TCPDump is a free software designed by Van Jacobson, Sally Floyd ,Vern Paxson and Steven McCanne in 1988. It uses the libpcap library in C to read packets from a network interface card

What is libpcap?

libpcap is the abbreviation of packet capture library which is an application programming interface (API ) that is used to capture network traffic. The libpcap is written in c so therefore in order for android which uses JAVA , we must use a wrapper to translate a library existing interface to the compatible interface. The wrapper enable android which uses JAVA to use libpcap c code.

Example of some wrapper program readily available would be jpcap,jnetpcap,jpcap and pcap4j.

These are all wrapper libraries made by other developers to enable c code to be use in java.

**Creating our own TCPDUMP**

How we planned to achieve this is to use libpcap and write the general code that does also the same as TCPDUMP and we cross compile it into a ARM compatible structure and use the ARM binary in our applications. Why do we use ARM? As most mobile uses ARM processor instead of intel processor . The difference between linux ARM and linux x86 is that internally their binary code is different thus software done on x86 must be pre compile to match the target architecture.

**Difficulties in creating our own TCPDUMP**

Although these libraries are readily available for us to use in android we face one problem.

The problem would be if we want to sniff packets we would need root permission. Giving root permission would not be a problem if we were to work in environment such as LINUX or WINDOWS.

In android any access to protected resources or services is guarded by the application permission framework, all access in native code needs to be analysed, and the required permissions should be identified. Whatever permissions the native code may need should be published for developers, so that they can include these permissions in their applications Manifest file. Native code should not rely on code that need access as this wont be available on standard android[4]

The above sentence is a research that we did when we came across that using a wrapper library that need route access violate that rule .Therefore we went to look for another alternative which is to use a ARM binary library for ARM architecture . If we were to use a binary in android we can give root to process and therefore bypassing the no rooting to native code problem.

The next problem would be even thought we had create our own arm binary, the binary should have no problem executing the code in the NEXUS 5 we are using for our Android sniffer application but when we tried to execute the code, the code show denied no permission even though the phone is rooted and in root permission. No available solution has been found for this problem for now.

**Setting up environment in linux**

wget http://www.tcpdump.org/release/libpcap-1.8.1.tar.gz

tar -xf libpcap-1.8.1.tar.gz

sudo apt-get update

sudo apt-get install flex

sudo apt-get install bison

cd libpcap-1.8.1

./configure –prefix=/usr

Make

Sudo make install

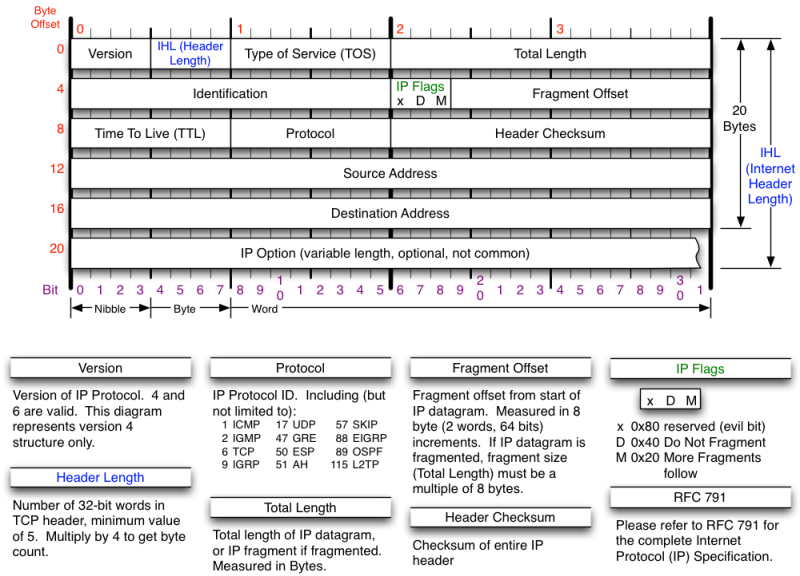
Following the above step will compiled the libpcap giving libpcap.a and libpcap.so file.

