/\*==================================================================================

Assignment :- A-1

Title :- Implementation of Packet sniffer. Program should identify header of each protocol.Use multi- core programming.

Roll\_No. :- 54

==================================================================================\*/

#include<netinet/in.h>

#include<errno.h>

#include<netdb.h>

#include<stdio.h> //For standard things

#include<stdlib.h> //malloc

#include<string.h> //strlen

#include<netinet/ip\_icmp.h> //Provides declarations for icmp header

#include<netinet/udp.h> //Provides declarations for udp header

#include<netinet/tcp.h> //Provides declarations for tcp header

#include<netinet/ip.h> //Provides declarations for ip header

#include<netinet/if\_ether.h> //For ETH\_P\_ALL

#include<net/ethernet.h> //For ether\_header

#include<sys/socket.h>

#include<arpa/inet.h>

#include<sys/ioctl.h>

#include<sys/time.h>

#include<sys/types.h>

#include<unistd.h>

void ProcessPacket(unsigned char\* , int);

void print\_ip\_header(unsigned char\* , int);

void print\_tcp\_packet(unsigned char \* , int );

void print\_udp\_packet(unsigned char \* , int );

void print\_icmp\_packet(unsigned char\* , int );

void PrintData (unsigned char\* , int);

FILE \*logfile;

struct sockaddr\_in source,dest;

int tcp=0,udp=0,icmp=0,others=0,igmp=0,total=0,i,j;

int main()

{

int saddr\_size , data\_size;

struct sockaddr saddr;

unsigned char \*buffer = (unsigned char \*) malloc(65536); //Its Big!

logfile=fopen("log.txt","w");

if(logfile==NULL)

{

printf("Unable to create log.txt file.");

}

printf("Starting...\n");

int sock\_raw = socket( AF\_PACKET , SOCK\_RAW , htons(ETH\_P\_ALL)) ;

//setsockopt(sock\_raw , SOL\_SOCKET , SO\_BINDTODEVICE , "eth0" , strlen("eth0")+ 1 );

if(sock\_raw < 0)

{

//Print the error with proper message

perror("Socket Error");

return 1;

}

while(1)

{

saddr\_size = sizeof saddr;

//Receive a packet

data\_size = recvfrom(sock\_raw , buffer , 65536 , 0 , &saddr , (socklen\_t\*)&saddr\_size);

if(data\_size <0 )

{

printf("Recvfrom error , failed to get packets\n");

return 1;

}

//Now process the packet

ProcessPacket(buffer , data\_size);

}

close(sock\_raw);

printf("Finished");

return 0;

}

void ProcessPacket(unsigned char\* buffer, int size)

{

//Get the IP Header part of this packet , excluding the ethernet header

struct iphdr \*iph = (struct iphdr\*)(buffer + sizeof(struct ethhdr));

++total;

switch (iph->protocol) //Check the Protocol and do accordingly...

{

case 1: //ICMP Protocol

++icmp;

print\_icmp\_packet( buffer , size);

break;

case 2: //IGMP Protocol

++igmp;

break;

case 6: //TCP Protocol

++tcp;

print\_tcp\_packet(buffer , size);

break;

case 17: //UDP Protocol

++udp;

print\_udp\_packet(buffer , size);

break;

default: //Some Other Protocol like ARP etc.

++others;

break;

}

printf("TCP : %d UDP : %d ICMP : %d IGMP : %d Others : %d Total : %d\r", tcp , udp , icmp , igmp , others , total);

}

void print\_ethernet\_header(unsigned char\* Buffer, int Size)

{

struct ethhdr \*eth = (struct ethhdr \*)Buffer;

fprintf(logfile , "\n");

fprintf(logfile , "Ethernet Header\n");

fprintf(logfile , " |-Destination Address : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n", eth->h\_dest[0] , eth-

>h\_dest[1] , eth->h\_dest[2] , eth->h\_dest[3] , eth->h\_dest[4] , eth->h\_dest[5] );

fprintf(logfile , " |-Source Address : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n", eth->h\_source[0] , eth-

>h\_source[1] , eth->h\_source[2] , eth->h\_source[3] , eth->h\_source[4] , eth->h\_source[5] );

fprintf(logfile , " |-Protocol : %u \n",(unsigned short)eth->h\_proto);

}

void print\_ip\_header(unsigned char\* Buffer, int Size)

{

print\_ethernet\_header(Buffer , Size);

