Assignment #2: SNMP Management Station

**Functional Specification:**

For this assignment, we used the net-snmp C library to create a SNMP capable management station. This management station has the capability to obtain a list of device interfaces, device IP neighbors, and traffic on each interface. Before running this program, it is assumed that there is already a SNMP agent running (with a known community name) and that the net-snmp library is installed.

1. The user is given four different prompts to specify how often traffic should be extracted (time interval), the number of samples to be observed for traffic, the IP address of the SNMP agent to be observed, and the community name of the SNMP agent.
2. With the given community name and IP address of the SNMP agent, the program will initiate a session with the SNMP agent to obtain the desired data.
3. The program should output a table of device interfaces (numbered starting from 0) with their corresponding IP addresses.
4. The program should output a table of IP neighbors for all device interfaces, which is designated by the interface number. This interface number should correspond to the device interfaces in the table of device interfaces [see 3].
5. The program should output the incoming traffic and outcoming traffic for each interface in bytes/second.

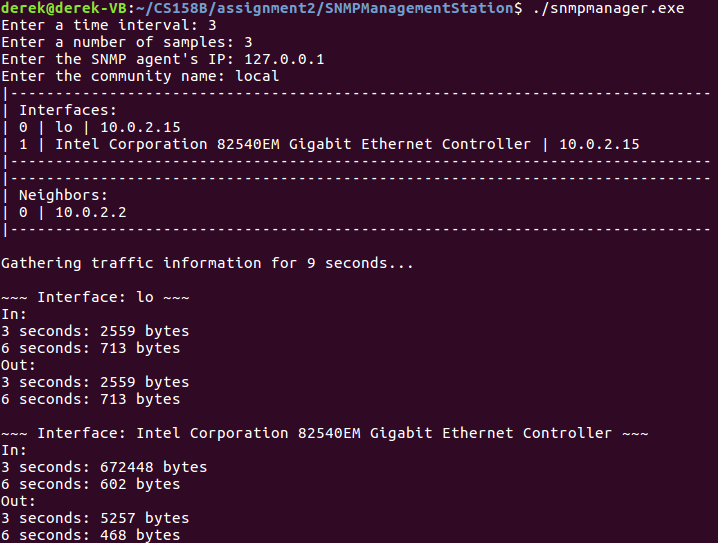


Figure 1: Program Demo

**Design Specification:**

To send the requests to the SNMP agent, we wrote a method called snmp\_walk, which inputs a netsnmp\_session and a char pointer. This method will construct a SNMP PDU with the specified parameters (allowing for multiple requests in a single request), send the request, and process the response, returning a netsnmp\_pdu.

*Obtaining Device Interface List:*  we use the method snmp\_walk, passing in ifDescr as first\_oid to obtain all the names of the interfaces. The same method is used again for the IP addresses for the interfaces, but passing in ifAddr as the first\_oid.

*Obtaining IP Neighbors to Device Interface:* The method, snmp\_walk, but instead use ipNetToMediaNetAddress for our first\_oid. The ipNetToMediaNetAddress is within the ipNetToMediaTable, which is the IP Address Translation Table. This allows us to obtain the IP addresses of the neighbors of the device interfaces. If, on the other hand, we wanted to obtain the Layer 2 addresses (MAC), we would use the ipNetToMediaPhysAddress.

*Obtaining Traffic:* Using the MIB objects ifInOctets and ifOutOctets, we use snmp\_walk() to obtain the pdu from the SNMP agent. This pdu is then passed as an argument into getTrafficFromPDU(), and processed to obtain a list of the interfaces with the corresponding traffic rates. This traffic polling is done in a loop (with the number of samples), calling sleep() with the specified time interval.

**Accuracy Analysis for Traffic**:

The accuracy of the traffic is reliant on the user-given parameters, primarily the time interval in which samples are taken. In the given program demo [see Figure 1], we can see that the management station polls every 3 seconds for traffic information (via snmp walk using ifInOctets and ifOutOctets). If within the time interval between polling, there is a sudden spike in traffic, this would not be reflected in the traffic statistics of our application. Thus, with smaller time intervals between polling, there would be better results.

Notice however, that with more frequent polling, it would also result in more processing for the SNMP agent (which could strain the resources). If the resources of the SNMP agent are unable to keep up with the number of polling requests, this could also result in a delay in processing the polling requests and obtaining the data (ifInOctets and ifOutOctets). This time delay would mean that it is possible that the time interval between each polling request may differ (more time delay if there is a strain in resources).