The .a file are static libraries while .so are dynamic libraries.The difference between the two is, in a static libraries , if you used code stored inside them, it is taken from them and embedded into your own binary while in dynamic libraries, the code is not taken and embedded into your own library . It is referenced so the binary will depend on them and the code from the so file is loaded at runtime.

**Type of IP Packets**

1. TCP/IP
2. UDP
3. ICMP (INTERNET CONTROL MESSAGE PROTOCOL)
4. IGMP (INTERNET GROUP MANAGEMENT PROTOCOL)
5. IGRP ( INTERIOR GATEWAY ROUTING PROTOCOL)
6. ESP (ENCAPSULATING SECURITY PAYLOAD)
7. AH (AUTHENTICATION HEADER)

**IP Packet Header**

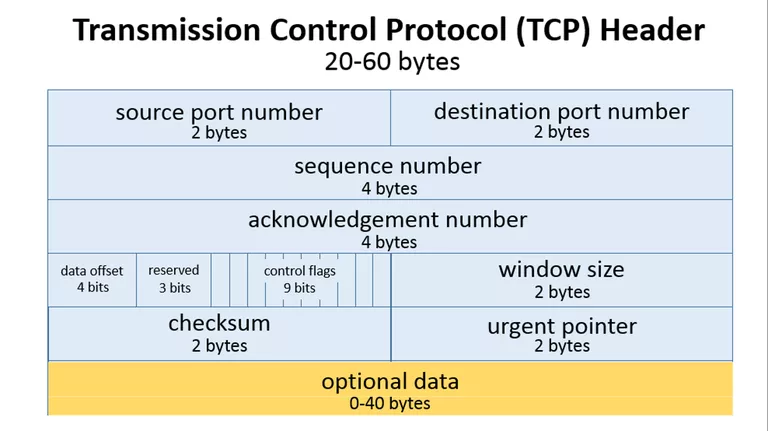
**[5]**

IP header usually about 20 bytes . The source address and destination address mean from where the ip packet comes from and where the packet is trying to reach.

The identification field is used to uniquely identify the group of fragments of a single ipdatagram.

The Types of service has now been redefined by RFC 2474 for Differentiated service and Explicit congestion notification(ECN). Differentiated service (DSCP ) is used when technologies require real time data streaming is needed . ECN is defined in RFC3186 and allows end to end notification of network congestion without dropping packets . It is an optional feature which is only effective if there is underlying network.

**Transport control protocol(TCP)**

[2]

TCP header usually requires a minimum of 20 bytes anything.

As shown above the source port takes up around 2 bytes, destination port takes up around 2 bytes,

Sequence number takes up 4 bytes. Sequence number is used to mark the ordering of a group of message . Acknowledgment number takes up 4 bytes are used by the sender and receiver to communicate the sequence number of messages.

Data offset field takes up 4 bits and store the total size of a TCP header in multiples of four bytes.A header using the optional TCP field has a data offset of 5 , while a header using the maximum-sized optional field has a data offset of 15 .

Reserved uses 3 bits and the value is always zero .

Control flags uses 9 bits and TCP use 6 standard and 3 extend control flags to manage data flow in specific situations. Each flag is 1 bit in size . The flags are SYN , ACK , FIN ,URG , PSH , RST ,ECE ,CWR ,NS flags.

SYN aka synchronous flag is used as a first step in establishing a 3 way handshake between 2 host

ACK aka acknowledgment is used to acknowledge the successful receipt of a packet. It is use to tell the sender it has receive the initial packet.

FIN aka as Finished . Tell the receiver that sender has no more data to send. Therefore this is the last packet sent to receiver.

URG aka urgent . This flag is to the receiver to process the urgent packet first before processing other packet.

PSH stands for push. It is similar to URG flag . it is used to tell the receiver to process the packets with PSH flag instead of buffering them.

RST stands for reset flag. This flag is sent when a packet is sent to a particular host that was no expecting it.

