ttl=124 time=68.285 ms
84 bytes from 10.250.84.218 icmp_seq=3 ttl=124 time=52.250 ms
84 bytes from 10.250.84.218 icmp_seq=4 ttl=124 time=74.095 ms
84 bytes from 10.250.84.218 icmp_seq=5 ttl=124 time=56.239 ms
```

Fig. 3.1.3 Pinging from PC3 to PC1, PC2, and localhost.



```
C:\WINDOWS\system32>ping 11.47.4.2 (PC1)

Pinging 11.47.4.2 with 32 bytes of data:
Reply from 11.47.4.2: bytes=32 time=83ms TTL=60
Reply from 11.47.4.2: bytes=32 time=97ms TTL=60
Reply from 11.47.4.2: bytes=32 time=53ms TTL=60
Reply from 11.47.4.2: bytes=32 time=69ms TTL=60

Ping statistics for 11.47.4.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 53ms, Maximum = 97ms, Average = 75ms
```

Fig.3.1.4 Pinging from localhost to PC1.



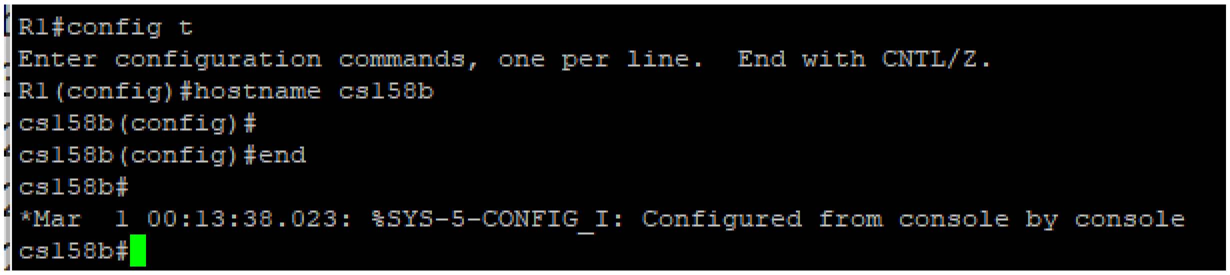
```
Administrator: Command Prompt
C:\WINDOWS\system32>ping 12.47.4.2 (PC 2)

Pinging 12.47.4.2 with 32 bytes of data:
Reply from 12.47.4.2: bytes=32 time=67ms TTL=60
Reply from 12.47.4.2: bytes=32 time=84ms TTL=60
Reply from 12.47.4.2: bytes=32 time=83ms TTL=60
Reply from 12.47.4.2: bytes=32 time=44ms TTL=60

Ping statistics for 12.47.4.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 44ms, Maximum = 84ms, Average = 69ms

C:\WINDOWS\system32>ping 11.47.7.2 (PC3)

Pinging 11.47.7.2 with 32 bytes of data:
Reply from 11.47.7.2: bytes=32 time=76ms TTL=60
Reply from 11.47.7.2: bytes=32 time=53ms TTL=60
Reply from 11.47.7.2: bytes=32 time=72ms TTL=60
Reply from 11.47.7.2: bytes=32 time=71ms TTL=60

Ping statistics for 11.47.7.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 53ms, Maximum = 76ms, Average = 68ms
```

Fig. 3.1.5 Pinging from localhost to PC2 and PC3.

3.2. Able to receive syslog from network devices

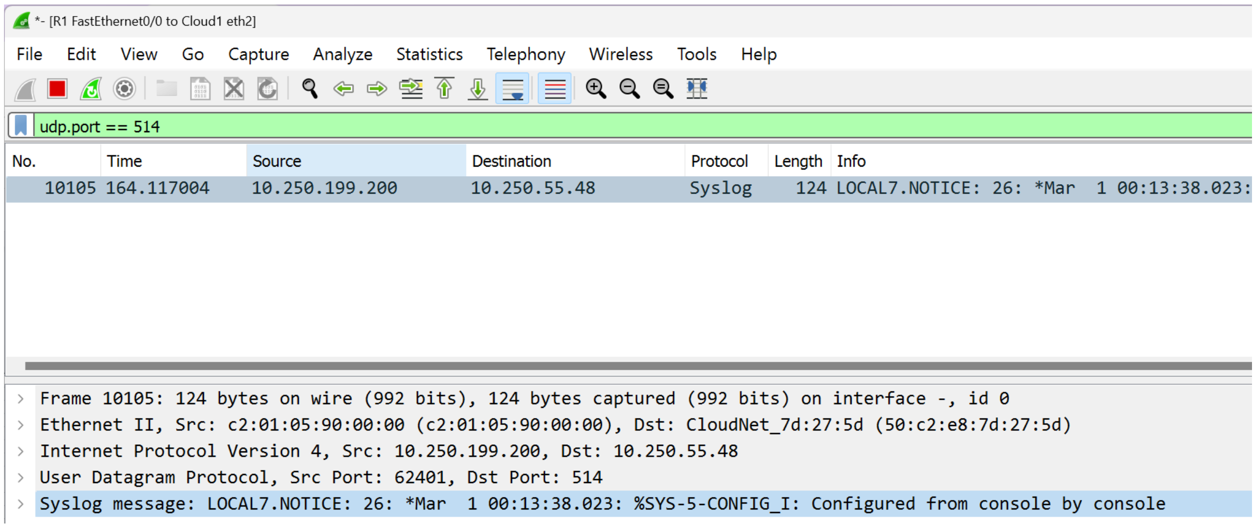
First, we open a console for R1 in GNS3 and make a basic modification to change the hostname of the router. After making this change, we expect to see a syslog message to be sent out.

Figure 3.2.1 shows that the syslog message has successfully been sent out in Wireshark. The message is being sent out from 10.250.199.200, the router IP from the cloud, into 10.250.55.48, which is the local computer IP. The UDP port being used is 514. The timestamp in the message content matches up with the timestamp shown in Figure 3.2.0, meaning that syslog is correctly received.



```
R1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#hostname cs158b
cs158b(config)#
cs158b(config)#end
cs158b#
*Mar  1 00:13:38.023: %SYS-5-CONFIG_I: Configured from console by console
cs158b#
```

Fig. 3.2.0 - GNS3 modifications



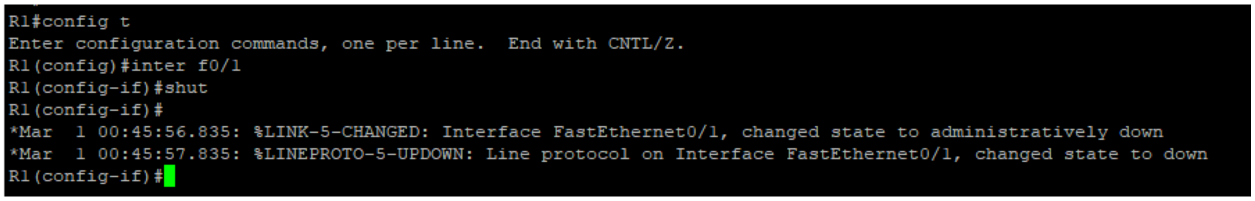
*- [R1 FastEthernet0/0 to Cloud1 eth2]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
udp.port == 514

No.    Time        Source          Destination   Protocol Length Info
 10105 164.117004  10.250.199.200  10.250.55.48  Syslog      124 LOCAL7.NOTICE: 26: *Mar  1 00:13:38.023:

> Frame 10105: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface -, id 0
> Ethernet II, Src: c2:01:05:90:00:00 (c2:01:05:90:00:00), Dst: CloudNet_7d:27:5d (50:c2:e8:7d:27:5d)
> Internet Protocol Version 4, Src: 10.250.199.200, Dst: 10.250.55.48
> User Datagram Protocol, Src Port: 62401, Dst Port: 514
> Syslog message: LOCAL7.NOTICE: 26: *Mar  1 00:13:38.023: %SYS-5-CONFIG_I: Configured from console by console
```

