THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

THADOMAL SHAHANI ENGINEERING COLLEGE

Bandra (W.), Mumbai - 400050

Division : C2

Batch : C23

Roll Number : 1902112

Certify that Mr. **Tushar Nankani**

of **Computer** Department, Semester **V** with

Roll No. **1902112** has completed a course of the necessary experiments in the subject **Computer Network Lab (CSL502)** under my supervision in the Thadomal Shahani Engineering College Laboratory in the **year 2021 - 2022**

Prof. Nabanita Mandal
Teacher In-Charge

Dr. Tanuja Sarode
Head of the Department

Dr. GT Thampi
Principal

Date 23/10/2021

List of Experiments
T.E. (Computer) Sem V
Subject: Computer Network Lab (CSL502)

Title of Experiments

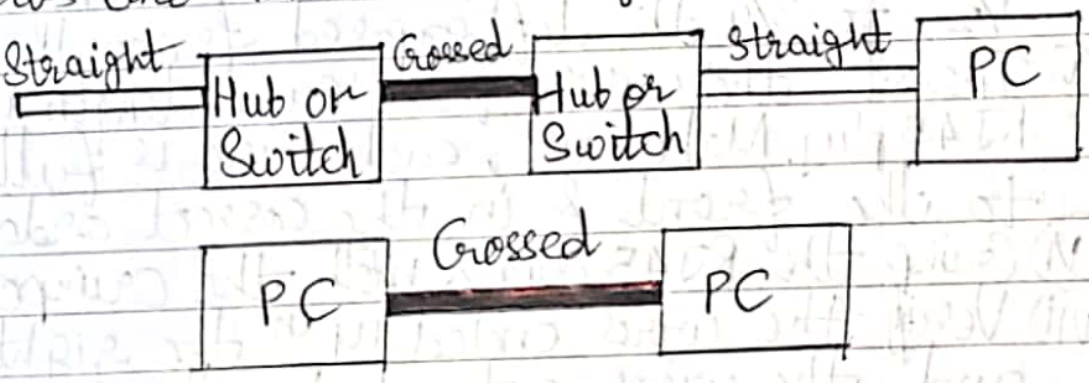
	3) Discuss Persistent and Non-Persistent Protocols used in Transport and Application layers of TCP/IP protocol suite		

CN

Experiment 1

- Study of RJ45 and CAT 5 cabling

CAT5 cable : CAT is an ethernet network cable standard defined by the electronic industries association and telecommunication industry association. It is the fifth generation of twisted pair EIA/TIA cabling. It is a twisted pair ethernet technology and the most popular CAT5 contains 4 pair of copper wires. It supports speeds upto 100 mbps. CAT5 cables runs are limited to 100 m. They consist of 4 twisted pair of copper wire terminated by RJ45 connectors. Computers hooked up to LANs are connected using CAT5 cables. They can be either crossed cables or straight - through cables. The following diagram shows the normal use of both.



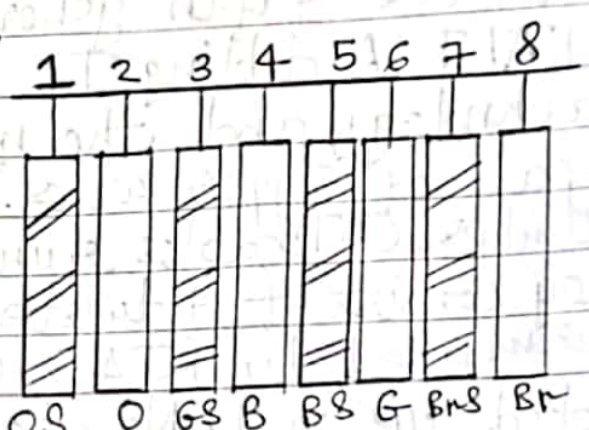
RJ45 Connectors: RJ45 is a standard type of connector for network cables. RJ45 connectors are most commonly seen with ethernet cables featuring 8 pins to which the wire strands of a cable interface electrically. Standard RJ-45 pinouts define the arrangement of the individual wires needed when attaching connectors to a cable.

Date __/__/____

Procedure for making CAT-5 cable:

(i) Strip off about 1-2 inches of exposed ethernet cable.
(ii) Untwine the pairs, do not untwine them beyond what is exposed.
(iii) Align the coloured wires according to the diagram below.

1) Orange-Striped
2) Orange
3) Green Striped
4) Blue
5) Blue Striped
6) Green
7) Brown Striped
8) Brown



(iv) Trim all the wires to the same length, about 1/2 to 3/4 th left exposed from the sheath
(v) Insert the wires of the same length into the RJ45 plug. Make sure, each wire is fully inserted to the front & in the correct order.
(vi) Crimp the RJ45 plug & with the crimper tool.
(vii) Verify the wires ended up in the right order and the wires extend to the front of the RJ45 plug & make good contact with the metal contacts in the RJ45 plug.
(viii) Cut the ethernet cable to length make sure it is more than enough for your needs.
(ix) Next we insert our cable into the cable tester. If there are 4 green lights, then our cable works properly

Conclusion: Thus we have studied RJ45 & CAT5.

Page No.

Program for Hamming Code is:

```
import java.util.*;

class hamming {

 public static int XOR(StringBuilder arr) {  int count = 0;

 char one = '1';

 for(int i = 0; i < arr.length(); i++){  if(arr.charAt(i) == one){

 count++;

 }

 }

 if(count % 2 == 0){

 return 0;

 }

 return 1;

 }

 public static int XOR(int arr[]) {  int count = 0;

 for(int i = 0; i < arr.length; i++) {  if(arr[i] == 1) {

 count++;

 }

 }

 if(count % 2 == 0){

 return 0;

 }

 return 1;

 }

 public static void main(String[] args) {

 Scanner sc = new Scanner(System.in);

 StringBuilder sb = new StringBuilder("");  System.out.print("Enter the data bits : ");

 sb.append(sc.nextLine());

 int noOfRedundantBits = 0;

 int len = sb.length();

 while(Math.pow(2, noOfRedundantBits) <= len +  noOfRedundantBits + 1){
```

```
noOfRedundantBits++;
}

//System.out.println(sb);
System.out.println("\n" + "Number of Redundant Bits  required will be : " + noOfRedundantBits);

int[] parityBits = new int[noOfRedundantBits];
int[][] checkBitsSender = { {6,5,3,2,0} , {6,4,3,1,0} ,  {5,4,3}, {2,1,0} };

for(int i = 0; i < parityBits.length; i++){
StringBuilder sb1 = new StringBuilder("");  for(int j = 0; j < checkBitsSender[i].length; j++){  int num = checkBitsSender[i][j];
char ch = sb.charAt(num);
sb1.append(ch);
}
System.out.println("P[" + (int)(Math.pow(2,i)) +"] =  XOR(" + sb1 + ")");
parityBits[i] = XOR(sb1);
}

for(int i = 0; i < parityBits.length; i++){
System.out.println("P[" + (int)(Math.pow(2,i)) +"] = " +  parityBits[i]);
}
sb.insert(3, parityBits[3]);
sb.insert(7, parityBits[2]);
sb.insert(9, parityBits[1]);
sb.insert(10, parityBits[0]);
System.out.print("How many error bits you want to  insert (1/2) : ");
int choice = sc.nextInt();
if(choice == 1){
System.out.print("\n" + "Data bits that will be sent is  : ");
System.out.println(sb + "\n");
```

```
System.out.print("At what position you want to add  error bit : ");
int orgPos = sc.nextInt();
int pos = Math.abs(sb.length() - orgPos);  if(sb.charAt(pos) == '1') {
sb.setCharAt(pos, '0');
} else {
sb.setCharAt(pos, '1');
}
System.out.println("\n" + "Data bits received on  Receiver side " + sb);
int receiver[] = new int[parityBits.length];
int[][] checkBitsReceiver = { {10,8,6,4,2,0}  ,{9,8,5,4,1,0} , {7,6,5,4}, {3,2,1,0} };
for(int i = 0; i < receiver.length; i++) {
StringBuilder sb2 = new StringBuilder("");  for(int j = 0; j < checkBitsReceiver[i].length; j++){  int num
= checkBitsReceiver[i][j];
char a = sb.charAt(num);
sb2.append(a);
}
System.out.println("C[" + (int)(Math.pow(2,i)) +"] =  XOR(" + sb2 + ")");
receiver[i] = XOR(sb2);
}
for(int i = 0; i < receiver.length; i++){
System.out.println("C[" + (int)(Math.pow(2,i)) +"] = "  + receiver[i]);
}
System.out.print("Error has occured at Bit : ");  for(int i = receiver.length - 1;i >= 0; i--)
{  System.out.print(receiver[i]);
}
System.out.println("\n");
System.out.println("Error can be resolved by  changing the bit : " + orgPos + "\n");
System.out.print("After making changes we have data  bits as : ");
if(sb.charAt(pos) == '1') {
sb.setCharAt(pos, '0');
} else {
sb.setCharAt(pos, '1');
```

```
}
System.out.print(sb);
} else {
int addParity = XOR(parityBits);
char c = Character.forDigit(addParity, 10);  sb.insert(0, c);
System.out.println("Data bits that will be sent is : " +  sb + "\n");
System.out.print("At what position you want to add  error bits : ");
int[] errorLoc = new int[2];
errorLoc[0] = Integer.parseInt(sc.next());  errorLoc[1] = Integer.parseInt(sc.next());
int pos = Math.abs(sb.length() - errorLoc[0]);  if(sb.charAt(pos) == '1') {
sb.setCharAt(pos, '0');
} else {
sb.setCharAt(pos, '1');
}
int pos2 = Math.abs(sb.length() - errorLoc[1]);  if(sb.charAt(pos2) == '1') {
sb.setCharAt(pos2, '0');
} else {
sb.setCharAt(pos2, '1');
}
System.out.println("Data bits received by Receiver is " + sb + "\n");
int finalSender = XOR(sb);
int receiver2[] = new int[parityBits.length];
int[][] checkBitsReceiver2 = { {11,9,7,5,3,1}  ,{10,9,6,7,2,1} , {8,7,6,5,0}, {4,3,2,1,0} };
for(int i = 0; i < receiver2.length; i++) {
StringBuilder sb2 = new StringBuilder("");  for(int j = 0; j < checkBitsReceiver2[i].length; j++){  int
num = checkBitsReceiver2[i][j];  char a = sb.charAt(num);
sb2.append(a);
}
System.out.println("C[" + (int)(Math.pow(2,i)) +"] =  XOR(" + sb2 + ")");
receiver2[i] = XOR(sb2);
}
```

```
for(int i = 0; i < receiver2.length; i++){
System.out.println("C[" + (int)(Math.pow(2,i)) +"] = "  + receiver2[i]);
}
int finalReceiver = XOR(receiver2);
System.out.println("P = " + finalSender);  System.out.println("C = " + finalReceiver);
if(finalReceiver == 0 && finalSender == 0) {  System.out.println("No Error");
} else if(finalReceiver != 0 && finalSender == 1) {
System.out.println("Single Error can be  corrected!!!");
} else if(finalReceiver != 0 && finalSender == 0) {
System.out.println("Double Error cannot be  corrected");
} else if(finalReceiver == 0 && finalSender == 1) {  System.out.println("Error in P[12] bit");  }
}
sc.close();
}
}
```

Output for 1-bit Error Detection and Correction

```
Enter the data bits : 1101001

Number of Redundant Bits  required will be : 4
P[1] =  XOR(10101)
P[2] =  XOR(10111)
P[4] =  XOR(001)
P[8] =  XOR(011)
P[1] = 1
P[2] = 0
P[4] = 1
P[8] = 0
How many error bits you want to  insert (1/2) : 1

Data bits that will be sent is  : 11001001101

At what position you want to add  error bit : 3

Data bits received on  Receiver side 11001001001
C[1] =  XOR(100101)
C[2] =  XOR(000111)
C[4] =  XOR(1001)
C[8] =  XOR(0011)
C[1] = 1
C[2] = 1
C[4] = 0
C[8] = 0
Error has occured at Bit : 0011

Error can be resolved by  changing the bit : 3

After making changes we have data  bits as : 11001001101
```

Output for 2-bit Error Detection

```
Enter the data bits : 1101001

Number of Redundant Bits  required will be : 4
P[1] =  XOR(10101)
P[2] =  XOR(10111)
P[4] =  XOR(001)
P[8] =  XOR(011)
P[1] = 1
P[2] = 0
P[4] = 1
P[8] = 0
How many error bits you want to  insert (1/2) : 2
Data bits that will be sent is : 011001001101

At what position you want to add  error bits : 4 8
Data bits received by Receiver is 011011000101

C[1] =  XOR(110101)
C[2] =  XOR(010011)
C[4] =  XOR(00010)
C[8] =  XOR(10110)
C[1] = 0
C[2] = 1
C[4] = 1
C[8] = 1
P = 0
C = 1
Double Error cannot be  corrected
```

Date __/__/__ Experiment 2
CN

Aim: To code a program for Hamming Code for 1-bit error detection and correction & 2-bit error detection.

Theory: Hamming code is a set of error-correction codes that can be used to detect & correct the errors that can occur when the data is moved or stored from the sender to the receiver.

Some basic concepts related to hamming code are:

Redundant Bits
Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer.

Parity Bits
A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data is even or odd. Parity bits are used for error detection.

Algorithm:
① Write the bit positions starting from 1 in binary form (1, 10, 11, 100) etc.
② All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8 etc.)
③ All the other bit positions are marked as data

Date __/__/____

bits.

④ Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.

→ Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11) etc.

→ Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the LSB (2, 3, 6, 7, 10, 11, etc.)

→ Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the 3rd position from the least significant bit (4-7, 12-15, 20-23) etc.

→ Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the 4th position from the least Significant bit bits (8-15, 24-31, 40-47, etc.)

→ In general, each parity bit covers all bits whose the bitwise AND of the parity position & the bit position is non-zero.

⑤ Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks.

⑥ Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Page No.

```
public static int XOR(StringBuilder arr) {
   int count = 0;     char one = '1';
  for(int i = 0; i < arr.length(); i++){
       if(arr.charAt(i) == one){
           count++;
       }
  }
  if(count % 2 == 0){
       return 0;
  }
  return 1;
}
public static int XOR(int arr[]) {
   int count = 0;
  for(int i = 0; i < arr.length; i++) {
       if(arr[i] == 1) {
          count++;
       }
  }
  if(count % 2 == 0){
    return 0;
  }
  return 1;
}
public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  StringBuilder sb = new StringBuilder("");
  System.out.print("Enter the data bits : ");
  sb.append(sc.nextLine());
```

```
    int noOfRedundantBits = 0;
    int len = sb.length();
    while(Math.pow(2, noOfRedundantBits) <= len + noOfRedundantBits + 1){
        noOfRedundantBits++;
    }
    //System.out.println(sb);
    System.out.println("\n" + "Number of Redundant Bits required will be : " +
noOfRedundantBits);
    int[] parityBits = new int[noOfRedundantBits];
int[][] checkBitsSender = { {6,5,3,2,0} , {6,4,3,1,0} , {5,4,3}, {2,1,0} };
    for(int i = 0; i < parityBits.length; i++){
        StringBuilder sb1 = new StringBuilder("");
        for(int j = 0; j < checkBitsSender[i].length; j++){
            int num = checkBitsSender[i][j];
            char ch = sb.charAt(num);
            sb1.append(ch);
        }
        System.out.println("P[" + (int)(Math.pow(2,i)) +"] = XOR(" + sb1 + ")");
        parityBits[i] = XOR(sb1);
    }
    for(int i = 0; i < parityBits.length; i++){
        System.out.println("P[" + (int)(Math.pow(2,i)) +"] = " + parityBits[i]);
    }
     sb.insert(3, parityBits[3]);
     sb.insert(7, parityBits[2]);
     sb.insert(9, parityBits[1]);
     sb.insert(10, parityBits[0]);
    System.out.print("How many error bits you want to insert (1/2) : ");
     int choice = sc.nextInt();
```

```
   if(choice == 1){
      System.out.print("\n" + "Data bits that will be sent is : ");
      System.out.println(sb + "\n");
      System.out.print("At what position you want to add error bit : ");
      int orgPos = sc.nextInt();
      int pos = Math.abs(sb.length() - orgPos);
      if(sb.charAt(pos) == '1') {
         sb.setCharAt(pos, '0');
      }
 else {
         sb.setCharAt(pos, '1');
      }
      System.out.println("\n" + "Data bits received on Receiver side " + sb);
       int receiver[] = new int[parityBits.length];
       int[][] checkBitsReceiver = { {10,8,6,4,2,0} ,{9,8,5,4,1,0} , {7,6,5,4}, {3,2,1,0} };
         for(int i = 0; i < receiver.length; i++) {
          StringBuilder sb2 = new StringBuilder("");
          for(int j = 0; j < checkBitsReceiver[i].length; j++){
             int num = checkBitsReceiver[i][j];
             char a = sb.charAt(num);
             sb2.append(a);
          }
          System.out.println("C[" + (int)(Math.pow(2,i)) +"] = XOR(" + sb2 + ")");
receiver[i] = XOR(sb2);
      }

      for(int i = 0; i < receiver.length; i++){
         System.out.println("C[" + (int)(Math.pow(2,i)) +"] = " + receiver[i]);
      }
```

```
        System.out.print("Error has occured at Bit : ");
         for(int i = receiver.length - 1;i >= 0; i--) {
            System.out.print(receiver[i]);
        }
        System.out.println("\n");
        System.out.println("Error can be resolved by changing the bit : " + orgPos + "\n");
        System.out.print("After making changes we have data bits as : ");
        if(sb.charAt(pos) == '1') {             sb.setCharAt(pos, '0');
        }
else {
            sb.setCharAt(pos, '1');
        }
        System.out.print(sb);
    }
else {
         int addParity = XOR(parityBits);
         char c = Character.forDigit(addParity, 10);
         sb.insert(0, c);
        System.out.println("Data bits that will be sent is : " + sb + "\n");
        System.out.print("At what position you want to add error bits : ");
        int[] errorLoc = new int[2];
        errorLoc[0] = Integer.parseInt(sc.next());
        errorLoc[1] = Integer.parseInt(sc.next());
        int pos = Math.abs(sb.length() - errorLoc[0]);
        if(sb.charAt(pos) == '1') {
             sb.setCharAt(pos, '0');
        }
else {
            sb.setCharAt(pos, '1');
```

```
        }
        int pos2 = Math.abs(sb.length() - errorLoc[1]);         if(sb.charAt(pos2) == '1') {
sb.setCharAt(pos2, '0');
        }
 else {
            sb.setCharAt(pos2, '1');
        }
        System.out.println("Data bits received by Receiver is " + sb + "\n");
        int finalSender = XOR(sb);
        int receiver2[] = new int[parityBits.length];
        int[][] checkBitsReceiver2 = { {11,9,7,5,3,1} ,{10,9,6,7,2,1} , {8,7,6,5,0}, {4,3,2,1,0} };
        for(int i = 0; i < receiver2.length; i++) {
            StringBuilder sb2 = new StringBuilder("");
            for(int j = 0; j < checkBitsReceiver2[i].length; j++){
                int num = checkBitsReceiver2[i][j];
                char a = sb.charAt(num);
                sb2.append(a);
            }
            System.out.println("C[" + (int)(Math.pow(2,i)) +"] = XOR(" + sb2 + ")");
            receiver2[i] = XOR(sb2);
        }
        for(int i = 0; i < receiver2.length; i++){
            System.out.println("C[" + (int)(Math.pow(2,i)) +"] = " + receiver2[i]);
        }
        int finalReceiver = XOR(receiver2);
        System.out.println("P = " + finalSender);
        System.out.println("C = " + finalReceiver);
        if(finalReceiver == 0 && finalSender == 0) {
            System.out.println("No Error");
```

```
        } else if(finalReceiver != 0 && finalSender == 1) {
            System.out.println("Single Error can be corrected!!!");
        } else if(finalReceiver != 0 && finalSender == 0) {
            System.out.println("Double Error cannot be corrected");
        } else if(finalReceiver == 0 && finalSender == 1) {
            System.out.println("Error in P[12] bit");
        }
    }
    sc.close();
  }
}
```

Output:

```
Enter the data bits : 1001100

Number of Redundant Bits required will be : 4
P[1] = XOR(00101)
P[2] = XOR(01101)
P[4] = XOR(011)
P[8] = XOR(001)
P[1] = 0
P[2] = 1
P[4] = 0
P[8] = 1
How many error bits you want to insert (1/2) : 1

Data bits that will be sent is : 10011100010

At what position you want to add error bit : 4

Data bits received on Receiver side 10011101010
C[1] = XOR(000101)
C[2] = XOR(101101)
C[4] = XOR(1011)
C[8] = XOR(1001)
C[1] = 0
C[2] = 0
C[4] = 1
C[8] = 0
Error has occured at Bit : 0100

Error can be resolved by changing the bit : 4

After making changes we have data bits as : 10011100010
```

Date __/__/__ Experiment 3

CN

Aim: Write a program to implement Cyclic Redundancy Code (CRC) to find the codeword & to check whether the received data is error free or not.