ECE is responsible for indicating if a TCP peer is ECN capable

CWR flag stands for Congestion window reduced is used by sending host to indicate it received a packet with ECE flag set.

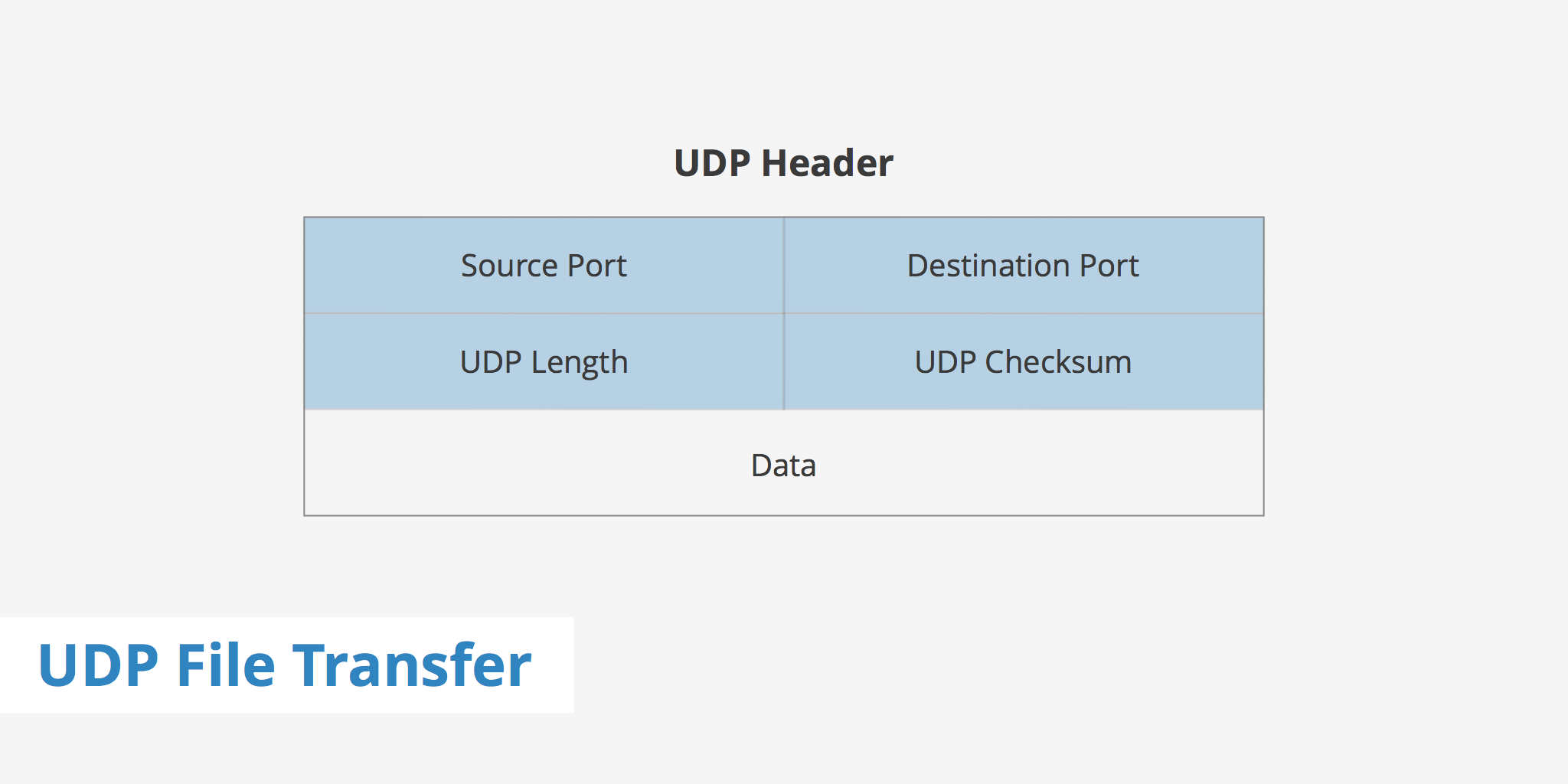
NS stands for Nonce Sum , it is an experimental flag used to help protect accidental malicious concealment of packets from sender.

