UNX511NZA Assignment 01

Name	Aryan Khurana
Email	akhurana22@myseneca.ca
Student ID	145282216
Professor	Mohammed Nooruddin
Date	November 24, 2024

Questions

Q1) In a few sentences, describe the general architecture of your entire layout with the network monitor and interface monitors. What does each do?

Answer 01

- **Network Monitor:** The network monitor is the central system that coordinates and manages multiple interface monitors. It listens for user input, queries for the number and names of network interfaces, and starts interface monitor processes. It controls the monitoring process by instructing the interface monitors to monitor specific statistics at regular intervals. It also handles link-down events, sending instructions to bring interfaces back up, and manages graceful shutdowns.
- **Interface Monitors:** Each network interface has a corresponding interface monitor. These monitors are responsible for reading statistics from the `/sys/class/net/<interface-name>` directory, such as received and transmitted bytes, errors, and link state. If an interface goes down, the monitor sends a message to the network monitor, which instructs it to bring the link back up. The interface monitor also manages graceful shutdowns when it receives a shutdown signal from the network monitor.

Q2) Could the interface monitor and network monitor all be contained within one process, if so how?

Answer 02

Yes, it is possible for the interface monitor and network monitor to be contained within one process, though it might not be the most efficient solution. In this case, the network monitor could spawn separate threads for each interface, allowing each thread to independently monitor its respective network interface and report back to the main network monitor. However, using multiple processes allows for better isolation, scalability, and resource management.

Q3) Could the interface monitor and network monitor all be contained within one process, assuming 128 network interfaces running at several Giga-bits per second, which require a polling interval of one millisecond per interface

Answer 03

No, running 128 network interfaces at several Gbps with a polling interval of one millisecond per interface within a single process would be extremely inefficient and likely impractical. Each polling operation would need to be handled in real time, which would overwhelm a single process due to the high concurrency and the need for precise timing. A multi-process or multi-threaded approach is necessary to distribute the load efficiently and ensure the system can scale to handle the volume of traffic and the required polling frequency.

Q4) What is a software defined network? In doing so, describe the applications layer, the control layer, and the forwarding layer

Answer 04

- **Software Defined Network (SDN)** is a networking approach that centralizes network control in a software-based controller, decoupling the control plane (which makes decisions about where traffic is sent) from the data plane (which forwards traffic). SDN enables more flexible and efficient management of network resources.
- **Applications Layer:** This layer is where the network applications reside, interacting with the network to provide services such as monitoring, security, and optimization. It uses the SDN controller to communicate network requirements and receive network data.
- **Control Layer:** The control layer is the brain of the SDN, where the SDN controller resides. This layer is responsible for making decisions about how data should flow through the network based on a global view of the network's state. The controller programs the network devices (such as switches and routers) to forward traffic according to these decisions.
- **Forwarding Layer:** This is the layer where the actual physical network devices, such as switches and routers, reside. These devices forward data packets based on instructions provided by the control layer. The devices are simplified as they rely on the SDN controller for decision-making, enabling more efficient and flexible networking.

In SDN, the focus is on programmability and centralized control, allowing networks to be dynamically adjusted to meet changing needs and conditions, such as traffic loads or security requirements.
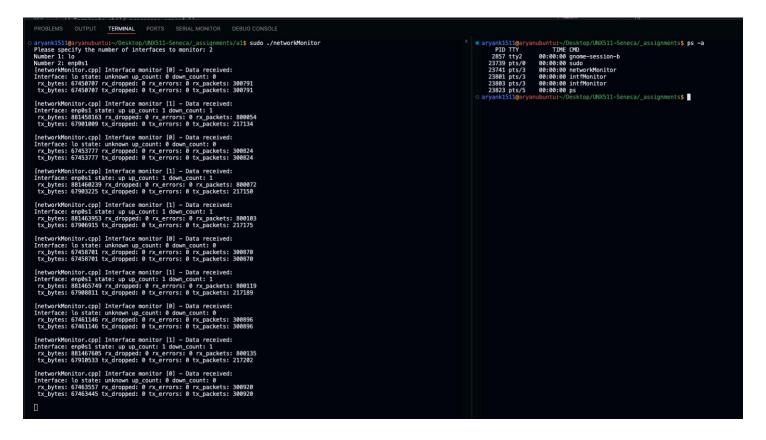
Screenshots

