**1. INTRODUCTION**

* ***Socket Programming:***

A socket is one of the most fundamental technologies of computer networking. Sockets allow applications to communicate using standard mechanisms built into network hardware and operating systems. Although network software may seem to be a relatively new "Web" phenomenon, socket technology actually has been employed for roughly two decades.

Socket interfaces can be divided into three categories.

1.Perhaps the most commonly-used type, the **stream socket**, implements "connection-oriented" semantics.

2.**Datagram sockets** offer "connection-less" semantics.

3. The third type of socket -- the so-called **raw socket** -- bypasses the library's built-in support for standard protocols like TCP and UDP. Raw sockets are used for custom low-level protocol development.

In our packet sniffer we have made use of the raw packet sniffers using the following code:

sock\_raw = socket(AF\_INET , SOCK\_RAW , IPPROTO\_TCP);

if(sock\_raw < 0)

{

printf("Socket Error\n");

return 1;

}

while(1)

{

saddr\_size = sizeof saddr;

//Receive a packet

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

if(data\_size <0 )

{

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

return 1;

}

//Now process the packet

ProcessPacket(buffer , data\_size);

}

Here buffer contains the raw packet. Process the packet byte by byte, based on type of packet.

***1.2*** ***Packet Sniffer:***

A packet sniffer (also known as a network analyzer, protocol analyzer or packet analyzer, or for particular types of [networks](http://en.wikipedia.org/wiki/Telecommunications_network), an Ethernet sniffer or wireless sniffer) is a computer program or a piece of computer hardware that can intercept and log traffic passing over a digital network or part of a network. As [data streams](http://en.wikipedia.org/wiki/Data_stream) flow across the network, the sniffer captures each [packet](http://en.wikipedia.org/wiki/Packet_(information_technology)) and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content.

Packet sniffers work by intercepting and logging network traffic that they can 'see' via the wired or wireless network interface that the packet [sniffing software](http://netsecurity.about.com/cs/generalsecurity/g/def_pacsniff.htm) has access to on its host computer.

On a wired network, what can be captured depends on the structure of the network. A [packet sniffer](http://netsecurity.about.com/cs/hackertools/a/aafreepacsniff.htm) might be able to see traffic on an entire network or only a certain segment of it, depending on how the [network switches](http://compnetworking.about.com/od/hardwarenetworkgear/g/bldef_switch.htm) are configured, placed, etc. On wireless networks, packet sniffers can usually only capture one channel at a time unless the host computer has multiple wireless interfaces that allow for multichannel capture.

Once the raw packet data is captured, the packet sniffing software must analyze it and present it in human-readable form so that the person using the packet sniffing software can make sense of it.

 The person [analyzing the data](http://netsecurity.about.com/od/securitytoolprofiles/p/aapranalyzer.htm) can view details of the 'conversation' happening between two or more nodes on the network. Network technicians can use this information to determine where a fault lies, such as determining which device failed to respond to a network request.

Hackers can use sniffers to eavesdrop on unencrypted data in the packets to see what information is being exchanged between two parties. They can also capture information such as passwords and authentication tokens (if they are sent in the clear). Hackers can also capture packets for later playback in replay, man-in-the-middle, and packet injection attacks that some systems may be vulnerable to.

**1.2.1 Applications of Packet Sniffing**:

The versatility of packet sniffers means they can be used to:

* Analyze network problems
* Detect [network intrusion](http://en.wikipedia.org/wiki/Network_intrusion_detection_system) attempts
* Detect network misuse by internal and external users
* Gain information for effecting a network intrusion
* Isolate exploited systems
* Monitor WAN bandwidth utilization
* Monitor network usage (including internal and external users and systems)
* Gather and report network statistics
* Filter suspect content from network traffic
* Serve as primary data source for day-to-day network monitoring and management
* Spy on other network users and collect sensitive information such as login details
* Debug client/server communications
* Verify adds, moves and changes

**Disadvantages of packet Sniffer:**

Negative usage of a sniffer is well known as its harms to network security:

Catching password, which is the main reason for most illegal uses of sniffing tool;

Capturing special and private information of transactions, like username, credit ID, account, and password;

Recording email or instant message and resuming its content;

Some Sniffers can even modify target the computer's information and damage the system;

Interrupting the security of a network or to gain higher level authority.

