Attendance (3)	In-time Check (2)	Oral (5)	Total (10)	Dated Sign

Assignment No:	
Title :	WrWrite a program to Implement a packet sniffing tool in C++/Java/Python.
Roll No :	
Class :	T.E(C.O)
Date :	
Subject:	Programming Lab II
Signature:	

Title :

Title: packet formats captured through Wireshark for wired network

OBJECTIVES:

1.To understand packet formats captured through Wireshark for wired network.

Problem Statement: Write a program to analyze following packet formats captured through Wireshark for wired network. 1. Ethernet 2. IP 3.TCP 4. UDP.

Outcomes: Captured packet format through Wireshark

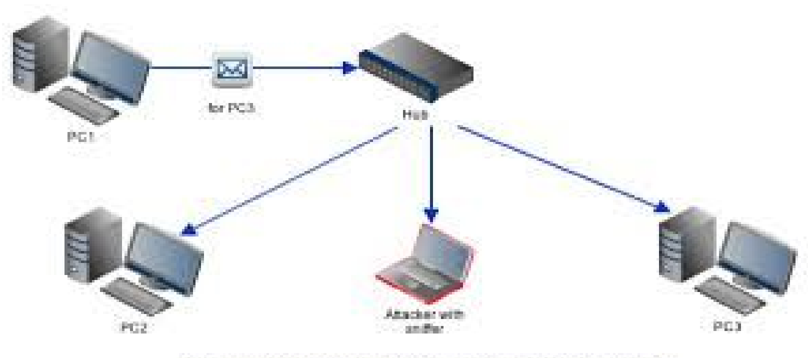
Software: C/C++

Theory :

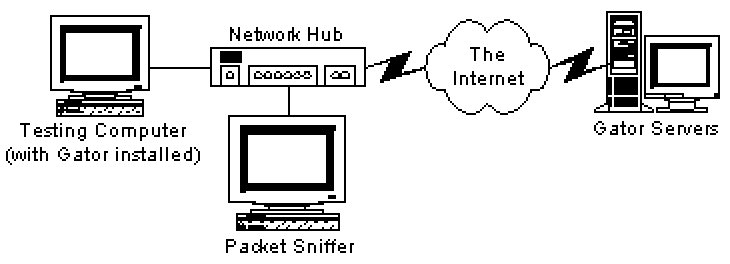
Packet Sniffer:

A packet sniffer (also known as a network analyzer, protocol analyzer or for particular types of networks, 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 flow across the network, the sniffer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content.

A packet sniffer is a wire-tap device that plugs into computer networks and eavesdrops on the network traffic.



Packet addressed to PC3 is forwarded by the hub to other hosts in the network



Computer-network administrators have used packet sniffers for years to monitor their networks and perform diagnostic tests or troubleshoot problems. Essentially, a packet sniffer is a program that can see all of the information passing over the network it is connected to. As data streams back and forth on the network, the program looks at, or "sniffs," each packet. A packet is a part of a message that has been broken up.

Normally, a computer only looks at packets addressed to it and ignores the rest of the traffic on the network. But when a packet sniffer is set up on a computer, the sniffer's network interface is set to promiscuous mode. This means that it is looking at everything that comes through. The amount of traffic largely depends on the location of the computer in the network. A client system out on an isolated branch of the network sees only a small segment of the network traffic, while the main domain server sees almost all of it.

- A packet sniffer can usually be set up in one of two ways:

1. Unfiltered - captures all of the packets.
2. Filtered - captures only those packets containing specific data elements

How does Sniffing Works ?

- Ethernet was built around a "shared" principle: all machines on a local network share the same wire.
- This implies that all machines are able to "see" all the traffic on the same wire.
- Thus, Ethernet hardware is built with a "filter" that ignores all traffic that doesn't belong to it. It does this by ignoring all frames whose MAC address doesn't match.
- A sniffer program turns off this filter, putting the Ethernet hardware into "promiscuous mode". Thus, Mark can see all the traffic among all machines, as long as they are on the same Ethernet wire.

