CS158B F2021 - Team 3
30 November 2021

CS158B Final Project

Team 3: Jade Webb, Brian Eappen, Mikhail Sumawan, Khoa Phan

Introduction

This project demonstrates a proof of concept for a small network from the view of a Network Management Consultant. It features a network topology design and configured network devices. The purpose of the project is to demonstrate how networking projects are carried out using various modern networking technologies.

The project must meet a set of requirements as listed below:

1. *Discover entire network*
    a. *Each network device has the following:*
        i. *Disable DNS lookup*
        ii. *Router name*
        iii. *Domain name cs158b.com*
        iv. *Encrypted privileged EXEC password cisco*
        v. *Console access password cisco*
        vi. *Local admin account: admin/cisco*
        vii. *Login on vty lines to user local database*
        viii. *VTY lines accept telnet all*
        ix. *Encrypted clear text password*
        x. *Banner “Welcome to CS158B”*
2. *Receive syslog from network devices*
3. *Receive SNMP messages from network devices*
4. *Receive event when interface or device is down*
5. *Run SNMP mibwalk*
6. *Compile a new SNMP mib when a customer purchases a new device*
7. *Make changes through SNMP set operation*
8. *Monitor device resources (memory, bandwidth, cpu utilization, etc.)*
9. *Run python automation script to query and set device attributes*

CS158B F2021 - Team 3
30 November 2021

Proposed Solution

Our solution features a variety of tools used in modern networking projects. As they were introduced by Professor Paul Nguyen, a Cisco employee, it is likely that these tools are currently used in the industry by network managers at Cisco and elsewhere.

Tools Used

a) GNS3: Graphical Network Simulator-3 is a network simulator. It simulates the interaction between the local computer and a network system of configurable switches, routers and other PCs. Carrying out an actual networking procedure with physical devices is frankly expensive and impractical, so GNS3 solves this issue. GNS3 can help the user understand networking functionality by simply using their personal computer, assuming that the computer has enough memory.
b) Virtual Machine Client (VMWare Fusion or Virtualbox): VirtualBox and VMWare are emulators that allow the local host computer to become an ideal machine to run GNS3. One of the main purposes of this project is to study the communication process between the local system of the local computer and the system loaded on the virtual machine through the GNS3 VM server. The virtual machine takes up memory space from the local computer but it is technically a separate system in this case.
c) MG-Soft MIB Browser: Enterprise devices/agents communicate with each other using SNMP (Simple Network Management Protocol). The Management Information Base, or MIB, file is provided from the enterprise in order for the devices to function. In this project, the enterprises are Cisco and IBM. From these files, a MIB tree is constructed that allows for device querying, filtering, and monitoring.
d) PRTG Network Monitor: PRTG is an online, Windows-exclusive network monitoring software. Specifically, it monitors connected interfaces, system health, and utilization of memory, bandwidth and cpu for the simulated devices in the GNS3 virtual machine.
e) Wireshark: Wireshark is a packet analyzation tool that can be used to filter packet traffic on a specified local network which can be captured for troubleshooting and network analysis.

CS158B F2021 - Team 3
30 November 2021

Topology

The topology of our designed GNS3 network is illustrated in Figure 1. There are some key points to this network:

- Network used: 10.7.0.0/24
- 13 routers, 4 Virtual PCs
- 3 different platforms are used: 3620, 3640, 3725
    - 3620: R6-R7
    - 3640: R12-R13
    - 3725: R1-R5, R8-R11
- Network devices are divided into two halves. One half (blue) uses the RIP version 2 protocol, the other half (red) uses the OSPF area 0 protocol
- R1 is the bridge connecting the RIP side with the OSPF side, and acts as the Ethernet router for the entire GNS3 network
    - R1 is assigned a DHCP IP address of 192.168.1.117
    - The Cloud allows this network to communicate with the host computer which has an IP address of 192.168.1.71

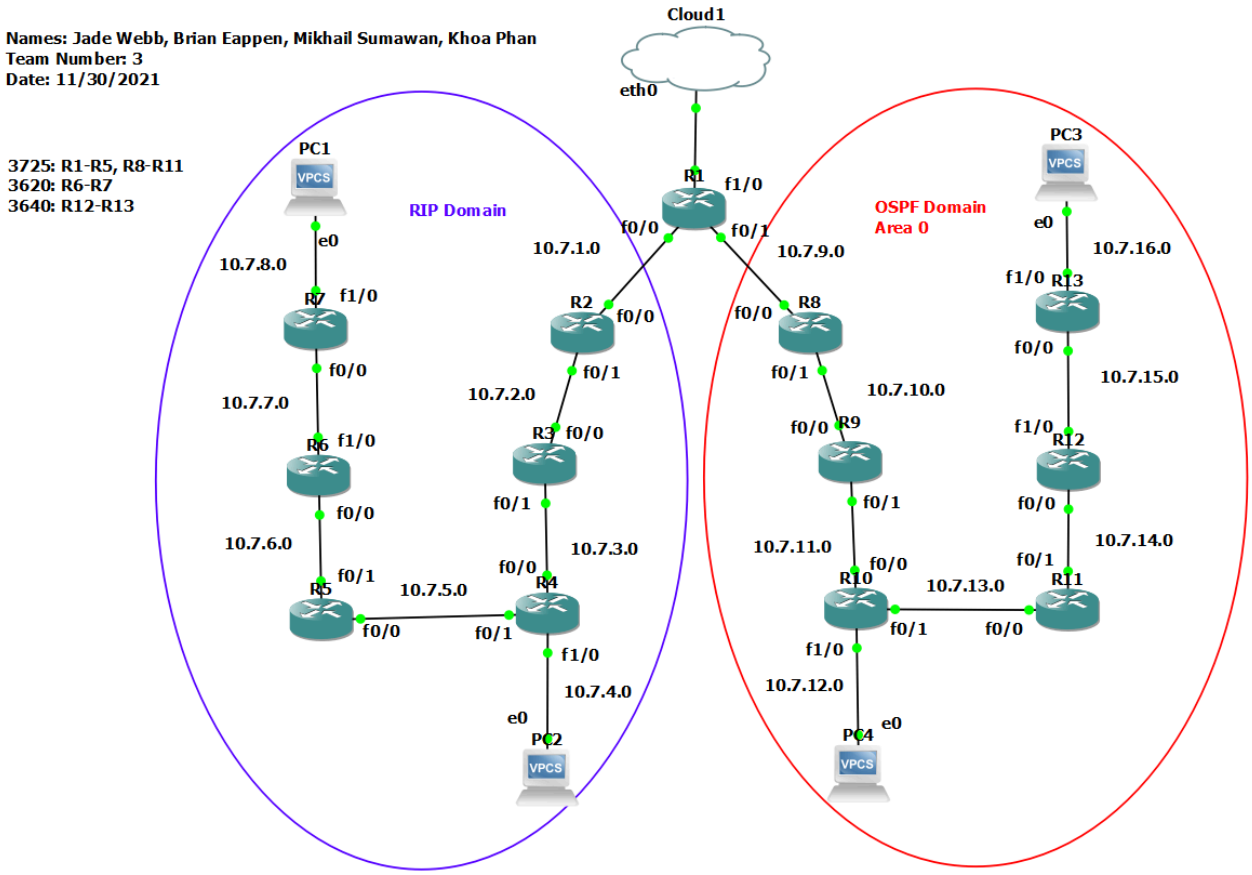


Figure 1: Topology of the GNS3 network

CS158B F2021 - Team 3
30 November 2021

Protocols Used

2 routing protocols were used:

- RIP Version 2: Routing Information Protocol (RIP) is a distance vector protocol used to route data packets throughout a network. It measures a route based on the number of hops between routers.
    - Classless
        - Triggers routing updates only when topology changes
        - Includes subnet mask information in routing updates
        - Uses less bandwidth than classful
    - Summarization
    - Security
- OSPF Area 0: Open Shortest Path First (OSPF) is a link state routing protocol used to efficiently route data packets throughout a network. It measures a route based on the number of hops between routers, as well as factors including bandwidth, delay, and load.
    - Classless
    - No max hop count
    - Updates routing tables incrementally
        - More efficient bandwidth use
    - Routers are have a relationship with neighboring adjacent routers
        - Incremental updates passed on by neighbors
        - Routers only compute the topology of their neighborhood area

Connectivity Testing

Connectivity within the GNS3 Network was tested by pinging between routers in the RIP domain and routers in the OSPF domain to ensure the network protocols were being redistributed properly. Additionally, all PCs were tested to ensure they could ping each other. The cloud's connectivity was tested by pinging between the host computer and every device in the GNS3 network. Lastly, the IP route of R1 was analyzed to ensure all 16 subnets were configured correctly and reachable throughout the network. The subnet labels are as follows:

- C: directly connected through FastEthernet cable
- R: connected via RIP
- O: connected via OSPF
- S*: static route

Connectivity was tested further by ensuring the PRTG Network Monitor could discover, ping, and monitor all routers in the network. All 13 devices and their individual sensors were troubleshooted to ensure an up (green) status when viewed on the PRTG Devices tab. Additionally, the network's ability to be viewed in the MG-Soft MIB Browser was tested by ensuring all GNS3 devices were configured with SNMP, including public read-only, private read-write, and enabled traps. By specifying R1's IP address as the remote SNMP agent, the system was able to be contacted and queried in the browser. This connectivity was able to be confirmed further by observing the packet traffic through UDP port 161 in Wireshark.

Feature Validation

Able to discover the entire network and verify individual device configuration. Able to ping between PC in RIP domain and PC in OSPF domain. Able to ping between host computer and PCs in each domain.