In this screenshot, you can see how I am able to monitor two interfaces **lo** and **enp0s1** using the network monitor and **CTRL + C** is being handled gracefully to shut down the program.

```
aryank1511@aryanubuntu:~/Desktop/UNX511-Seneca/_assignments/a1$ sudo ./networkMonitor
Please specify the number of interfaces to monitor: 2
Number 1: lo
Number 2: enp0s1
[networkMonitor.cpp] Interface monitor [0] - Data received:
Interface: lo state: unknown up_count: 0 down_count: 0
 rx_bytes: 67107811 rx_dropped: 0 rx_errors: 0 rx_packets: 297238
 tx_bytes: 67107811 tx_dropped: 0 tx_errors: 0 tx_packets: 297238

[networkMonitor.cpp] Interface monitor [1] - Data received:
Interface: enp0s1 state: up up_count: 1 down_count: 1
 rx_bytes: 881186796 rx_dropped: 0 rx_errors: 0 rx_packets: 797730
 tx_bytes: 67630343 tx_dropped: 0 tx_errors: 0 tx_packets: 215078

[networkMonitor.cpp] Interface monitor [1] - Data received:
Interface: enp0s1 state: up up_count: 1 down_count: 1
 rx_bytes: 881188686 rx_dropped: 0 rx_errors: 0 rx_packets: 797745
 tx_bytes: 67632079 tx_dropped: 0 tx_errors: 0 tx_packets: 215090

[networkMonitor.cpp] Interface monitor [0] - Data received:
Interface: lo state: unknown up_count: 0 down_count: 0
 rx_bytes: 67110316 rx_dropped: 0 rx_errors: 0 rx_packets: 297262
 tx_bytes: 67110316 tx_dropped: 0 tx_errors: 0 tx_packets: 297262

[networkMonitor.cpp] Interface monitor [0] - Data received:
Interface: lo state: unknown up_count: 0 down_count: 0
 rx_bytes: 67113094 rx_dropped: 0 rx_errors: 0 rx_packets: 297291
 tx_bytes: 67113094 tx_dropped: 0 tx_errors: 0 tx_packets: 297291

[networkMonitor.cpp] Interface monitor [1] - Data received:
Interface: enp0s1 state: up up_count: 1 down_count: 1
 rx_bytes: 881190980 rx_dropped: 0 rx_errors: 0 rx_packets: 797764
 tx_bytes: 67634099 tx_dropped: 0 tx_errors: 0 tx_packets: 215106

[networkMonitor.cpp] Interface monitor [1] - Data received:
Interface: enp0s1 state: up up_count: 1 down_count: 1
 rx_bytes: 881192710 rx_dropped: 0 rx_errors: 0 rx_packets: 797779
 tx_bytes: 67635821 tx_dropped: 0 tx_errors: 0 tx_packets: 215119

[networkMonitor.cpp] Interface monitor [0] - Data received:
Interface: lo state: unknown up_count: 0 down_count: 0
 rx_bytes: 67115210 rx_dropped: 0 rx_errors: 0 rx_packets: 297312
 tx_bytes: 67115210 tx_dropped: 0 tx_errors: 0 tx_packets: 297312

^C
[networkMonitor.cpp] CTRL-C - shutting down
[networkMonitor.cpp] Sending SIGUSR1 signal to child process (PID: 23266)...
[networkMonitor.cpp] Sending SIGUSR1 signal to child process (PID: 23267)...
[intfMonitor.cpp] Shutting down
[intfMonitor.cpp] Shutting down
[networkMonitor.cpp] A child process has exited.
[networkMonitor.cpp] A child process has exited.
[networkMonitor.cpp] Closed monitor socket 4
[networkMonitor.cpp] Closed monitor socket 5
[networkMonitor.cpp] Master socket closed.
[networkMonitor.cpp] Socket path unlinked successfully.
aryank1511@aryanubuntu:~/Desktop/UNX511-Seneca/_assignments/a1$ 
```

Here you can see the output of the network monitor side-by-side with the **ifconfig** command.

Here, you can see how there is one **networkMonitor** process and two **intfMonitor** child processes since we are monitoring two network interfaces.

Lastly, you can see how the interface monitor turns the interface back up if it goes down.