Theory:

Cyclic Redundancy Check (CRC)

An error detection mechanism in which a special number is appended to a block of data in order to detect any changes introduced during storage (or transmission). The CRC is recalculated on retrieval (or reception) & compared to the value originally transmitted, which can reveal certain types of error. For example, a single corrupt bit may cancel each other out. CRC is more powerful than VRC & LRC in detecting errors. It is not based on binary addition like VRC and LRC. Rather it is based on binary division. At the sender side, the data unit to be transmitted is divided by a predetermined divisor (binary number) in order to obtain the remainder. The remainder is called CRC. The CRC has one bit less than the divisor. It means that if CRC is of n bits, divisor is of n+1 bit. The sender appends this CRC to the end of data unit to be transmitted is divided by predetermined divisor i.e. remainder becomes zero. At the destination, the incoming data unit i.e. data+

Page No.

Date __/__/____

CRC is divided by the same number (predetermined binary divisor). If the remainder after division is zero, then there is no error in the data unit & receiver accepts it. If remainder after division is 0 then there is no error in the data unit & receiver accepts it. If remainder after division is not 0 then the data unit & receiver accepts it. If remainder after division is not zero, it indicates that the data unit has been damaged in transit & ∴ it is rejected. This technique is more powerful than the parity check & checksum error detection.

The various steps followed in the CRC method are :-

1. A string of n 0s is appended to the data unit. The length of predetermined divisor is n+1.

2. The newly formed data unit i.e. original data + string of n 0s are divided by the divisor using binary division & remainder is obtained. The remainder is called CRC.

3. Now, string of n 0s appended to data unit is replaced by the CRC remainder (which is also of n bit).

4. The data unit + CRC is then transmitted to receiver.

Page No.

5. The receiver on receiver it divides data unit + CRC by the same divisor & checks the remainder.

6. If the remainder of division is zero, receiver assume that there is no error in data and it accepts it.

7. If remainder is non-zero, then there is an error in data and receiver rejects it.

EXAMPLE

Data is to be transmitted is 1001 & predetermined divisor is 1011. The procedure given below is used:

String of 3 zeroes is appended to 1011 as divisor is of 4 bits. Now newly formed data is 1011000.

1. Data unit 1011000 is divided by 1011.

```
          1010   (Quotient)
        _______
1011 | 1001000
       1011
       -----
        01000
        0000
        ----
        1000
        1011
        ----
         011   ← Remainder 3 bits
```

Data	CRC
1001	110

2. During this process of division, whenever the leftmost bit of dividend or remainder is 0, we use a string of 0s of same length as divisor. Thus, in this case, divisor 1011 is replaced by 0000.

3. At the receiver side, data received is 1001110.

4. This data is again divided by a divisor 1011

5. The remainder obtained is 000; there is no error.

```
            1010
1011 ) 1001110
       1011
       ----
        0101
        0000
        ----
         1011
         1011
         ----
          0000
          0000
          ----
           000    Remainder is 0, no error
```

- CRC can detect all the burst error that affect an odd number of bits.

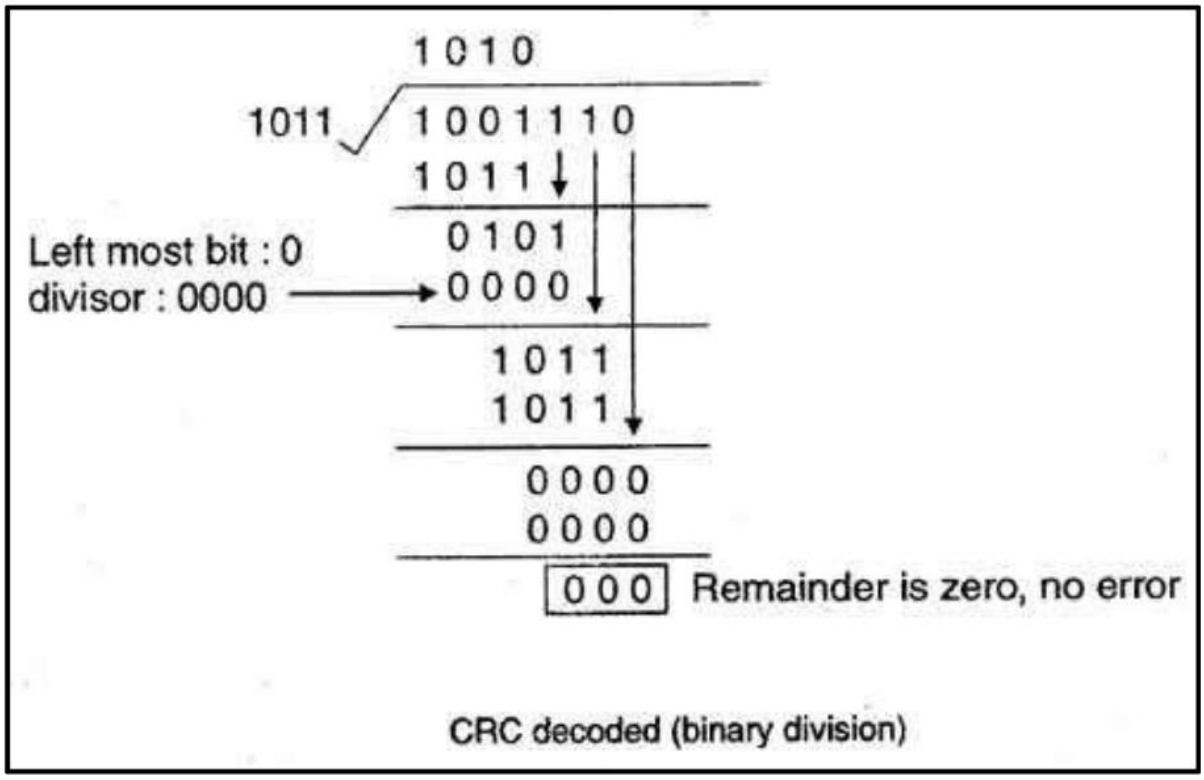
- The probability of error detection & the types of detectable errors depends on the choice of divisor.

Thus, two major requirement of CRC are:

(a) CRC should have exactly 1 bit less than divisor

(b) Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.

CONCLUSION: Thus, we implemented to the CRC program wherein we found codeword & detected whether the received data is error free or not.



CRC decoded (binary division)

• CRC can detect all the burst errors that affect an odd number of bits.

• The probability of error detection and the types of detectable errors depends on the choice of divisor.

Thus, two major requirement of CRC are:

(a) CRC should have exactly one bit less than divisor.

(b) Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.

Conclusion: Thus, we implemented the CRC program wherein we found codeword and detected whether the received data is error free or not.

Program:

```
import java.util.Scanner; public class CRC {

public static int XOR(int x, int y) { if(x == y) {

return 0; }

return 1; }
```

```
public static int flip(int x) { if(x == 0) {
return 1;
}
return 0; }
public static int[] moduloDivision(String data, int[] dividend, int[] divisor) {
for(int i = 0; i < data.length() ; i++) {
if(dividend[i] == 1){
for(int j = 0; j < divisor.length; j++) {
dividend[i + j] = XOR(dividend[i + j], divisor[j]); }
} }
return dividend; }
public static void displayCRC(String data, int[] dividend)
{ System.out.print("CRC is : ");
for(int i = data.length(); i < dividend.length; i++) {
System.out.print(dividend[i]); }
System.out.println(); }
public static void displayCheckSum(String data, int[] dividend) {
System.out.print("Checksum code is : "); for(int i = 0; i < data.length(); i++) {
dividend[i] = data.charAt(i) - '0'; }
for(int i = 0; i < dividend.length; i++) { System.out.print(dividend[i]);
} System.out.println();
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter data bits : "); String data = sc.next();
System.out.print("Enter check bits : "); String check = sc.next();
int dividend[] = new int[data.length() + (check.length() - 1)];
int divisor[] = new int[check.length()];
for(int i = 0; i < data.length(); i++) { dividend[i] = data.charAt(i) - '0';
}
for(int i = 0; i < check.length(); i++) { divisor[i] = check.charAt(i) - '0';
```

```
}
// calculating remainder(CRC)
dividend = moduloDivision(data, dividend, divisor);
//display remainder displayCRC(data, dividend);
// display checksum displayCheckSum(data, dividend);
//asking for change in checksum
System.out.print("Do you want to put error bit in checksum (0/1) : ");
int choice = sc.nextInt();
if(choice == 1) {
System.out.print("How many error bits you want to change : ");
int select = sc.nextInt();
System.out.print("Enter the bit number you want to change : ");
String change = sc.next();
for(int i = 0; i < select; i++) {
dividend[change.charAt(i) - '0'] = flip(dividend[change.charAt(i) - '0']);
}
dividend = moduloDivision(data, dividend, divisor);
displayCRC(data, dividend);
System.out.println("We see that remainder is not 0. Hence data is corrupted!!");
} else {
System.out.println("CRC obtained at receiver side is zero");
System.out.println("Data sent without corruption"); }
}
}
```

Output:

```
C:\Users\91961\Desktop>java CRC
Enter data bits : 1001
Enter check bits : 1011
Do you want to put error bit in checksum (0/1) : 1
How many error bits you want to change : 3
Enter the bit number you want to change : 246
CRC is : 101
We see that remainder is not 0. Hence data is corrupted!!

C:\Users\91961\Desktop>java CRC
Enter data bits : 1101011111
Enter check bits : 10011
Do you want to put error bit in checksum (0/1) : 0
CRC obtained at receiver side is zero
Data sent without corruption
```

Date __/__/__ Experiment 4

CN

Aim: Go-Back-N flow control:-

Theory: Go Back-N protocal, also called Go-Back-N Automatic Repeat request, is a data link layer protocal that uses a sliding window method for reliable & sequential delivery of data frames. It is a case of sliding window protocal having to send window size of N & receiving window size of 1.

Go-Back-N ARQ provides for sending multiple frames before receiving the acknowledgement for the 1st frame. The frames are sequentially numbered & a finite number of frames. The maximum number of frames that can be sent depends upon the size of the sending window. If the acknowledgement of a frame is not received within an agreed upon time period, all frames starting from that frame are retransmitted.

The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an n-bit field, then the range of sequence numbers that can be assigned to is 0 to 2n-1. Consequently the size of the sending window is 2n-1. Thus, in order to accomodate a sending window size of 2n-1, a n-bit sequence number is chosen.

Page No.

Code :

```
import java.util.Scanner;
public class goBackN {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
int count;
System.out.print("Enter window size : ");
int windowSize = sc.nextInt();
System.out.print("Enter total frames to be sent : ");
int totalFrames = sc.nextInt();
//initializing array with data frames
int[] senderFrames = new int[totalFrames];
for(int i = 0; i < senderFrames.length; i++) {
senderFrames[i] = i;
}
//displayin data frames
for(int i = 0; i < senderFrames.length; i++) {
System.out.print(senderFrames[i] + " | ");
```

```
}
//displaying sender window
System.out.println();
System.out.print("Do you want to start sending frames (0/1) : ");
int choice = sc.nextInt();
System.out.println();
if(choice == 1) {
int ptrOnWindowLeftSender = 0;
int ptrOnWindowLeftReceiver = 0;
int totalSentFrames = 0;
while(ptrOnWindowLeftSender < totalFrames){
//Sender side
count = 0;
System.out.println("At Sender End:");
for(int i = ptrOnWindowLeftSender; (i <
totalFrames && count < windowSize); i++) {
System.out.println("Sent frame[" + (i+1) + "]");
ptrOnWindowLeftSender++;
totalSentFrames++;
count++;
}
//ptrOnWindowLeft = 0;
System.out.println();
//Receiver side
System.out.println("At Receiver end: ");
int j = 0;
count = 0;
for(int i = ptrOnWindowLeftReceiver; (i <
totalFrames && count < windowSize) ; i++) {
System.out.print("Did you receive frame[" +
(i+1) +"] (y/n) : ");
```

```
char yN = sc.next().charAt(0);
if(yN == 'n'){
System.out.println("Frames will again be sent from frame no. " + (i+1));
System.out.println();
ptrOnWindowLeftSender = i;
break;
} else {
j++;
ptrOnWindowLeftReceiver++;
}
count++;
}
if(j == windowSize){
System.out.println("All Frames from this window sent without errors. Sending
next frames...");
System.out.println();
}
}
System.out.println();
System.out.println("All frames are sent." + "\n" + "Total no. of frames sent
including retransmission is " +totalSentFrames);
}
}
}
```

Output:

```
Enter window size : 4
Enter total frames to be sent : 10
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
Do you want to start sending frames (0/1) : 1

At Sender End:
Sent frame[1]
Sent frame[2]
Sent frame[3]
Sent frame[4]

At Receiver end:
Did you receive frame[1] (y/n) : y
Did you receive frame[2] (y/n) : n
Frames will again be sent from frame no. 2

At Sender End:
Sent frame[2]
Sent frame[3]
Sent frame[4]
Sent frame[5]

At Receiver end:
Did you receive frame[2] (y/n) : y
Did you receive frame[3] (y/n) : y
Did you receive frame[4] (y/n) : y
Did you receive frame[5] (y/n) : y
All Frames from this window sent without errors. Sending next frames...

At Sender End:
Sent frame[6]
Sent frame[7]
Sent frame[8]
Sent frame[9]
```

```
At Receiver end:
Did you receive frame[6] (y/n) : y
Did you receive frame[7] (y/n) : y
Did you receive frame[8] (y/n) : y
Did you receive frame[9] (y/n) : y
All Frames from this window sent without errors. Sending next frames...

At Sender End:
Sent frame[10]

At Receiver end:
Did you receive frame[10] (y/n) : y

All frames are sent.
Total no. of frames sent including retransmission is 13
```

Experiment 5

Aim: Build a simple network topology and configure it

Theory: Packet Tracer supplements physical equipment in the classroom by allowing students to create a network with an almost unlimited number of devices, encouraging practice, discovery, and troubleshooting. The simulation-based learning environment helps students develop 21st century skills such as decision making, creative and critical thinking, and problem solving. Packet Tracer complements the Networking Academy curricula, allowing instructors to easily teach and demonstrate complex technical concepts and networking systems design. Instructors can customize individual or multiuser activities, providing hands-on lessons for students that offer value and relevance in their classrooms. Students can build, configure, and troubleshoot networks using virtual equipment and simulated connections, alone or in collaboration with other students. Packet Tracer offers an effective, interactive environment for learning networking concepts and protocols. Most importantly, Packet Tracer helps students and instructors create their own virtual "network worlds" for exploration, experimentation, and explanation of networking concepts and technologies.