Figure 2: Device configuration according to specifications

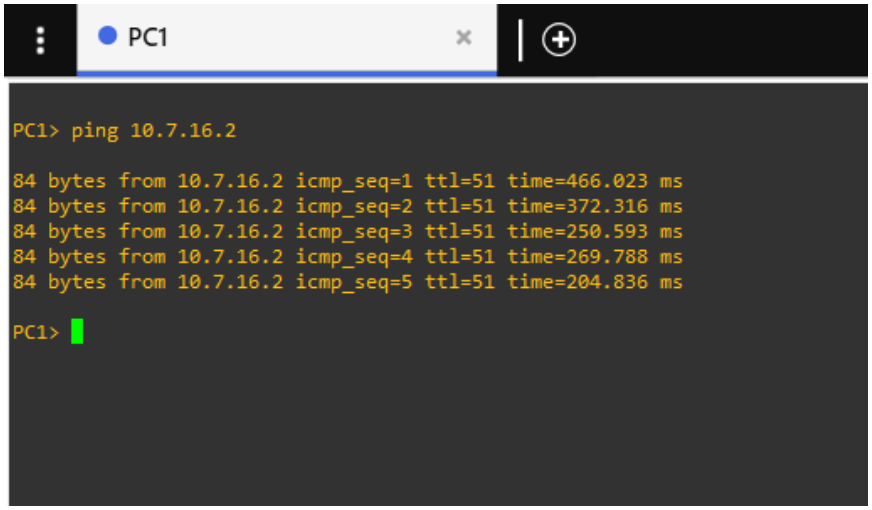


Figure 3: Pinging PC3 from PC1

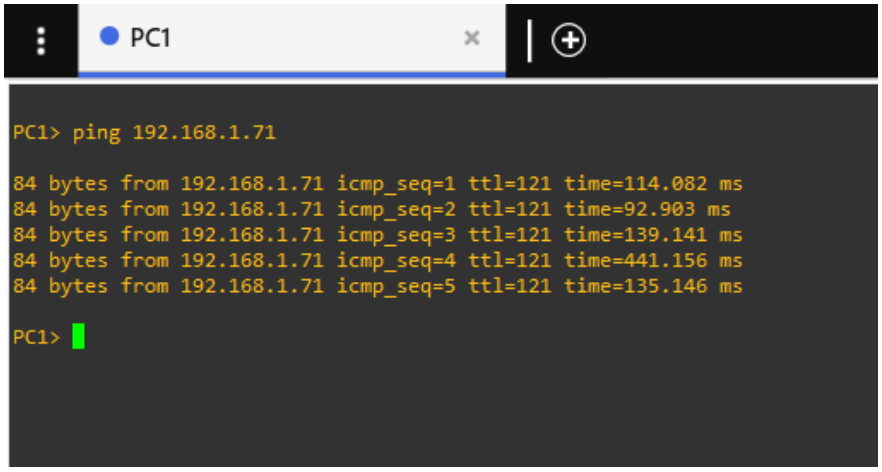


Figure 4: Pinging host computer from PC1

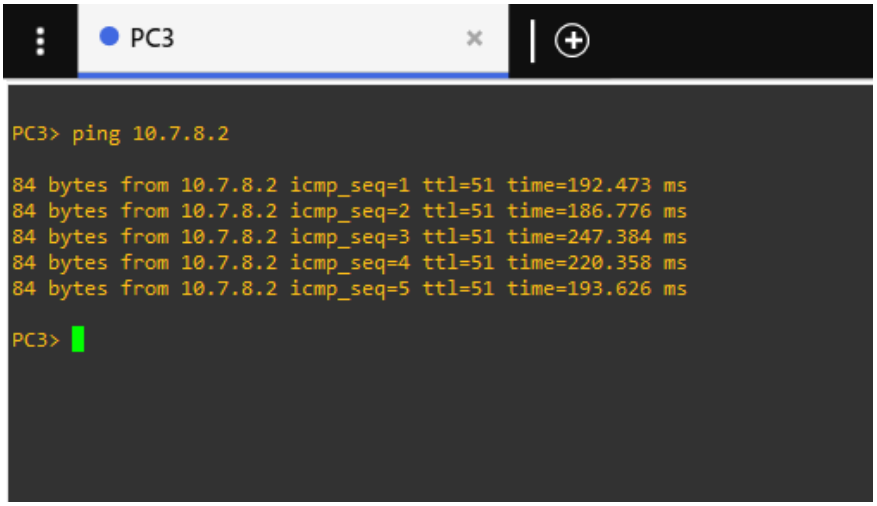


Figure 5: Pinging PC1 from PC3

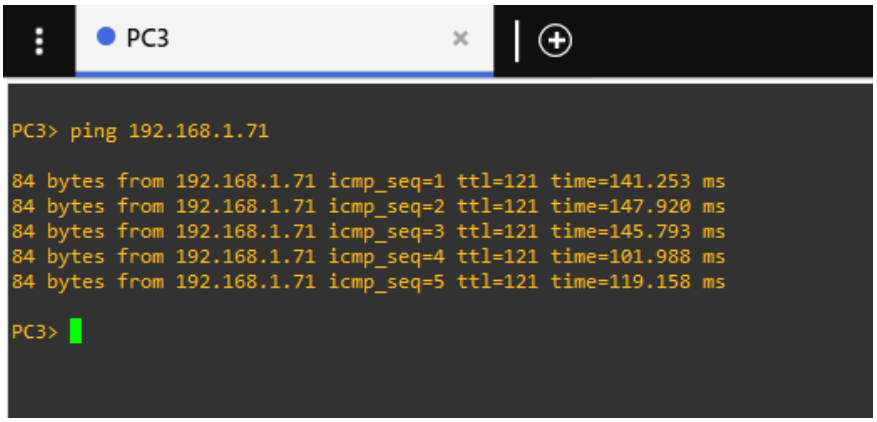


Figure 6: Pinging host computer from PC3

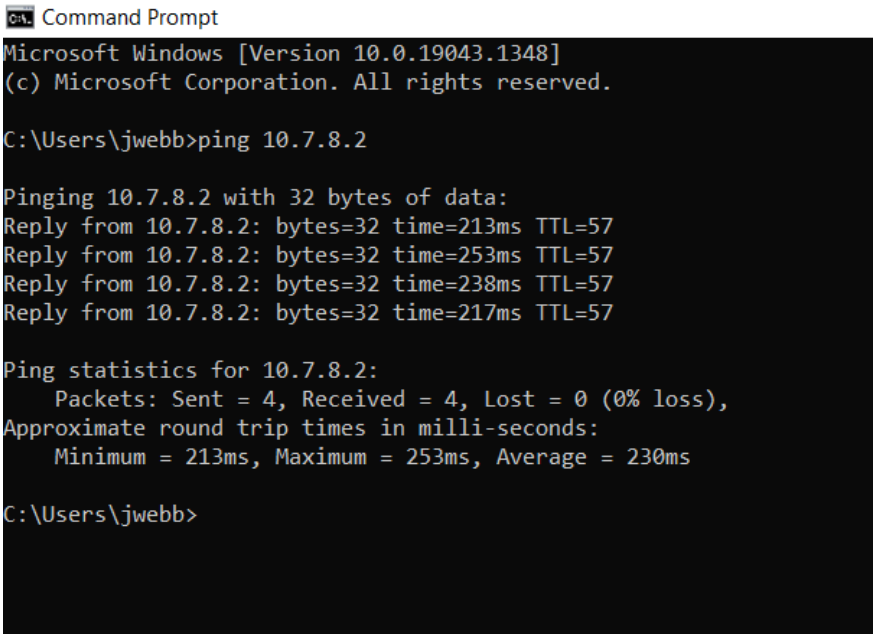


Figure 7: Pinging PC1 from host computer

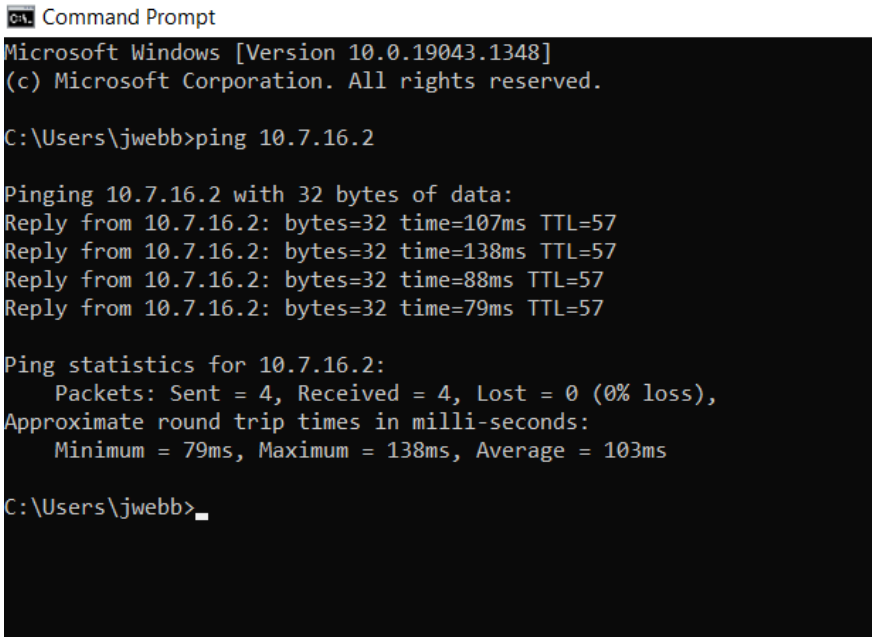


Figure 8: Pinging PC3 from host computer

Able to view IP interface and IP route, demonstrating connection to cloud and subnets.

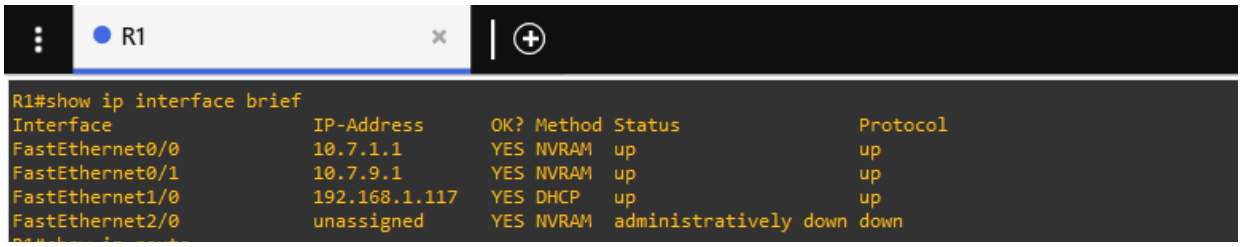


Figure 9: R1 IP interface

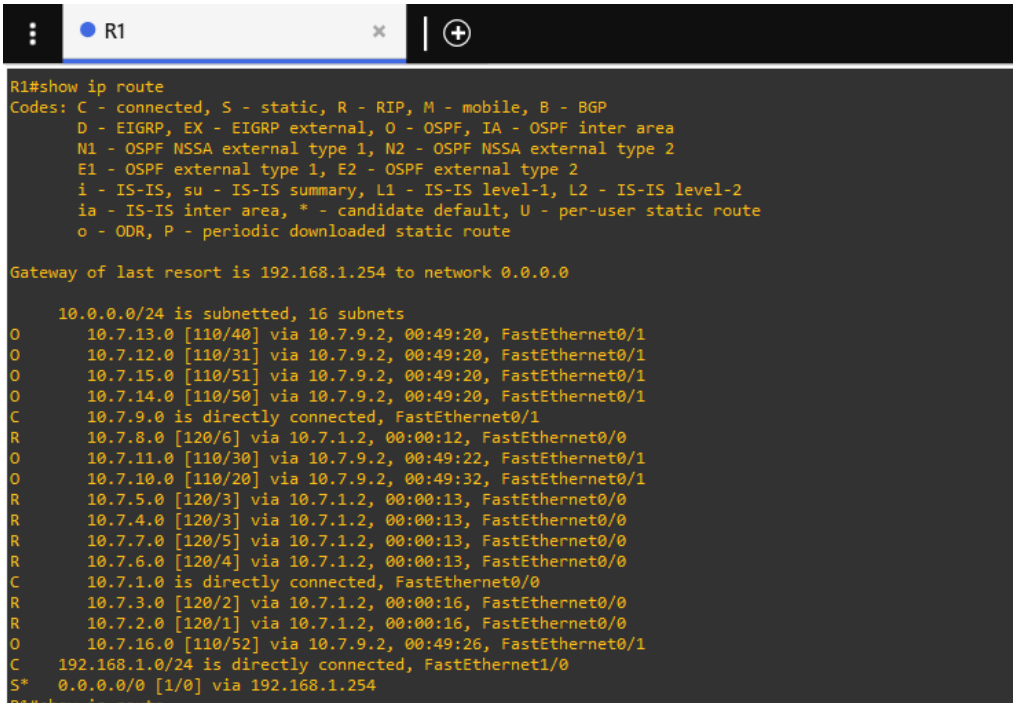


Figure 10: R1 IP route

CS158B F2021 - Team 3
30 November 2021

Able to monitor device resources for all 13 routers in GNS3 topology. All sensors except temperature are active and error-free.