Why we use packet snifing ?

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

- Analyse network problems.
- Detect network intrusionattempts.
- Gain information for effecting a network intrusion.
- Gather and report network statistics.
- Filter suspect content from network traffic.
- Debug client/server communications.

Capabilities:

On wired broadcast LANs, depending on the network structure (hub or switch), one can capture traffic on all or just parts of the network from a single machine within the network; however, there are some methods to avoid traffic narrowing by switches to gain access to traffic from other systems on the network (e.g., ARP spoofing). For network monitoring purposes, it may also be desirable to monitor all data packets in a LAN by using a network switch with a so-called monitoring port, whose purpose is to mirror all packets passing through all ports of the switch when systems (computers) are connected to a switch port. To use a network tap is an even more reliable solution than to use a monitoring port, since taps are less likely to drop packets during high traffic load.

On wireless LANs, one can capture traffic on a particular channel, or on several channels when using multiple adapters.

On wired broadcast and wireless LANs, to capture traffic other than unicast traffic sent to the machine running the sniffer software, multicast traffic sent to a multicast group to which that machine is listening, and broadcast traffic, the network adapter being used to capture the traffic must be put into promiscuous mode; some sniffers support this, others do not. On wireless LANs, even if the adapter is in promiscuous mode, packets not for the service set for which the adapter is configured will usually be ignored. To see those packets, the adapter must be in monitor mode.

When traffic is captured, either the entire contents of packets can be recorded, or the headers can be recorded without recording the total content of the packet. This can reduce storage requirements, and avoid legal problems, but yet have enough data to reveal the essential information required for problem diagnosis.

The captured information is decoded from raw digital form into a human-readable format that permits users of the protocol analyzer to easily review the exchanged information. Protocol analyzers vary in their abilities to display data in multiple views, automatically detect errors, determine the root causes of errors, generate timing diagrams, reconstruct TCP and UDP data streams, etc.

Some protocol analyzers can also generate traffic and thus act as the reference device; these can act as protocol testers. Such testers generate protocol-correct traffic for functional testing, and may also have the ability to deliberately introduce errors to test for the DUT's ability to deal with error conditions.

Protocol analyzers can also be hardware-based, either in probe format or, as is increasingly more common, combined with a disk array. These devices record packets (or a slice of the packet) to a disk array. This allows historical forensic analysis of packets without the users having to recreate any fault.

Uses:

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

- Analyze network problems
- Detect network intrusion attempts
- Detect network misuse by internal and external users
- Documenting regulatory compliance through logging all perimeter and endpoint traffic
- Gain information for effecting a network intrusion
- Isolate exploited systems
- Monitor WAN bandwidth utilization
- Monitor network usage (including internal and external users and systems)
- Monitor data-in-motion
- Monitor WAN and endpoint security status
- 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 or users cookies (depending on any content encryption methods that may be in use)
- Reverse engineer proprietary protocols used over the network
- Debug client/server communications
- Debug network protocol implementations
- Verify adds, moves and changes
- Verify internal control system effectiveness (firewalls, access control, Web filter, spam filter, proxy)

Packet capture can be used to fulfill a warrant from a law enforcement agency (LEA) to produce all network traffic generated by an individual. Internet service providers and VoIP providers in the United States must comply with CALEA (Communications Assistance for Law Enforcement Act) regulations. Using packet capture and storage, telecommunications carriers can provide the legally required secure and separate access to targeted network traffic and are able to use the same device for internal security purposes. Collection of data from a carrier system without a warrant is illegal due to laws about interception.

Ethernet:

Ethernet is a way of connecting computers together in a local area network. It has been the most widely used method of linking computers together in LANs since the 1990s. The basic idea of its design is that multiple computers have access to it and can send data at any time. This is comparatively easy to engineer.