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)(Buffer + sizeof(struct ethhdr) );

iphdrlen =iph->ihl\*4;

memset(&source, 0, sizeof(source));

source.sin\_addr.s\_addr = iph->saddr;

memset(&dest, 0, sizeof(dest));

dest.sin\_addr.s\_addr = iph->daddr;

fprintf(logfile , "\n");

fprintf(logfile , "IP Header\n");

fprintf(logfile , " |-IP Version : %d\n",(unsigned int)iph->version);

fprintf(logfile , " |-IP Header Length : %d DWORDS or %d Bytes\n",(unsigned int)iph->ihl,((unsigned

int)(iph->ihl))\*4);

fprintf(logfile , " |-Type Of Service : %d\n",(unsigned int)iph->tos);

fprintf(logfile , " |-IP Total Length : %d Bytes(Size of Packet)\n",ntohs(iph->tot\_len));

fprintf(logfile , " |-Identification : %d\n",ntohs(iph->id));

//fprintf(logfile , " |-Reserved ZERO Field : %d\n",(unsigned int)iphdr->ip\_reserved\_zero);

//fprintf(logfile , " |-Dont Fragment Field : %d\n",(unsigned int)iphdr->ip\_dont\_fragment);

//fprintf(logfile , " |-More Fragment Field : %d\n",(unsigned int)iphdr->ip\_more\_fragment);

fprintf(logfile , " |-TTL : %d\n",(unsigned int)iph->ttl);

fprintf(logfile , " |-Protocol : %d\n",(unsigned int)iph->protocol);

fprintf(logfile , " |-Checksum : %d\n",ntohs(iph->check));

fprintf(logfile , " |-Source IP : %s\n",inet\_ntoa(source.sin\_addr));

fprintf(logfile , " |-Destination IP : %s\n",inet\_ntoa(dest.sin\_addr));

}

void print\_tcp\_packet(unsigned char\* Buffer, int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)( Buffer + sizeof(struct ethhdr) );

iphdrlen = iph->ihl\*4;

struct tcphdr \*tcph=(struct tcphdr\*)(Buffer + iphdrlen + sizeof(struct ethhdr));

int header\_size = sizeof(struct ethhdr) + iphdrlen + tcph->doff\*4;

fprintf(logfile , "\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TCP Packet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

print\_ip\_header(Buffer,Size);

fprintf(logfile , "\n");

fprintf(logfile , "TCP Header\n");

fprintf(logfile , " |-Source Port : %u\n",ntohs(tcph->source));

fprintf(logfile , " |-Destination Port : %u\n",ntohs(tcph->dest));

fprintf(logfile , " |-Sequence Number : %u\n",ntohl(tcph->seq));

fprintf(logfile , " |-Acknowledge Number : %u\n",ntohl(tcph->ack\_seq));

fprintf(logfile , " |-Header Length : %d DWORDS or %d BYTES\n" ,(unsigned int)tcph->doff,(unsigned

int)tcph->doff\*4);

//fprintf(logfile , " |-CWR Flag : %d\n",(unsigned int)tcph->cwr);

//fprintf(logfile , " |-ECN Flag : %d\n",(unsigned int)tcph->ece);

fprintf(logfile , " |-Urgent Flag : %d\n",(unsigned int)tcph->urg);

fprintf(logfile , " |-Acknowledgement Flag : %d\n",(unsigned int)tcph->ack);

fprintf(logfile , " |-Push Flag : %d\n",(unsigned int)tcph->psh);

fprintf(logfile , " |-Reset Flag : %d\n",(unsigned int)tcph->rst);

fprintf(logfile , " |-Synchronise Flag : %d\n",(unsigned int)tcph->syn);

fprintf(logfile , " |-Finish Flag : %d\n",(unsigned int)tcph->fin);

fprintf(logfile , " |-Window : %d\n",ntohs(tcph->window));

fprintf(logfile , " |-Checksum : %d\n",ntohs(tcph->check));

fprintf(logfile , " |-Urgent Pointer : %d\n",tcph->urg\_ptr);

fprintf(logfile , "\n");

fprintf(logfile , " DATA Dump ");

fprintf(logfile , "\n");

fprintf(logfile , "IP Header\n");

PrintData(Buffer,iphdrlen);

fprintf(logfile , "TCP Header\n");

PrintData(Buffer+iphdrlen,tcph->doff\*4);

fprintf(logfile , "Data Payload\n");

PrintData(Buffer + header\_size , Size - header\_size );

fprintf(logfile , "\n###########################################################");