Packet Tracer can be used in a variety of ways:

- Group work
- Class work, Homework, and Distance Learning
- Formative assessment
- Hands-on lab reinforcement
- Lecture demonstrations
- Modeling and visualization of networking device algorithms and networking protocols
- Case studies
- Multi-user cooperative and competitive activities
- Competitions
- Problem-solving activities in concept-building, skill-building, design, and troubleshooting

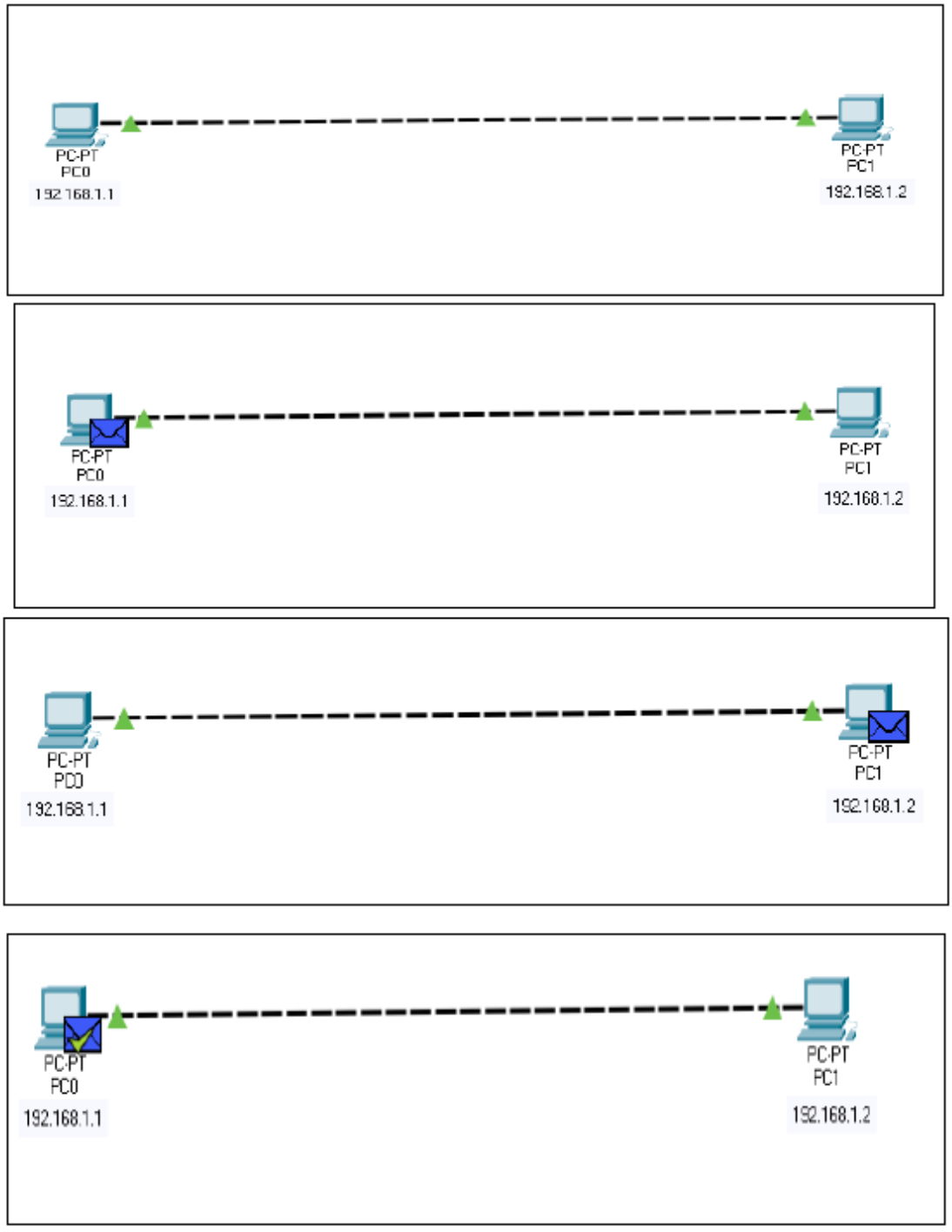
Key Features

- **Packet Tracer Workspaces:** Cisco Packet Tracer has two workspaces— logical and physical. The logical workspace allows users to build logical network topologies by placing, connecting, and clustering virtual network devices. The physical workspace provides a graphical physical dimension of the logical network, giving a sense of scale and placement in how network devices such as routers, switches, and hosts would look in a real environment. The physical view also provides geographic representations of networks, including multiple cities, buildings, and wiring closets. Figure 3. The physical workspace provides a graphical view of the logical network
- **Modular Devices:** Graphical representations visually simulate hardware and offer the ability to insert interface cards into modular routers and switches, which then become part of the simulation.
- **Multiuser Functionality:** Cisco Packet Tracer is a network-capable application, with a multiuser peer-to-peer mode that allows collaborative construction of virtual networks over a real network. The multiuser feature enables exciting collaborative and competitive interactions, providing the option to progress from individual to social learning and features

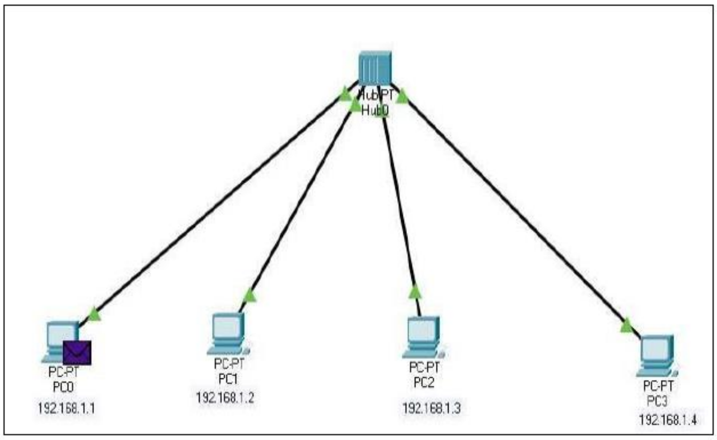
opportunities for collaboration, competition, remote instructor student interactions, social networking, and gaming.

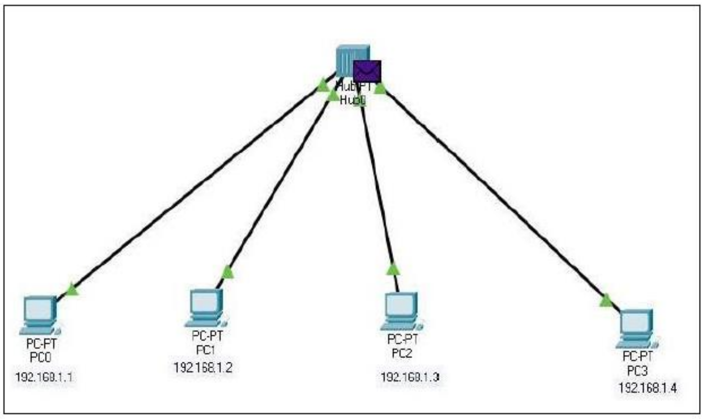
Output:

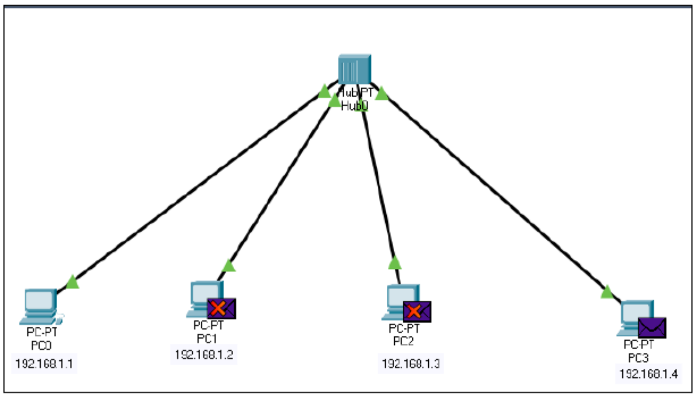
PC to PC

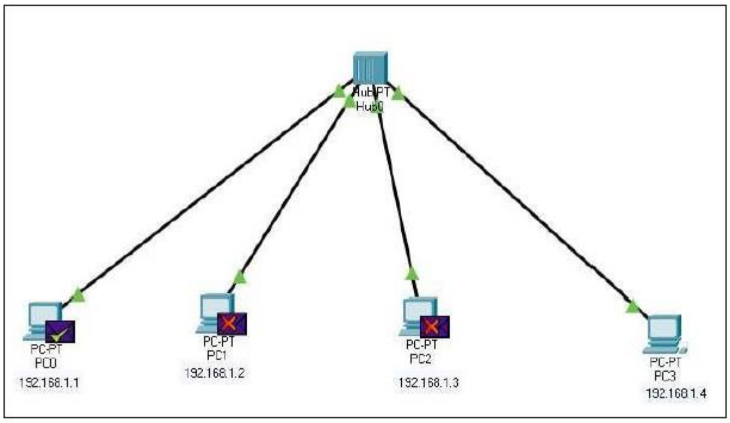


Hub to 4 PCs

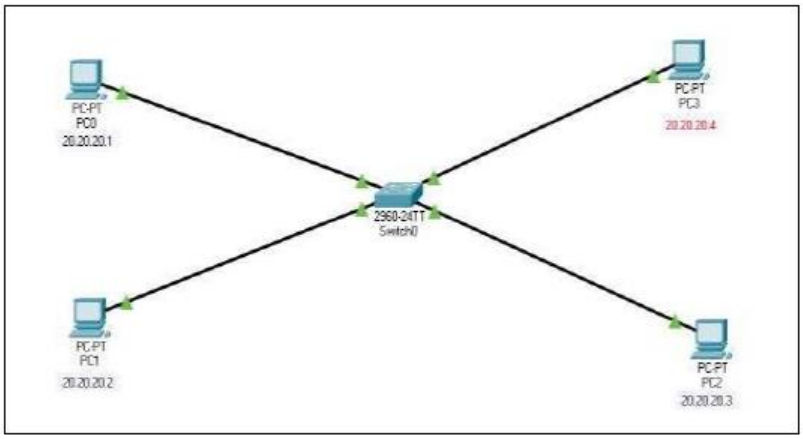

Hub-PT
Hub0
PC-PT
PC0
192.168.1.1
PC-PT
PC1
192.168.1.2
PC-PT
PC2
192.168.1.3
PC-PT
PC3
192.168.1.4

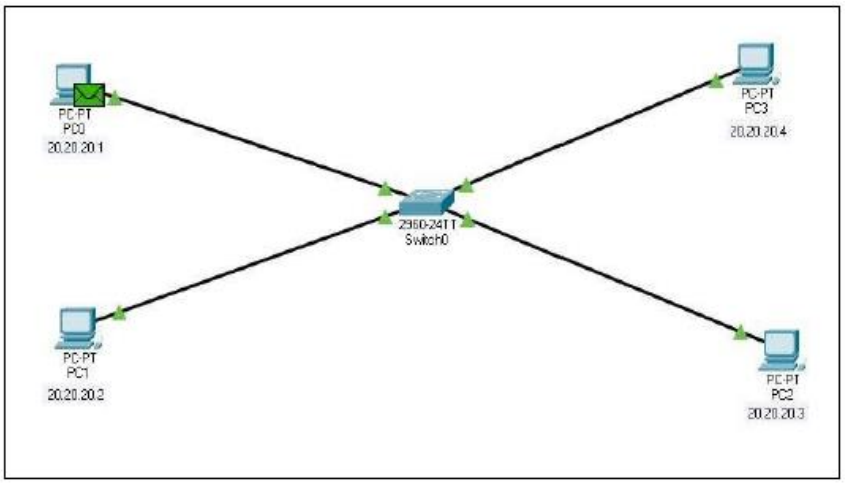

Hub-PT
Hub0
PC-PT
PC0
192.168.1.1
PC-PT
PC1
192.168.1.2
PC-PT
PC2
192.168.1.3
PC-PT
PC3
192.168.1.4

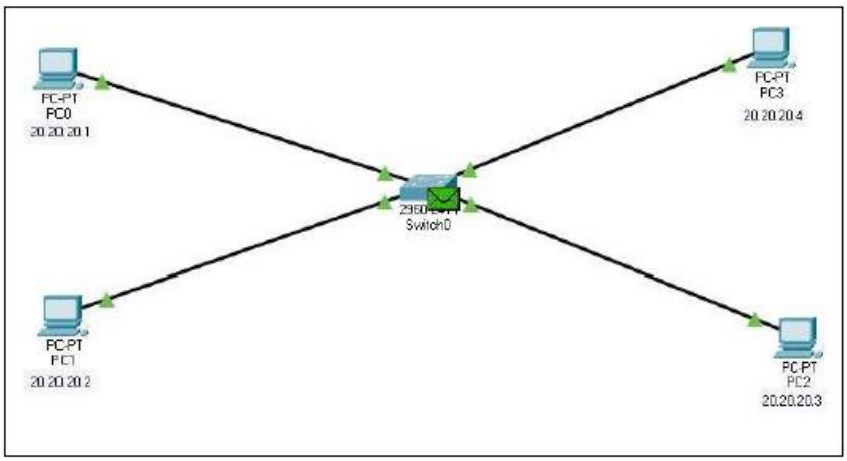

Hub-PT
Hub0
PC-PT
PC0
192.168.1.1
PC-PT
PC1
192.168.1.2
PC-PT
PC2
192.168.1.3
PC-PT
PC3
192.168.1.4

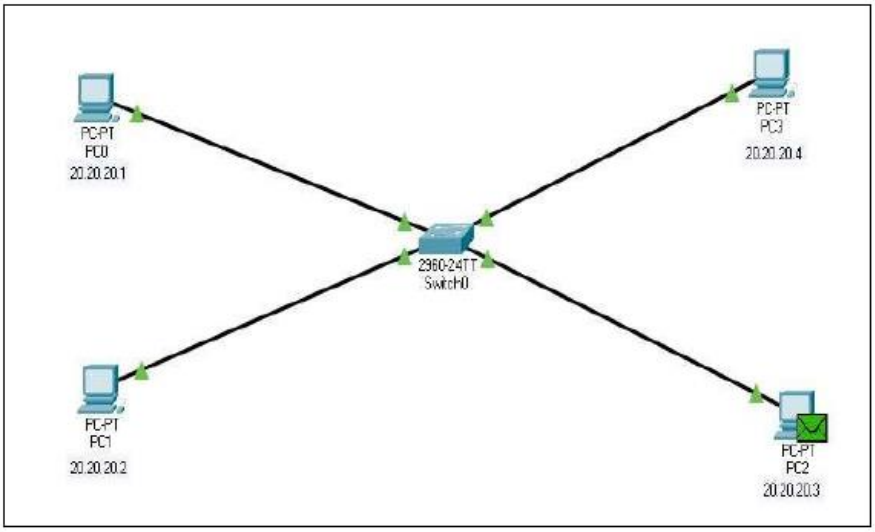

Hub-PT
Hub0
PC-PT
PC0
192.168.1.1
PC-PT
PC1
192.168.1.2
PC-PT
PC2
192.168.1.3
PC-PT
PC3
192.168.1.4

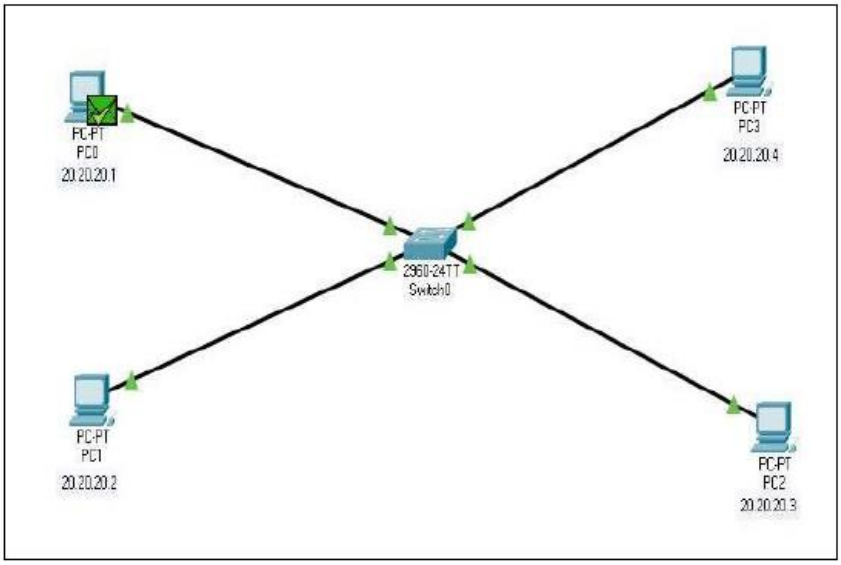
Switch to 4 PCs



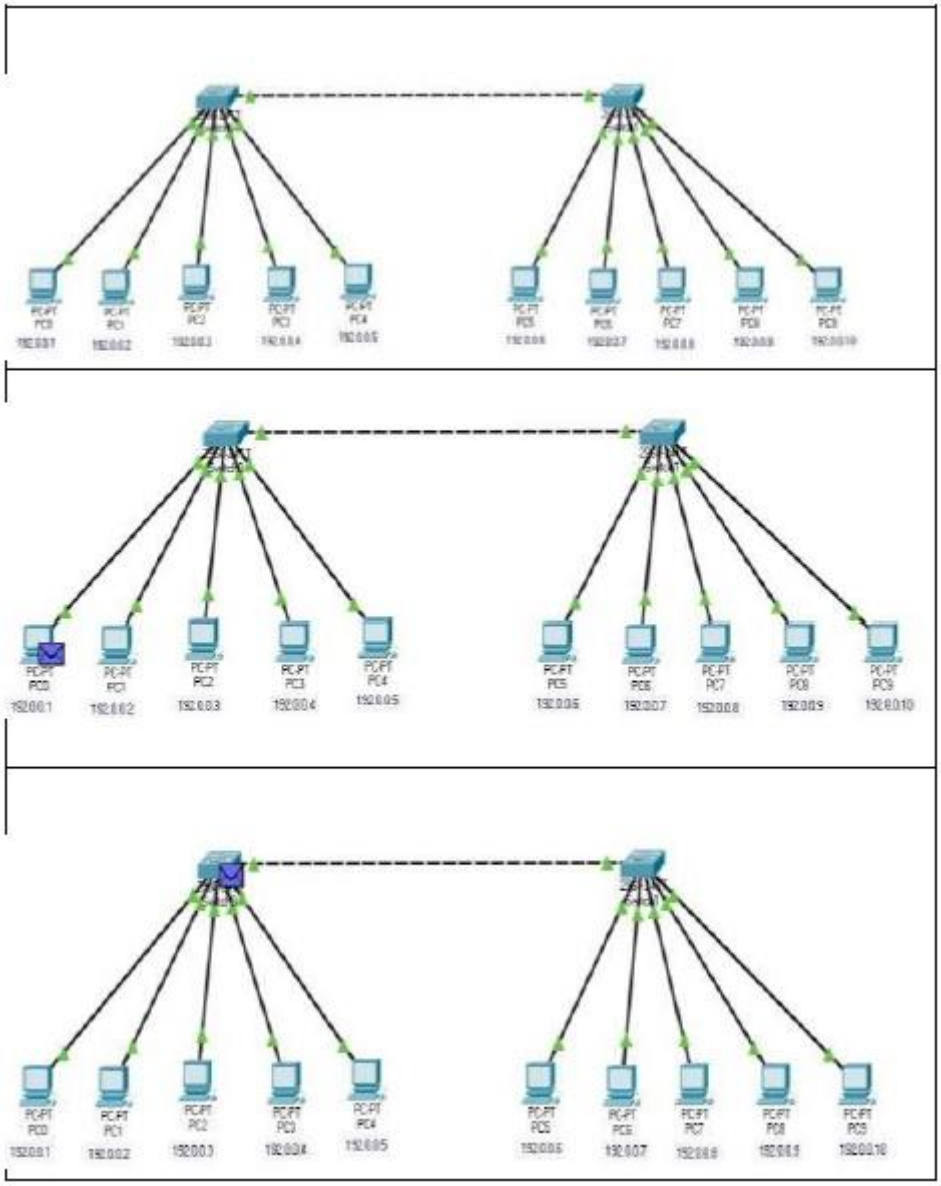


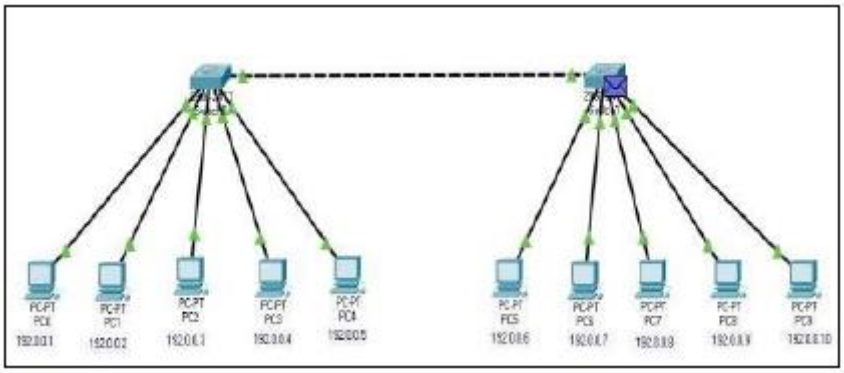


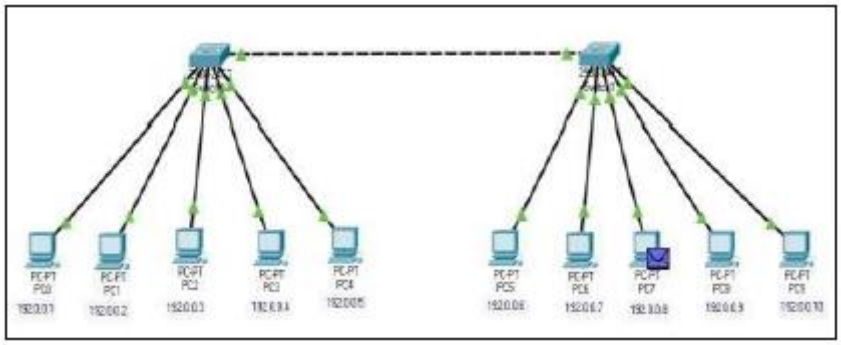

PC-PT
PC0
20.20.20.1
PC-PT
PC3
20.20.20.4
2960-24TT
Switch0
PC-PT
PC1
20.20.20.2
PC-PT
PC2
20.20.20.3

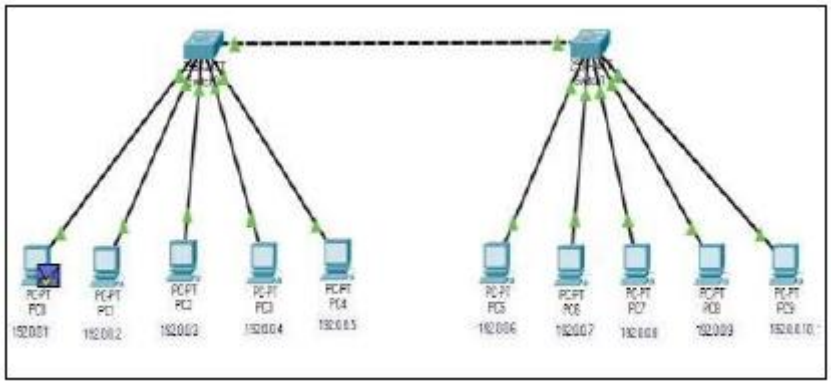

PC-PT
PC0
20.20.20.1
PC-PT
PC3
20.20.20.4
2960-24TT
Switch0
PC-PT
PC1
20.20.20.2
PC-PT
PC2
20.20.20.3

Switch to Switch

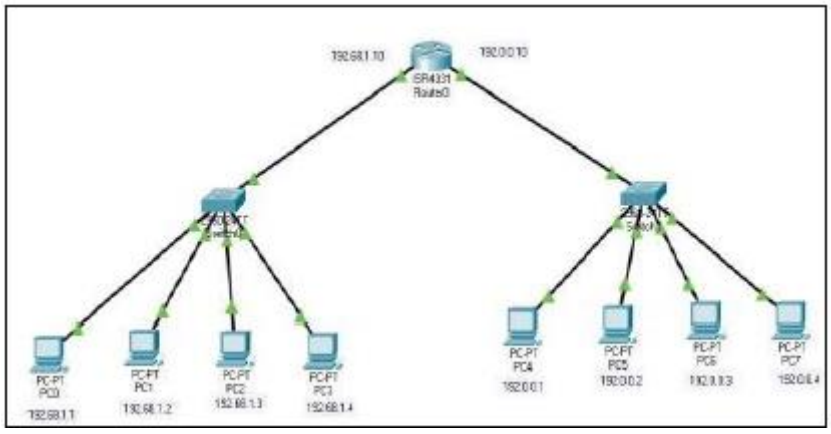


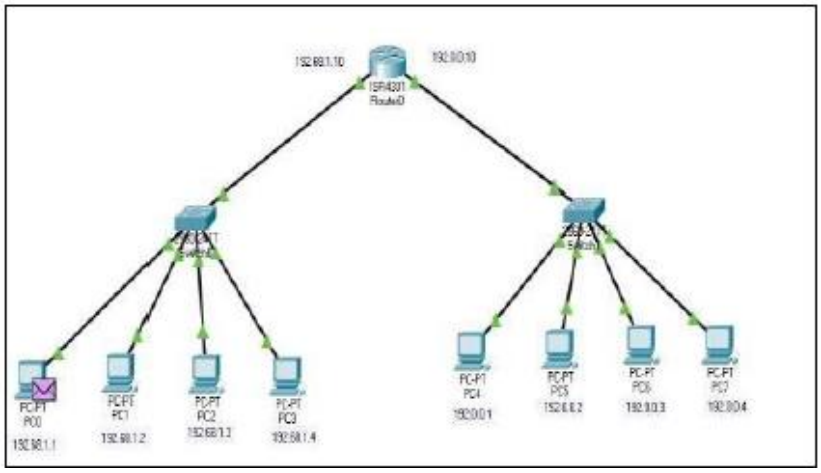


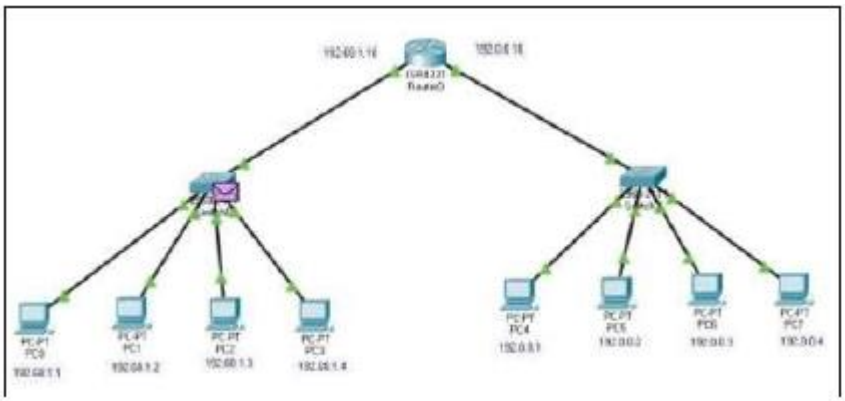


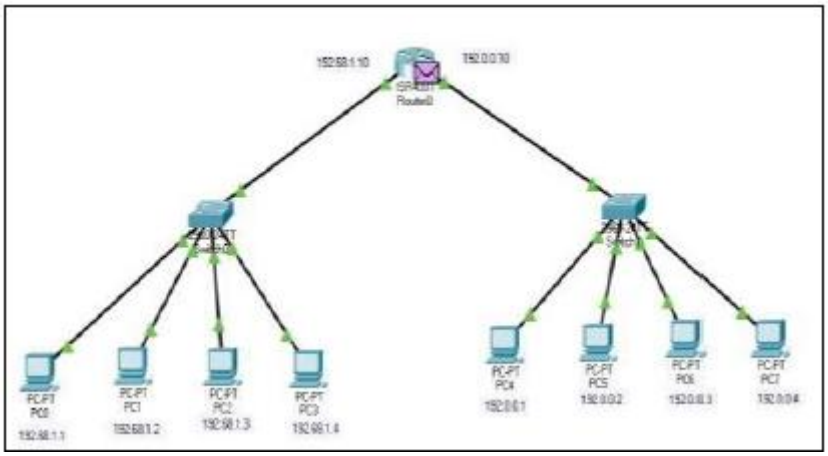


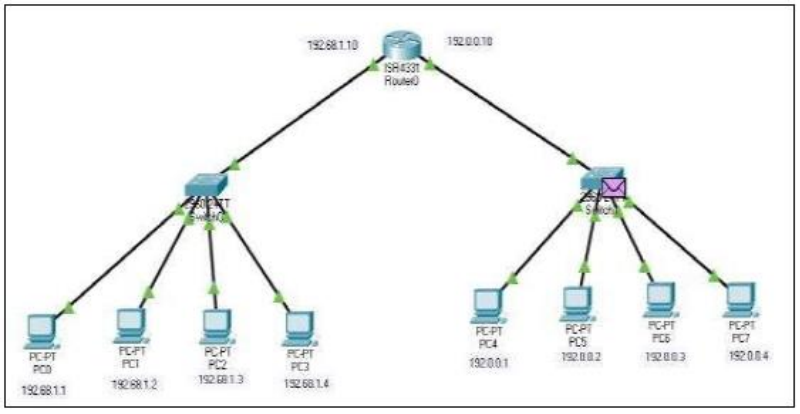
Two Switches connecting one router

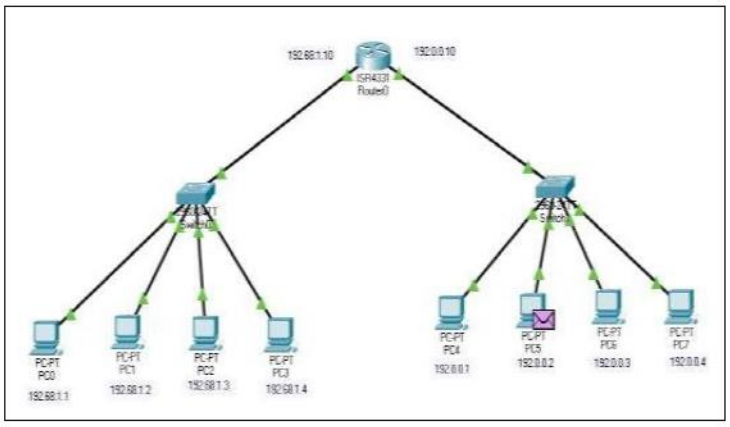


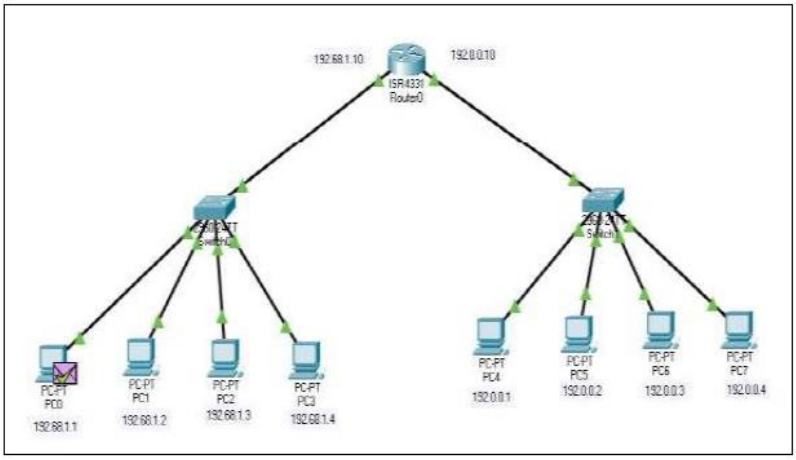











192.68.1.10
192.0.0.10
ISR4331
Router0
PC-PT
PC0
192.68.1.1
PC-PT
PC1
192.68.1.2
PC-PT
PC2
192.68.1.3
PC-PT
PC3
192.68.1.4
PC-PT
PC4
192.0.0.1
PC-PT
PC5
192.0.0.2
PC-PT
PC6
192.0.0.3
PC-PT
PC7
192.0.0.4

Experiment 6

AIM: Design VPN and Configure RIP/OSPF using Packet tracer.

THEORY:

Routing Information Protocol (RIP):

Routing Information Protocol (RIP) is a distance vector protocol that uses hop count as its primary metric. RIP defines how routers should share information when moving traffic among an interconnected group of local area networks (LANs). Routing Information Protocol was originally designed for Xerox PARC

Universal Protocol and was called GWINFO in the Xerox Network Systems (XNS) protocol suite in 1981. RIP, which was defined in RFC 1058 in 1988, is known for being easy to configure and easy to use in small networks.

Versions:

There are three versions of the Routing Information Protocol: RIPv1 and RIPv2

- RIPv1-- standardized in 1988 -- is also called Classful Routing Protocol because it does not send subnet mask information in its routing updates. On the other hand, RIPv2 -- standardized in 1998 -is called Classless Routing Protocol because it does send subnet mask information in its routing updates. RIPng is an extension of RIPv2 that was made to support IPv6.
- In RIPv1, routes are decided based on the IP destination and hop count. RIPv2 advanced this method and started to include subnet masks and gateways. Furthermore, the routing table in RIPv1 is broadcast to every station on the attached network whereas RIPv2 sends the routing table to a multicast address in an effort to reduce network traffic. Additionally, RIPv2 uses authentication for security.

Routing Information Protocol (RIP) working:

- RIP uses a distance vector algorithm to decide which path to put a packet on to get to its destination. Each RIP router maintains a routing table, which is a list of all the destinations the router knows how to reach.
- Each router broadcasts its entire routing table to its closest neighbours every 30 seconds.
- In this context, neighbours are the other routers to which a router is connected directly -- that is, the other routers on the same network segments as the selected router. The neighbours, in turn, pass the information on to their nearest neighbours, and so on, until all RIP hosts within the network have the same knowledge of routing paths.
- This shared knowledge is known as convergence.

- If a router receives an update on a route, and the new path is shorter, it will update its table entry with the length and next-hop address of the shorter path.
- If the new path is longer, it will wait through a "hold-down" period to see if later updates reflect the higher value as well. It will only update the table entry if the new, longer path has been determined to be stable.
- If a router crashes or a network connection is severed, the network discovers this because that router stops sending updates to its neighbors, or stops sending and receiving updates along the severed connection.

VLAN :

VLAN is a custom network which is created from one or more local area networks. It enables a group of devices available in multiple networks to be combined into one logical network. The result becomes a virtual LAN that is administered like a physical LAN. The full form of VLAN is defined as Virtual Local Area Network.

Without VLANs, a broadcast sent from a host can easily reach all network devices. Each and every device will process broadcast received frames. It can increase the CPU overhead on each device and reduce the overall network security. In case if you place interfaces on both switches into separate VLAN, a broadcast from host A can reach only devices available inside the same VLAN. Hosts of VLANs will not even be aware that the communication took place.

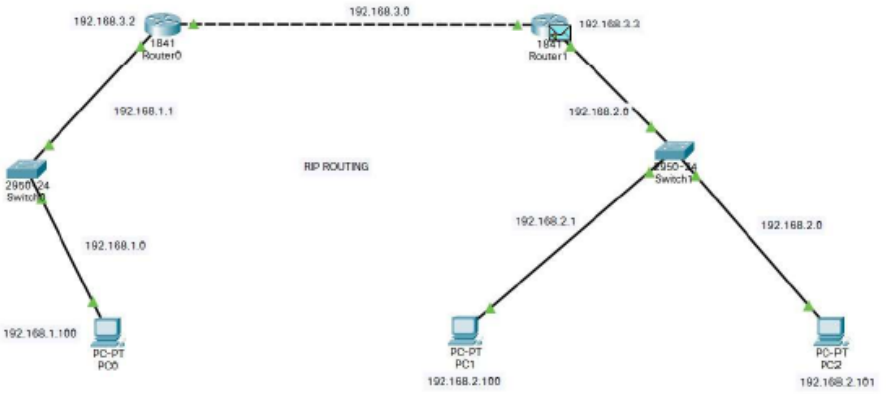
Characteristics of VLAN :

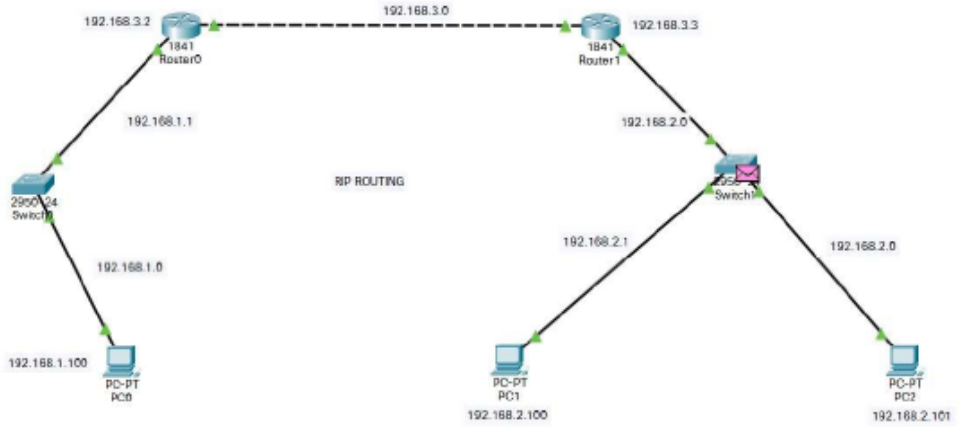
Here are the important characteristics of VLAN:

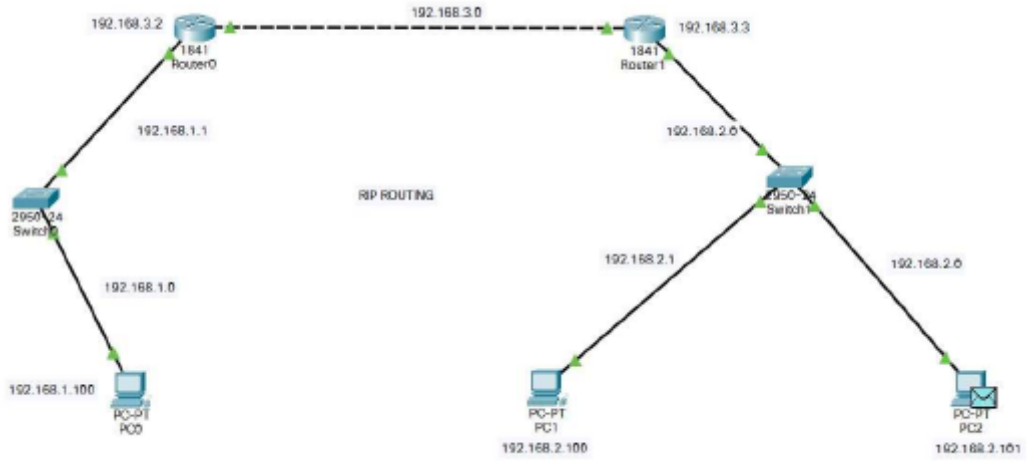
- Virtual LANs offer structure for making groups of devices, even if their networks are different.
- It increases the broadcast domains possible in a LAN.
- Implementing VLANs reduces the security risks as the number of hosts which are connected to the broadcast domain decreases.
- This is performed by configuring a separate virtual LAN for only the hosts having sensitive information.
- It has a flexible networking model that groups users depending on their departments instead of network location.
- Changing hosts/users on a VLAN is relatively easy. It just needs a new port-level configuration.