Fig. 3.2.1 - Verify syslog messages come through in Wireshark

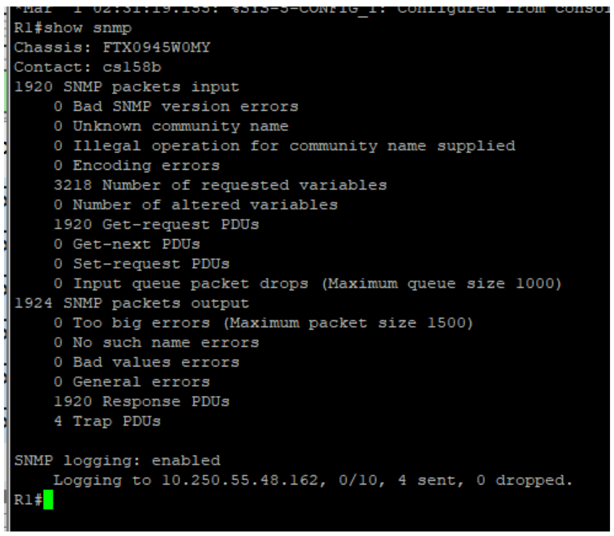
3.3. Able to receive snmp

First, we make an interface change in the console of R1 in GNS3 - we are shutting interface f0/1 down. To verify that snmp messages are sent out as a result of this, we can use the "show snmp" command in the GNS3 R1 console to see that a majority of the packets sent involve get-request and response. Similarly, this is reflected in Figure 3.3.2; we see that almost all packets in the screenshot are of info get-request and get-response in WireShark when we filter only packets being sent through the UDP port of 161.



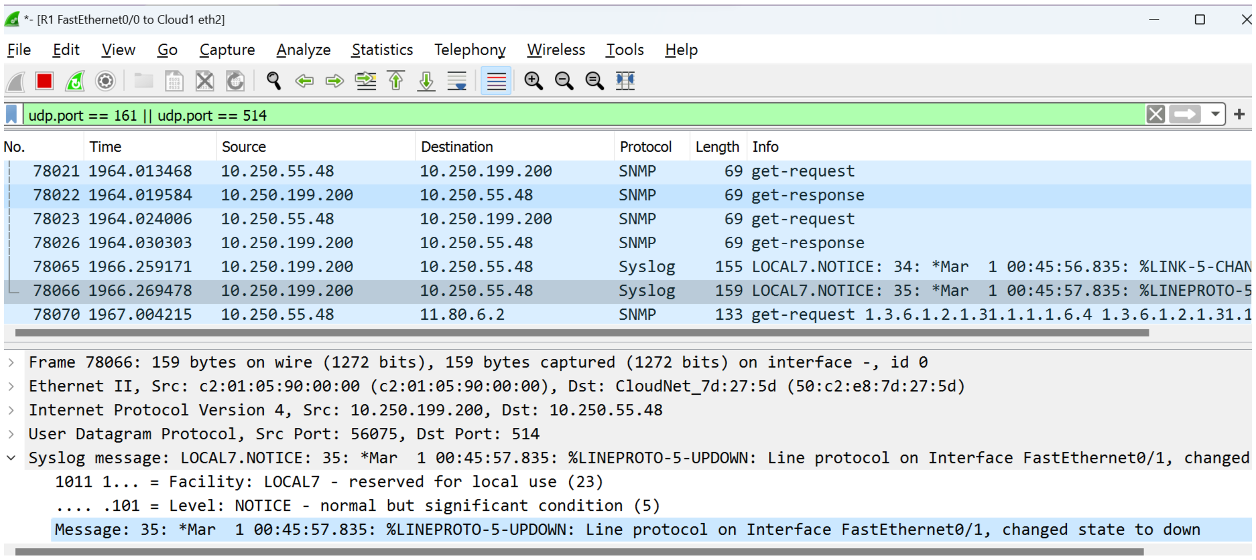
```
R1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#inter f0/1
R1(config-if)#shut
R1(config-if)#
*Mar  1 00:45:56.835: %LINK-5-CHANGED: Interface FastEthernet0/1, changed state to administratively down
*Mar  1 00:45:57.835: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to down
R1(config-if)#
```

Fig. 3.3.0 - Making interface change



```
*Mar  1 02:31:19.155: %SYS-5-CONFIG_I: Configured from consol
R1#show snmp
Chassis: FTX0945W0MY
Contact: cs158b
1920 SNMP packets input
    0 Bad SNMP version errors
    0 Unknown community name
    0 Illegal operation for community name supplied
    0 Encoding errors
    3218 Number of requested variables
    0 Number of altered variables
    1920 Get-request PDUs
    0 Get-next PDUs
    0 Set-request PDUs
    0 Input queue packet drops (Maximum queue size 1000)
1924 SNMP packets output
    0 Too big errors (Maximum packet size 1500)
    0 No such name errors
    0 Bad values errors
    0 General errors
    1920 Response PDUs
    4 Trap PDUs

SNMP logging: enabled
    Logging to 10.250.55.48.162, 0/10, 4 sent, 0 dropped.
R1#
```

Fig. 3.3.1 - SNMP messages in GNS3



*- [R1 FastEthernet0/0 to Cloud1 eth2]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
udp.port == 161 || udp.port == 514

No.     Time          Source          Destination     Protocol  Length Info
  78021 1964.013468   10.250.55.48    10.250.199.200  SNMP        69 get-request
  78022 1964.019584   10.250.199.200  10.250.55.48    SNMP        69 get-response
  78023 1964.024006   10.250.55.48    10.250.199.200  SNMP        69 get-request
  78026 1964.030303   10.250.199.200  10.250.55.48    SNMP        69 get-response
  78065 1966.259171   10.250.199.200  10.250.55.48    Syslog     155 LOCAL7.NOTICE: 34: *Mar  1 00:45:56.835: %LINK-5-CHAN
  78066 1966.269478   10.250.199.200  10.250.55.48    Syslog     159 LOCAL7.NOTICE: 35: *Mar  1 00:45:57.835: %LINEPROTO-5
  78070 1967.004215   10.250.55.48    11.80.6.2       SNMP       133 get-request 1.3.6.1.2.1.31.1.1.1.6.4 1.3.6.1.2.1.31.1