With more and more hackers using of packet sniffers, it has become one of the most important tools in the defense of cyber-attacks and cyber-crime

**2. TYPES OF PACKETS**

***2.1 TCP packet:***

The **Transmission Control Protocol** (**TCP**) is a core [protocol](http://en.wikipedia.org/wiki/Communications_protocol) of the [Internet Protocol Suite](http://en.wikipedia.org/wiki/Internet_Protocol_Suite). It originated in the initial network implementation in which it complemented the [Internet HYPERLINK "http://en.wikipedia.org/wiki/Internet\_Protocol" Protocol](http://en.wikipedia.org/wiki/Internet_Protocol) (IP). Therefore, the entire suite is commonly referred to as *TCP/IP*. TCP provides [reliable](http://en.wikipedia.org/wiki/Reliability_(computer_networking)), ordered, and [error-checked](http://en.wikipedia.org/wiki/Error_detection_and_correction) delivery of a stream of [octets](http://en.wikipedia.org/wiki/Octet_(computing)) between applications running on hosts communicating over an IP network. TCP is the protocol that major Internet applications such as the [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web), email, HYPERLINK "http://en.wikipedia.org/wiki/Remote\_administration"administrationand [file transfer](http://en.wikipedia.org/wiki/File_transfer) rely on.

Structure of TCP Header:

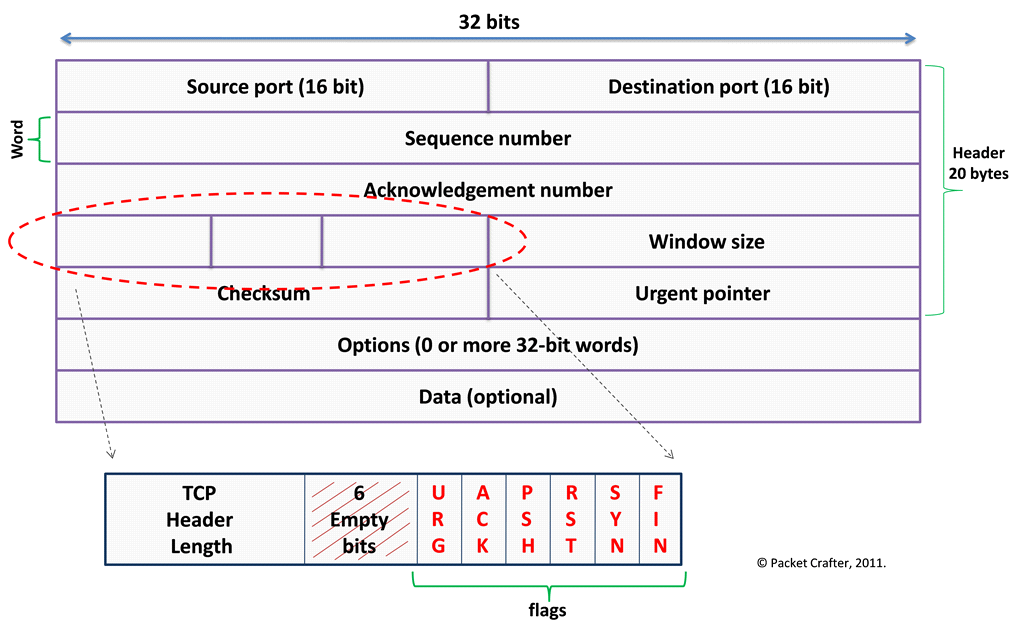


FIGURE: TCP Header Structure

Source port (16 bits)

identifies the sending port

Destination port (16 bits)

identifies the receiving port

Sequence number (32 bits)

has a dual role:If the SYN flag is set (1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1.If the SYN flag is clear (0), then this is the accumulated sequence number of the first data byte of this segment for the current session.

Acknowledgment number (32 bits)

if the ACK flag is set then the value of this field is the next sequence number that the receiver is expecting. This acknowledges receipt of all prior bytes (if any). The first ACKsent by each end acknowledges the other end's initial sequence number itself, but no data.

Data offset (4 bits)

specifies the size of the TCP header in 32-bit words. The minimum size header is 5 words and the maximum is 15 words thus giving the minimum size of 20 bytes and maximum of 60 bytes, allowing for up to 40 bytes of options in the header. This field gets its name from the fact that it is also the offset from the start of the TCP segment to the actual data.

Reserved (3 bits)

for future use and should be set to zero.