Output:

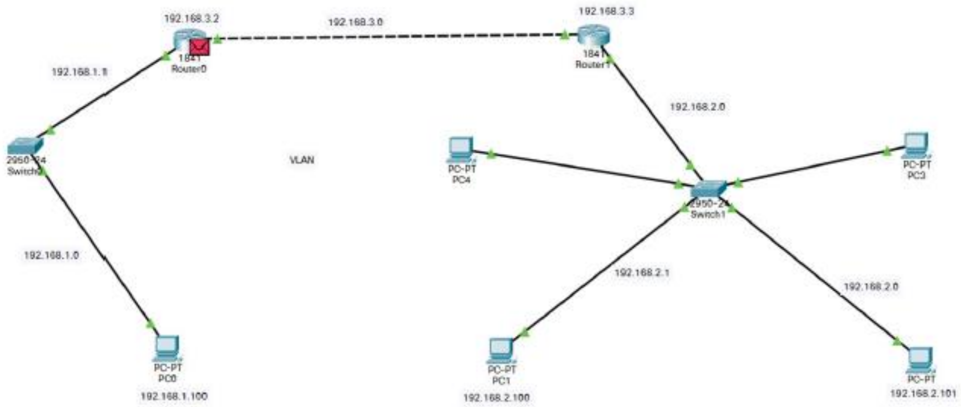
RIP ROUTING:

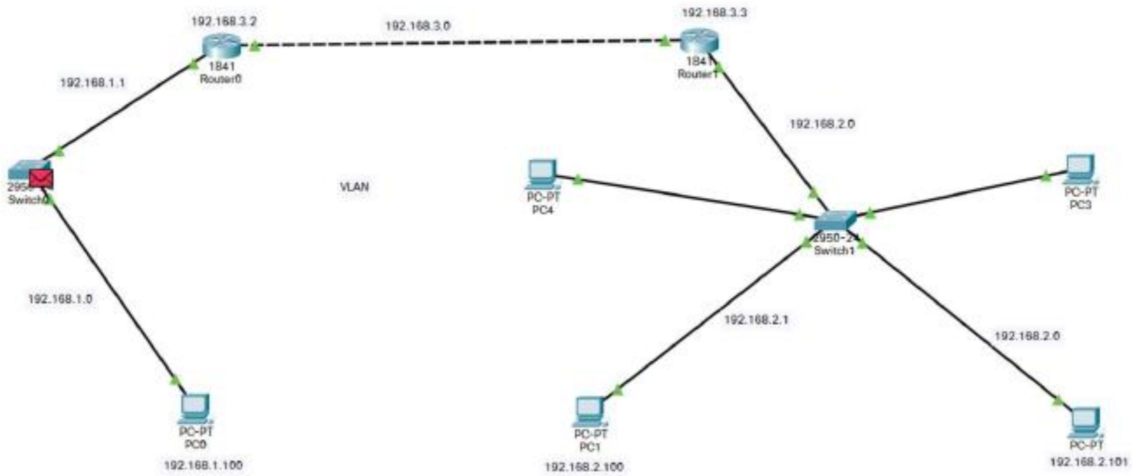

192.168.3.2
192.168.3.0
1841
Router0
1841
Router1
192.168.3.3
192.168.1.1
192.168.2.0
RIP ROUTING
2950-24
Switch0
2950-24
Switch1
192.168.2.1
192.168.2.0
192.168.1.0
192.168.1.100
PC-PT
PC0
PC-PT
PC1
192.168.2.100
PC-PT
PC2
192.168.2.101

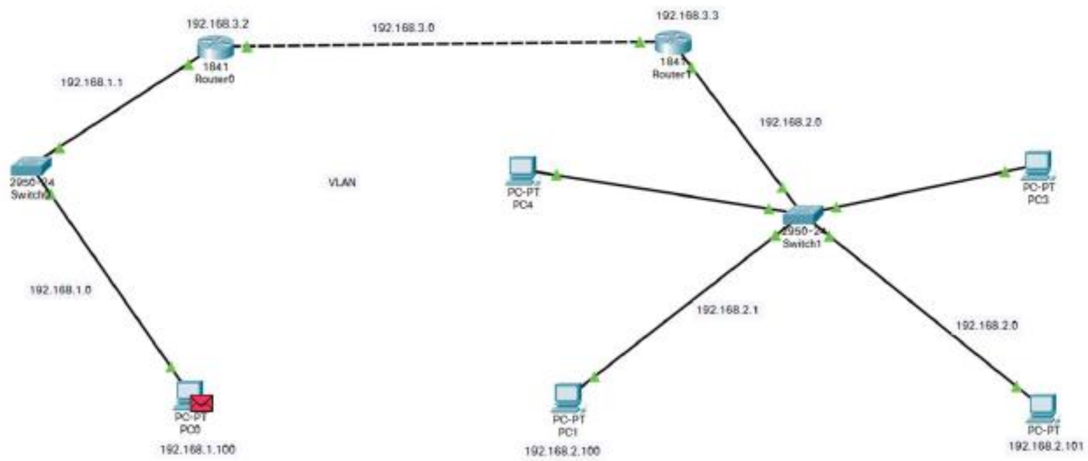

192.168.3.2
192.168.3.0
1841
Router0
1841
Router1
192.168.3.3
192.168.1.1
192.168.2.0
RIP ROUTING
2950-24
Switch0
Switch1
192.168.2.1
192.168.2.0
192.168.1.0
192.168.1.100
PC-PT
PC0
PC-PT
PC1
192.168.2.100
PC-PT
PC2
192.168.2.101


192.168.3.2
192.168.3.0
1841
Router0
1841
Router1
192.168.3.3
192.168.1.1
192.168.2.0
RIP ROUTING
2950-24
Switch0
Switch1
192.168.2.1
192.168.2.0
192.168.1.0
192.168.1.100
PC-PT
PC0
PC-PT
PC1
192.168.2.100
PC-PT
PC2
192.168.2.101

VLAN:







Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	Router0	PC0	ICMP		0.000	N	0	(edit)	(delete)

Date __/__/____ CN (Saathi)

Experiment 7

Aim: WAP to implement IPv4 addressing concept along with subnet.

Theory: An IP address is an address having information about how to reach a specific host, especially outside the LAN. An IP address is a 32-bit unique address having an address space of 2^{32}. Generally, there are 2 notations in which an IP address is written, dotted decimal notation & hexadecimal notation.

Classful Addressing
The 32 bit IP address is divided into 5 subclasses. These are:

1) Class A
2) Class B
3) Class C
4) Class D
5) Class E

Each of these classes has a valid range of IP addresses. Classes D and E are reserved for multicast & experimental purposes respectively. The order of bits in the first octet determines the classes of IP addresses.

IPv4 address is divided into 2 parts:

- Network ID
- Host IP

Date __/__/____

Saathi

The class of IP address is used to determine the bits used for network ID & host ID and the number of total networks & hosts possible in that particular class.
Each ISP or network administrator assigns an IP address to each device that is connected to its network.

Classful addressing
The 32 bit IP address are divided into 5 subclasses. Each of these classes has a valid range of IP addresses. Classes D & E are reserved for multicast and experimental purposes respectively. The order of bits in the first octet determines the classes of IP addresses.

1) Class A – Networks that contain a large number of hosts.
2) Class B – Networks that contain a medium to large number of hosts.
3) Class C – Networks that contain small-sized networks.
4) Class D – It is reserved for multicasting.
5) Class E – It is reserved for experimental & research

Program:

```
import math

def findClass(ip): if 0 <= ip[0] <= 127:

print("Network Address is : ", ip[0]) print('No. of IP addresses possible : ', 2 ** 24) return "A", '255.0.0.0'

elif 128 <= ip[0] <= 191: ip = [str(i) for i in ip] print("Network Address is : ", ".".join(ip[0:2])) print('No. of IP addresses possible : ', 2 ** 16) return "B", '255.255.0.0'

elif 192 <= ip[0] <= 223: ip = [str(i) for i in ip] print("Network Id is : ", ".".join(ip[0:3])) print('No. of IP addresses possible : ', 2 ** 8) return "C", '255.255.255.0'

elif 224 <= ip[0] <= 239:

print("In this Class, IP address is not divided into Network and Host ID") return "D"

else:

print("In this Class, IP address is not divided into Network and Host ID") return "E"

def Subnetting(ip, num, className, ip_addresses):

temp = 0 if className == "A":

place2 = ip_addresses / (256 ** 2) for i in range(num):

print(f"Subnet {i} => ") print(temp) print("Subnet Address : ", ip[0] + '.' + str(temp) + '.0' + '.0') temp += int(place2) print("Broadcast address : ", ip[0] + '.' + str(temp - 1) + '.255' + '.255') print("Valid range of host IP address : ", ip[0] + '.' + str(temp - int(place2)) + '.' + '0' + '.1' + '\t-\t' + ip[0] + '.' + str( temp - 1) + '.254' + '.254')

print()

elif className == "B":

place2 = ip_addresses / 256 for i in range(num):

print(f"\nSubnet {i} => ") print("Subnet Address : ", ".".join(ip[0:2]) + '.' + str(temp) + '.0') temp += int(place2) print("Broadcast address : ", ".".join(ip[0:2]) + '.' + str(temp - 1) + '.255') print("Valid range of host IP address : ",
```

".".join(ip[0:2]) + '.' + str(temp - int(place2)) + '.1\t-\t' + ".".join(ip[0:2]) + '.' + str(temp - 1) + '.254') print()

elif className == "C": for i in range(num):

print(f"\nSubnet {i} => ") print("Subnet Address : ", ".".join(ip[0:3]) + '.' + str(temp)) temp += int(ip_addresses) print("Broadcast address : ", ".".join(ip[0:3]) + '.' + str(temp - 1)) print("Valid range of host IP address : ",

".".join(ip[0:3]) + '.' + str(temp - int(ip_addresses) + 1) + '\t-\t' + ".".join(ip[0:3]) + '.' + str(temp - 2)) print()

else:

print("In this Class, IP address is not divided into Network and Host ID")

def subnetmask(num, network_mask): var = '1' * int(math.log(num, 2)) var1 = '0' * (8 - int(math.log(num, 2))) binary_num = var + var1 network_mask = network_mask.split('.') network_mask = [i for i in network_mask if i != '0'] network_mask.append(str(int(binary_num, 2))) while len(network_mask) < 5: network_mask.append('0') print('Subnet Mask – ', ".".join(network_mask[0:4]))

ip = input("Enter the IP address : ") ip = ip.split(".") ip = [int(i) for i in ip]

lst = findClass(ip) networkClass = lst[0] print("Given IP address belongs to class : ", networkClass)

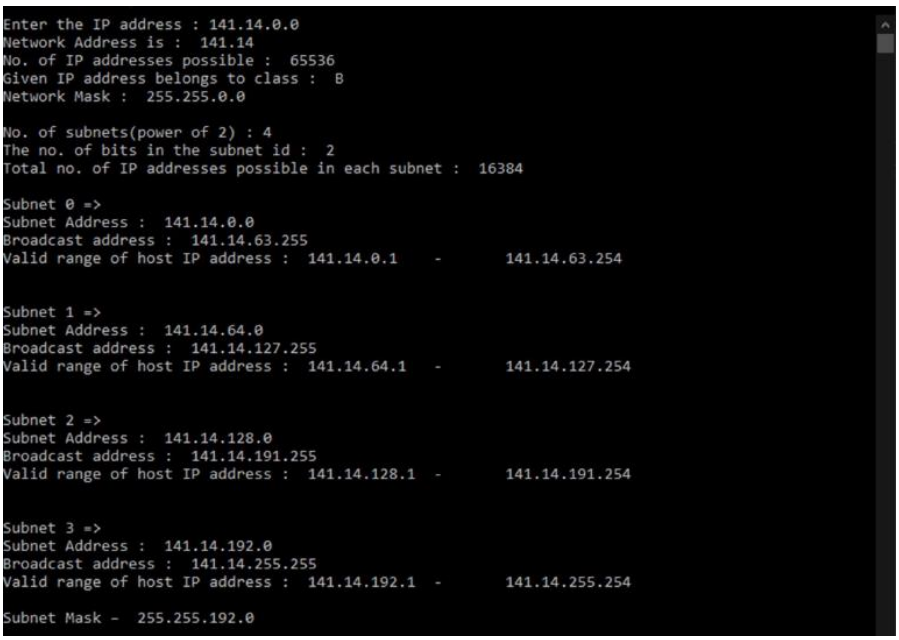
ip = [str(i) for i in ip]

network_mask = lst[1] print('Network Mask : ', network_mask) num_subnet = int(input('\nNo. of subnets(power of 2) : ')) num_ip = int(2 ** (8 * (68 - ord(networkClass))) / num_subnet) print('The no. of bits in the subnet id : ', int(math.log(num_subnet, 2))) if ord(networkClass) < 68:

print('Total no. of IP addresses possible in each subnet : ', num_ip)

Subnetting(ip, num_subnet, networkClass, num_ip) subnetmask(num_subnet, network_mask)

Output:



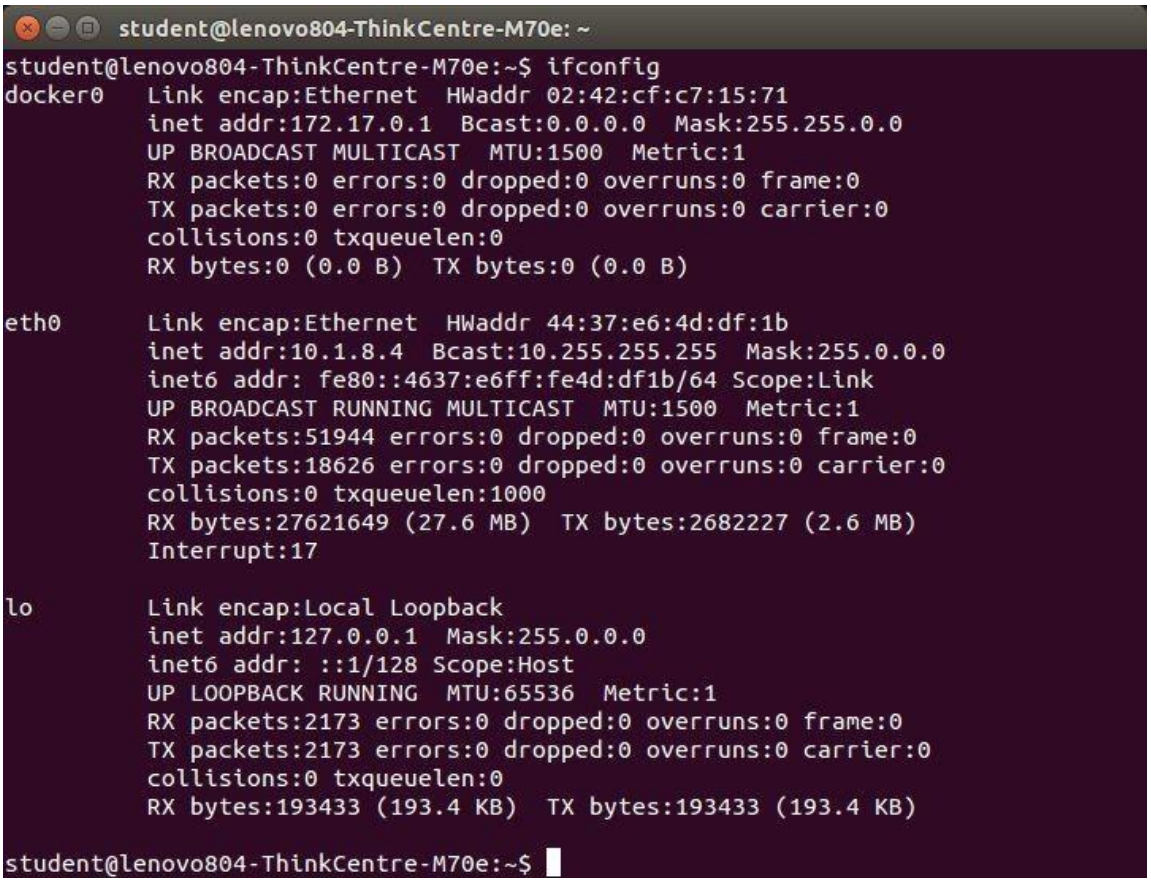
EXPERIMENT NO.8

AIM: Use basic networking commands in Linux (ping, tracert, nslookup, netstat, ARP, RARP, ip, ifconfig, dig, route)

THEORY:

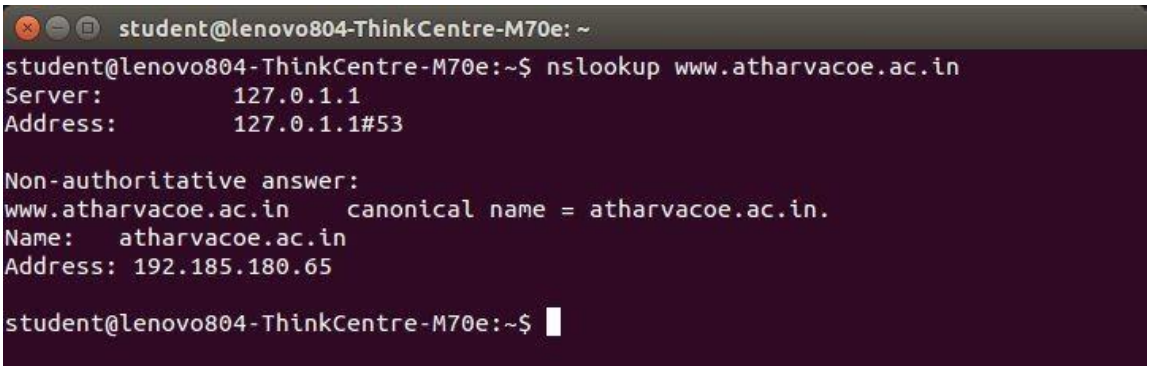
1. **ifconfig**

ifconfig(interface configuration) command is used to configure the kernel-resident network interfaces. It is used at the boot time to set up the interfaces as necessary. After that, it is usually used when needed during debugging or when you need system tuning. Also, this command is used to assign the IP address and netmask to an interface or to enable or disable a given interface.



