Packet Capturing Tool on Windows
Design Document

By – Siddharth Sundewar

2021177557

1. Overview:

1.1 Requirements:

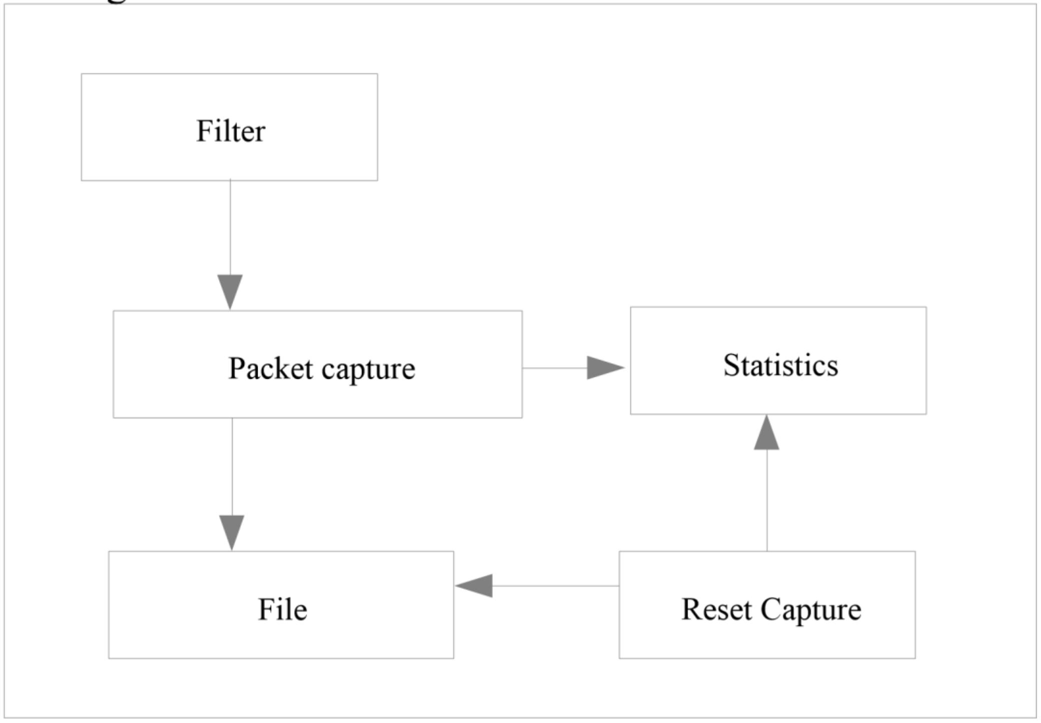
- Capture & display/log details of all incoming packets
- Display/log to be formatted
- Support for filters - user should be able to specify to log only packets from certain parameters like: - IP address
- Protocol type
- Port number
- Maintain and print statistics whenever requested

1.2 Scope:

The tool is capable of capturing and processing all packets that arrive on the system's default interface and display the contents of the packets in a formatted manner in a file. The tool also allows the user to set filters for the packets which he/she wants to log into the file. The user can also view the statistics whenever required and also perform a reset of the resources.

Design of The System

2. Block Diagram:



Packet Capture:

The capturing and processing of packets takes place in this block. For this purpose, the winpcap library is used. The winpcap library provides functions for selecting the interface on which the user wants to capture packets, set the NIC card into promiscuous mode (promiscuous mode enables your system to capture the packets which are not intended for your system also) and start a live capture for the specified interface. Init and Ununit are the sub blocks withing the Packet Capture block.

Init:

The message queue and the semaphore are created and the file is opened.

Uninit:

The message queue is closed and deleted. The file is closed. The threads are terminated. The semaphore is closed and removed and the application is terminated.

File:

The packets obtained from the Packet Capture block are logged into a file. The packets are formatted as per the different headers in the packet and then logged into the file. The user is required to look into this file in order to view the packets. The maximum number of packets the file can store is set to be 500 after which the file is cleared before logging the forthcoming packets.

Filter:

This block allows the user to set the filter depending on the packets he/she wishes to log into the file. Functions of the **winpcap** library are used for this purpose , which allow dynamic setting of the filter. The user can perform filtering by providing the IP address, Port No. or the protocol type.

