Leung Wai Chan, Alexander Hwang

HW2: SNMP

Functional Specification:

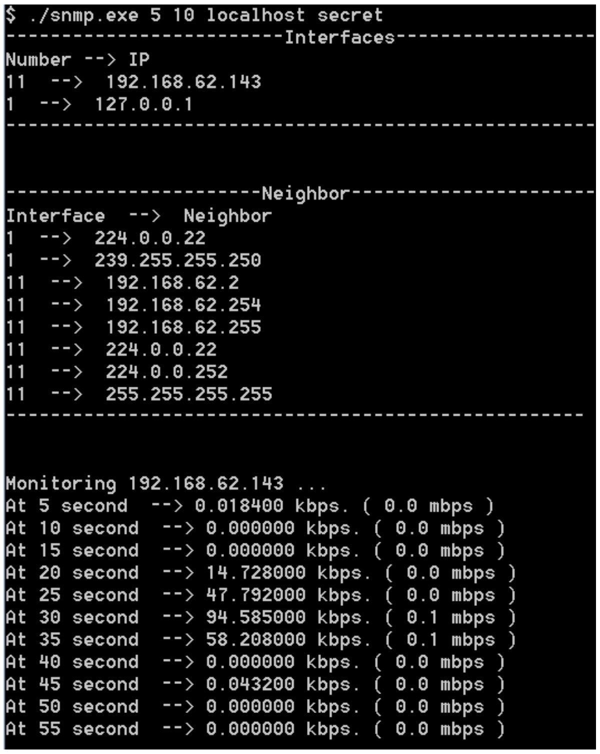
The program will discover the interface in the device, discover the IP neighbors, and monitor the agent interface's traffic given time interval between polling, number of samples take, ip address of agent, and community.

Design Specification

The programming language used is C with netsnmp library. All functions to get MIB object are by netsnmp functions . The program will set up connection and session. Then, for each snmp method call, the PDU request will be generated. Upon receiving the response from the agent, some business logic will be ran and parsed. When finished , the session need to be cleared so that next call will be running properly.

Testing

1) 5 seconds interval , 10 samples , localhost , secret community name. No network browsing activity



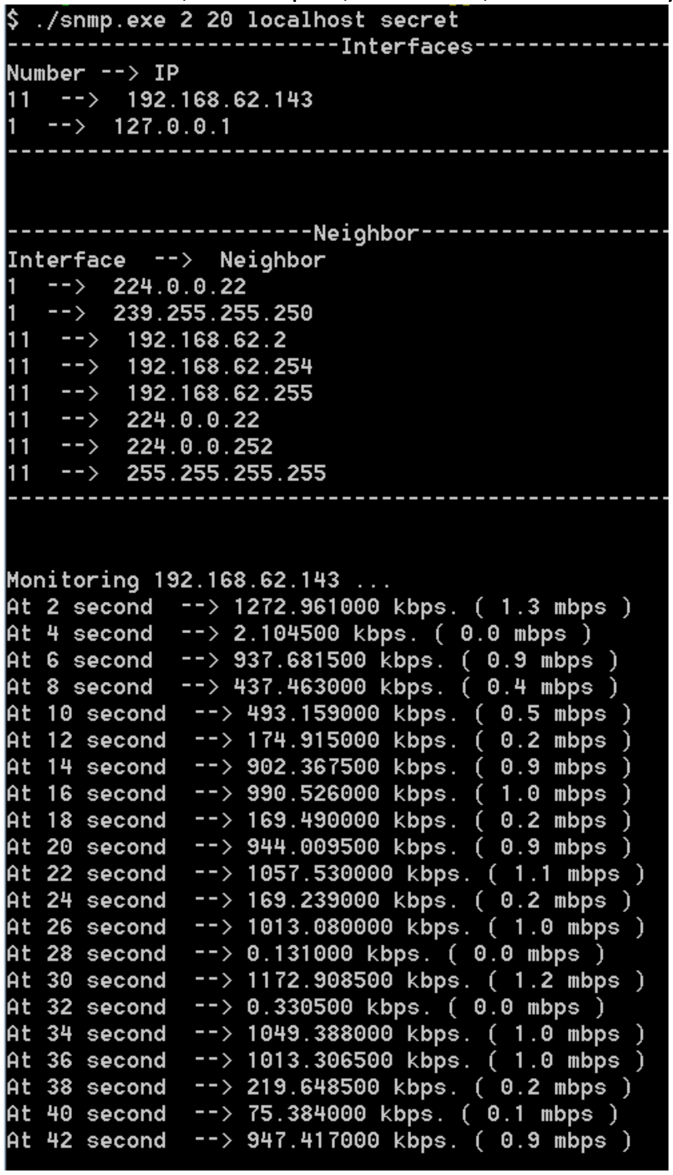
```
$ ./snmp.exe 5 10 localhost secret
------------------------Interfaces-----------------
Number --> IP
11  -->  192.168.62.143
1  -->  127.0.0.1
---------------------------------------------------



----------------------Neighbor---------------------
Interface  -->  Neighbor
1  -->  224.0.0.22
1  -->  239.255.255.250
11  -->  192.168.62.2
11  -->  192.168.62.254
11  -->  192.168.62.255
11  -->  224.0.0.22
11  -->  224.0.0.252
11  -->  255.255.255.255
--------------------------------------------------


Monitoring 192.168.62.143 ...
At 5 second  --> 0.018400 kbps. ( 0.0 mbps )
At 10 second  --> 0.000000 kbps. ( 0.0 mbps )
At 15 second  --> 0.000000 kbps. ( 0.0 mbps )
At 20 second  --> 14.728000 kbps. ( 0.0 mbps )
At 25 second  --> 47.792000 kbps. ( 0.0 mbps )
At 30 second  --> 94.585000 kbps. ( 0.1 mbps )
At 35 second  --> 58.208000 kbps. ( 0.1 mbps )
At 40 second  --> 0.000000 kbps. ( 0.0 mbps )
At 45 second  --> 0.043200 kbps. ( 0.0 mbps )
At 50 second  --> 0.000000 kbps. ( 0.0 mbps )
At 55 second  --> 0.000000 kbps. ( 0.0 mbps )
```

