Ex.no: 5
09/08/24

Practical 5

Aim:

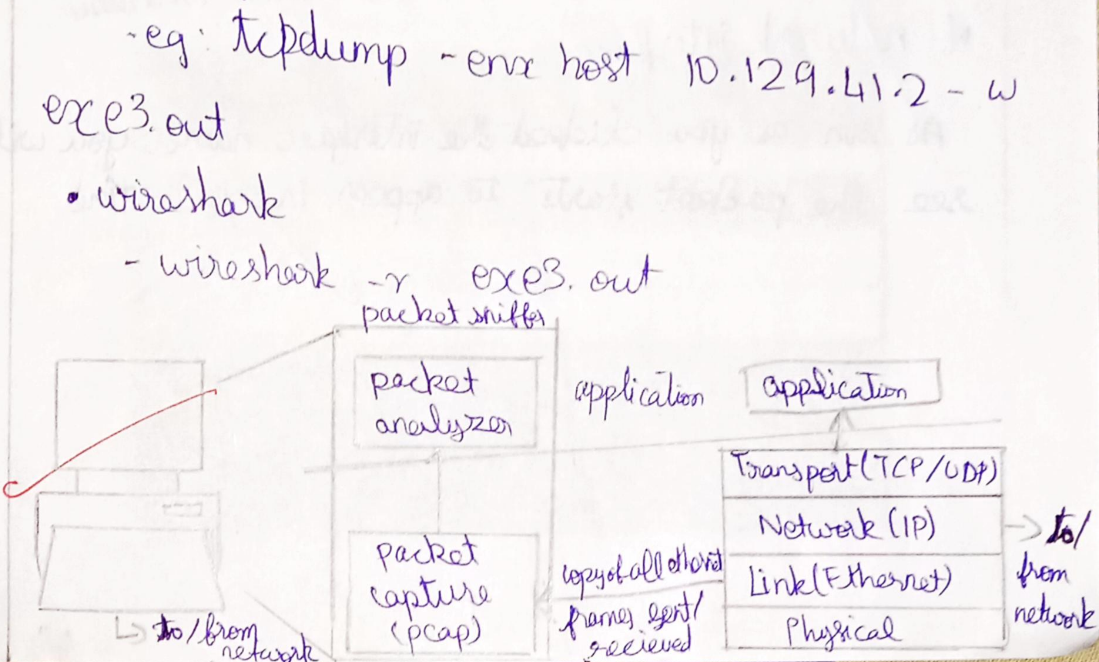
Experiment on Packet Capture tool : wireshark

Packet Sniffer

- Sniff message being sent/recieved from/by computer.
- Stores & Display content of various protocol
- Passive Program
  * never send packet itself
  * no packet addressed to it
  * recieved a copy of all packet

Packet Sniffer Structure Diagonostic Tools

- Tcpdump
  - eg: tcpdump -enx host 10.129.41.2 -w exe3.out
- wireshark
  - wireshark -r exe3.out



26/08/24

Aim:

Write a program to implement error detection and correction using HAMMING Code concept. Make a test run to input data stream and verify error correction feature

Error Correction at Data Link Layer:

Hamming Code is a set of error - correction codes that can be used ~~for~~ to detect and correct errors that occur when data is transmitted from sender to receiver.

Create sender program with below feature

1) Input to sender file should be text of any ~~text~~ length Program should convert text to binary.

2) Apply hamming code concept on binary data

3) Save output in a file called channel.

Create receiver program with below feature

1) Should read input from channel file

2) Apply hamming code on binary data to check for error

3) If there is an error, display the position of the error.

4) Else remove the redundant bits and convert the binary data to ascii and display the output

Student Observation

sender code.py

```
def send_data(bit, length):
    r= calredundant ( length)
    pos = posRedundantBit( bit, r, length)
    par = findParity (pos, r)
    return par
def calredundant(l):
    for i in range(l):
        if (2**i >= l+i+1):
            return i

def posRedundantBit(bit, r, l):
    j = int(0)
    res = [ ]
    reverse_bit = bit[::-1]
    rev_len = 0
    for i in range (1, l+r+1):
        if (pow(2,j) == i):
            res.append('0')
            j+=1
        else
            res.append(reverse_bit[rev_len])
            rev_len +=1
    return res
def findParity(pos, r):
    red = 0
    while red <= r:
        count = 0
    for i in range(len(pos)):
        if i+1 == pow(2, red):
            step = pow(2, red)
            for temp in range(i, len(pos), step * 2):
                j = step
                while j > 0 and temp < len(pos):
                    if pos[temp] == '1':
                        count += 1
                    temp += 1
                    j -= 1
            if count%2 != 0:
                pos[i] = '1'
            else
                pos[i] = '0'
    red += 1
  return pos[::-1]
def write_data(message):
    with open("channel.txt","w") as file:
        for bichar in message:
            file.write(bichar+'\n')

message = input("Enter your message:")
bit = [format(ord(char), '08b') for char in str(message)]
length = len(bit)
hamming_code = ~~send~~( send_data(bit, length)
send = write_data(hamming_code)
```

receiver.py

```
def detectError(received):
    r = calredundant (len(received))
    received = list (received[::-1])
    error_position = 0
    for red in range(r):
        count = 0
        for i in range (len( received)):
            if i+1 == pow(2, red):
                step = pow(2, red)
                for temp in range (i, len(received), step*2):
                    j = step
                    while j>0 and temp < len(received):
                        if received[temp] == '1':
                            count += 1
                        temp += 1
                        j -= 1
        if count % 2 != 0:
            error_position += pow(2, red)
    return error_position
def calredundant(l):
    for i in range(l):
        if(2**i >= l+i+1):
            ~~return~~ i
def receiver(received):
    error_position = detectError(received)
    if Error_position = 0:
        print("No error detected")
```

Output:

```
sender-code.py
 Enter your message: Prathap

receiver-code.py
 Error detected at position 24
 Message is: Prathap
```

21/8/24

Result:

Thus the implementation of hamming code for error detection and correction is successfully established. and output is verified