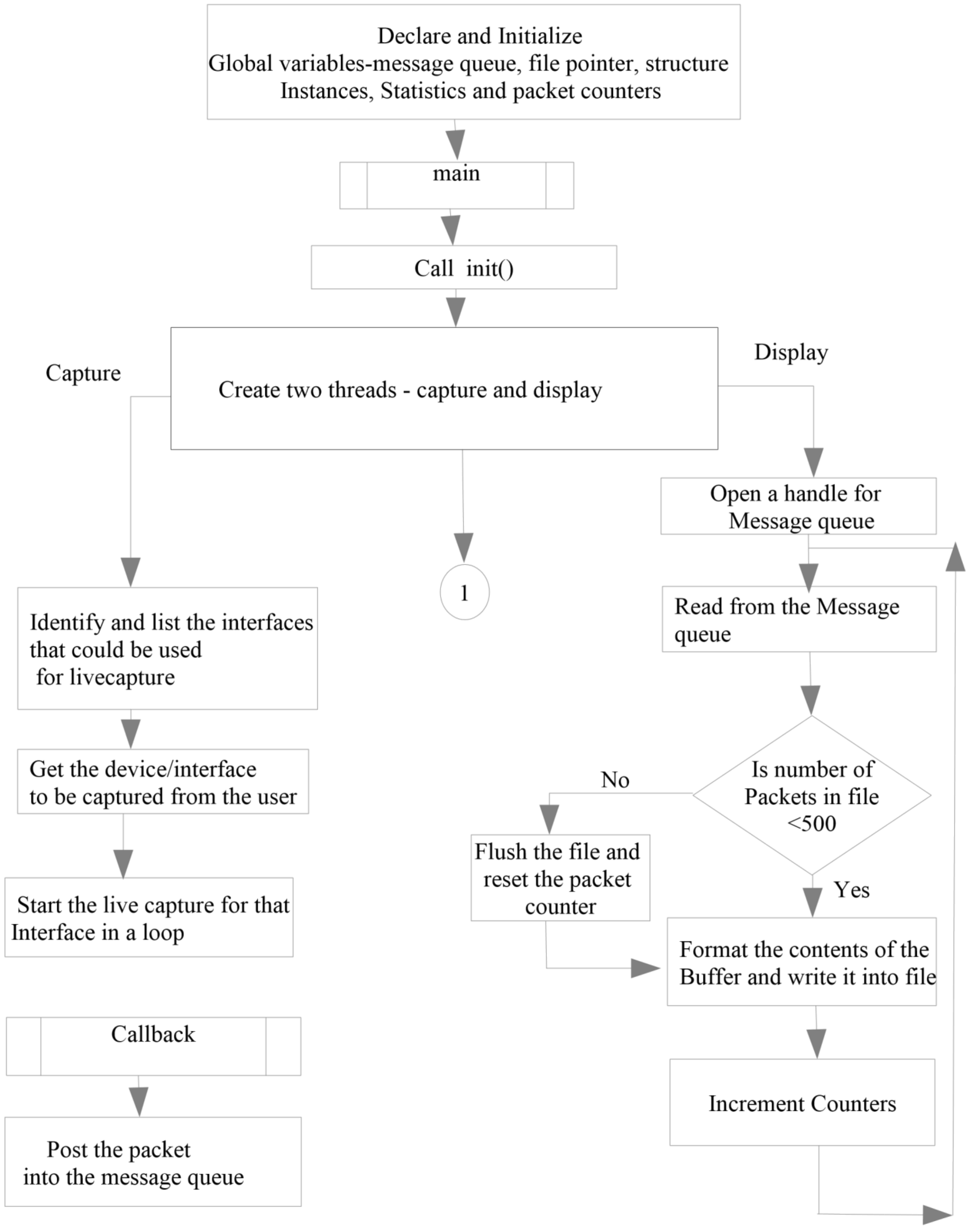
Statistics:

This block displays the counters for packets of different protocol types on user's request.