Window size is a value to regulate how much data is send before an acknowledgment from reciever is required. This value cannot be too small or too big as being too small will slow performance while too big will cause network link to be saturated making it unusable for other application

Checksum is used to help receiver check if the data receive has been corrupted or tempered with.

Urgent pointer field is often set to zero or ignored.[3]

**User Datagram Protocol(UDP)**

[1]

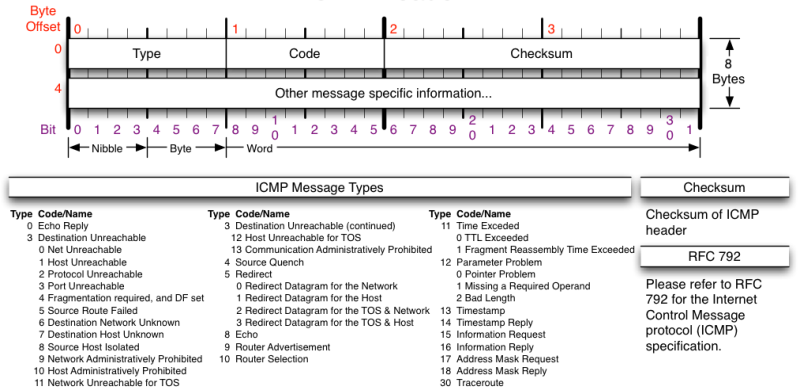
UDP header is much smaller than TCP . It is 8 bytes in total.

As shown above it only has four field . Source port, Destination Port, UDP length and UDP checksum

UDP length is used to identify the length of the header itself

UDP checksum is the same as TCP checksum. It is used to check if a UDP packet data is corrupted or been tampered with.

**ICMP packet header**

**[5]**

Currently our program is unable to sniff ICMP packets due to not enough time . However by the end of the project period , ICMP packet would be sniff by pcbin .

ICMP is not a transport protocol that sends data between systems . It is used to troubleshoot internet connections by network admin.

**Implementation**

Main functions of a libpap library that enables packet sniffer.

1. pcap\_lookupdev

* find the default device on which to capture the packet.
* Pcap\_lookupdev returns a pointer to a string giving the name of a network device suitable for use with other main function to create a packet sniffer.
* However in our program we would like to use our own interface that we specified therefore this function is not needed.

1. Pcap\_create(device, errbuff)

* Used to create a packet capture handle to look at the packet on the network

1. Pcap\_set\_promisc(handler,int )

* to set promiscuous mode on a non activated handle . If int is any non zero value , promiscuous mode will be set otherwise it will not be set.

4) pcap\_activate(handler)

* using pcap\_create just creates the handler in order to use the handler , we would need to activate it

5)pcap\_datalink

* get the link layer header type

6)pcap\_lookupnet ()

* get the netmask and network number of IPv4 for a device

7)pcap\_compile()

* used to compile the string str in to filter program.Program is a pointer to a bpf\_program struct .optimize controls whether optimization on the resulting code is performed.
* Return 0 on success and -1 on failure

8)pcap\_setfilter()

* Used to specify a filter program which is apointer to a bpf\_program struct ,usually the result of a call to pcap\_compile()
* Return 0 on success -1 on failure.

9)pcap\_loop()