Flags (9 bits) (aka Control bits)

contains 9 1-bit flags

Window size (16 bits)

the size of the receive window, which specifies the number of window size units (by default, bytes) (beyond the sequence number in the acknowledgment field) that the sender of this segment is currently willing to receive.

Checksum (16 bits)

The 16-bit [checksum](http://en.wikipedia.org/wiki/Checksum) field is used for error-checking of the header and data.

Urgent pointer (16 bits)

if the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte.

Options (Variable 0–320 bits, divisible by 32)

The length of this field is determined by the data offset field. Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable).

Padding

The TCP header padding is used to ensure that the TCP header ends and data begins on a 32 bit boundary. The padding is composed of zeros.

***2.2 ICMP Packet:***

Internet Control Message Protocol (ICMP) is an error reporting and diagnostic utility and is considered a required part of any IP implementation. Understanding ICMP and knowing what can possibly generate a specific type of ICMP is useful in diagnosing network problems.

ICMPs are used by routers, intermediary devices, or hosts to communicate updates or error information to other routers, intermediary devices, or hosts.   
  
Each ICMP message contains three fields that define its purpose and provide a checksum. They are TYPE, CODE, and CHECKSUM fields. The TYPE field identifies the ICMP message, the CODE field provides further information about the associated TYPE field, and the CHECKSUM provides a method for determining the integrity of the message.

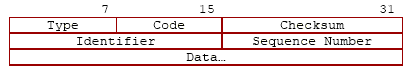


FIGURE:Structure of ICMP packet Header

Type:  8 bits.Specifies the format of the ICMP message.

Code: 8 bits.Further qualifies the ICMP message.

ICMP Header Checksum: 16 bits.Checksum that covers the ICMP message. This is the 16-bit one's complement of the one's complement sum of the ICMP message starting with the [*Type*](http://www.networksorcery.com/enp/protocol/icmp.htm) field. The checksum field should be cleared to zero before generating the checksum.

Data:. Variable length.Contains the data specific to the message type indicated by the [*Type*](http://www.networksorcery.com/enp/protocol/icmp.htm) and [*Code*](http://www.networksorcery.com/enp/protocol/icmp.htm) fields.

***2.3 ARP Packet:***

The Address Resolution Protocol (ARP) is a [telecommunication](http://en.wikipedia.org/wiki/Telecommunication) protocol used for resolution of [network layer](http://en.wikipedia.org/wiki/Network_layer) addresses into [link layer](http://en.wikipedia.org/wiki/Link_layer) addresses, a critical function in multiple-access networks.

ARP is used to convert a network address (e.g. an [IPv4 address](http://en.wikipedia.org/wiki/IPv4_address)) to a physical address such as an [Ethernet address](http://en.wikipedia.org/wiki/Ethernet_address) (also known as a [MAC address](http://en.wikipedia.org/wiki/MAC_address)). ARP has been implemented with many combinations of network and data link layer technologies, such as [IPv4](http://en.wikipedia.org/wiki/IPv4).

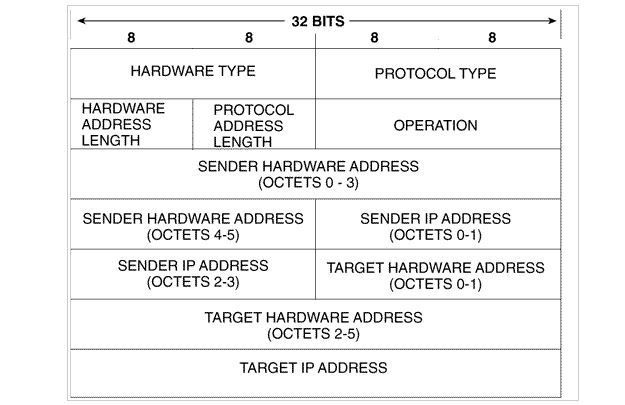


FIGURE:Structure of ARP packet.

The ARP message format is straightforward and consists of the following fields:

Hardware type:The type of MAC address being sought

Protocol type:The Layer-3 protocol in use

Hardware size:The length of the MAC address

Protocol size:The length of the protocol address

OpCode:The type of ARP message

Sender MAC address:The MAC address of the machine sending the request

Sender IP address:The protocol address of the machine sending the ARP request

Target MAC address:The MAC address being sought

Target IP address:The protocol address of the destination

The terms hardware address and protocol address are used as general descriptions, but operationally these will almost always be Ethernet six-byte hardware addresses and IP four-byte addresses. The OpCode will be either a request or a reply.