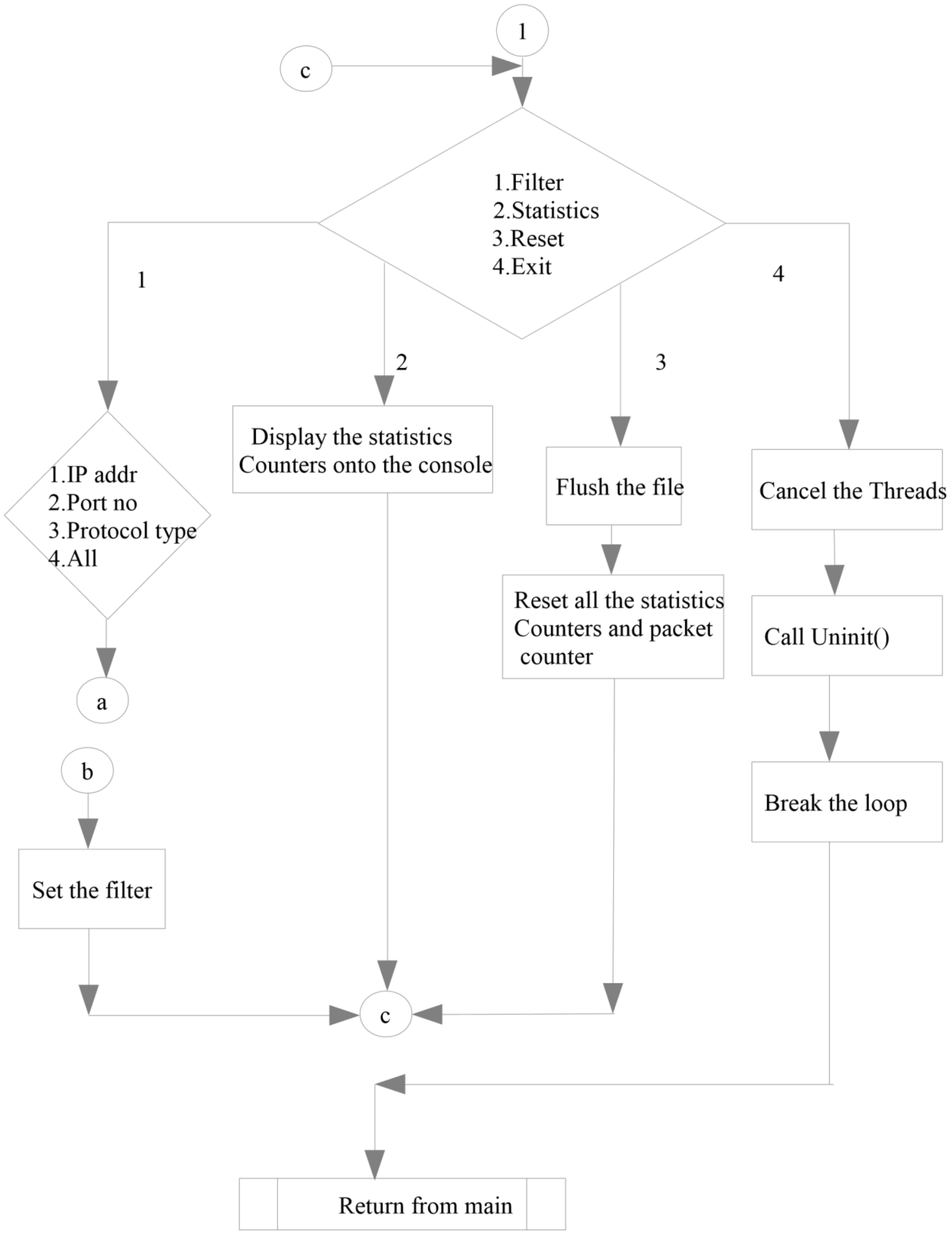
Reset Capture:

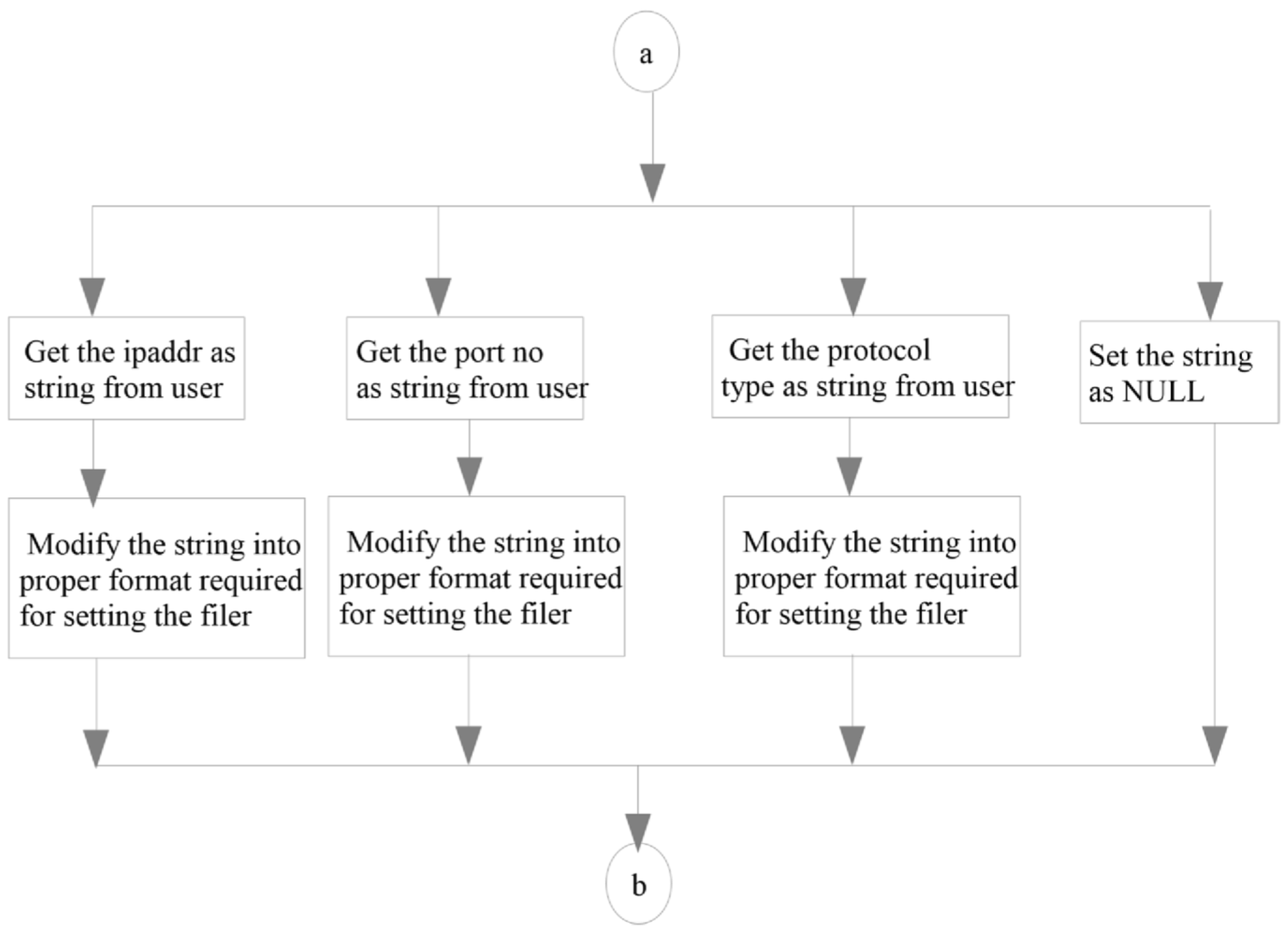
This block comes into the picture when the user requires a reset. All the counters are reset to 0 and the contents of the file are cleared and a fresh log of packets into the file is started.

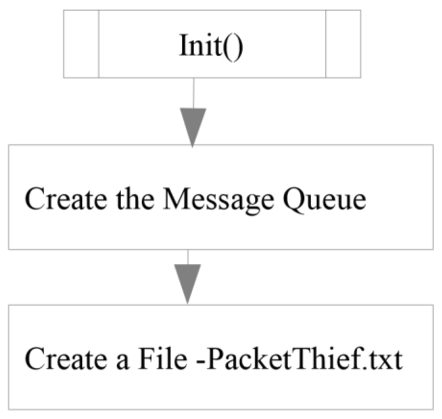
3. Flow Diagram

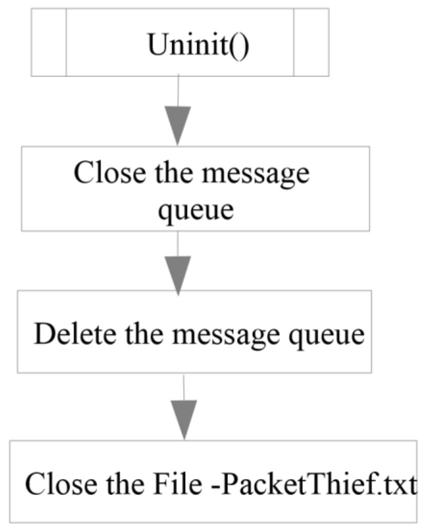


Declare and Initialize
Global variables-message queue, file pointer, structure Instances, Statistics and packet counters
main
Call init()
Capture
Create two threads - capture and display
Display
Open a handle for Message queue
1
Identify and list the interfaces that could be used for livecapture
Read from the Message queue
Get the device/interface to be captured from the user
No
Is number of Packets in file <500
Flush the file and reset the packet counter
Yes
Start the live capture for that Interface in a loop
Format the contents of the Buffer and write it into file
Callback
Increment Counters
Post the packet into the message queue