2) 2 sec interval , 20 samples, localhost , secret. With youtube video running



```
$ ./snmp.exe 2 20 localhost secret
-------------------------Interfaces--------------
Number --> IP
11  -->  192.168.62.143
1  -->  127.0.0.1
--------------------------------------------------



-----------------------Neighbor------------------
Interface  -->  Neighbor
1  -->  224.0.0.22
1  -->  239.255.255.250
11  -->  192.168.62.2
11  -->  192.168.62.254
11  -->  192.168.62.255
11  -->  224.0.0.22
11  -->  224.0.0.252
11  -->  255.255.255.255
--------------------------------------------------



Monitoring 192.168.62.143 ...
At 2 second  --> 1272.961000 kbps. ( 1.3 mbps )
At 4 second  --> 2.104500 kbps. ( 0.0 mbps )
At 6 second  --> 937.681500 kbps. ( 0.9 mbps )
At 8 second  --> 437.463000 kbps. ( 0.4 mbps )
At 10 second  --> 493.159000 kbps. ( 0.5 mbps )
At 12 second  --> 174.915000 kbps. ( 0.2 mbps )
At 14 second  --> 902.367500 kbps. ( 0.9 mbps )
At 16 second  --> 990.526000 kbps. ( 1.0 mbps )
At 18 second  --> 169.490000 kbps. ( 0.2 mbps )
At 20 second  --> 944.009500 kbps. ( 0.9 mbps )
At 22 second  --> 1057.530000 kbps. ( 1.1 mbps )
At 24 second  --> 169.239000 kbps. ( 0.2 mbps )
At 26 second  --> 1013.080000 kbps. ( 1.0 mbps )
At 28 second  --> 0.131000 kbps. ( 0.0 mbps )
At 30 second  --> 1172.908500 kbps. ( 1.2 mbps )
At 32 second  --> 0.330500 kbps. ( 0.0 mbps )
At 34 second  --> 1049.388000 kbps. ( 1.0 mbps )
At 36 second  --> 1013.306500 kbps. ( 1.0 mbps )
At 38 second  --> 219.648500 kbps. ( 0.2 mbps )
At 40 second  --> 75.384000 kbps. ( 0.1 mbps )
At 42 second  --> 947.417000 kbps. ( 0.9 mbps )
```

Deliverable analysis:

I used different approaches to get the results, including getbulk, get, and getnext. Getbulk is very convenient as list of OID object can be retrieved with 1 request which saves bandwidth when lots of OID access are necessary. Although it has this advantage, it suffers from another issue that usually it is returning more than OID we actually need.
Getnext is very useful when I want to retrieve to multiples columns' values as oppsed to getbulk which it iterates through each row before it goes to another column. The getnext gives us better control to manipulate the operation.
To find out if we step over at the end of the row , I used OID's data type to determine as fortunately in this project there is no 2 consecutives column with same type. Otherwise, I have to figure the OID index to find out if we are at the end of the column.

As for the traffic monitoring features, the shorter the interval the more accuracy of the data we can get but that also means a lot request need to be made. That’s a trade off. I’d say the result is pretty accurate for my program as the speed increases as I opened other software that takes network consumption or watched video.