}

void print\_udp\_packet(unsigned char \*Buffer , int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)(Buffer + sizeof(struct ethhdr));

iphdrlen = iph->ihl\*4;

struct udphdr \*udph = (struct udphdr\*)(Buffer + iphdrlen + sizeof(struct ethhdr));

int header\_size = sizeof(struct ethhdr) + iphdrlen + sizeof udph;

fprintf(logfile , "\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*UDP Packet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

print\_ip\_header(Buffer,Size);

fprintf(logfile , "\nUDP Header\n");

fprintf(logfile , " |-Source Port : %d\n" , ntohs(udph->source));

fprintf(logfile , " |-Destination Port : %d\n" , ntohs(udph->dest));

fprintf(logfile , " |-UDP Length : %d\n" , ntohs(udph->len));

fprintf(logfile , " |-UDP Checksum : %d\n" , ntohs(udph->check));

fprintf(logfile , "\n");

fprintf(logfile , "IP Header\n");

PrintData(Buffer , iphdrlen);

fprintf(logfile , "UDP Header\n");

PrintData(Buffer+iphdrlen , sizeof udph);

fprintf(logfile , "Data Payload\n");

//Move the pointer ahead and reduce the size of string

PrintData(Buffer + header\_size , Size - header\_size);

fprintf(logfile , "\n###########################################################");

}

void print\_icmp\_packet(unsigned char\* Buffer , int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)(Buffer + sizeof(struct ethhdr));

iphdrlen = iph->ihl \* 4;

struct icmphdr \*icmph = (struct icmphdr \*)(Buffer + iphdrlen + sizeof(struct ethhdr));

int header\_size = sizeof(struct ethhdr) + iphdrlen + sizeof icmph;

fprintf(logfile , "\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ICMP Packet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

print\_ip\_header(Buffer , Size);

fprintf(logfile , "\n");

fprintf(logfile , "ICMP Header\n");

fprintf(logfile , " |-Type : %d",(unsigned int)(icmph->type));

if((unsigned int)(icmph->type) == 11)

{

fprintf(logfile , " (TTL Expired)\n");

}

else if((unsigned int)(icmph->type) == ICMP\_ECHOREPLY)

{

fprintf(logfile , " (ICMP Echo Reply)\n");

}

fprintf(logfile , " |-Code : %d\n",(unsigned int)(icmph->code));

fprintf(logfile , " |-Checksum : %d\n",ntohs(icmph->checksum));

//fprintf(logfile , " |-ID : %d\n",ntohs(icmph->id));

//fprintf(logfile , " |-Sequence : %d\n",ntohs(icmph->sequence));

fprintf(logfile , "\n");

fprintf(logfile , "IP Header\n");

PrintData(Buffer,iphdrlen);

fprintf(logfile , "UDP Header\n");

PrintData(Buffer + iphdrlen , sizeof icmph);

fprintf(logfile , "Data Payload\n");

//Move the pointer ahead and reduce the size of string

PrintData(Buffer + header\_size , (Size - header\_size) );

fprintf(logfile , "\n###########################################################");

}

void PrintData (unsigned char\* data , int Size)

{

int i , j;

for(i=0 ; i < Size ; i++)

{

if( i!=0 && i%16==0) //if one line of hex printing is complete...

{

fprintf(logfile , " ");

for(j=i-16 ; j<i ; j++)

{

if(data[j]>=32 && data[j]<=128)

fprintf(logfile , "%c",(unsigned char)data[j]); //if its a number or alphabet

else fprintf(logfile , "."); //otherwise print a dot

}

fprintf(logfile , "\n");

}

if(i%16==0) fprintf(logfile , " ");

fprintf(logfile , " %02X",(unsigned int)data[i]);

if( i==Size-1) //print the last spaces

{

for(j=0;j<15-i%16;j++)

{

fprintf(logfile , " "); //extra spaces

}

fprintf(logfile , " ");

for(j=i-i%16 ; j<=i ; j++)

{

if(data[j]>=32 && data[j]<=128)

{

fprintf(logfile , "%c",(unsigned char)data[j]);

}

else

{

fprintf(logfile , ".");

}

}

fprintf(logfile , "\n" );

}

}

}

/\* OUTPUT

administrator@administrator-OptiPlex-3010:~$ sudo su

[sudo] password for administrator:

root@administrator-OptiPlex-3010:/home/administrator# cd Desktop/

root@administrator-OptiPlex-3010:/home/administrator/Desktop# gcc pc.c

root@administrator-OptiPlex-3010:/home/administrator/Desktop# ./a.out

Starting...

TCP : 900 UDP : 4549 ICMP :^Z4 IGMP : 0 Others : 1564 Total : 7036

\*/