1
c
1.Filter
2.Statistics
3.Reset
4.Exit
1
2
3
4
1.IP addr
2.Port no
3.Protocol type
4.All
Display the statistics Counters onto the console
Flush the file
Cancel the Threads
Reset all the statistics Counters and packet counter
Call Uninit()
a
b
Break the loop
Set the filter
c
Return from main


a
Get the ipaddr as string from user
Get the port no as string from user
Get the protocol type as string from user
Set the string as NULL
Modify the string into proper format required for setting the filer
Modify the string into proper format required for setting the filer
Modify the string into proper format required for setting the filer
b


Init()
Create the Message Queue
Create a File -PacketThief.txt


Uninit()
Close the message queue
Delete the message queue
Close the File -PacketThief.txt

3.2 Flow diagram explanation

1.Declare and initialize the global variables – message queue,file pointer,statistics and packet counters.
2.Create the message queue using mq_open() function and the semaphore and open the file in init().
3.Create two additional threads by name capture and display using pthread_create().

3.1 In the capture thread,

a) Identify the interface using pcap_lookupdev() function.
b) Create the live capture handle using pcap_create() for that interface.
c) Set promiscuous mode on the handle using pcap_set_promisc().
d)Set the snapshot length on the handle using pcap_set_snaplen().
e)Set the buffer size that will be used on the handle,using pcap_set_buffer_size().
f)Activate the created handle for capturing using pcap_activate().
g)Process the packets from the live capture in a loop by using pcap_loop().
h)pcap_loop() in turn calls a callback function where the packet is moved into the message queue using mq_send().

3.2 In the display thread,

a) Declare and initialize the structure instances.
b)Open a file.
c)Read from the message queue using mq_receive() and store it in a buffer.
d)Check the number of packets stored in the file,if it is less than 500 then format the contents and write into the file,if not flush the file and reset the packet counter. e)Increment all the counters
f)Repeat from step (c) till the application terminates.

3.3 In the main() thread,

a)Display the menu-1.Filter 2.Statistics 3.Reset 4.Exit.
case FILTER:
Get the netmask, using pcap_lookupnet(), which is a parameter to be used in pcap_compile().
Compile the user input to filter expression using pcap_compile().
Set the filter using pcap_setfilter().
Go to (a).
case STATISTICS:
Display the statistics counters onto the console.
Go to (a). case RESET:
Flush the file.
Reset all the counters.
Go to (a).
Case EXIT:
Cancel the threads and call uninit().
Close the message queue using mq_close() and remove the message queue using mq_unlink(), close the file and close the semaphore in uninit(). Return from main().

4. Libraries

4.1 WINPCAP library :

Need : PacketThief application is built on winpcap library for packet capture .The **winpcap** library provides us with the APIs required for packet capture namely setting the NIC card into promiscuous mode , creating handle for a particular interface on which we want to capture packets and setting the filter for receiving only the packets you require from the wire. The **winpcap functions used in PacketThief** are listed below:

1. **pcap_lookupdev :find the default device on which to capture**

SYNOPSIS #include <pcap/pcap.h>

char errbuf[PCAP_ERRBUF_SIZE]; char

*pcap_lookupdev(char *errbuf);

DESCRIPTION
pcap_lookupdev() returns a pointer to a string giving the name of a network device suitable for use with pcap_create() and pcap_activate(), or with pcap_open_live(), and with pcap_lookupnet(). If there is an error, NULL is returned and errbuf is filled in with an appropriate error message. errbuf is assumed to be able to hold at least PCAP_ERRBUF_SIZE chars.

RETURN VALUE
pcap_lookupdev() returns a string denoting the default interface of the system.