> Frame 78066: 159 bytes on wire (1272 bits), 159 bytes captured (1272 bits) on interface -, id 0
> Ethernet II, Src: c2:01:05:90:00:00 (c2:01:05:90:00:00), Dst: CloudNet_7d:27:5d (50:c2:e8:7d:27:5d)
> Internet Protocol Version 4, Src: 10.250.199.200, Dst: 10.250.55.48
> User Datagram Protocol, Src Port: 56075, Dst Port: 514
v Syslog message: LOCAL7.NOTICE: 35: *Mar  1 00:45:57.835: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed
    1011 1... = Facility: LOCAL7 - reserved for local use (23)
    .... .101 = Level: NOTICE - normal but significant condition (5)
    Message: 35: *Mar  1 00:45:57.835: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to down
```

Fig. 3.3.2 - SNMP messages in Wireshark

3.4. Able to receive event when interface or device is down

As seen in Figure 3.4.1, when a device is running properly, all sensors will have a green check mark next to them. When a device is down, all sensors will be down and the Ping sensor will raise an alert (seen in Figure 3.4.2). When such an event occurs, PRTG will automatically send out notification messages to the admin that the device is down. This allows users to monitor the devices on their network and be notified when any interface or device is down. Specific notification messages and methods can also be configured within PRTG, such as when the syslog receiver or SNMP trap receiver receives an alert from the devices.

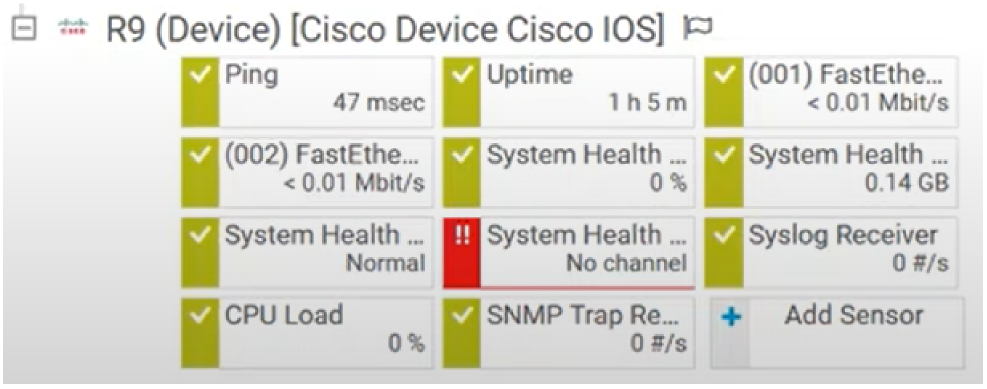

R9 (Device) [Cisco Device Cisco IOS]
Ping 47 msec
Uptime 1 h 5 m
(001) FastEthe... < 0.01 Mbit/s
(002) FastEthe... < 0.01 Mbit/s
System Health ... 0 %
System Health ... 0.14 GB
System Health ... Normal
System Health ... No channel
Syslog Receiver 0 #/s
CPU Load 0 %
SNMP Trap Re... 0 #/s
Add Sensor

Fig. 3.4.1 Screenshot from PRTG Network Monitor when device R9 up and running.

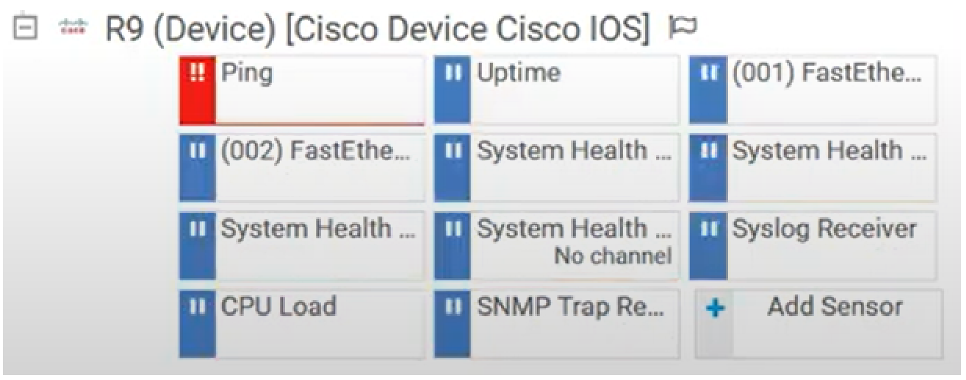
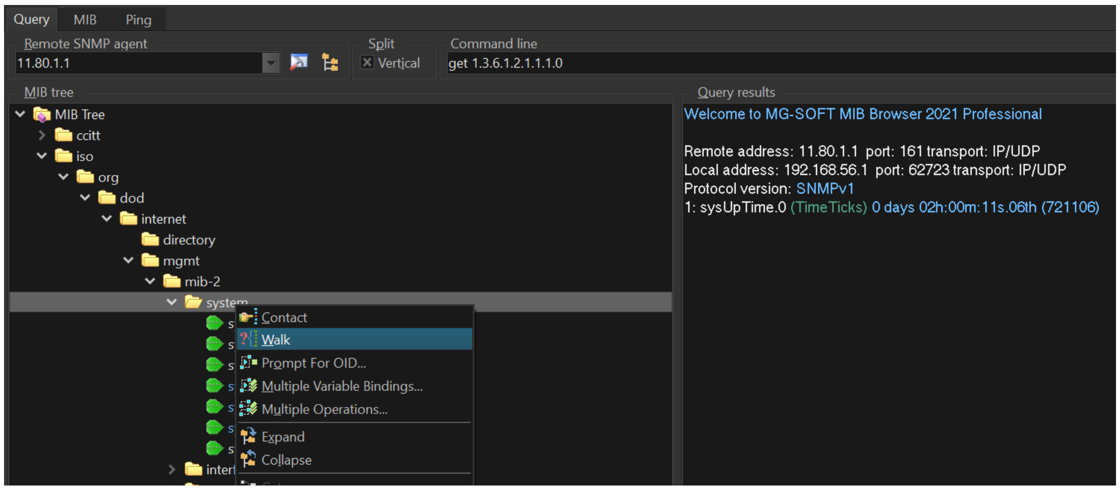


Fig. 3.4.2 Screenshot from PRTG Network Monitor when device R9 is down.

3.5. Able to run snmp mibwalk

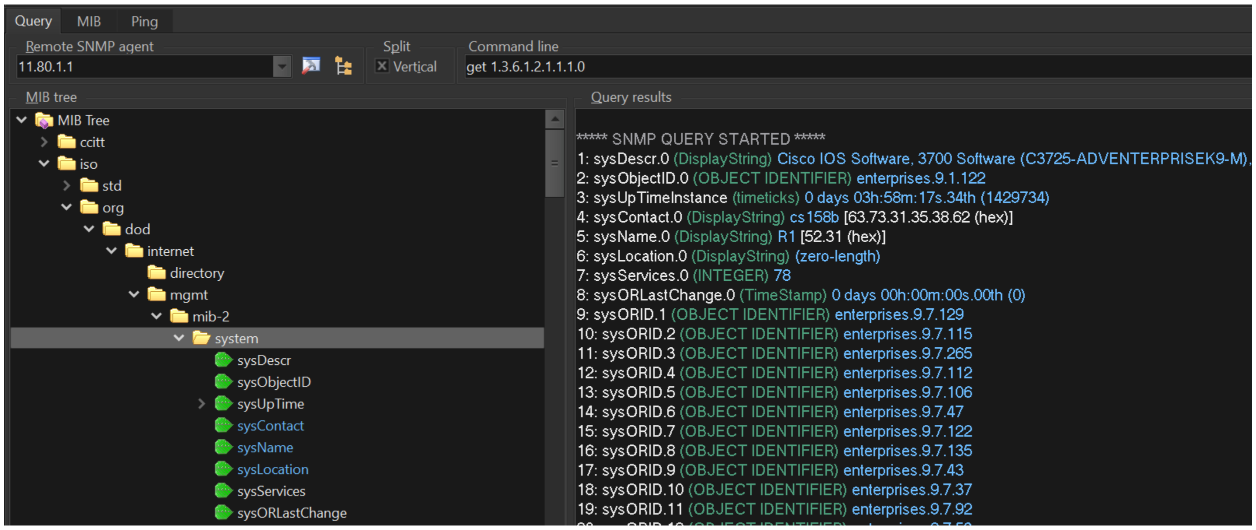
With the remote SNMP agent of 11.80.1.1 (R1), we first use the contact command, then use the walk command by right clicking on the system folder in MgSoft MIB Browser. The query results are shown in Figure 3.5.1. The results show the information of all the variables of the system, particularly for R1. It takes 3 seconds in total to run the entire MIBwalk.



```
get 1.3.6.1.2.1.1.1.0
```

```
Welcome to MG-SOFT MIB Browser 2021 Professional

Remote address: 11.80.1.1  port: 161 transport: IP/UDP
Local address: 192.168.56.1  port: 62723 transport: IP/UDP
Protocol version: SNMPv1
1: sysUpTime.0 (TimeTicks) 0 days 02h:00m:11s.06th (721106)
```

Fig. 3.5.0 - Run command



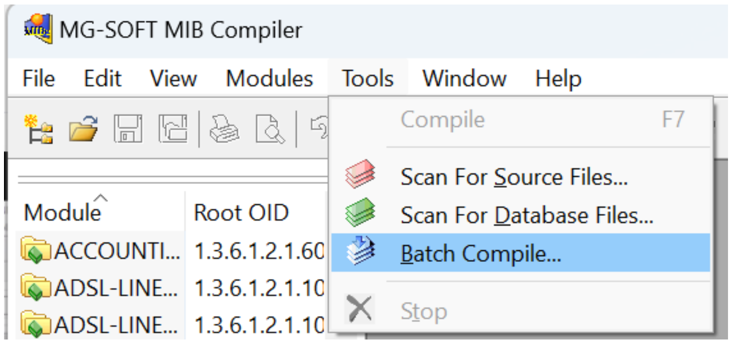
```
get 1.3.6.1.2.1.1.1.0
```