***2.4 UDP Packet:***

The User Datagram Protocol (UDP) is a transport layer protocol defined for use with the [IP](http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/ip.html) network layer protocol. It provides a [best-effort](http://www.erg.abdn.ac.uk/users/gorry/course/arq-pages/best-effort.html) datagram service to an [End System (IP host)](http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/es-is.html).

The service provided by UDP is an unreliable service that provides no guarantees for delivery and no protection from duplication (e.g. if this arises due to software errors within an [Intermediate System](http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/es-is.html) (IS)). The simplicity of UDP reduces the overhead from using the protocol and the services may be adequate in many cases.

UDP provides a minimal, unreliable, best-effort, message-passing transport to applications and upper-layer protocols

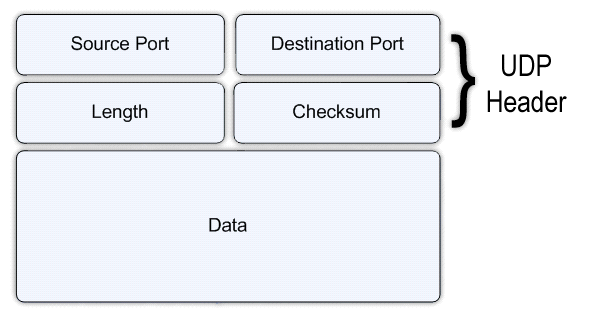


FIGURE: Structure of UDP Header

The UDP header consists of four fields each of 2 bytes in length:

**Source Port** (UDP packets from a client use this as a [service access point (SAP)](http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/sap.html) to indicate the session on the local client that originated the packet. UDP packets from a server carry the server SAP in this field).

**Destination Port**(UDP packets from a client use this as a [service access point (SAP)](http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/sap.html) to indicate the service required from the remote server. UDP packets from a server carry the client SAP in this field).

**UDP length**(The number of bytes comprising the combined UDP header information and payload data).

**UDP Checksum**(A [checksum](http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/ip-cksum.html) to verify that the end to end data has not been corrupted by [routers](http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/router.html) or [bridges](http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/bridge.html) in the network or by the processing in an end system. The algorithm to compute the checksum is the Standard Internet Checksum algorithm. This allows the receiver to verify that it was the intended destination of the packet, because it covers the IP addresses, port numbers and protocol number, and it verifies that the packet is not truncated or padded, because it covers the size field. Therefore, this protects an application against receiving corrupted payload data in place of, or in addition to, the data that was sent. In the cases where this check is not required, the value of 0x0000 is placed in this field, in which case the data is not checked by the receiver.

***2.5 DHCP Packet:***

Dynamic Host Configuration Protocol (DHCP) is a standard protocol that allows a server to dynamically distribute IP addressing and configuration information to clients. Normally the DHCP server provides the client with at least this basic information:

* IP Address
* Subnet Mask
* Default Gateway

Other information can be provided as well, such as Domain Name Service (DNS) server addresses and Windows Internet Name Service (WINS) server addresses. The system administrator configures the DHCP server with the options that are parsed out to the client.

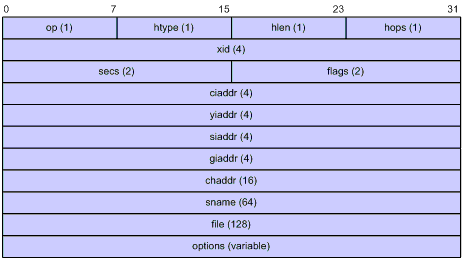


FIGURE: Structure of DHCP packet

Structure of DHCP packet:

op: Message type defined in option field. 1 = REQUEST, 2 = REPLY

 htype,hlen: Hardware address type and length of a DHCP client.

 hops: Number of relay agents a request message traveled.

xid: Transaction ID, a random number chosen by the client to identify an IP address allocation.

secs: Filled in by the client, the number of seconds elapsed since the client began address acquisition or renewal process. Currently this field is reserved and set to 0.

flags: The leftmost bit is defined as the BROADCAST (B) flag. If this flag is set to 0, the DHCP server sent a reply back by unicast; if this flag is set to 1, the DHCP server sent a reply back by broadcast. The remaining bits of the flags field are reserved for future use.

ciaddr: Client IP address.

yiaddr: 'your' (client) IP address, assigned by the server.

siaddr: Server IP address, from which the clients obtained configuration parameters.

giaddr: The first relay agent IP address a request message traveled.

chaddr: Client hardware address

sname: The server host name, from which the client obtained configuration parameters.

file: Bootfile name and routing information, defined by the server to the client.

options: Optional parameters field that is variable in length, which includes the message type, lease, DNS IP address, WINS IP address and so forth.