Figure 11: PRTG Group R1-R7

Figure 12: PRTG Group R8-R13

CS158B F2021 - Team 3
30 November 2021

Able to receive SNMP and contact the system.

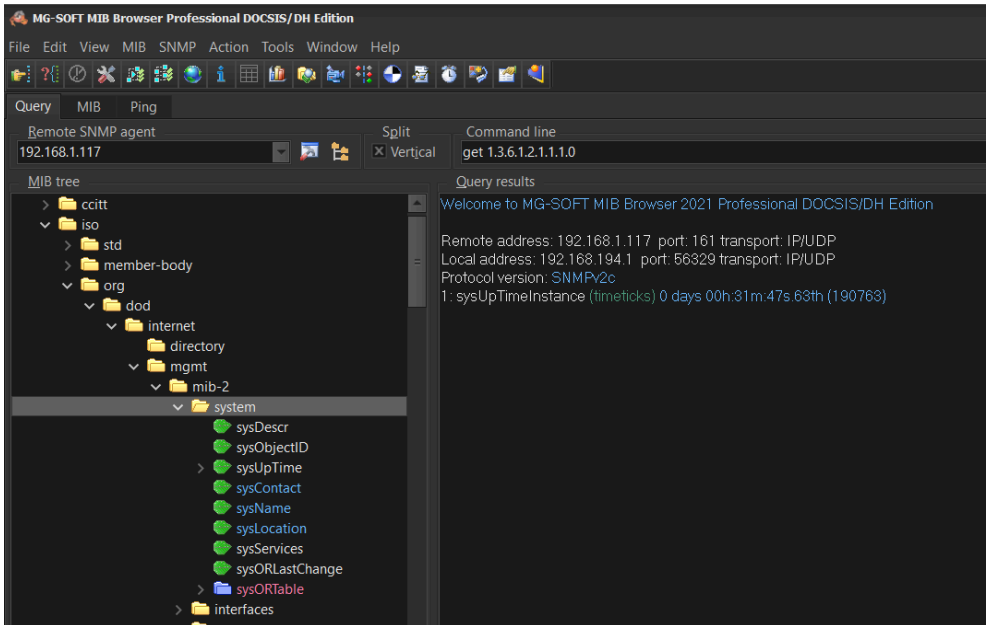


Figure 13: MIB Browser: System - Contact

Able to perform an SNMP mibwalk upon the system.

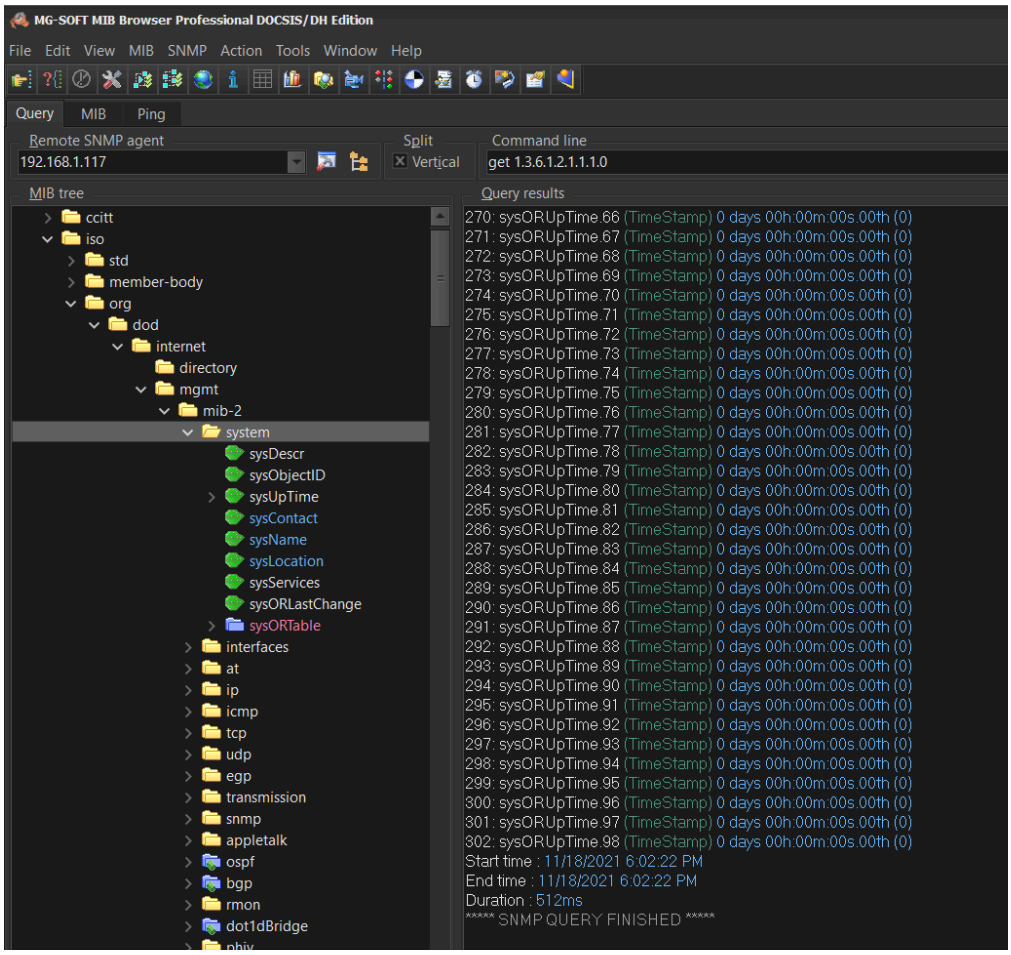


Figure 14: MIB Browser: System - Walk

Able to perform SNMP queries.