```
***** SNMP QUERY STARTED *****
1: sysDescr.0 (DisplayString) Cisco IOS Software, 3700 Software (C3725-ADVENTERPRISEK9-M),
2: sysObjectID.0 (OBJECT IDENTIFIER) enterprises.9.1.122
3: sysUpTimeInstance (timeticks) 0 days 03h:58m:17s.34th (1429734)
4: sysContact.0 (DisplayString) cs158b [63.73.31.35.38.62 (hex)]
5: sysName.0 (DisplayString) R1 [52.31 (hex)]
6: sysLocation.0 (DisplayString) (zero-length)
7: sysServices.0 (INTEGER) 78
8: sysORLastChange.0 (TimeStamp) 0 days 00h:00m:00s.00th (0)
9: sysORID.1 (OBJECT IDENTIFIER) enterprises.9.7.129
10: sysORID.2 (OBJECT IDENTIFIER) enterprises.9.7.115
11: sysORID.3 (OBJECT IDENTIFIER) enterprises.9.7.265
12: sysORID.4 (OBJECT IDENTIFIER) enterprises.9.7.112
13: sysORID.5 (OBJECT IDENTIFIER) enterprises.9.7.106
14: sysORID.6 (OBJECT IDENTIFIER) enterprises.9.7.47
15: sysORID.7 (OBJECT IDENTIFIER) enterprises.9.7.122
16: sysORID.8 (OBJECT IDENTIFIER) enterprises.9.7.135
17: sysORID.9 (OBJECT IDENTIFIER) enterprises.9.7.43
18: sysORID.10 (OBJECT IDENTIFIER) enterprises.9.7.37
19: sysORID.11 (OBJECT IDENTIFIER) enterprises.9.7.92
```

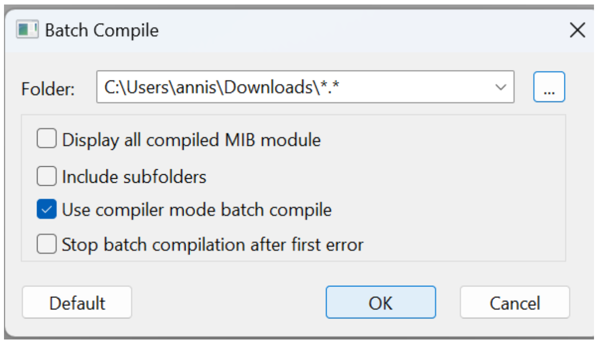
Fig. 3.5.1 - Walk query results

3.6. Able to compile a new snmp mib when a new device is purchased

We use the batch compile command found in the tools tab of the Mg Soft MIB compiler. After providing the folder that the SNMP MIB source code is in, use the default options and press OK. Afterwards, a new tab titled "Compiled MIB Modules" will pop up with the compiled modules. Pressing save all with the default options again should be sufficient to save the new SNMP MIB for future use.

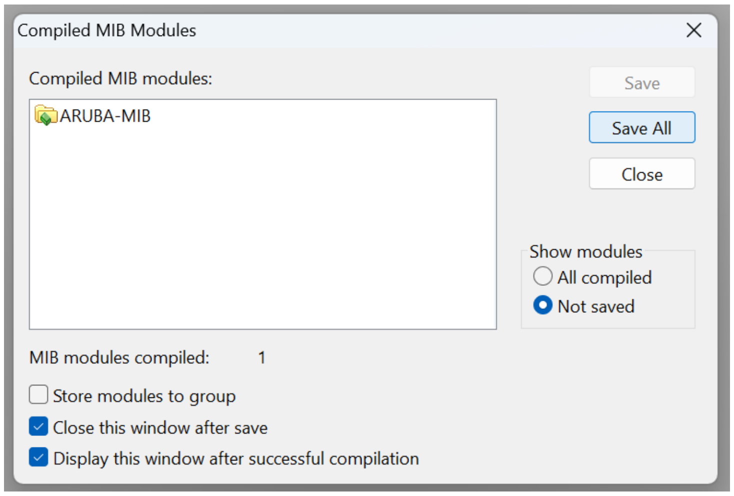


Module	Root OID
ACCOUNTI...	1.3.6.1.2.1.60
ADSL-LINE...	1.3.6.1.2.1.10
ADSL-LINE...	1.3.6.1.2.1.10



Folder: C:\Users\annis\Downloads\*.*

- [ ] Display all compiled MIB module
- [ ] Include subfolders
- [x] Use compiler mode batch compile
- [ ] Stop batch compilation after first error



Compiled MIB modules:
ARUBA-MIB

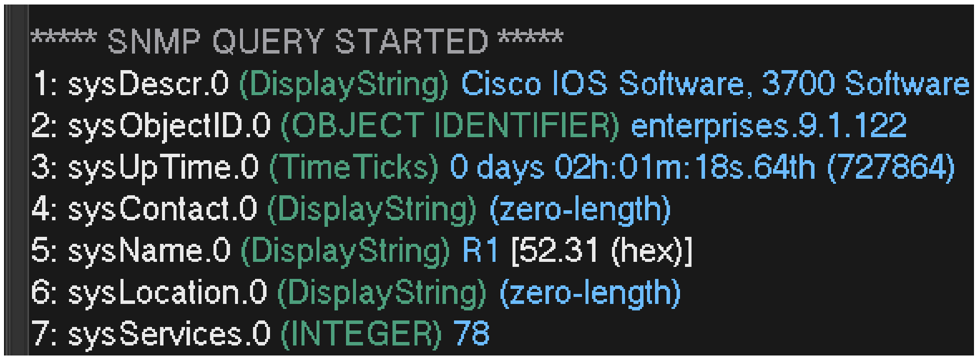
Show modules
○ All compiled
● Not saved

MIB modules compiled: 1

- [ ] Store modules to group
- [x] Close this window after save
- [x] Display this window after successful compilation

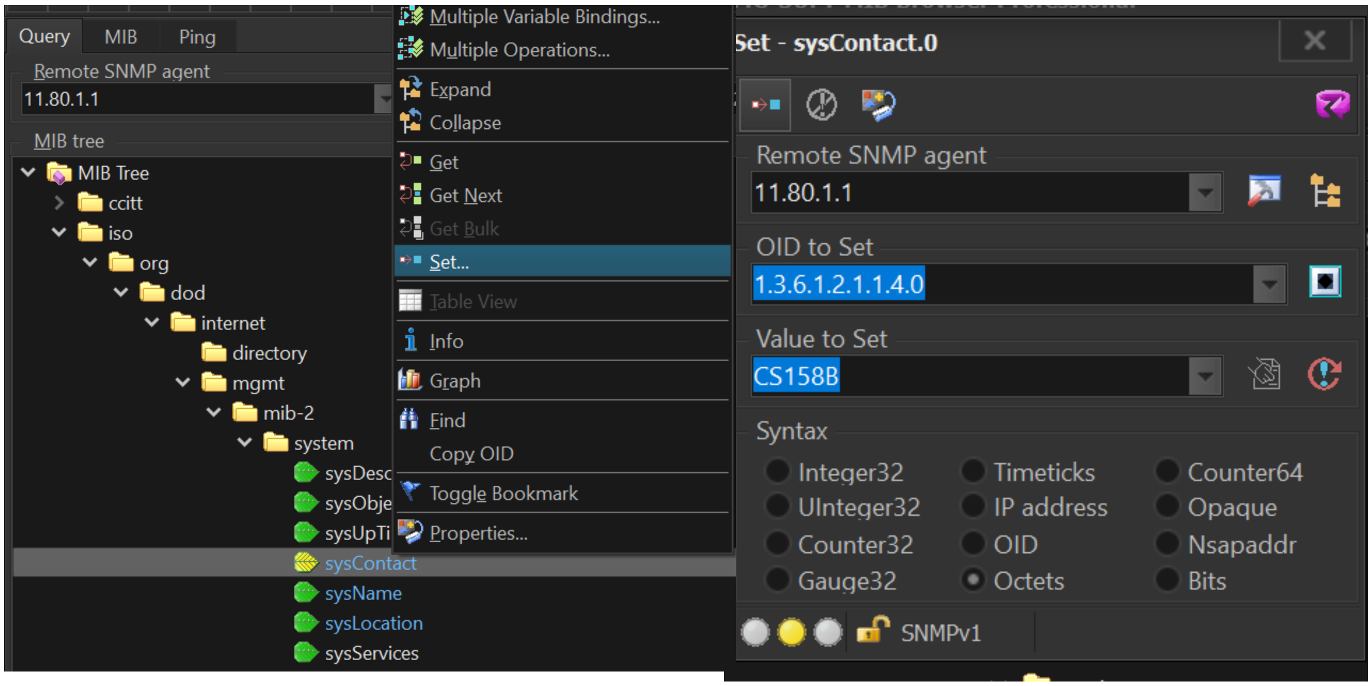
3.7. Able to make changes through snmp set operation

Figure 3.7.0 shows the original values. The goal of this validation demo is to change the contact variable. Originally, it was a DisplayString of a length of 0. Then, in MIB Browser we can right click on the sysContact variable to set it to a new value, like “CS158B”. Figure 3.7.2 shows the results of setting the variable - sysContact is successfully set to CS158B.