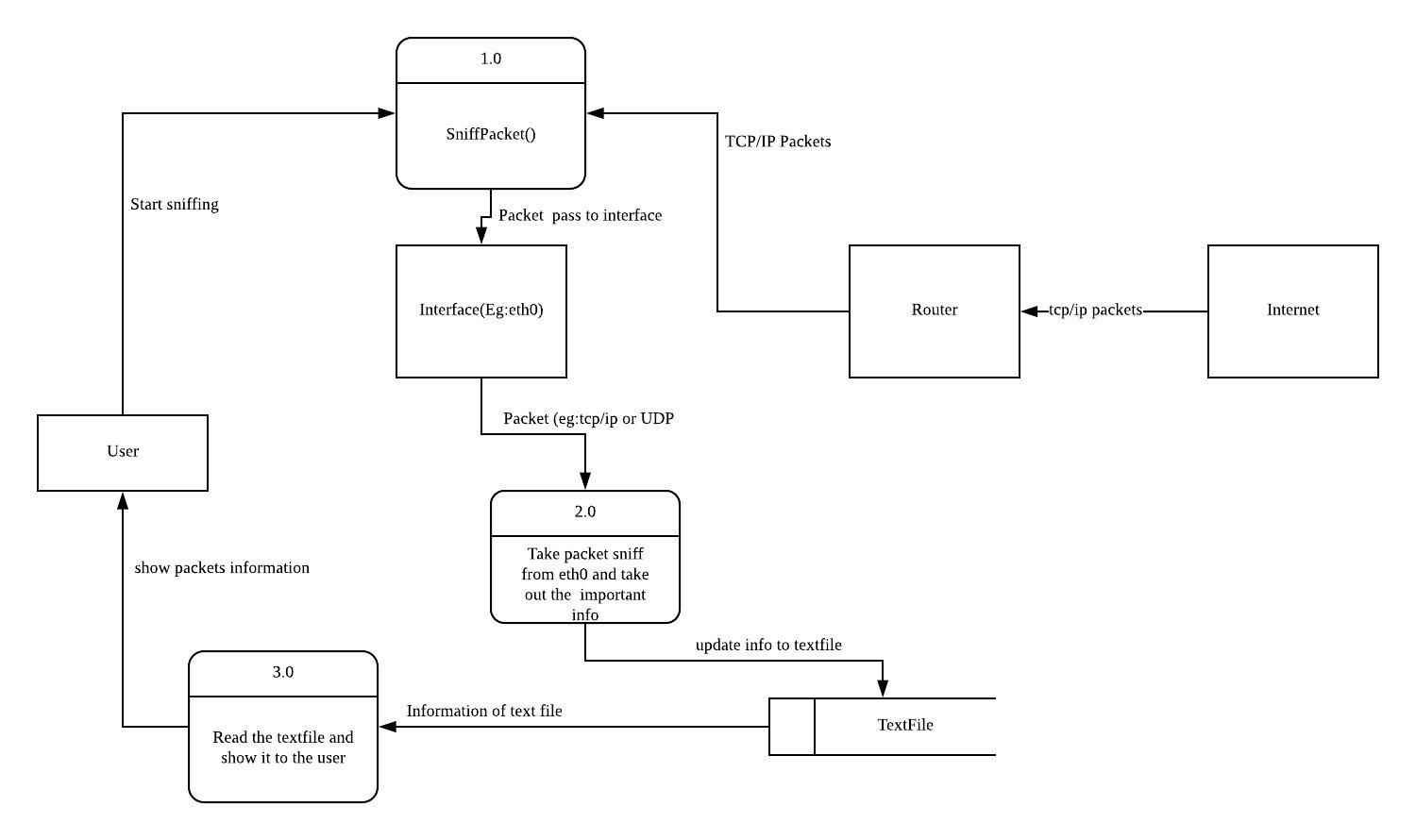
- loop mean that if will keep sniffing packet until it is specified to stop.

-it loops a specified function that is placed on the third argument

10)pcap\_close()

- close the handler once it has completed its job.

**DATA FLOW DIAGRAM for pcbin**



**Future updates**

* Enable sniffing of all other packet for eg (ARP packets, DNS and ICMP packets)
* Implement it together with libnet library to send packet to other device .

(Libnet is required to break wep)

1. <https://www.keycdn.com/support/udp-file-transfer/>
2. <https://www.lifewire.com/tcp-headers-and-udp-headers-explained-817970>
3. <https://www.keycdn.com/support/tcp-flags/>
4. <https://opensourceforu.com/2013/09/what-a-native-developer-should-know-about-android-security/>
5. <https://nmap.org/book/tcpip-ref.html>