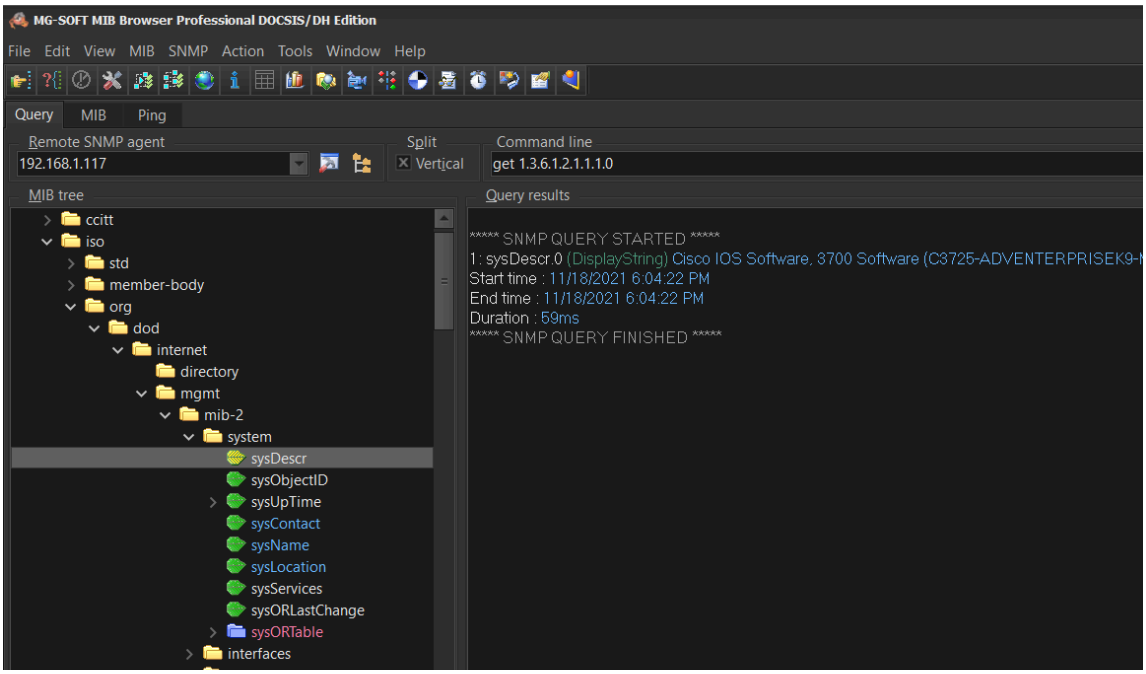


Figure 15: MIB Browser: System Description - Walk

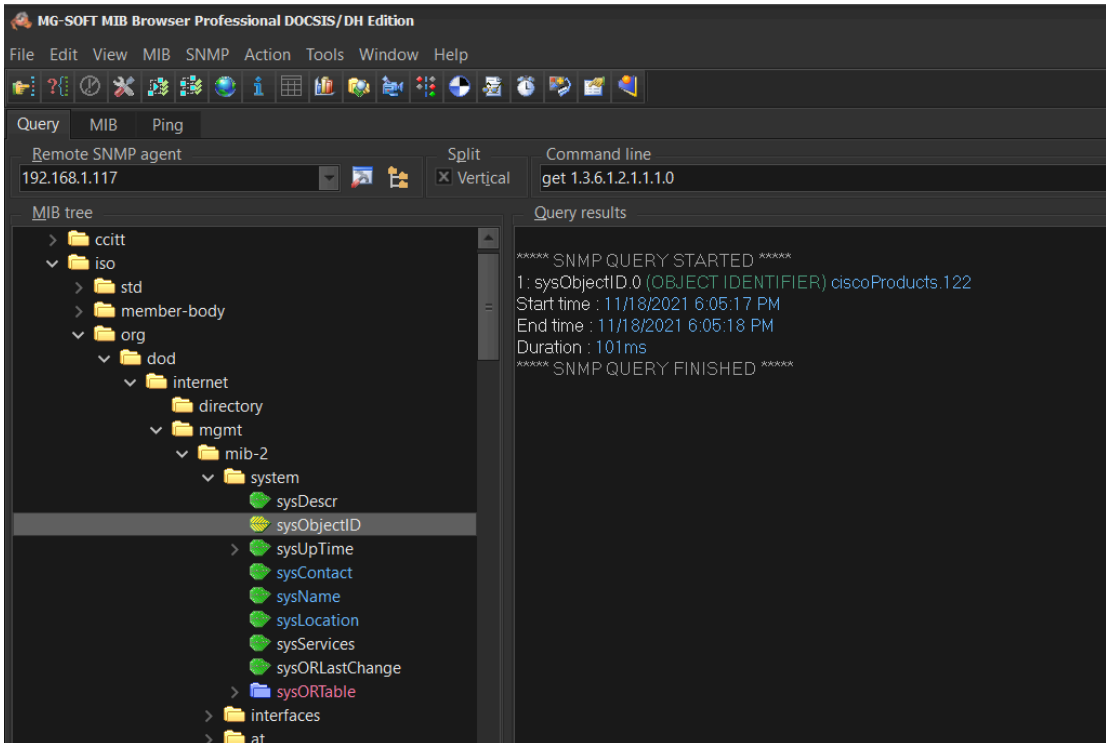


Figure 16: MIB Browser: System ObjectID - Walk

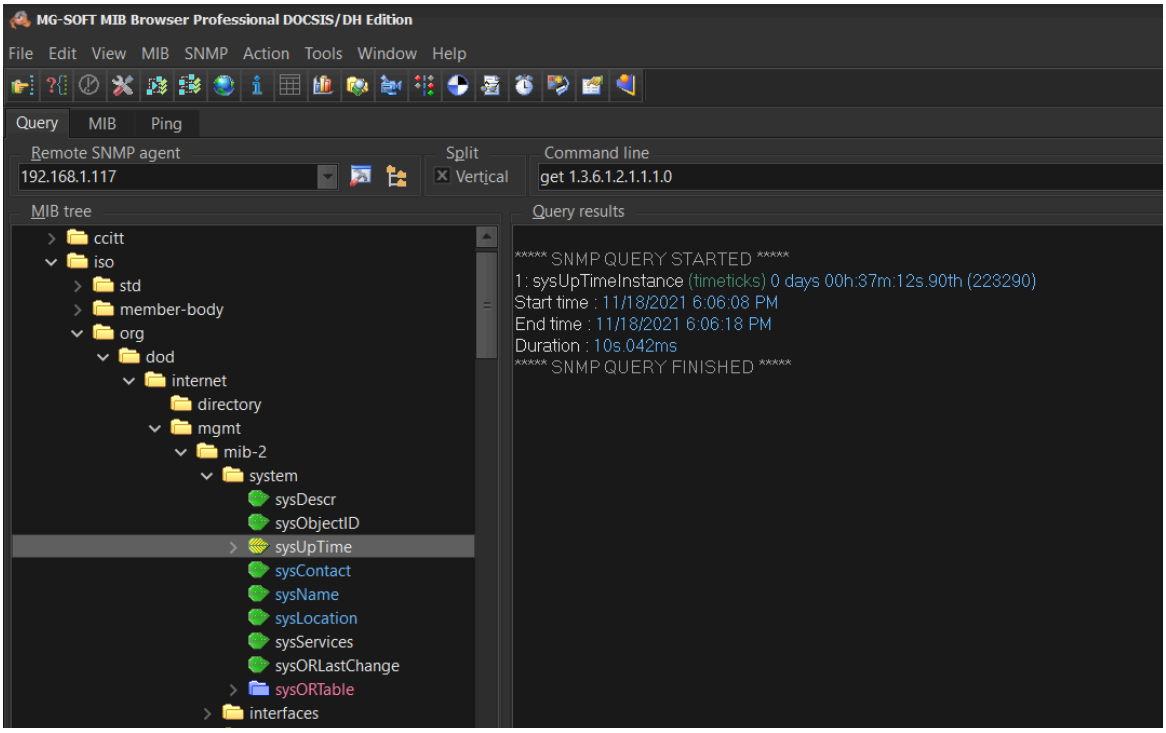


Figure 17: MIB Browser: System UpTime - Walk

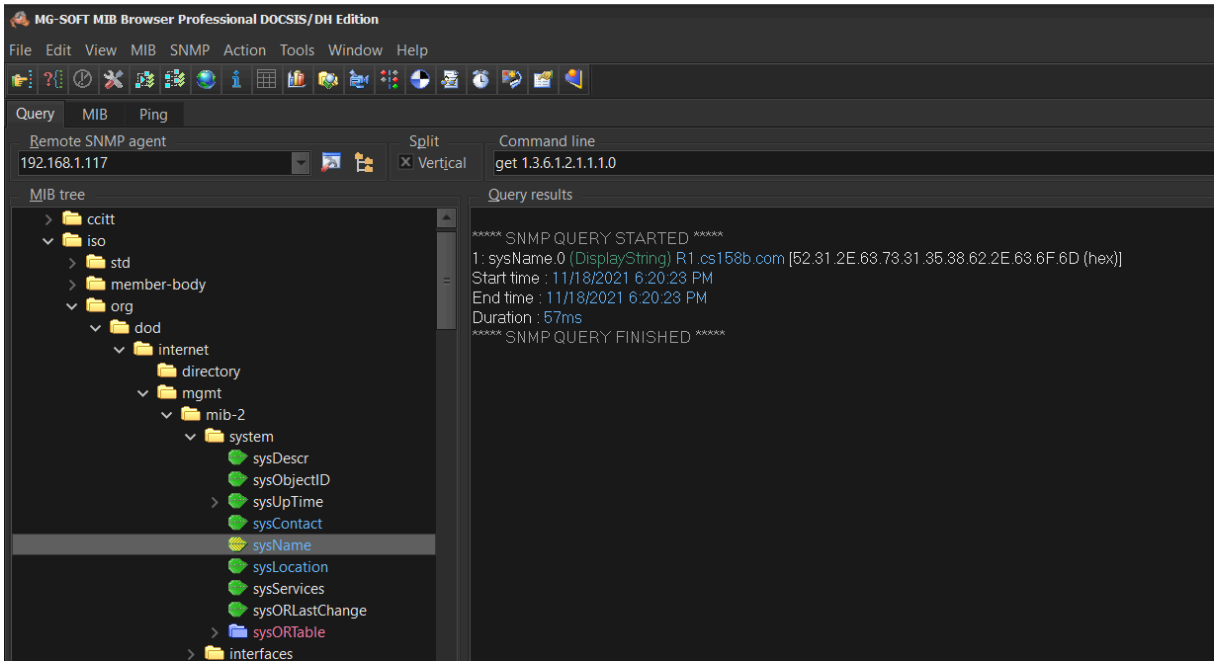


Figure 18: MIB Browser: System Name - Walk

Able to make changes to system contact through SNMP set operation.