```
***** SNMP QUERY STARTED *****
1: sysDescr.0 (DisplayString) Cisco IOS Software, 3700 Software
2: sysObjectID.0 (OBJECT IDENTIFIER) enterprises.9.1.122
3: sysUpTime.0 (TimeTicks) 0 days 02h:01m:18s.64th (727864)
4: sysContact.0 (DisplayString) (zero-length)
5: sysName.0 (DisplayString) R1 [52.31 (hex)]
6: sysLocation.0 (DisplayString) (zero-length)
7: sysServices.0 (INTEGER) 78
```

Fig. 3.7.0 - Original values



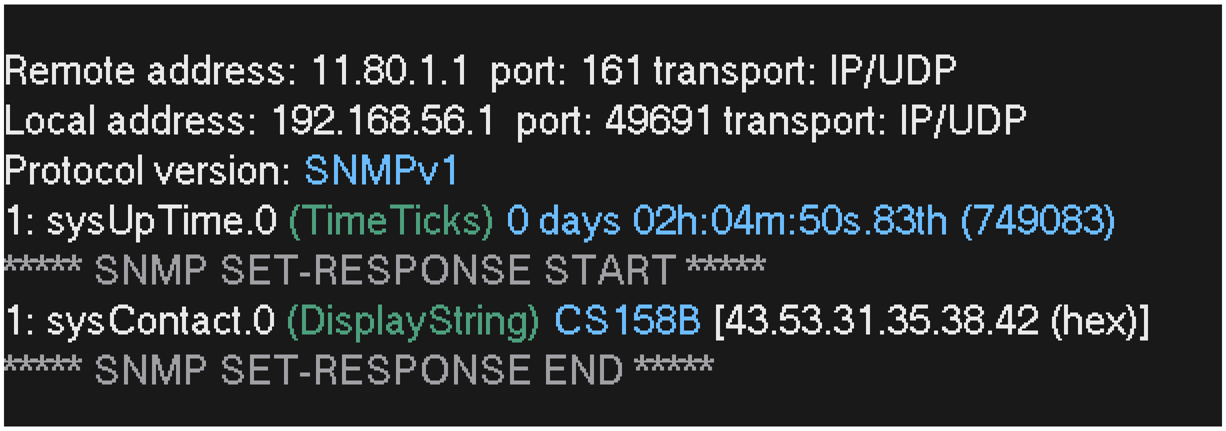
Remote SNMP agent: 11.80.1.1

OID to Set: 1.3.6.1.2.1.1.4.0

Value to Set: CS158B

Syntax: [✓ Octets]

Fig. 3.7.1 - Set operation on contact variable



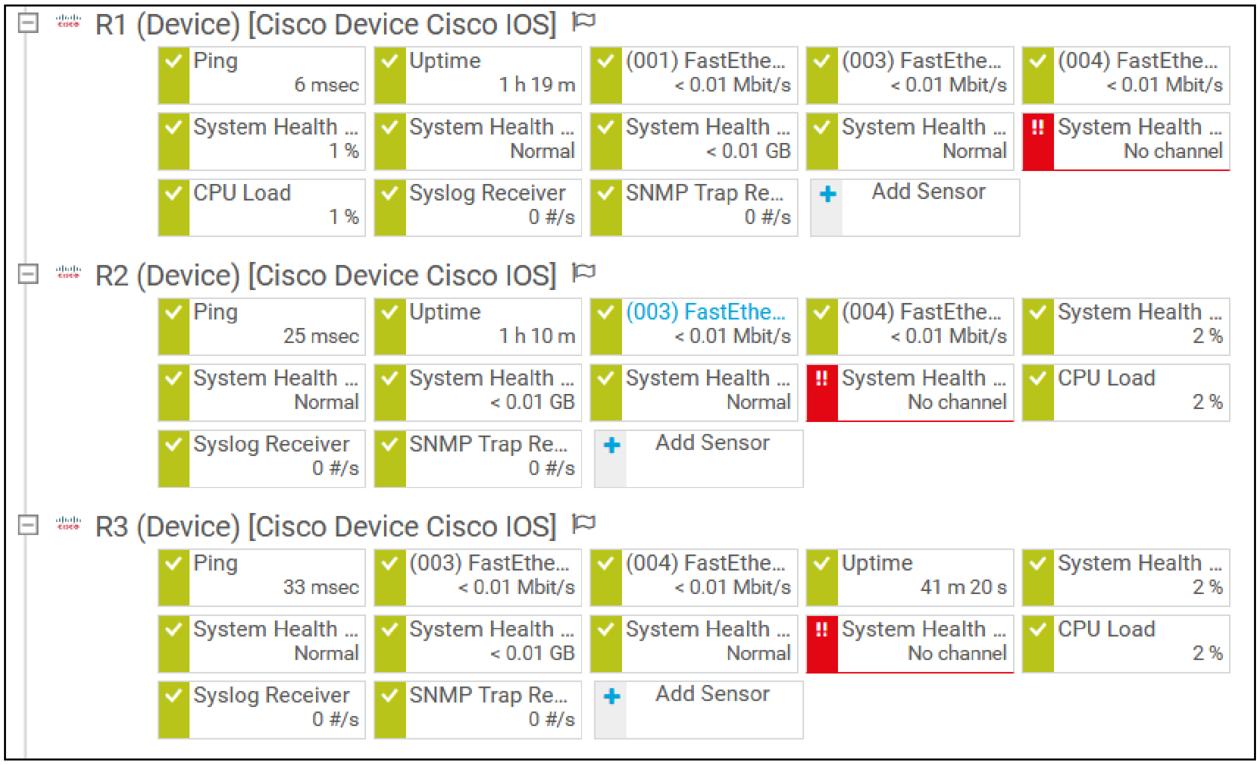
```
Remote address: 11.80.1.1  port: 161 transport: IP/UDP
Local address: 192.168.56.1  port: 49691 transport: IP/UDP
Protocol version: SNMPv1
1: sysUpTime.0 (TimeTicks) 0 days 02h:04m:50s.83th (749083)
***** SNMP SET-RESPONSE START *****
1: sysContact.0 (DisplayString) CS158B [43.53.31.35.38.42 (hex)]
***** SNMP SET-RESPONSE END *****
```

Fig. 3.7.2 - Changed contact value

3.8. Able to monitor device resources

In the screenshots below, you can see how devices from our network are displayed on PRTG. Each device comes with sensors for ping, uptime, system health, CPU load, and other customizable sensors. The two that we are focusing on are syslog receiver and SNMP trap receiver. Since all devices are up and running properly, the sensors all display green check marks. When syslog receiver (see section 3.2 for more details) and SNMP trap receiver (see section 3.3 for more details) get any messages from the device, it will be stored and displayed in PRTG. Similarly, users will be notified whenever a device is down (see section 3.4 for more details).

In addition, we can also use SNMP MIBwalk via the MGSoft Browser, as shown in section 3.5. By using SNMP walk, we are easily able to see the values of the variables for any remote router we input as well as all of the changes and timestamps that particular attributes of the SNMP agent was changed (in the case demonstrated in Figure 3.5.1, we are monitoring resources for R1).



R1 (Device) [Cisco Device Cisco IOS]

✓ Ping 6 msec	✓ Uptime 1 h 19 m	✓ (001) FastEthe... < 0.01 Mbit/s	✓ (003) FastEthe... < 0.01 Mbit/s	✓ (004) FastEthe... < 0.01 Mbit/s
✓ System Health ... 1 %	✓ System Health ... Normal	✓ System Health ... < 0.01 GB	✓ System Health ... Normal	‼ System Health ... No channel
✓ CPU Load 1 %	✓ Syslog Receiver 0 #/s	✓ SNMP Trap Re... 0 #/s	+ Add Sensor	

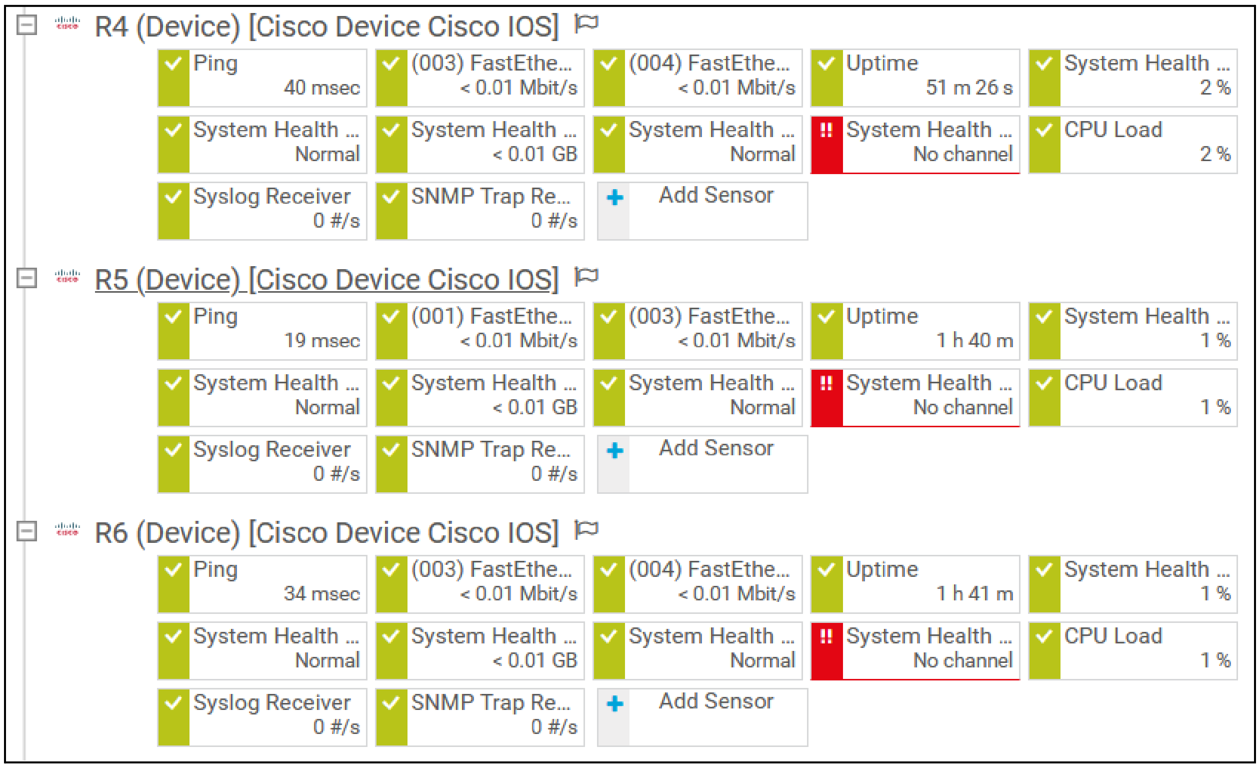
R2 (Device) [Cisco Device Cisco IOS]

✓ Ping 25 msec	✓ Uptime 1 h 10 m	✓ (003) FastEthe... < 0.01 Mbit/s	✓ (004) FastEthe... < 0.01 Mbit/s	✓ System Health ... 2 %
✓ System Health ... Normal	✓ System Health ... < 0.01 GB	✓ System Health ... Normal	‼ System Health ... No channel	✓ CPU Load 2 %
✓ Syslog Receiver 0 #/s	✓ SNMP Trap Re... 0 #/s	+ Add Sensor		

R3 (Device) [Cisco Device Cisco IOS]

✓ Ping 33 msec	✓ (003) FastEthe... < 0.01 Mbit/s	✓ (004) FastEthe... < 0.01 Mbit/s	✓ Uptime 41 m 20 s	✓ System Health ... 2 %
✓ System Health ... Normal	✓ System Health ... < 0.01 GB	✓ System Health ... Normal	‼ System Health ... No channel	✓ CPU Load 2 %
✓ Syslog Receiver 0 #/s	✓ SNMP Trap Re... 0 #/s	+ Add Sensor		

Fig. 3.8.1 Device resources of R1, R2, and R3 monitored on PRTG.



R4 (Device) [Cisco Device Cisco IOS]

✓ Ping 40 msec	✓ (003) FastEthe... < 0.01 Mbit/s	✓ (004) FastEthe... < 0.01 Mbit/s	✓ Uptime 51 m 26 s	✓ System Health ... 2 %
✓ System Health ... Normal	✓ System Health ... < 0.01 GB	✓ System Health ... Normal	‼ System Health ... No channel	✓ CPU Load 2 %
✓ Syslog Receiver 0 #/s	✓ SNMP Trap Re... 0 #/s	+ Add Sensor		