**3. IMPLEMENTATION IN C**

The packet sniffer in the assignment has been implemented using raw sockets.

A raw socket is made using the socket function as follows:

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

The *socket*() function shall create an unbound socket in a communications domain, and return a file descriptor *sock\_raw* that can be used in later function calls that operate on sockets.

AF\_SOCKET describes the communication domain.

SOCK\_RAW specifies the socket type.

*htons(ETH\_P\_ALL)* When protocol is set to *htons(ETH\_P\_ALL),* then all protocols are

received.

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

The *recvfrom*() function shall receive a message from a connection-mode or connectionless-mode socket. It is normally used with connectionless-mode sockets because it permits the application to retrieve the source address of received data.

The *recvfrom*() function takes the following arguments:

*Socket-raw*: Specifies the socket file descriptor.

*Buffer*:Points to the buffer where the message should be stored.

*65536*:Specifies the length in bytes of the buffer pointed to by the *buffer* argument.

*flags*:Specifies the type of message reception.

*&saddr:*A null pointer, or points to a **s**ockaddr structure in which the sending address is to be stored.

*&saddr\_size:*Specifies the length of the sockaddr structure pointed to by the *address* argument.

The *recvfrom*() function shall return the length of the message written to the buffer pointed to by the *buffer* argument.

The Process\_Packet() function is then called which based on the protocol calls the respective function for each of the packets that is ARP, TCP, ICMP, IGMP and UDP and further prints their details in a textfile called log.txt.

Struct iphdr is used to access the IP Header elements of each of the packets.It contains the following field:

|  |  |
| --- | --- |
| \_\_u8 | [tos](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |
| \_\_u16 | [tot\_len](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |
| \_\_u16 | [id](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |
| \_\_u16 | [frag\_off](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |
| \_\_u8 | [ttl](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |
| \_\_u8 | [protocol](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |
| \_\_u16 | [check](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |
| \_\_u32 | [saddr](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |
| \_\_u32 | [daddr](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structiphdr.html) |

Each of these is accessed and printed to logfile.txt using fprint().

Ethernet header information is printed using print\_ethernet\_header() function using struct ethhdr.

The fields are:

unsigned char h\_dest[ETH\_ALEN];

unsigned char h\_source[ETH\_ALEN];

unsigned short h\_proto;

For the ARP packet a slightly different approach is used.

descr = pcap\_open\_live(argv, MAXBYTES2CAPTURE, 0, 512, errbuf);

**pcap\_open\_live()** is used to obtain a packet capture handle to look at packets on the network. *Argv* is a string that specifies the network device to open.

MAXBYTES2CAPTURE specifies the snapshot length to be set on the handle.

*0* indicates the interface is not to be put into promiscuous mode.

*512* specifies the read timeout in milliseconds.

The data captured is stored in errbuf.

**It** returns a *pcap\_t descr* on success and **NULL** on failure.

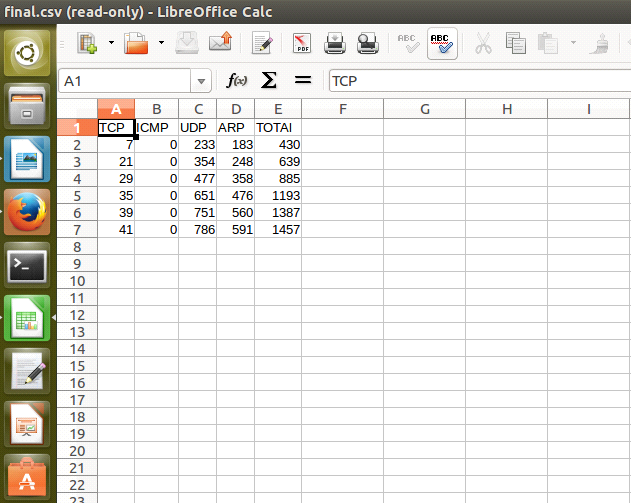
**packet = pcap\_next(descr,&pkthdr));**