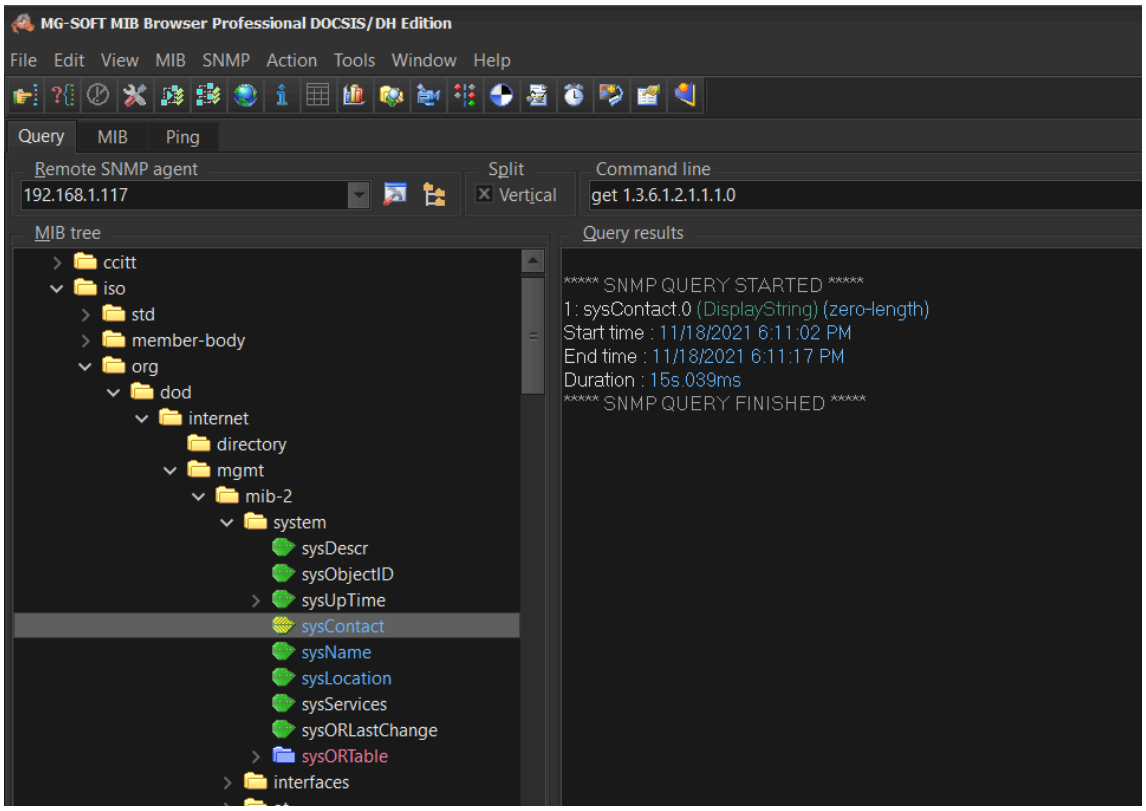


Figure 19: MIB Browser: System Contact - Walk (before Set)

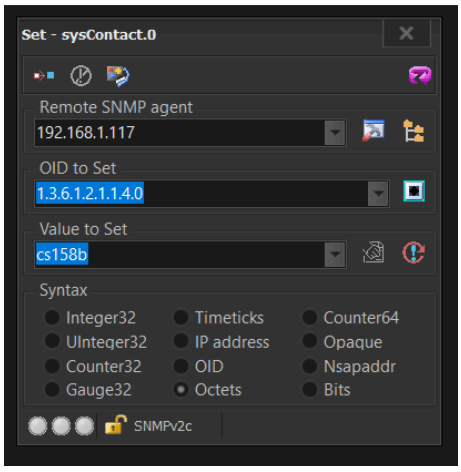


Figure 20: MIB Browser: System Contact - Set

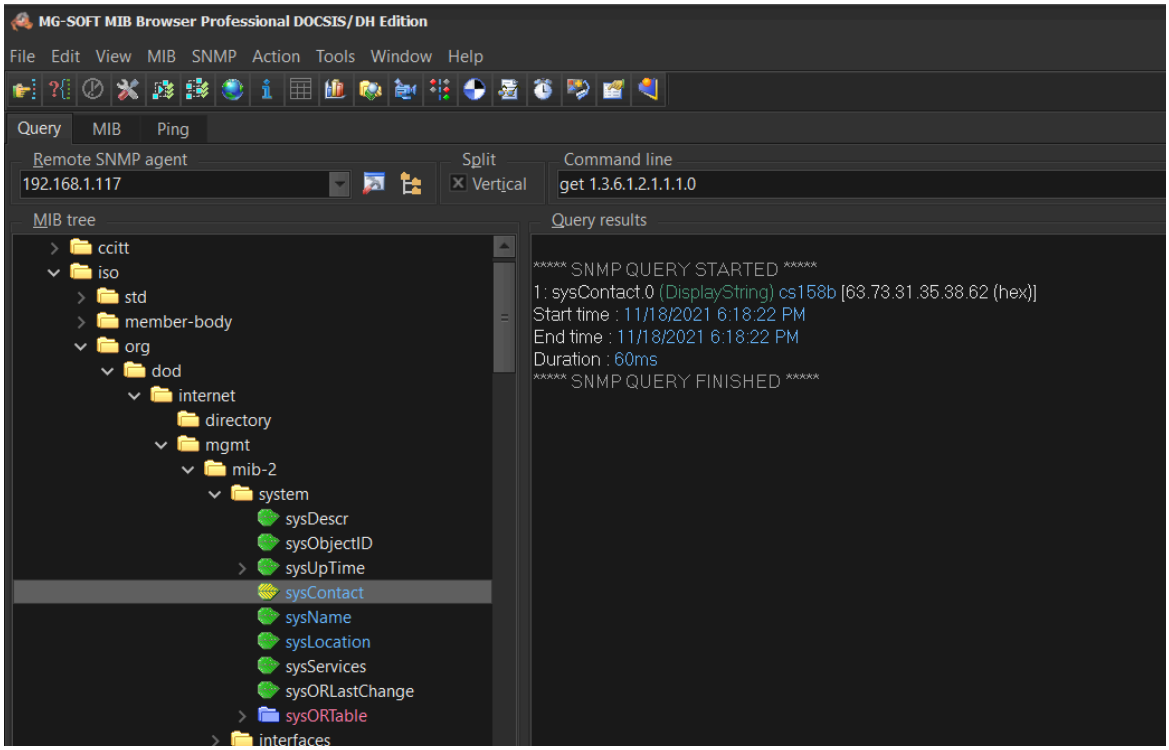


Figure 21: MIB Browser: System Contact - Walk (after Set)

Able to compile a new SNMP mib when a customer purchases a new device.