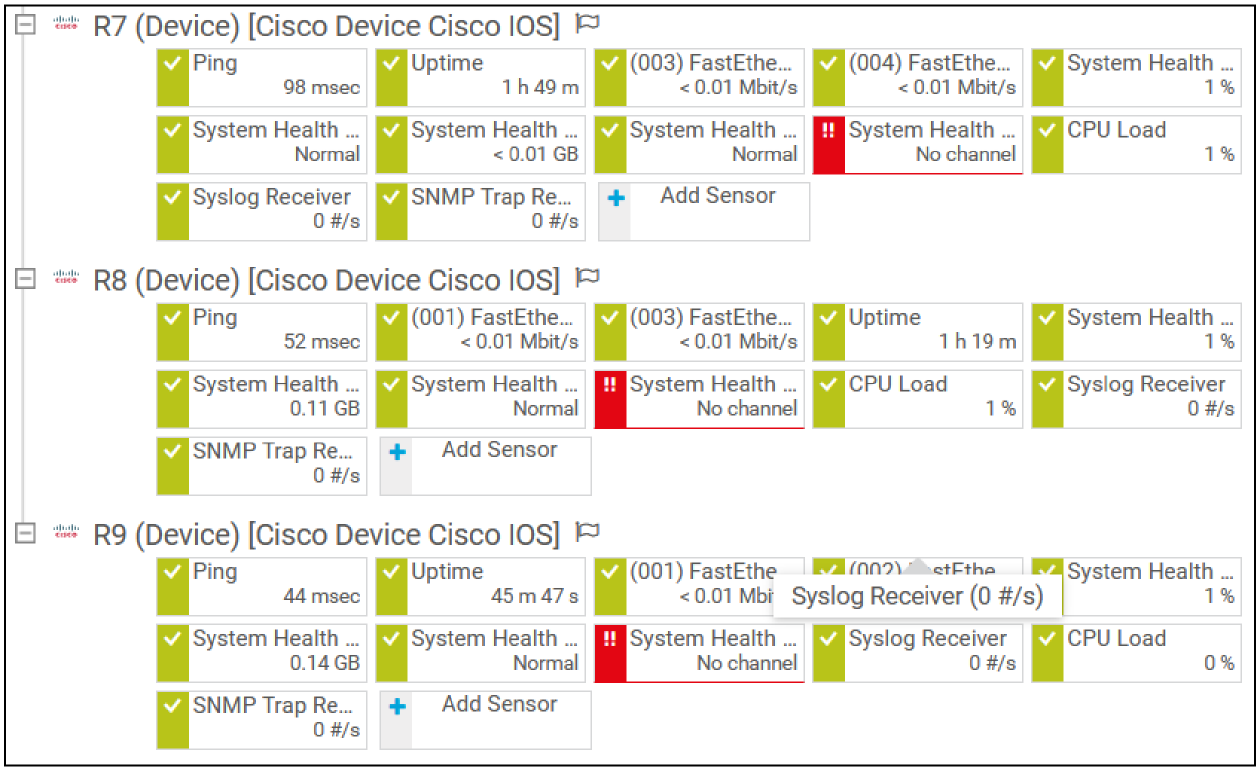
R5 (Device) [Cisco Device Cisco IOS]

✓ Ping 19 msec	✓ (001) FastEthe... < 0.01 Mbit/s	✓ (003) FastEthe... < 0.01 Mbit/s	✓ Uptime 1 h 40 m	✓ System Health ... 1 %
✓ System Health ... Normal	✓ System Health ... < 0.01 GB	✓ System Health ... Normal	‼ System Health ... No channel	✓ CPU Load 1 %
✓ Syslog Receiver 0 #/s	✓ SNMP Trap Re... 0 #/s	+ Add Sensor		

R6 (Device) [Cisco Device Cisco IOS]

✓ Ping 34 msec	✓ (003) FastEthe... < 0.01 Mbit/s	✓ (004) FastEthe... < 0.01 Mbit/s	✓ Uptime 1 h 41 m	✓ System Health ... 1 %
✓ System Health ... Normal	✓ System Health ... < 0.01 GB	✓ System Health ... Normal	‼ System Health ... No channel	✓ CPU Load 1 %
✓ Syslog Receiver 0 #/s	✓ SNMP Trap Re... 0 #/s	+ Add Sensor		

Fig. 3.8.2 Device resources of R4, R5, and R6 monitored on PRTG.



R7 (Device) [Cisco Device Cisco IOS]

✓ Ping 98 msec	✓ Uptime 1 h 49 m	✓ (003) FastEthe... < 0.01 Mbit/s	✓ (004) FastEthe... < 0.01 Mbit/s	✓ System Health ... 1 %
✓ System Health ... Normal	✓ System Health ... < 0.01 GB	✓ System Health ... Normal	!! System Health ... No channel	✓ CPU Load 1 %
✓ Syslog Receiver 0 #/s	✓ SNMP Trap Re... 0 #/s	+ Add Sensor		

R8 (Device) [Cisco Device Cisco IOS]

✓ Ping 52 msec	✓ (001) FastEthe... < 0.01 Mbit/s	✓ (003) FastEthe... < 0.01 Mbit/s	✓ Uptime 1 h 19 m	✓ System Health ... 1 %
✓ System Health ... 0.11 GB	✓ System Health ... Normal	!! System Health ... No channel	✓ CPU Load 1 %	✓ Syslog Receiver 0 #/s
✓ SNMP Trap Re... 0 #/s	+ Add Sensor			

R9 (Device) [Cisco Device Cisco IOS]

✓ Ping 44 msec	✓ Uptime 45 m 47 s	✓ (001) FastEthe < 0.01 Mbi	✓ (002) stEthe	✓ System Health ... 1 %
✓ System Health ... 0.14 GB	✓ System Health ... Normal	!! System Health ... No channel	✓ Syslog Receiver 0 #/s	✓ CPU Load 0 %
✓ SNMP Trap Re... 0 #/s	+ Add Sensor			

Syslog Receiver (0 #/s)

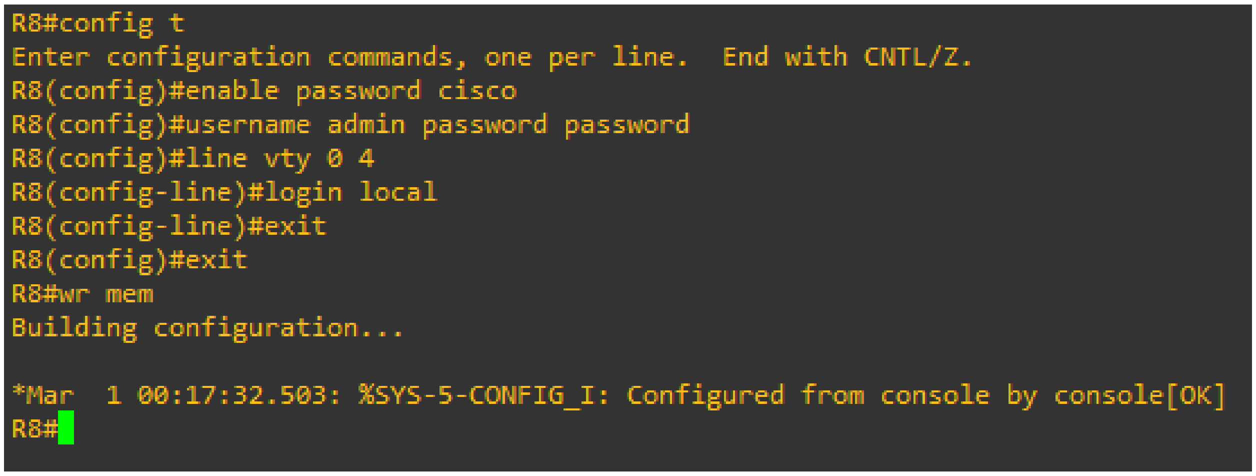
Fig. 3.8.3 Device resources of R7, R8, and R9 monitored on PRTG.

3.9. Able to run automation script to query and set device attributes

Automation scripts are very useful for setting up devices because when the user tries to set up the devices it may cause human errors such as typing wrong commands. Also, by having automation scripts, it will be helpful for setting up multiple devices at once. For example, if a user wants to set up multiple devices, the user needs to write many repetitive commands. However, by using the automation script, the user will only need to type the key parameters and not have to run whole commands.

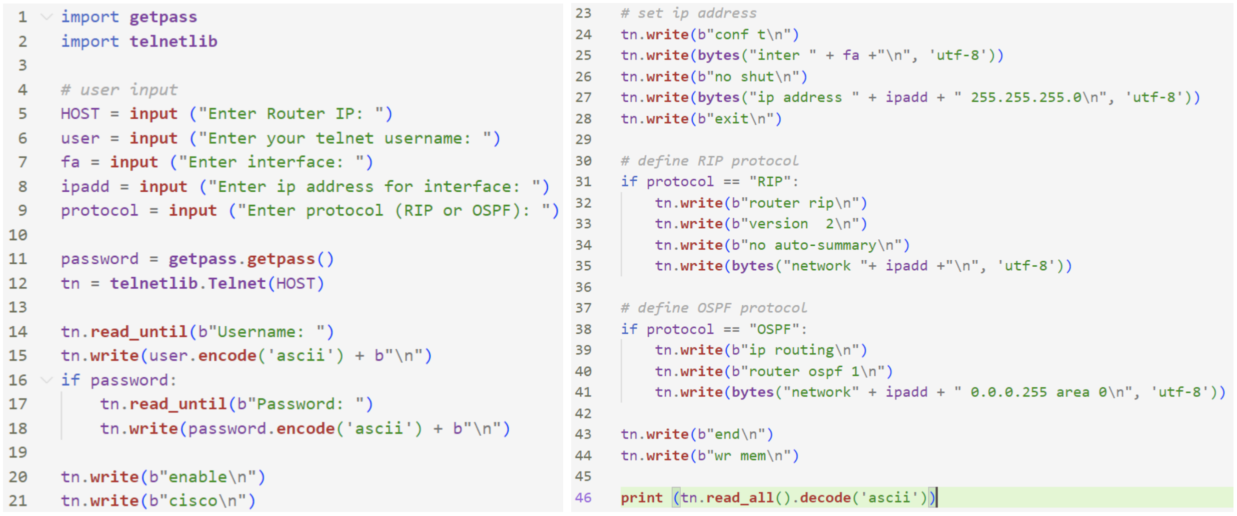
To run automation scripts in GNS3, the device must first enable password and telnet with the following commands in Figure 3.9.1. Next, running the python script (Figure 3.9.2) will prompt the users for inputs. As seen in Figure 3.9.3, the user must first enter the IP address of the router and their telnet username to start setting up the device. The script automatically configures the targeted device with the user-specified ip address for the interface and the type of protocol (can choose between RIP or OSPF).

In the script displayed in Figure 3.9.2, the user selects an interface and sets the ip address for it. Next, the script will set the device to RIP or OSPF protocol with the appropriate parameters. Finally, the changes are committed and written to memory. After running the script, the device will be set up and running. Figure 3.9.4 shows the commands used to automatically configure R8. Using an automation script greatly reduces the time it takes to configure each device and completely eliminates mistakes that humans are likely to make when doing repetitive tasks.