If two computers send data at the same time, a collision will occur. When this happens, the data sent is not usable. In general, both computers will stop sending, and wait a random amount of time, before they try again. A special protocol was developed to deal with such problems. It is called Carrier sense multiple access with collision detection or CSMA/CD.

Internet Protocol

The **Internet Protocol** (**IP**) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

IP has the task of delivering packets from the source host to the destination host solely based on the IP addresses in the packet headers. For this purpose, IP defines packet structures that encapsulate the data to be delivered. It also defines addressing methods that are used to label the datagram with source and destination information.

Historically, IP was the connectionless datagram service in the original *Transmission Control Program* introduced by Vint Cerf and Bob Kahn in 1974; the other being the connection-oriented Transmission Control Protocol (TCP). The Internet protocol suite is therefore often referred to as TCP/IP.

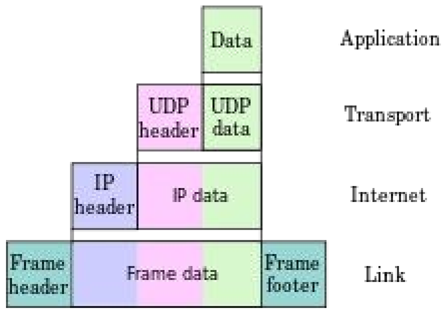
The first major version of IP, Internet Protocol Version 4 (IPv4), is the dominant protocol of the Internet. Its successor is Internet Protocol Version 6 (Ipv6).

Function:

The Internet Protocol is responsible for addressing hosts, encapsulating data into datagrams (including fragmentation and reassembly) and routing datagrams from a source host to a destination host across one or more IP networks.For these purposes, the Internet Protocol defines the format of packets and provides an addressing system.

Each datagram has two components: a header and a payload. The IP header includes source IP address, destination IP address, and other metadata needed to route and deliver the datagram. The payload is the data that is transported. This method of nesting the data payload in a packet with a header is called encapsulation.

IP addressing entails the assignment of IP addresses and associated parameters to host interfaces. The address space is divided into subnetworks, involving the designation of network prefixes. IP routing is performed by all hosts, as well as routers, whose main function is to transport packets across network boundaries. Routers communicate with one another via specially designed routing protocols, either interior gateway protocols or exterior gateway protocols, as needed for the topology of the network.



TCP Header Format

TCP segments are sent as internet datagrams. The Internet Protocol header carries several information fields, including the source and destination host addresses [2]. A TCP header follows the internet header, supplying information specific to the TCP protocol. This division allows for the existence of host level protocols other than TCP.

TCP Header Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                      TCP Header Format

          Note that one tick mark represents one bit position.

                          Figure 3.
```

Source Port: 16 bits

The source port number.

Destination Port: 16 bits

The destination port number.

Sequence Number: 32 bits

The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

Acknowledgment Number: 32 bits

If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.

Data Offset: 4 bits

The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an ntegral number of 32 bits long.

Reserved: 6 bits

Reserved for future use. Must be zero.

Control Bits: 6 bits (from left to right):

URG: Urgent Pointer field significant
ACK: Acknowledgment field significant
PSH: Push Function
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: No more data from sender

Window: 16 bits

The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

Checksum: 16 bits

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

The checksum also covers a 96 bit pseudo header conceptually prefixed to the TCP header. This pseudo header contains the Source Address, the Destination Address, the Protocol, and TCP length. This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/Network interface in the arguments or results of calls by the TCP on the IP.