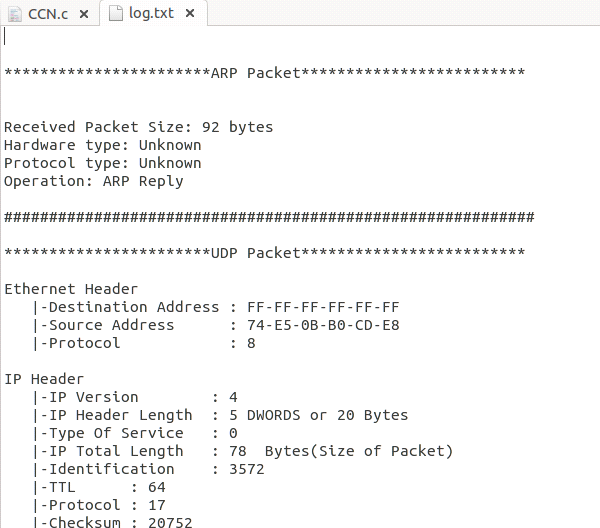
**pcap\_next** reads the next packet and returns a success/failure indication. If the packet was read without problems, the pointer pointed to by the *pkt\_header* argument is set to point to the *pkthdr* struct for the packet.

Struct arphdr which is a user defined structure is then accessed to gain information about the ARP packet.

The numbers of all these packets is updated into the excel sheet every 1 minute using a clock() function to update every time the difference between start time and end time becomes more than 60.

Finally depending on the entries in the excel sheet a line graph is plotted to understand at what rate each of the packet is received.





**4. Other Assignment Details**

***4.1 Explain the process of ping and associated packet details by taking an example.***

The Internet [Ping](http://www.ping127001.com/pingpage.htm) command bounces a small [*packet*](http://www.livinginternet.com/i/iw_packet.htm) off a domain or IP address to test network communications, and then tells how long the packet took to make the round trip. The Ping command is one of the most commonly used utilities on the [*Internet*](http://www.livinginternet.com/) by both people and automated programs.

**4.1.1. How Ping works**.

The Internet Ping program works much like a sonar echo-location, sending a small packet of information containing an [*ICMP*](http://www.livinginternet.com/i/ia_rfc_net.htm) ECHO\_REQUEST to a specified computer, which then sends an ECHO\_REPLY packet in return. The IP address 127.0.0.1 is set by convention to always indicate your own computer. Therefore, a ping to that address will always ping yourself and the delay should be very short.

**4.1.2 ICMP packet structure.**

|  |
| --- |
|  |

FIGURE: ICMP echo packet structure

**4.1.2 Case Study**  
  
Let's examine a case study where two machines called Paris and Berlin ping a machine called "[www.google.ch](http://www.google.ch/)".  
The  sniffer is located on the way between Berlin and "[www.google.ch](http://www.google.ch/)". It will be used to capture the packets content.  
Paris is a Linux Ubuntu machine and Berlin a Microsoft XP Machine.

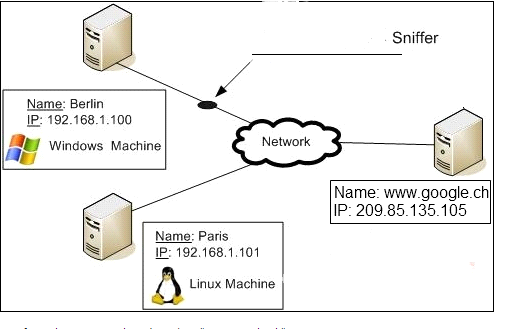


FIGURE: Configuration of machines

Paris: ping [www.google.ch](http://www.google.ch/)

Ping from the Windows machine (Berlin) to "[www.google.ch](http://www.google.ch/)":  
  
Berlin: ping [www.google.ch](http://www.google.ch/) 

|  |
| --- |
| Pinging [www.google.ch](http://www.google.ch/) [209.85.135.105] with 32 bytes of data:   Reply from 209.85.135.105: bytes=32 time=18 ms TTL=250  Reply from 209.85.135.105: bytes=32 time=21 ms TTL=250  Reply from 209.85.135.105: bytes=32 time=20 ms TTL=250  Reply from 209.85.135.105: bytes=32 time=33 ms TTL=250   Ping statistics for 209.85.135.105:        Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  Approximate round trip times in milli-seconds:        Minimum = 18ms, Maximum = 33ms, Average = 23ms |
|  |
|  |

The sniffer gives us the following info:

The packets have been recognized as ICMP packets.