```
R8#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#enable password cisco
R8(config)#username admin password password
R8(config)#line vty 0 4
R8(config-line)#login local
R8(config-line)#exit
R8(config)#exit
R8#wr mem
Building configuration...

*Mar  1 00:17:32.503: %SYS-5-CONFIG_I: Configured from console by console[OK]
R8#
```

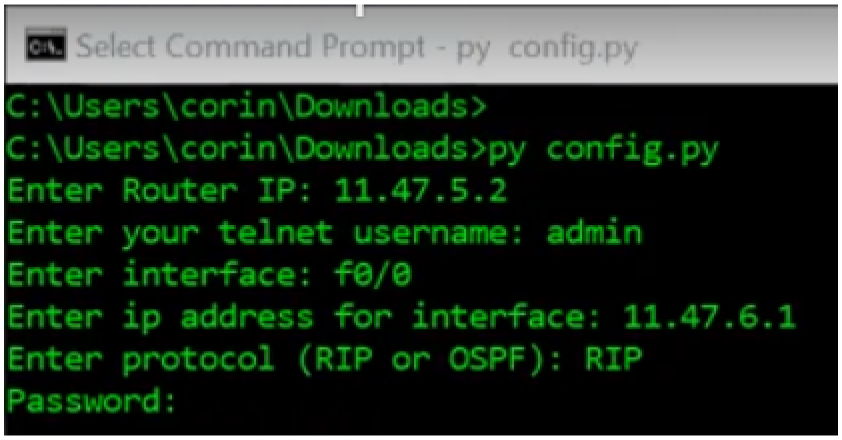
Fig. 3.9.1 Enabling password and telnet in R8.