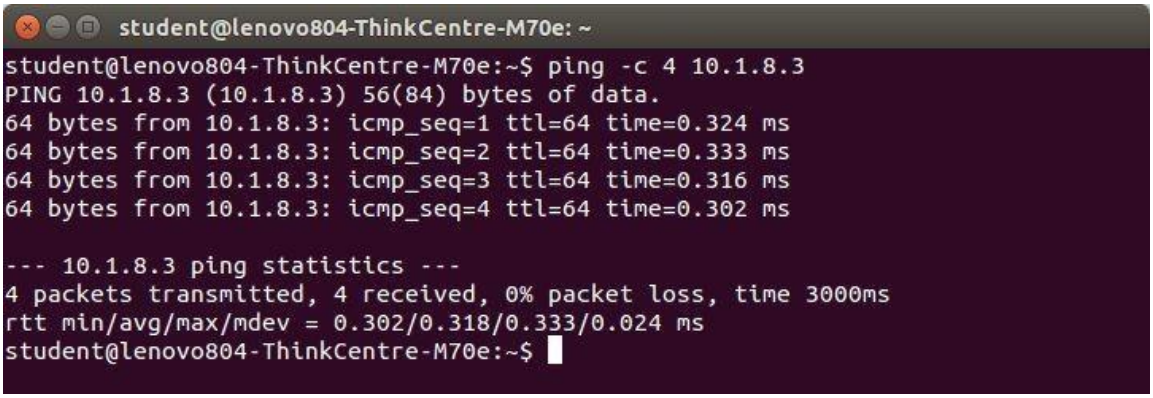
2. **NSLOOKUP**

Nslookup (stands for “Name Server Lookup”) is a useful command for getting information from DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record. It is also used to troubleshoot DNS related problems.



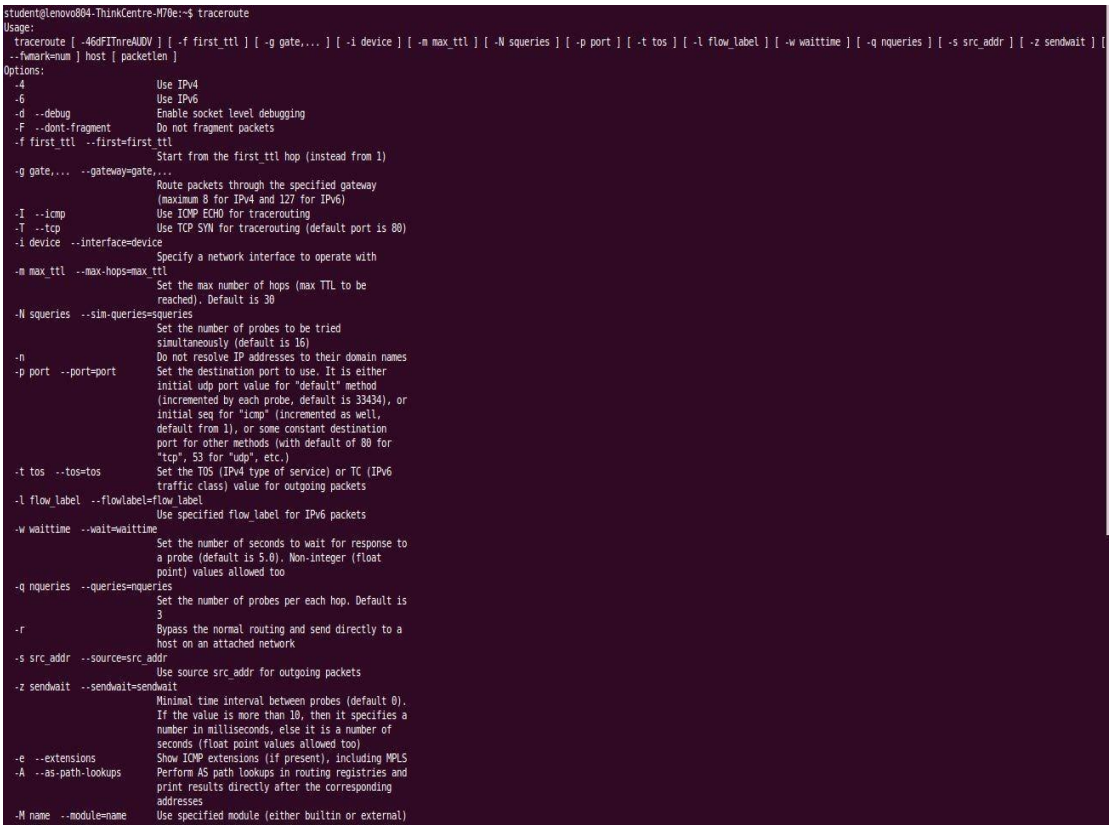
3. **Ping**

PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host. This command takes as input the IP address or the URL and sends a data packet to the specified address with the message “PING” and get a responsefrom the server/host this time is recorded which is called latency. Fast ping low latency means faster connection. Ping uses ICMP(Internet Control Message Protocol) to send an ICMP echo message to the specified host if that host is available then it sends ICMP reply message. Ping is generally measured in millisecond every modern operating system has this ping pre-installed.



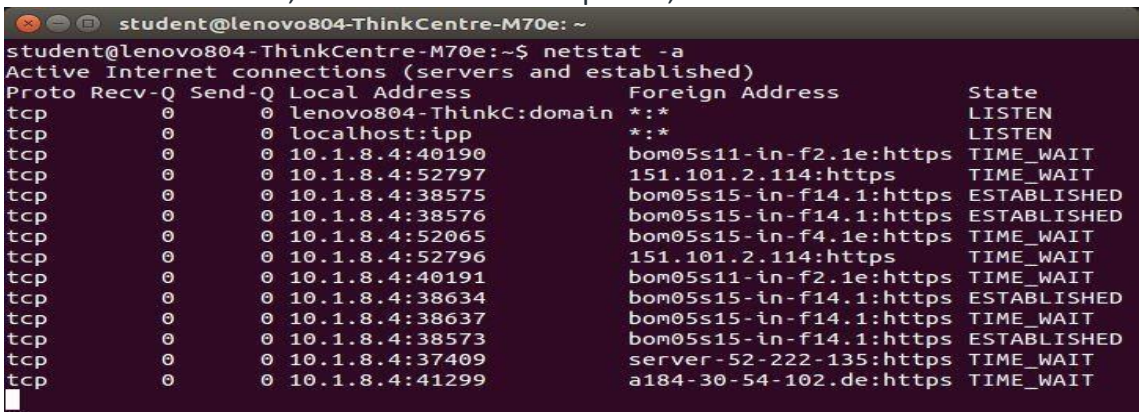
4. TRACEROUTE

traceroute command in Linux prints the route that a packet takes to reachthe host. This command is useful when you want to know about the route and about all the hops that a packet takes. Below image depicts how traceroute command is used to reach the Google(172.217.26.206) host from the local machine and it also prints detail about all the hops that it visits in between.



5. Netstat

Netstat command displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc.,

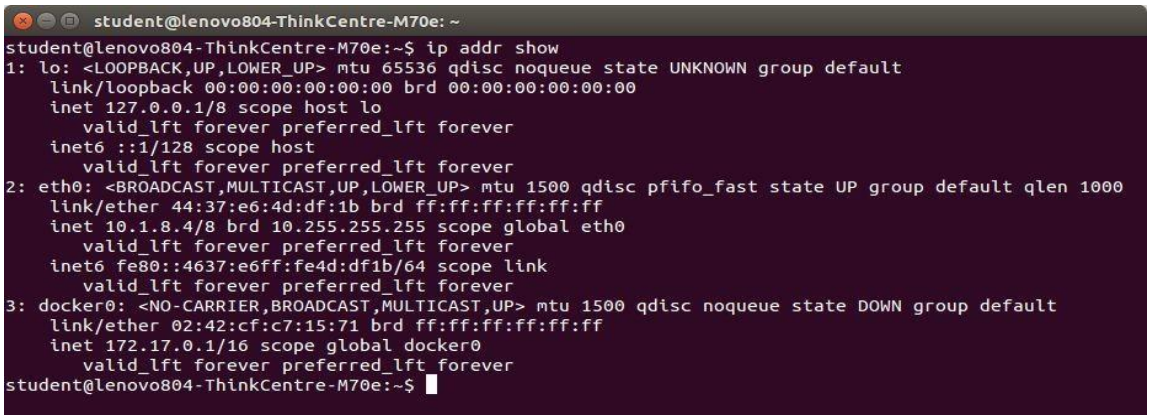


6. ARP

arp command manipulates the System's ARP cache. It also allows a complete dump of the ARP cache. ARP stands for Address Resolution Protocol. The primary function of this protocol is to resolve the IP address of a system to its mac address, and hence it works between level 2(Data link layer) and level 3(Network layer).

7. IP

ip command in Linux is present in the net-tools which is used for performing several network administration tasks. IP stands for Internet Protocol. This command is used to show or manipulate routing, devices, and tunnels. It is similar to *ifconfig* command but it is much more powerful with more functions and facilities attached to it. *ifconfig* is one of the deprecated commands in the

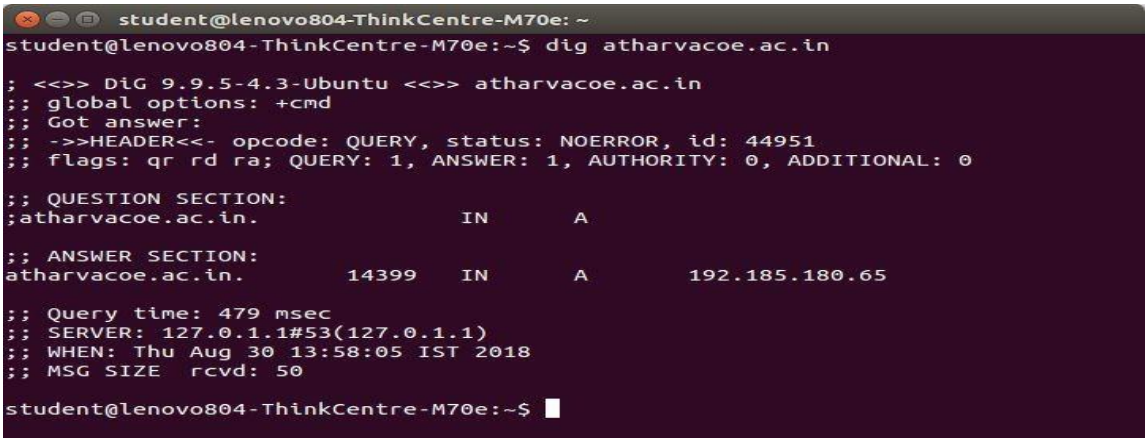


net-

tools of Linux that has not been maintained for many years. ip command is usedto perform several tasks like assigning an address to a network interface or configuring network interface parameters.
It can perform several other tasks like configuring and modifying the default andstatic routing, setting up tunnel over IP, listing IP addresses and property information, modifying the status of the interface, assigning, deleting and settingup IP addresses and routes.

8. **Dig**

   dig command stands for ***Domain Information Groper***. It is used for retrieving information about DNS name servers. It is basically used by network administrators. It is used for verifying and troubleshooting DNS problems and to perform DNS lookups. Dig command replaces older toolssuch as nslookup and the host.



CONCLUSION: Hence, in this experiment, we have successfully studied some important networking command and also implemented them in Linux

EXPERIMENT NO.9

AIM: Use Wire shark to understand the operation of TCP/IP layers:

- Ethernet Layer: Frame header, Frame size etc.
- Data Link Layer: MAC address, ARP (IP and MAC address binding)
- Network Layer: IP Packet (header, fragmentation), ICMP (Query and Echo)
- Transport Layer: TCP Ports, TCP handshake segments etc.

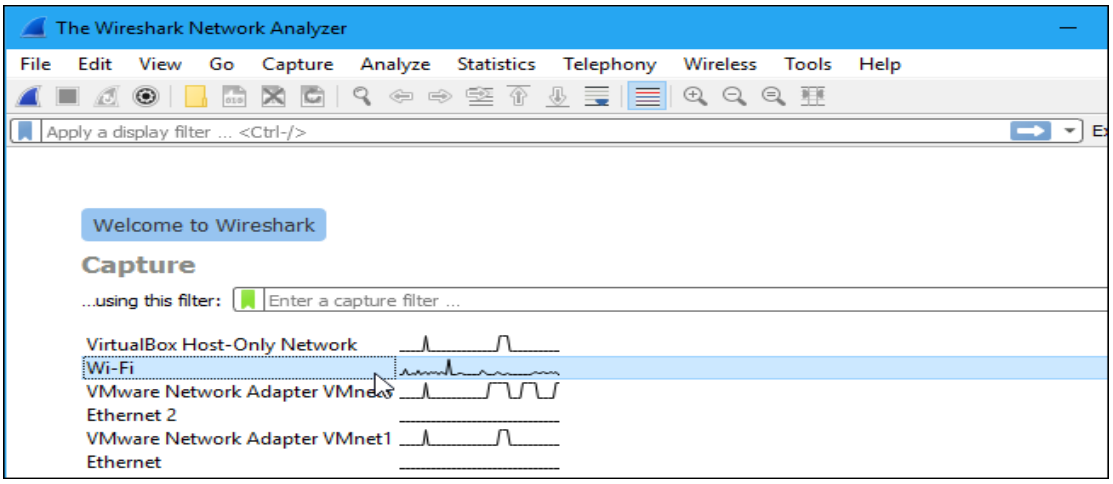
Application Layer: DHCP, FTP, HTTP header formats

THEORY:

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other features thatlet you dig deep into network traffic and inspect individual packets.

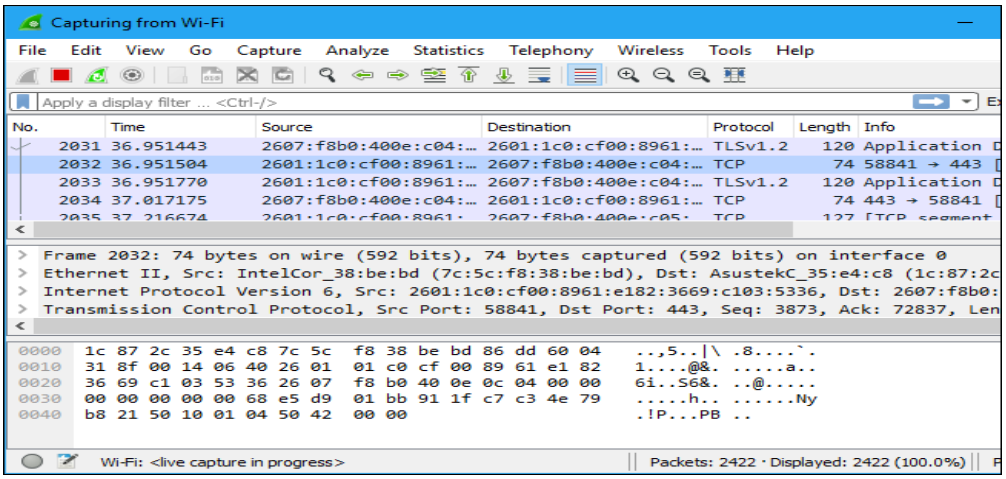
Capturing Packets

After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface. You can configure advanced features by clicking Capture > Options, but this isn't necessary for now.

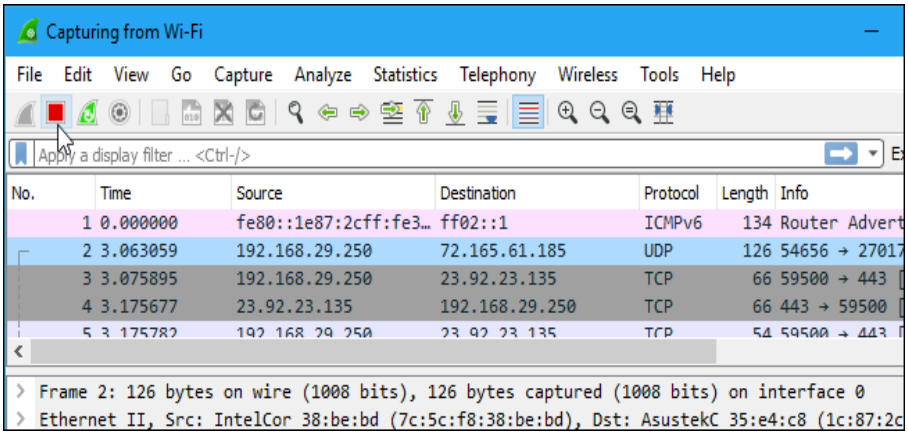


As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the "Enable promiscuousmode on all interfaces" checkbox is activated at the bottom of this window.