2. **pcap_create :create a live capture handle**

SYNOPSIS #include <pcap/pcap.h> char

errbuf[PCAP_ERRBUF_SIZE]; pcap_t

*pcap_create(const char *source, char *errbuf);

DESCRIPTION
pcap_create() is used to create a packet capture handle to look at packets on the network. source is a string that specifies the network device to open; on Linux systems with 2.2 or later kernels, a source argument of "any" or NULL can be used to capture packets from all interfaces.

The returned handle must be activated with pcap_activate() before packets can be captured with it; options for the capture, such as promiscuous mode, can be set on the handle before activating it.

RETURN VALUE
pcap_create() returns a pcap_t * on success and NULL on failure. If NULL is returned, errbuf is filled in with an appropriate error message. errbuf is assumed to be able to hold at least PCAP_ERRBUF_SIZE chars.

3. **pcap_set_promisc : set promiscuous mode for a not-yet-activated capture handle**

SYNOPSIS   #include <pcap/pcap.h>   int
pcap_set_promisc(pcap_t *p, int promisc);

DESCRIPTION
pcap_set_promisc() sets whether promiscuous mode should be set on a capture handle when the handle is activated. If promisc is non zero, promiscuous mode will be set, otherwise it will not be set.
RETURN VALUE
pcap_set_promisc() returns 0 on success or PCAP_ERROR_ACTIVATED if called on a capture handle that has been activated.

4. **pcap_set_snaplen : set the snapshot length for a not-yet-activated capture handle**

SYNOPSIS   #include <pcap/pcap.h>   int pcap_set_snaplen(pcap_t *p,
int snaplen);

DESCRIPTION
pcap_set_snaplen() sets the snapshot length to be used on a capture handle when the handle is activated to snaplen.

RETURN VALUE
pcap_set_snaplen() returns 0 on success or PCAP_ERROR_ACTIVATED if called on a capture handle that has been activated.

5. **pcap_set_buffer_size : set the buffer size for a not-yet-activated capture handle**

SYNOPSIS   #include <pcap/pcap.h>   int pcap_set_buffer_size(pcap_t *p, int
buffer_size);

DESCRIPTION
pcap_set_buffer_size() sets the buffer size that will be used on a capture handle when the handle is activated to buffer_size, which   is in units of bytes.

RETURN VALUE
pcap_set_buffer_size() returns 0 on success or PCAP_ERROR_ACTIVATED if called on a capture handle that has been activated.

6. **pcap_activate : activate a capture handle**

SYNOPSIS   #include <pcap/pcap.h>   int
pcap_activate(pcap_t *p);

DESCRIPTION
pcap_activate() is used to activate a packet capture handle to look at packets on the network, with the options that were set on the handle being in effect.

RETURN VALUE

pcap_activate() returns 0 on success without warnings. If PCAP_WARNING or PCAP_ERROR is returned, pcap_geterr() or pcap_perror() may be called with p as an argument to fetch or display a message describing the warning or error.

7. **pcap_lookupnet : find the IPv4 network number and netmask for a device**

SYNOPSIS #include <pcap/pcap.h> char errbuf[PCAP_ERRBUF_SIZE]; int

pcap_lookupnet(const char *device, bpf_u_int32 *netp,bpf_u_int32 *maskp, char *errbuf);

DESCRIPTION

pcap_lookupnet() is used to determine the IPv4 network number and mask associated with the network device device. Both netp and maskp are bpf_u_int32 pointers.

RETURN VALUE

pcap_lookupnet() returns 0 on success and -1 on failure. If -1 is returned, errbuf is filled in with an appropriate error message. errbuf is assumed to be able to hold at least PCAP_ERRBUF_SIZE chars.

8. pcap_compile :compile a filter expression

SYNOPSIS #include
<pcap/pcap.h>

int pcap_compile(pcap_t *p, struct bpf_program *fp, const char *str, int optimize, bpf_u_int32 netmask);

DESCRIPTION

pcap_compile() is used to compile the string str into a filter program. Man page pcap-filter(7) gives more details about the syntax of that string. fp is a pointer to a bpf_program struct and is filled in by pcap_compile(). optimize controls whether optimization on the resulting code is performed. netmask specifies the IPv4 netmask of the network on which packets are being captured; it is used only when checking for IPv4 broadcast addresses in the filter program.

If the netmask of the network on which packets are being captured isn't known to the program, or if packets are being captured on the Linux "any" pseudo-interface that can capture on more than one network, a value of PCAP_NETMASK_UNKNOWN can be supplied; tests for IPv4 broadcast addresses will fail to compile, but all other tests in the filter program will be OK.