```python
import getpass
import telnetlib

# user input
HOST = input ("Enter Router IP: ")
user = input ("Enter your telnet username: ")
fa = input ("Enter interface: ")
ipadd = input ("Enter ip address for interface: ")
protocol = input ("Enter protocol (RIP or OSPF): ")

password = getpass.getpass()
tn = telnetlib.Telnet(HOST)

tn.read_until(b"Username: ")
tn.write(user.encode('ascii') + b"\n")
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")

tn.write(b"enable\n")
tn.write(b"cisco\n")
```

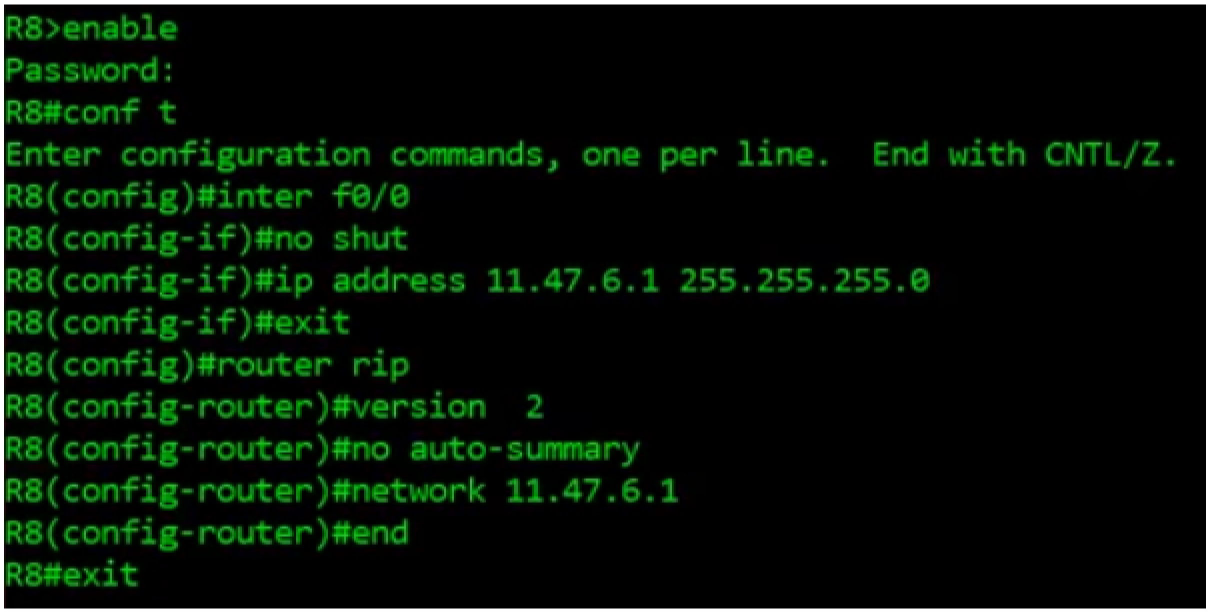
```python
# set ip address
tn.write(b"conf t\n")
tn.write(bytes("inter " + fa +"\n", 'utf-8'))
tn.write(b"no shut\n")
tn.write(bytes("ip address " + ipadd + " 255.255.255.0\n", 'utf-8'))
tn.write(b"exit\n")

# define RIP protocol
if protocol == "RIP":
    tn.write(b"router rip\n")
    tn.write(b"version  2\n")
    tn.write(b"no auto-summary\n")
    tn.write(bytes("network "+ ipadd +"\n", 'utf-8'))

# define OSPF protocol
if protocol == "OSPF":
    tn.write(b"ip routing\n")
    tn.write(b"router ospf 1\n")
    tn.write(bytes("network" + ipadd + " 0.0.0.255 area 0\n", 'utf-8'))

tn.write(b"end\n")
tn.write(b"wr mem\n")

print (tn.read_all().decode('ascii'))
```

Fig. 3.9.2 Automation python script for setting device attributes.



```
C:\Users\corin\Downloads>
C:\Users\corin\Downloads>py config.py
Enter Router IP: 11.47.5.2
Enter your telnet username: admin
Enter interface: f0/0
Enter ip address for interface: 11.47.6.1
Enter protocol (RIP or OSPF): RIP
Password:
```

Fig. 3.9.3 Running the automation python script.



```
R8>enable
Password:
R8#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#inter f0/0
R8(config-if)#no shut
R8(config-if)#ip address 11.47.6.1 255.255.255.0
R8(config-if)#exit
R8(config)#router rip
R8(config-router)#version  2
R8(config-router)#no auto-summary
R8(config-router)#network 11.47.6.1
R8(config-router)#end
R8#exit
```

Fig. 3.9.4 Commands for configuring R8 are automatically entered by the script.

4 Conclusion

Through using GNS3, VirtualBox, MG-Soft, and PRTG, we are able to develop a network that has 12 devices and 3 different platforms. Different platforms we used for the network are routers c3620, c3640, and c3725. By setting up all the devices as proposed in the topology, we were able to set up 2 protocols, RIP and OSPF. After we implemented the necessary features, we used SNMP MIB Browser, Wireshark, and automation python scripts to test and validate that our developed small network satisfies the company's requirements.

Links

- Automation python script (https://drive.google.com/file/d/1gxxrV28JIwbzjZj83VI26Qy70Kt2Ukjg/view?usp=share_link)
- GNS3 project (https://drive.google.com/file/d/1ZT8JYoYpfjaC_fHM7BpR2DCJIzgRz6MH/view?usp=share_link)