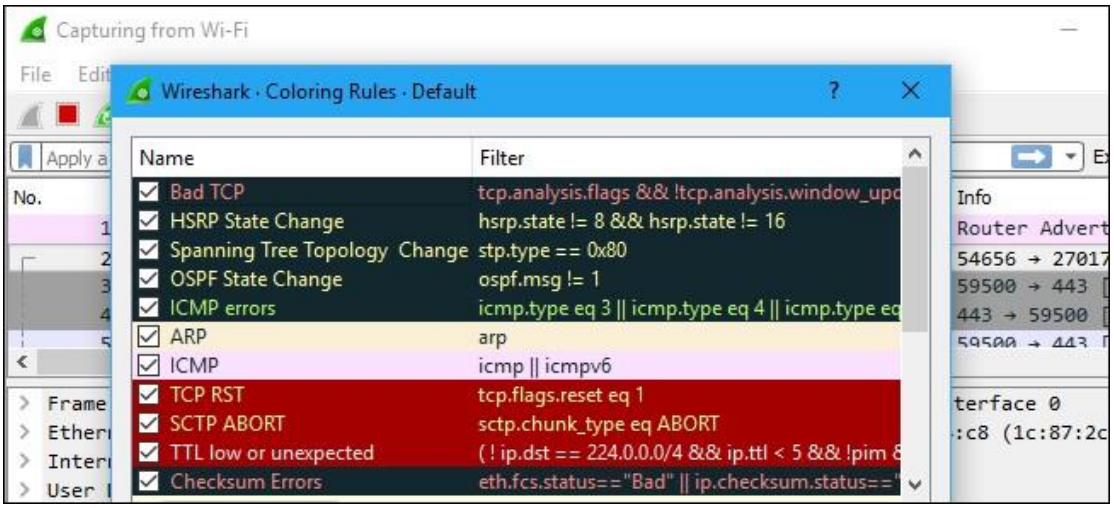
Click the red “Stop” button near the top left corner of the window when you want to stop capturing traffic.



Color Coding

You’ll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order.

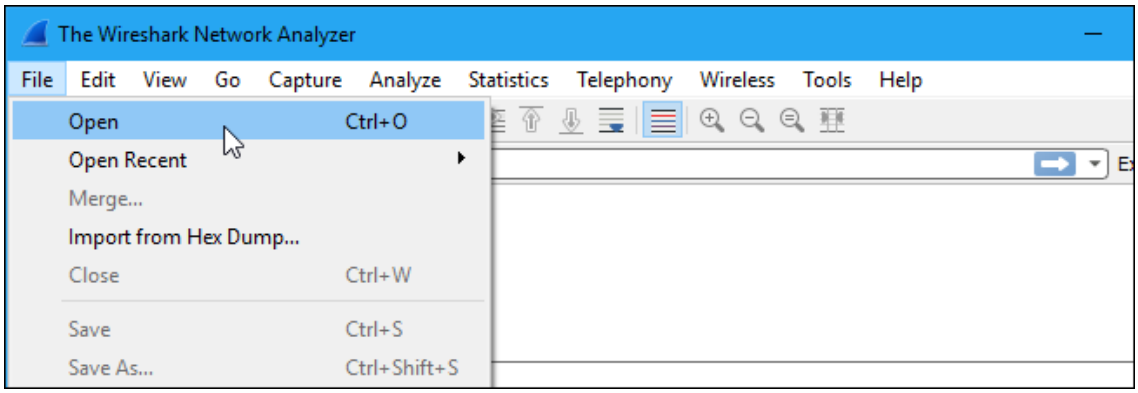
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.



Sample Captures

If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

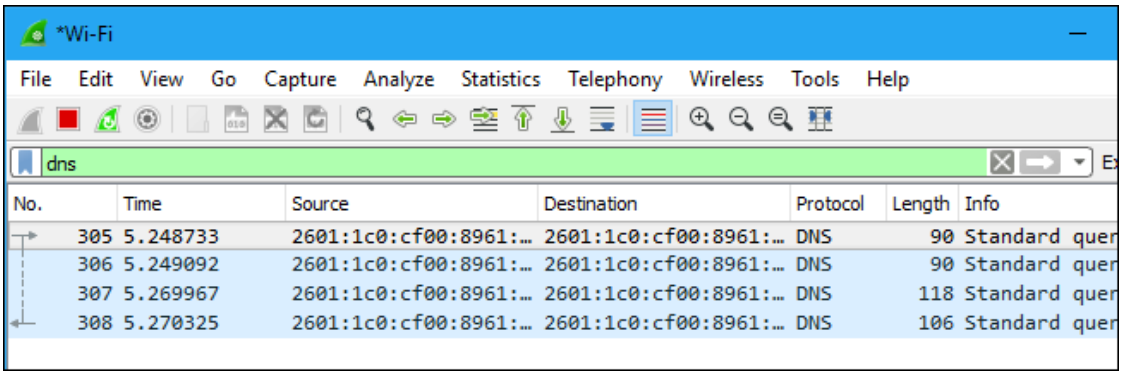
You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.



Filtering Packets

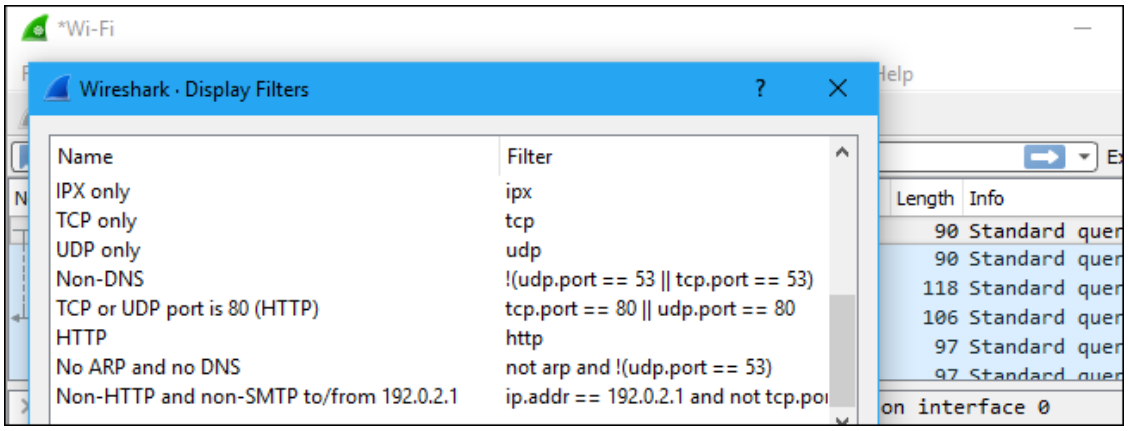
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type "dns" and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.



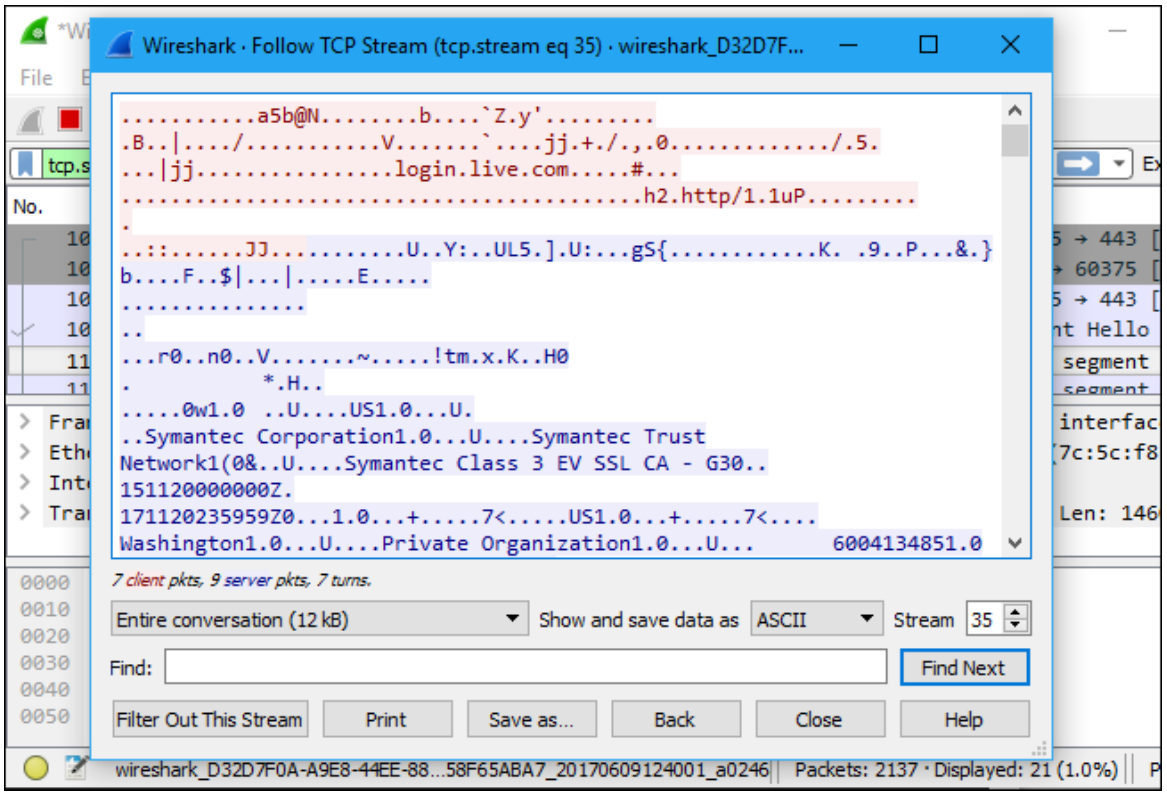
You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

For more information on Wireshark's display filtering language, read the [Building display filter expressions] page in the official Wireshark documentation.

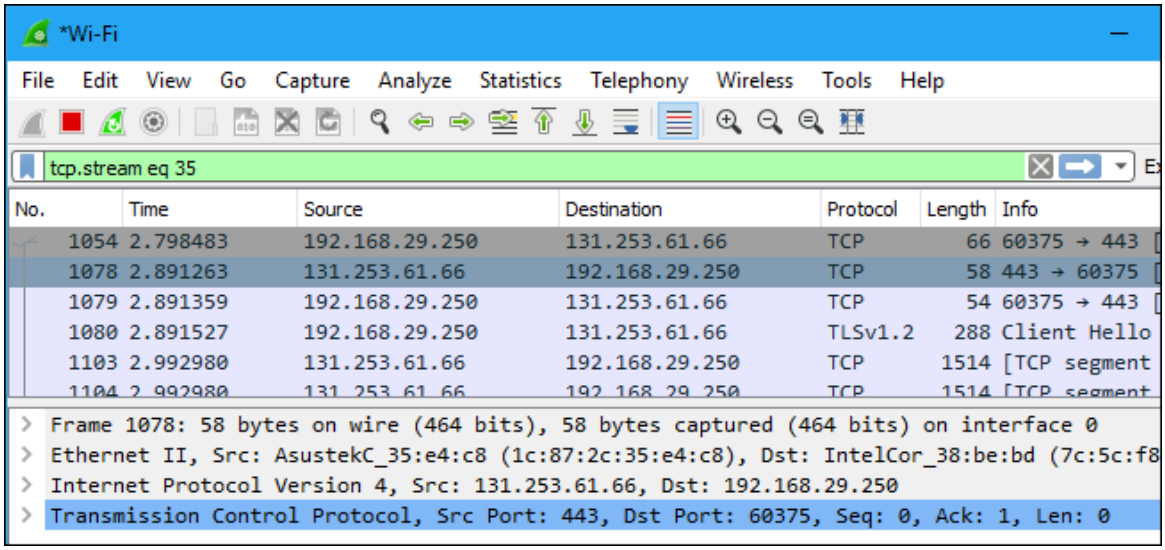


Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

You'll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.

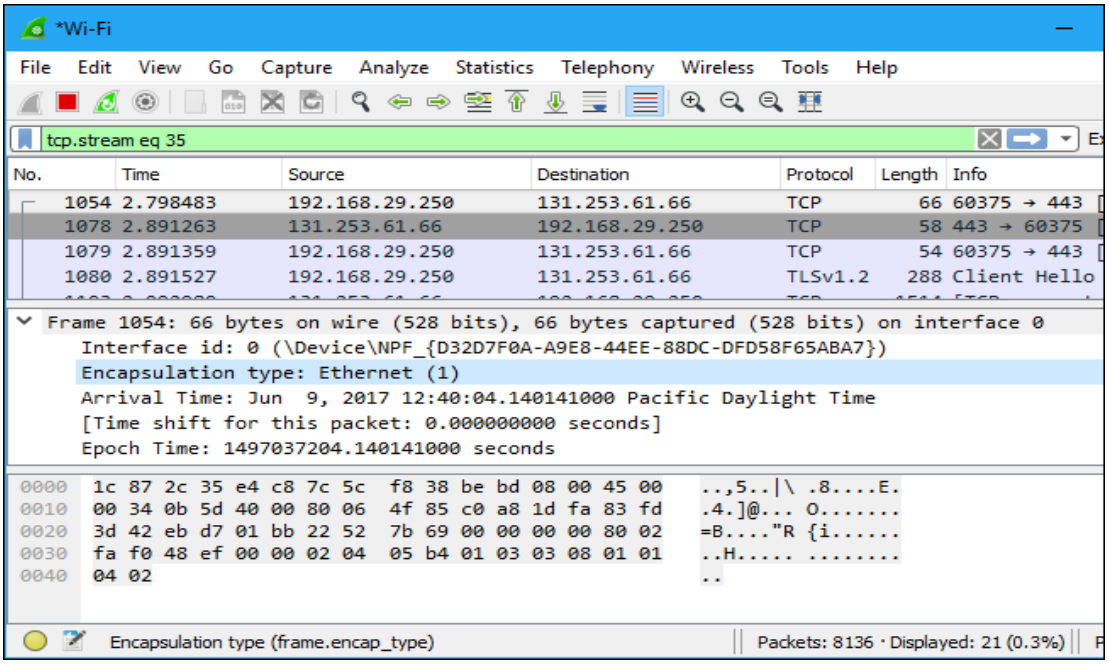


Close the window and you'll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.

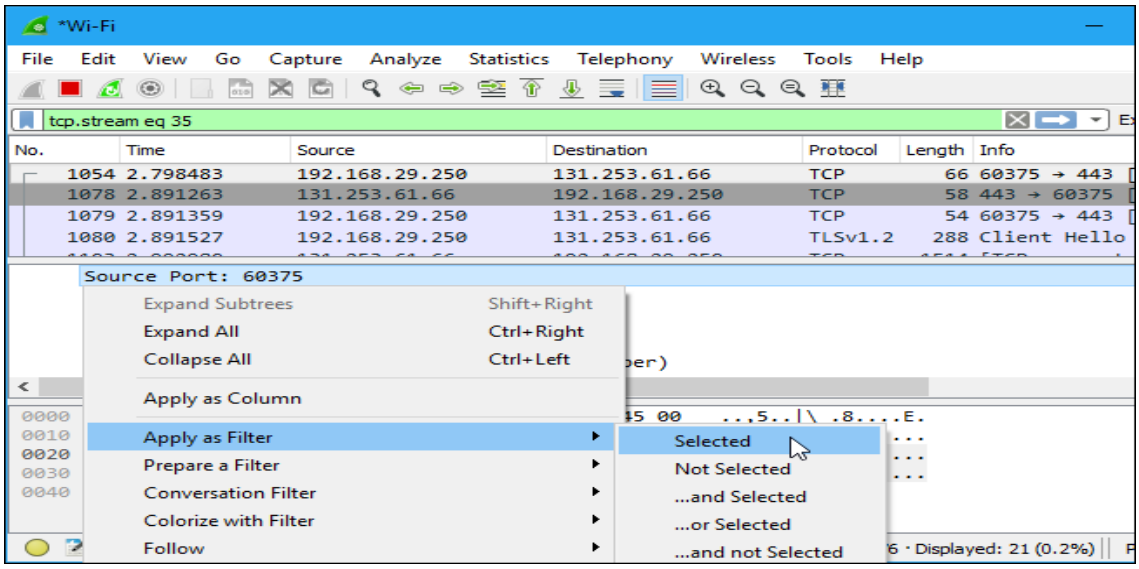


Inspecting Packets

Click a packet to select it and you can dig down to view its details.



You can also create filters from here — just right-click one of the details and use the Applyas Filter submenu to create a filter based on it.



Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

CONCLUSION: Thus, we have studied the working of Wire Shark

EXPERIMENT
NO.10

AIM: Java program for Socket Programming

THEORY:
Java Socket Programming

- Java Socket programming is used for communication between the applications runningon different JRE.
- Java Socket programming can be connection-oriented or connection-less.
- Socket and ServerSocket classes are used for connection-oriented socket programmingand DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

a. IP Address of Server, and
b. Port number.

Here, we are going to make one-way client and server communication. In this application, clientsends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket.

The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method ofServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

#Socket class

A socket is simply an endpoint for communications between the machines. The Socketclass can be used to create a socket.

#ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used toestablish communication with the clients.