- Four ICMP packets have been sent (echo\_request) and four ICMP packets have been received (echo\_reply).

- A packet value of 74 bytes which is composed of the headers (42 bytes) and the ICMP data part (32 bytes by default on Windows).

Besides the original ping program, ping might simply mean the action of checking if a remote node is responding, this might be done on several layers in a protocol stack - e.g. [ARP ping](http://en.wikipedia.org/wiki/Arping) for testing hosts on a local network. The term ping might be used on higher protocol layers and APIs as well, e.g. the act of checking if a database is up, done at the database layer.

***4.2 Explain the process of Web access and TCP by considering an appropriate example.***

**4.2.1 TCP/IP Protocols For the Web:**

Web browsers and servers use TCP/IP protocols to connect to the Internet. Common TCP/IP protocols are:

1.HTTP - Hyper Text Transfer Protocol HTTP takes care of the communication between a web server and a web browser. HTTP is used for sending requests from a web client (a browser) to a web server, returning web content (web pages) from the server back to the client.

2.HTTPS - Secure HTTP HTTPS takes care of secure communication between a web server and a web browser. HTTPS typically handles credit card transactions and other sensitive data.

3.FTP - File Transfer Protocol FTP takes care of transmission of files between computers.

IP is a "connection-less" communication protocol.

IP does not occupy the communication line between two computers. This reduces the need for network lines. Each line can be used for communication between many different computers at the same time. With IP, messages (or other data) are broken up into small independent "packets" and sent between computers via the Internet. IP is responsible for "routing" each packet to the correct destination.

When an IP packet is sent from a computer, it arrives at an IP router.

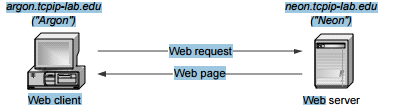
The IP router is responsible for "routing" the packet to the correct destination, directly or via another router.

The path the packet will follow might be different from other packets of the same communication. The router is responsible for the right addressing, depending on traffic volume, errors in the network, or other parameters.

Analogy: Communicating via IP is like sending a long letter as a large number of small postcards, each finding its own (often different) way to the receiver.

**4.2.2. Case study**

Neon is the web server that provides web access and Argon is the web client that accesses the web.



HTTP client and server:

Web browser runs an HTTP client program

• Web server runs an HTTP server program

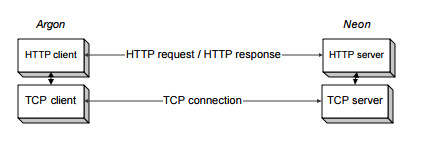
• HTTP client sends an HTTP request to HTTP server

• HTTP server responds with HTTP response

From HTTP to TCP :

• To send request, HTTP client program establishes an TCP connectionto the HTTP server Neon.

• The HTTP server at Neon has a TCP server running



Resolving hostnames and port numbers

• Since TCP does not work with hostnames and also would not know how to find the HTTP server program at Neon, two things must happen:

1. The name “neon.tcpip-lab.edu” must be translated into a 32-bit IP address.

2. The HTTP server at Neon must be identified by a 16-bit port number.

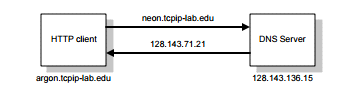
8 Translating a hostname into an IP address

• The translation of the hostname neon.tcpip-lab.edu into an IP address is done via a database lookup.

• The distributed database used is called the Domain Name System (DNS)

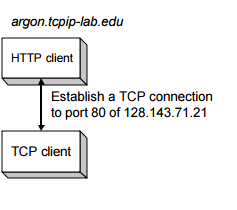
• All machines on the Internet have an IP address: argon.tcpip-lab.edu 128.143.137.144

neon.tcpip-lab.edu 128.143.71.21



Requesting a TCP Connection :

• The HTTP client at argon.tcpip-lab.edu requests the TCP client to establish a connection to port 80 of the machine with address 127.141.71.21

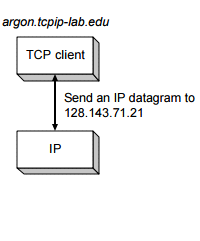


Invoking the IP protocol:

• The TCP client at Argon sends a request to establish a connection to port 80 at Neon

• This is done by asking its local IP module to send an IP datagram to 128.143.71.21

• (The data portion of the IP datagram contains the request to open a connection)



The datagram is then sent to IP router since both belong to different networks.