- IBM enterprise is compiled in MIB Compiler then imported into the MIB tree.
- Able to contact new IBM system through SNMP.

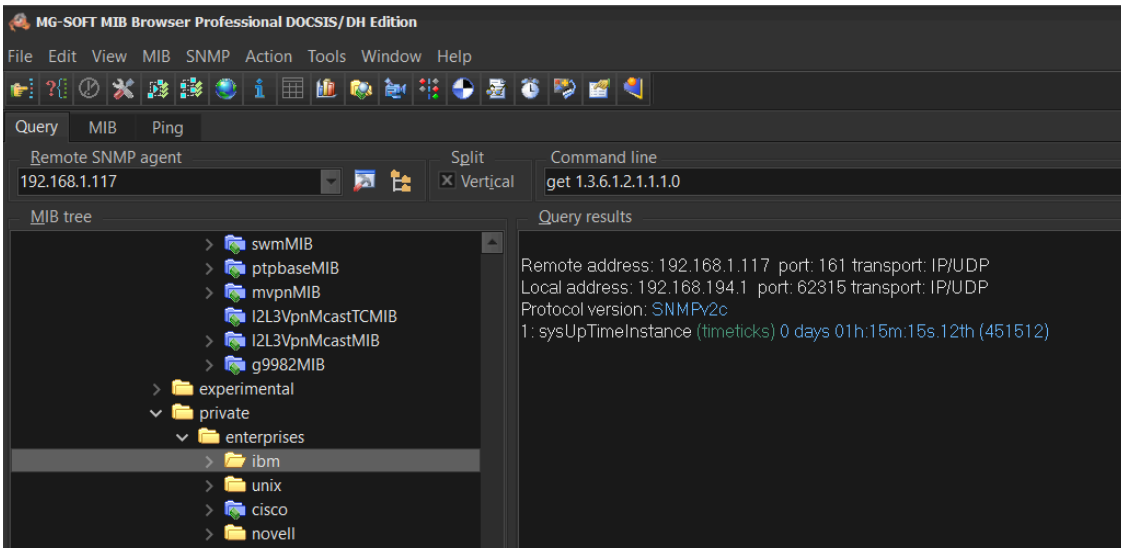


Figure 22: MIB Browser: new Enterprise SNMP mib file (from IBM) - Contact

Able to set SNMP Trap Alert through PRTG SNMP Trap Receiver.

- When R7 interface f0/0 is manually shut through the device console:
  - R7 is visibly down when viewed in PRTG
  - SNMP Trap Receiver sensor produces alerts

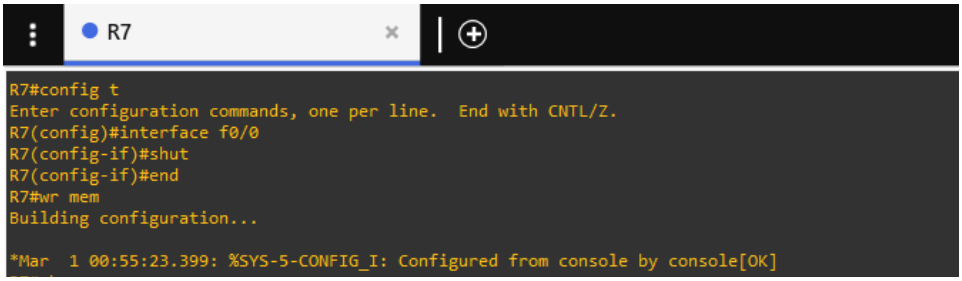
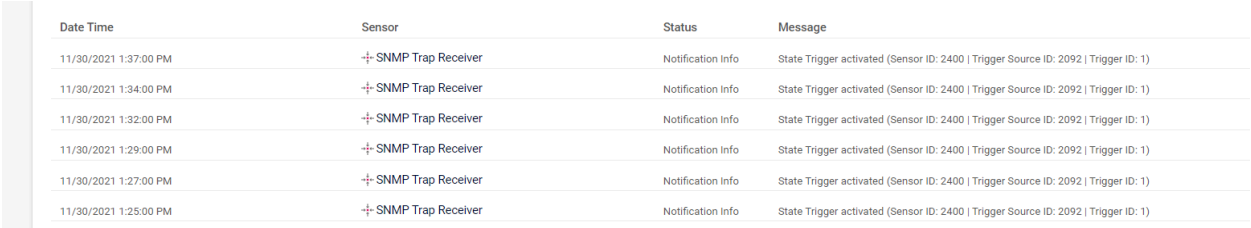


Figure 23: R7 shut interface f0/0

Figure 24: PRTG Group showing R7 down



Date Time	Sensor	Status	Message
11/30/2021 1:37:00 PM	SNMP Trap Receiver	Notification Info	State Trigger activated (Sensor ID: 2400 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:34:00 PM	SNMP Trap Receiver	Notification Info	State Trigger activated (Sensor ID: 2400 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:32:00 PM	SNMP Trap Receiver	Notification Info	State Trigger activated (Sensor ID: 2400 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:29:00 PM	SNMP Trap Receiver	Notification Info	State Trigger activated (Sensor ID: 2400 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:27:00 PM	SNMP Trap Receiver	Notification Info	State Trigger activated (Sensor ID: 2400 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:25:00 PM	SNMP Trap Receiver	Notification Info	State Trigger activated (Sensor ID: 2400 | Trigger Source ID: 2092 | Trigger ID: 1)

Figure 25: PRTG SNMP Trap Receiver Messages

Able to receive syslog messages from network devices through PRTG Syslog Receiver.

Date Time	Sensor	Status	Message
11/30/2021 1:46:00 PM	Syslog Receiver	Notification Info	State Trigger activated (Sensor ID: 2401 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:43:05 PM	Syslog Receiver	Notification Info	State Trigger activated (Sensor ID: 2401 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:41:00 PM	Syslog Receiver	Notification Info	State Trigger activated (Sensor ID: 2401 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:39:00 PM	Syslog Receiver	Notification Info	State Trigger activated (Sensor ID: 2401 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:37:00 PM	Syslog Receiver	Notification Info	State Trigger activated (Sensor ID: 2401 | Trigger Source ID: 2092 | Trigger ID: 1)
11/30/2021 1:34:05 PM	Syslog Receiver	Notification Info	State Trigger activated (Sensor ID: 2401 | Trigger Source ID: 2092 | Trigger ID: 1)

Figure 26: PRTG Syslog Receiver Messages

Attempted to additionally send an email notification to PRTG admin but unable to resolve MX server issue.