```
+--------+--------+--------+--------+
|           Source Address          |
+--------+--------+--------+--------+
|         Destination Address       |
+--------+--------+--------+--------+
|  zero  |  PTCL  |    TCP Length   |
+--------+--------+--------+--------+
```

The TCP Length is the TCP header length plus the data length in octets (this is not a explicitly transmitted quantity, but is computed), and it does not count the 12 octets of the pseudo header.

Urgent Pointer: 16 bits

This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the URG control bit set.

Options: variable

Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum. An option may begin on any octet boundary. There are two cases for the format of an option:

Case 1: A single octet of option-kind.

Case 2: An octet of option-kind, an octet of option-length, and the actual option-data octets.

The option-length counts the two octets of option-kind and option-length as well as the option-data octets.

Note that the list of options may be shorter than the data offset field might imply. The content of the header beyond the End-of-Option option must be header padding (i.e., zero).

A TCP must implement all options. Currently defined options include (kind indicated in octal):

Kind	Length	Meaning
0	-	End of option list.
1	-	No-Operation.
2	4	Maximum Segment Size.

Specific Option Definitions

End of Option List

```
+--------+
|00000000|
+--------+
 Kind=0
```

This option code indicates the end of the option list. This might not coincide with the end of the TCP header according to the Data Offset field. This is used at the end of all options, not the end of each option, and need only be used if the end of the options would not otherwise coincide with the end of the TCP header.

No-Operation

```
+--------+
|00000001|
+--------+
 Kind=1
```

This option code may be used between options, for example, to align the beginning of a subsequent option on a word boundary. There is no guarantee that senders will use this option, so receivers must be prepared to process options even if they do not begin on a word boundary.

Maximum Segment Size