RETURN VALUE

pcap_compile() returns 0 on success and -1 on failure. If -1 is returned, pcap_geterr() or pcap_perror() may be called with p as an argument to fetch or display the error text. **9. pcap_setfilter : set the filter**

SYNOPSIS #include <pcap/pcap.h> int

pcap_setfilter(pcap_t *p, struct bpf_program *fp); DESCRIPTION

pcap_setfilter() is used to specify a filter program. fp is a pointer to a bpf_program struct,usually the result of a call to pcap_compile().
RETURN VALUE
pcap_setfilter() returns 0 on success and -1 on failure. If -1 is returned, pcap_geterr() or pcap_perror() may be called with p as an argument to fetch or display the error text. 10. **pcap_loop : process packets from a live capture or savefil**e

SYNOPSIS #include <pcap/pcap.h> typedef void (*pcap_handler)(u_char *user, const struct pcap_pkthdr *h, const u_char *bytes); int pcap_loop(pcap_t *p, int cnt , pcap_handler callback, u_char *user);

DESCRIPTION
pcap_loop() processes packets from a live capture or "savefile" until cnt packets are processed, the end of the "savefile" is reached when reading from a “savefile”, pcap_breakloop() is called, or an error occurs. It does not return when live read timeouts occur. A value of -1 or 0 for cnt is equivalent to infinity, so that packets are processed until another ending condition occurs.

A value of -1 or 0 for cnt causes all the packets received in one buffer to be processed when reading a live capture.

callback specifies a pcap_handler routine to be called with three arguments: a u_char pointer which is passed in the user argument to pcap_loop() , a const struct pcap_pkthdr pointer pointing to the packet time stamp and lengths, and a const u_char pointer to the first caplen (as given in the struct pcap_pkthdr a pointer to which is passed to the callback routine) bytes of data from the packet. .

RETURN VALUE
pcap_loop() returns 0 if cnt is exhausted, -1 if an error occurs, or -2 if the loop terminated due to a call to pcap_breakloop() before any packets were processed.
If -1 is returned, pcap_geterr() or pcap_perror() may be called with p as an argument to fetch or display the error text.

4.2 PTHREADs:

Need : PacketThief requires to read the packets from the interface , format the packets , log the packets in the file and also wait on user 's input for setting the filter to read only the specified packets. As these functions need to working simultaneously we felt the need for threads to carry out these functions individually with loose coupling.

The **Pthread functions used in PacketThief** are listed below:

1. **pthread_create : create a new thread**
2. **pthread_cancel : cancels the thread**

4.3 MESSAGE QUEUES:

Need : PacketThief formats the packets before logging them into the file . The formatting of the packets is time consuming . Hence we have used message queue to one end of which a thread would be sending the packets(as messages) and another thread picks up the packets and logs them into the file after formatting .

The **Message Queue functions used in PacketThief** are listed below:

1. **mq_open(): open a message queue**
2. **mq_send(): send a message to a message queue**
3. **mq_receive(): receive a message from a message queue**
4. **mq_close():close a message queue descriptor**
5. **mq_unlink(): remove a message queue**

OUTPUT

5. Sample Packet Format

```
Packet Number:1
Ethernet Header
Destination MAC: ff ff ff ff ff ff (broadcast)
Source MAC: 00 23 24 c1 ef be

Ipv4 Header
Version: 4
Source IP: 10.75.15.117
Destination IP: 10.75.15.255
Header Length: 20
Total Length: 78
Identification: 3ed8
Flags: 0
Fragmentation Offset: 0
Time to Live: 128
Protocol: 17 (UDP)
Header Checksum: c7bd

UDP Header
Source Port: 137
Destination Port: 137
Length: 58
Checksum: febc

Payload
8a bc 00 00 aa bb cc dd ee ff 11 22 33 44 55 66 77 88 99
```

NOTE :-
For the GUI, we will use the gtk library. A tool by the name glade can be used which allows development of the GUI and uses gtk at the backend. It is the library which can be directly used to develop a GUI for our ANSI C code for packet capturing tool.

FUTURE ENHANCEMENTS

- We can port the code with some minor changes to linux using libpcap instead of winpcap and as a result we can use this tool in any linux environment.
- We can also use raw sockets instead of winpcap to built the packet capturing tool.

REFERENCES

- http://www.winpcap.org/docs/default.htm
- http://en.wikipedia.org/wiki/Pcap
-