Creating Server:

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may alsochoose any other port number. The accept() method waits for the client. If clients connects withthe given port number, it returns an instance of Socket.

```
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection and waits for the client
```

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
Socket s=new Socket("localhost",6666);
```

Code:

MyServer.java file

```
import
java.io.*;
import
java.net.*;
public class
MyServer
```

```
public static void main(String[]
args){try
{
        ServerSocket ss=new
        ServerSocket(6666); Socket
        s=ss.accept();//establishes
        connection
        DataInputStream dis=new
        DataInputStream(s.getInputStream());String
        str=(String)dis.readUTF();
        System.out.println("m
        essage= "+str);
        ss.close();
        }
        catch(Exception e){System.out.println(e);}
    }
}
```

MyClient.java file

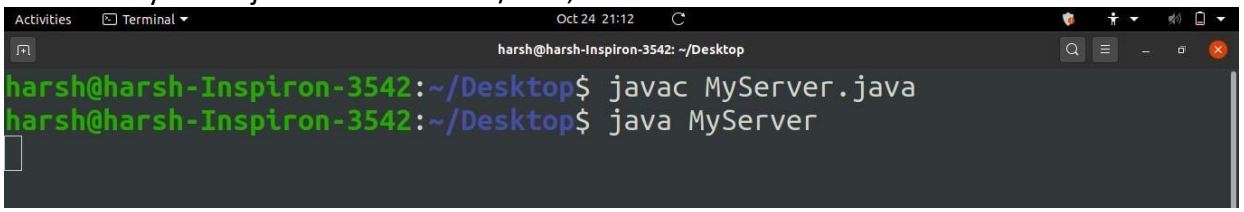
```
import
java.io.
*;
import
java.ne
t.*;
public
class
MyClie
nt
{
    public static void main(String[] args)
    {
        try
        {
        Socket s=new Socket("localhost",6666);
```

```
                    DataOutputStream dout=new
                    DataOutputStream(s.getOutputStream());
                    dout.writeUTF("Hello Server");
                    dout.flush();
                    dout.close();
                    s.close();
                    }catch(Exception e){System.out.println(e);}
                }
        }
```

Output:

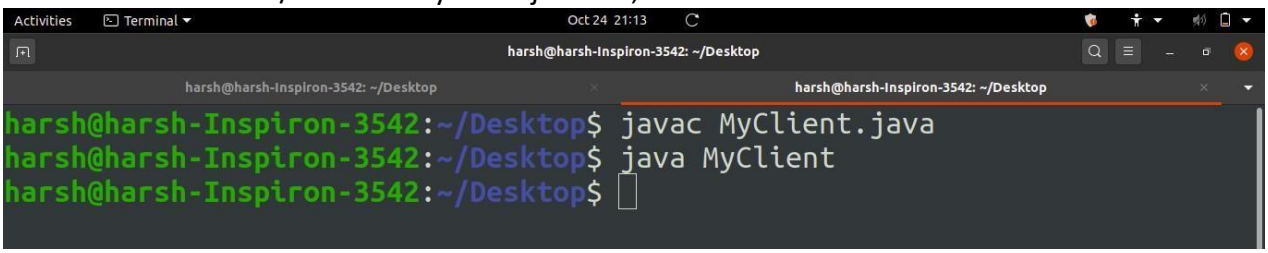
To execute this program open two command prompts and execute each program at eachcommand prompt as displayed in the below figures.

First run Myserver.java file in terminal/cmd,



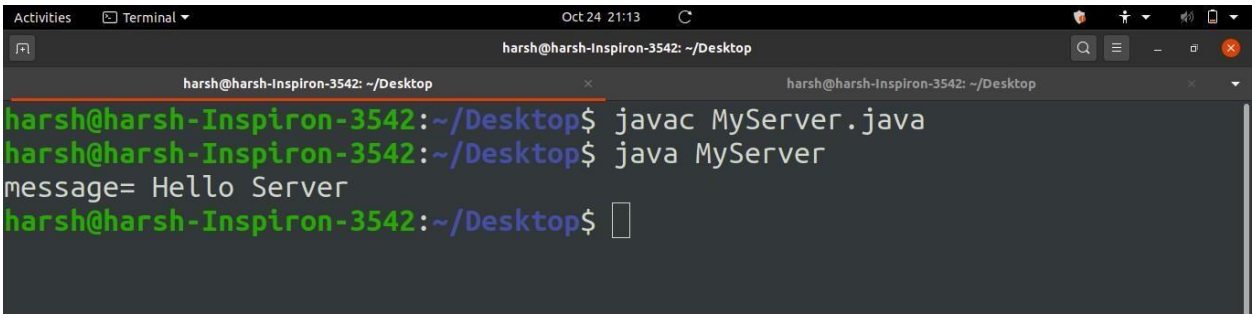
Running MyServer.java

Then in new terminal/cmd run MyClient.java file,



Running MyClient.java

As soon as you run MyClient program a message is sent to server anddisplayed in MyServer Terminal/CMD as shown below,



Message displayed in MyServer after running MyClient

CONCLUSION: So, in this experiment we have successfully understood the concept of Socket Programming and implemented it using Java Programming

DWM ASSIGNMENT - 01

Q. What is metadata? Explain data warehouse metadata with example.

A → • Metadata is simply defined as data about data. The data that is used to represent other data is known as metadata.

Eg:- The index of a book serves as metadata for the contents in the book. In other words, we can say that metadata is the summarized data that leads us to detailed data.

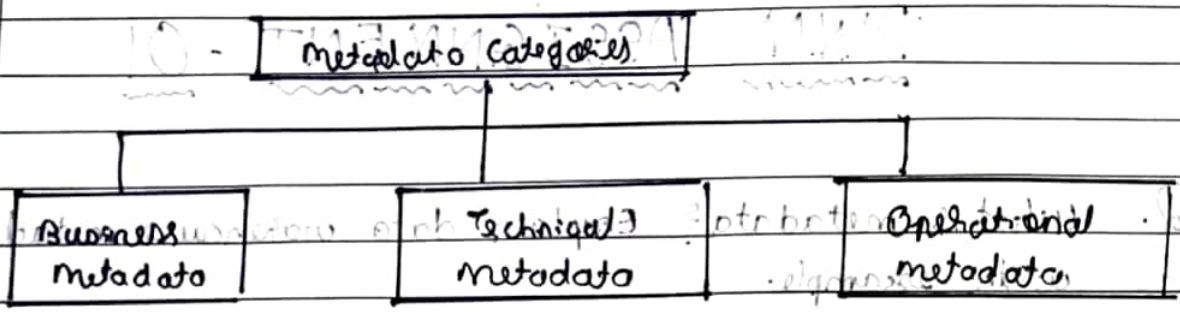
• In terms of data warehouse, we create metadata for data names and definitions of the given data. It acts like a roadmap to the data warehouse and defines the warehouse objects. It also acts as a directory which helps in the decision support system to locate the contents of a data warehouse.

• Metadata can be categorized into 3 categories:

⇒ Business metadata:- It has the data ownership information, business definition and changing policies.

⇒ Technical metadata:- It includes database system names, table and column names and sizes, data types and allowed values. Technical metadata also includes structural info like primary, foreign keys, attributes and indices.

⇒ Operational Metadata:- Includes currency of data and data lineage. Currency of data means whether the data is active, archived or purged.



- Role of Metadata :-

=> It acts as a directory.

=> This directory helps the decision support system to locate the contents of the data warehouse.

=> metadata helps in decision support system for mapping of data when data is transformed from operational environment to data warehouse environment.

=> It helps in summarization between current detailed data and highly summarized data.

=> metadata also helps in summarization between lightly detailed data and highly summarized data.

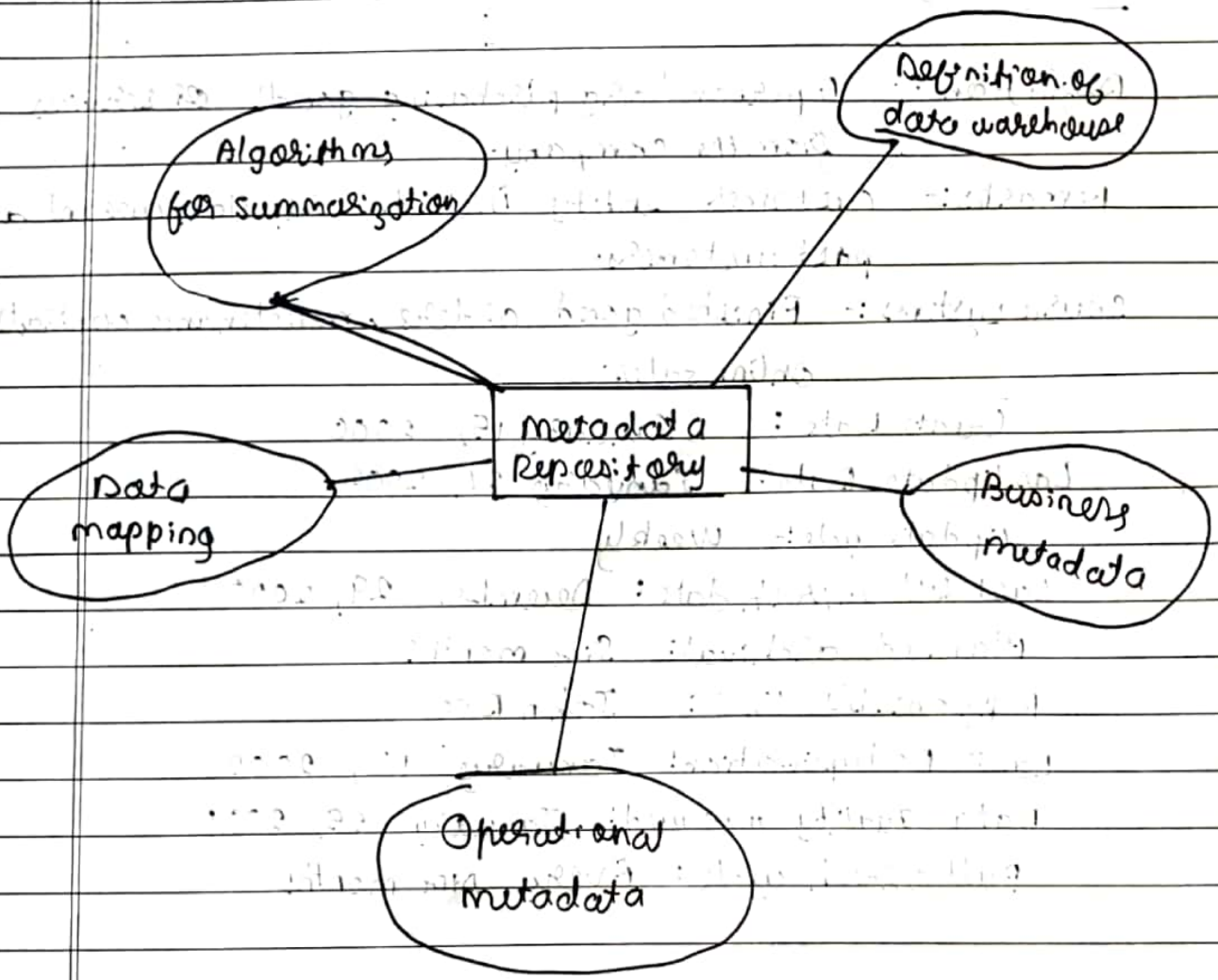
=> metadata is used for query tools.

=> metadata is used in extraction and cleaning tools.

=> metadata is used for transformation tools.

=> Plays an important role in loading functions.

Metadata Repository is an integral part of a data warehouse system. It has following metadata:-



Example:

Metadata for a 'Customer' entity:

Definition :- A person / org purchasing goods or services from the company.

Remarks :- Customer entity includes regular, current and past customers.

Source systems :- Finished good orders, maintenance contracts online sales.

Create Date : January 15, 2000

Last Update Date : January 21, 2002.

Update Cycle :- Weekly

Last full refresh date : December 29, 2000

Planned archival : Six months

Responsible User : John Doe

Last Deduplication : January 10, 2002

Data quality reviewed : January 25, 2001

Full refresh cycle : Every six months

Written Assignment 2

Q) Write short note on
(i) Ethernet
(ii) IPVG
(iii) SSH

(i) Ethernet:
Ethernet is a family of wired computer networking technologies commonly used in LAN, MAN and WAN.
Ethernet is widely used in homes and industry and interworks well with wireless wifi and interworks well with wireless wifi technologies. The internet protocol is commonly carried over ethernet and so it is considered one of the key technologies that make up the internet.

Ethernet is covered by IEEE 802.3 standard that defines what is commonly known as CSMA/CD protocol. Ethernet is currently by used for approximately 85% of the world's LAN connected PCs and workstations. Ethernet is the major LAN technology because of the following characteristics:
- It is easy to understand, implement, manage and maintain.
- It allows cost network implementations
- Provides extensive topological flexibility

for network installation
- Guarantees successful interconnection and operation of standard complient products regardless of manufacturer.

ii) IPV6

The need for IPv6 was first raised in 1990 but became more urgent by February 2011 when the Internet assigned number authority (IANA) had allocated the last 16 million IPv4 addresses to the regional internet registries. By September 2015, 4 of the 5 regional registries had run out of IPv4 addresses but used CIDR to extend IPv4 addresses as IPv6 had not been adopted yet. So this led to rise in IPv6 usage. The two protocols are not interoperable as the packet formatting is significantly different. However, most devices used little or no change to switch to IPv6. Therefore IPv6 is the next generation IP designed as successor to the IPv4. It is designed to enable high performance, scalable internet. There are 2^{128} possible ways. IPv6 is written in hexadecimal of 40 bytes in length and has only 8 fields. Source and destination addresses are 128 bits in length.

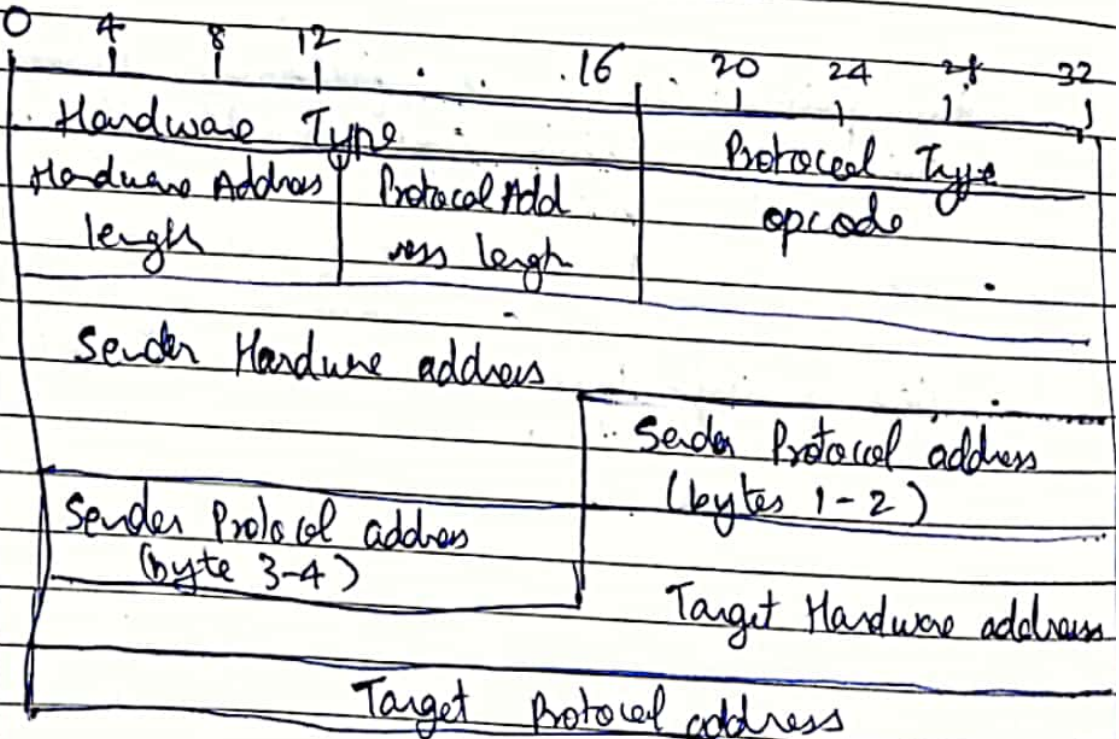
IPv6 has 3 different types of addresses:
- unicast
- anycast
- multicast

(iii) Secure shell or secure socket shell also known as SSH is access credential that is used in SSH protocol. It is a cryptographic network protocol that is used for transferring encrypted data over network. It allows you to connect to a server or multiple servers without having you to remember or enter your password for each system into another. It always comes in key pair:
- Public key: used for encryption
- Private key: used for decryption

Key pairs can be of following types:
- user key: Public and private keys will stay with user.
- Host key: Public and private keys will be on remote system.
- Session key: Used for large amount of data transmission.

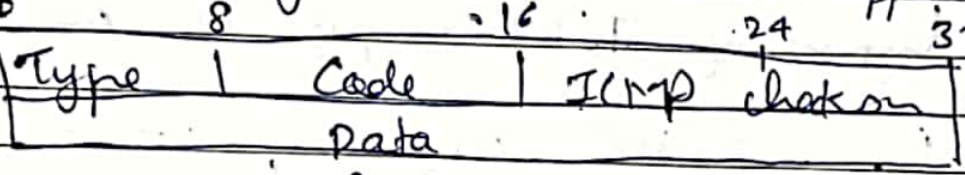
Q) Explain purpose of following protocols with their header format:
(i) ARP
(ii) ICMP
(iii) DNS

(i) ARP is a communication protocol used for discovering the link layer address, such as MAC address associated with a given internet layer address. This mapping is a critical function in the internet protocol suite. ARP has been implemented with many combinations of network and data by link layer technologies, such as IPv4, DEC Net and Xerox universal packet (PUP) using IEEE 802 standards.



ARP message packet

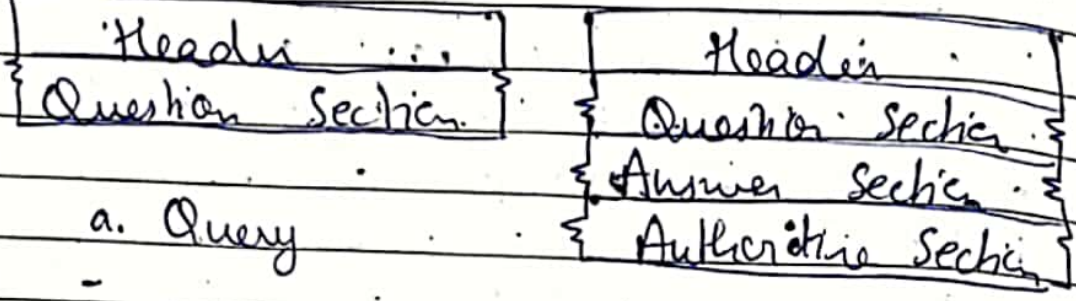
(ii) ICMP is a supporting protocol in internet protocol suite. It is used by network devices, including routers, to send error messages and operational info indicating success or failure when communicating with another IP address, for example, an error is indicated when a service is unavailable or host could not be reached. It differs from transport protocols as it is not typically used to exchange data between systems, nor it is regularly employed by end user network applications.



ICMP Header Packet

(iii) DNS is a hierarchial and decentralised naming system for computers, services or other resources connected to the internet or a private network. It associates various info with domain names assigned to each of the participating entities. The purpose of DNS is to translate a domain name into the appropriate IP address. This is done by looking up the DNS records of the request domain. There are typically steps in DNS lookup process that follow the info path from the originating web browser to the DNS server

and back again. In practice, DNS info is cached to reduce the DNS lookup response time.



Q) Discuss persistent and non-persistent protocols used transport and application layers of TCP/IP protocol suite.

→ Persistent HTTP connections are designed to allow multiple request/response exchange without establishing new connections for each. In TCP, this can considerably reduce network congestion and resource use by eliminating the need for the TCP 3-way handshake for each request one "introduction" b/w client and server is all that is needed.

Advantages:

- Lower CPU and memory usage.
- Allows for HTTP pipelining of requests and responses
- Reduced network congestion.

- Reduced latency for subsequent requests
- Errors can be reported without TCP connection termination
- Reduced round trips because of fewer new connection and TLS handshakes.

Disadvantages
- Resources may be inaccessible even when unused as connection may not be closed by the current client.

Non-persistent HTTP connections were the defacto means of communication in early HTTP implementations. It is not that this method was chosen over persistent this connection protocols rather persistent protocols did not exist yet. Modern web browsers like chrome, firefox, opera use persistent HTTP connections by default. As such, non-persistent connections are widely used in modern network applications. However, there are still some advantages

Advantages: unused resources are freed immediately after use

Disadvantages: 2 RTT's per object requested
Greater CPU overhead.