```
+--------+--------+---------+--------+
|00000010|00000100|   max seg size   |
+--------+--------+---------+--------+
 Kind=2   Length=4
```

Maximum Segment Size Option Data: 16 bits

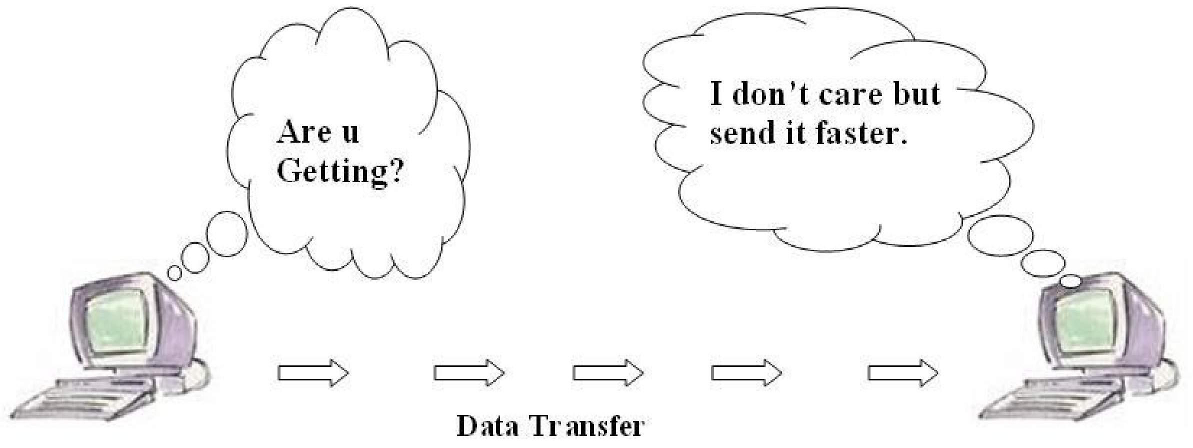
If this option is present, then it communicates the maximum receive segment size at the TCP which sends this segment. This field must only be sent in the initial connection request (i.e., in segments with the SYN control bit set). If this option is not used, any segment size is allowed.

Padding: variable

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.

What is UDP?

UDP

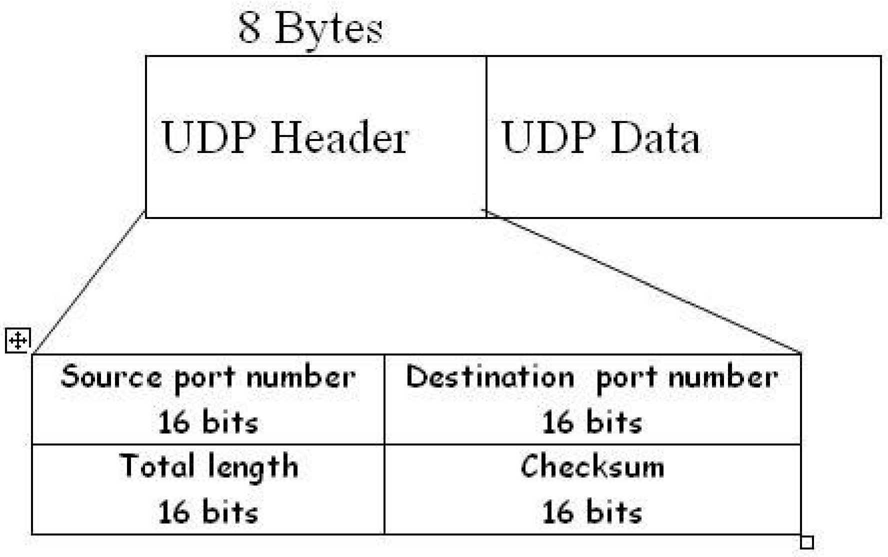


UDP is a connectionless and unreliable transport protocol.The two ports serve to identify the end points within the source and destination machines. User Datagram Protocol is used, in place of TCP, when a reliable delivery is not required.However, UDP is never used to send important data such as web-pages, database information, etc. Streaming media such as video,audio and others use UDP because it offers speed.

Why UDP is faster than TCP?

The reason UDP is faster than TCP is because there is no form of flow control. No error checking,error correction, or acknowledgment is done by UDP.UDP is only concerned with speed. So when, the data sent over the Internet is affected by collisions, and errors will be present.

UDP packet's called as user datagrams with 8 bytes header. A format of user datagrams is shown in figur 3. In the user datagrams first 8 bytes contains header information and the remaining bytes contains data.



***Source port number*:** This is a port number used by source host,who is transferring data. It is 16 bit longs. So port numbers range between 0 to 65,535.

***Destination port number*:** This is a port number used by Destination host, who is getting data. It is also 16 bits long and also same number of port range like source host.

length: Length field is a 16 bits field. It contains the total length of the user datagram, header and data.

***Checksum*:** The UDP checksum is optional. It is used to detect error fro the data. If the field is zero then checksum is not calculated. And true calculated then field contains 1.

Characteristics of UDP

The characteristics of UDP are given below.

• End-to-end. UDP can identify a specific process running on a computer.

• Unreliable, connectionless delivery (e.g. USPS)::

UDP uses a connectionless communication setup. In this UDP does not need to establish a connection before sending data. Communication consists only of the data segments themselves

• Same best effort semantics as IP

• No ack, no sequence, no flow control

• Subject to loss, duplication, delay, out-of-order, or loss of connection

• Fast, low overhead

1.Suit for reliable, local network

2.RTP(Real-Time Transport Protocol)

Packet sniffing tools:

1) WIRESHARK:

- Wireshark (known as Ethereal until a trademark dispute in Summer 2006) is a fantastic open source multi-platform network protocol analyzer.
- It allows you to examine data from a live network or from a capture file on disk. You can interactively browse the capture data, delving down into just the level of packet detail you need.
- Wireshark has several powerful features, including a rich display filter language and the ability to view the reconstructed stream of a TCP session.
- It also supports hundreds of protocols and media types. A tcpdump-like console version named tshark is included.
- One word of caution is that Wireshark has suffered from dozens of remotely exploitable security holes, so stay up-to-date and be wary of running it on untrusted or hostile networks (such as security conferences)

Conclusion :

hence we have successfully implemented the program for packet sniffing in python and studied the concepts of tcp header.