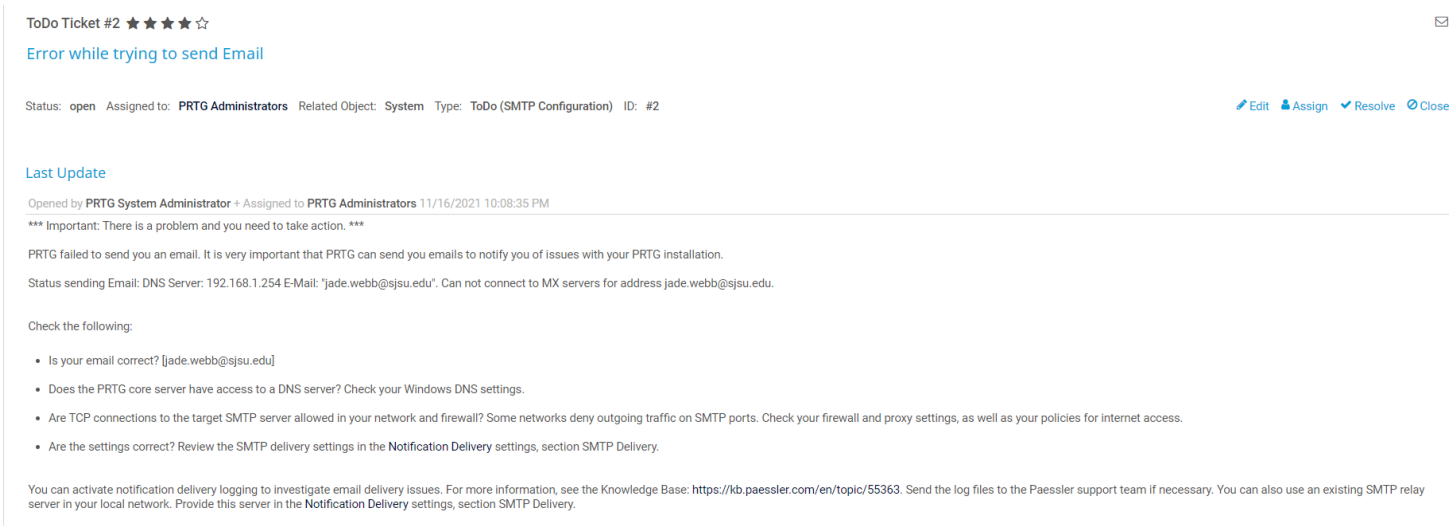


Figure 27: Error Message: Unable to send email notifications

Able to view packet traffic through UDP port 161 for R1, captured in Wireshark.

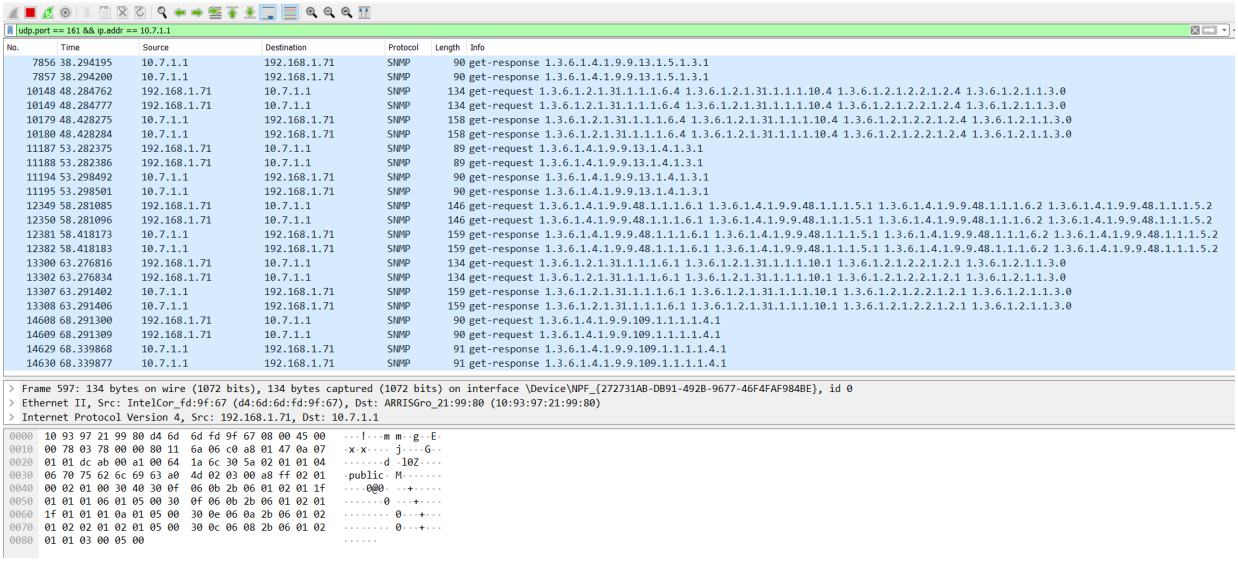


Figure 28: Wireshark SNMP UDP Port Activity

Able to define Python functions for automating get/set operations for network devices

```
C:\Users\jwebb>python
Python 3.9.9 (tags/v3.9.9:ccb0e6a, Nov 15 2021, 18:08:50) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from pysnmp import hlapi
>>> def get(target, oids, credentials, port=161, engine=hlapi.SnmpEngine(), context=hlapi.ContextData()):
...     handler = hlapi.getCmd(
...             engine,
...             credentials,
...             hlapi.UdpTransportTarget((target, port)),
...             context,
...             *construct_object_types(oids)
...     )
...     return fetch(handler, 1)[0]
```

Figure 29: Python Automation: get function

```
>>> def construct_object_types(list_of_oids):
...     object_types = []
...     for oid in list_of_oids:
...             object_types.append(hlapi.ObjectType(hlapi.ObjectIdentity(oid)))
...     return object_types
```

Figure 30: Python Automation: construct_obj_types function

```
...
>>> def fetch(handler, count):
...     result = []
...     for i in range(count):
...         try:
...             error_indication, error_status, error_index, var_binds = next(handler)
...             if not error_indication and not error_status:
...                 items = {}
...                 for var_bind in var_binds:
...                     items[str(var_bind[0])] = cast(var_bind[1])
...                 result.append(items)
...             else:
...                 raise RuntimeError('Got SNMP error: {0}'.format(error_indication))
...         except StopIteration:
...             break
...     return result
```

Figure 31: Python Automation: fetch function

```
...
>>> def cast(value):
...     try:
...         return int(value)
...     except (ValueError, TypeError):
...         try:
...             return float(value)
...         except (ValueError, TypeError):
...             try:
...                 return str(value)
...             except (ValueError, TypeError):
...                 pass
...     return value
```

Figure 32: Python Automation: cast function

CS158B F2021 - Team 3
30 November 2021

Conclusion

In conclusion, we were able to successfully design, build, configure, and monitor a network as part of our Proof of Concept to fulfill the role of newly hired Network Management Consultant. We were able to provide device configuration and network connectivity involving multiple platforms and protocols throughout the GNS3 topology. We were able to configure SNMP on all devices to enable network interaction through the MIB Browser including MIB tree creation, SNMP mibwalk, set and query operations, and compilation of a new device to add to the MIB tree. We were able to use PRTG to monitor device resources, SNMP traps, and SNMP syslogs. We were able to define Python functions to allow for the automation of basic SNMP operations regarding device attributes. In the end, we were able to successfully collaborate to design, develop, and troubleshoot our network to end up with a comprehensive Proof of Concept that is now ready for real-life development.