### Exp 1: Basic Networking commands

#### Linux:

* ifconfig
* ping firefox.com
* tracepath www.google.com
* traceroute google.com
* host www.firefox.com
* nslookup firefox.com
* netstat -rn
* ip addr show
* dig firefox.com

#### Windows:

* ipconfig
* ping google.com
* tracert www.lexis.com
* nslookup google.com
* netstat -tuln

#### 

### Exp 2: Simple Network Topology:

* Open .pkt file
* Select add simple Pdu from above toolbox (mail option)
* Select PC0
* Select PC1
* Run a simulation
* Pause when Successful status at bottom right corner

### Exp 3: Wireshark

* Open Wireshark
* Click on “Capture”
* Select Wifi/Ethernet
* It streams through various frames/packages
* Open any one.
* Bottom Left, you will see various descriptions such as Frame, Ethernet, Internet Protocol, User Diagram Protocol
* Expand everything one by one and present to teacher
* One for TCP and UDP Protocol.

### Exp 4: Hamming Code Implementation

* Open Online Compiler.
* **Enter:**   
  #include<iostream>

using namespace std;

int main() {

int data[10];

int dataatrec[10],c,c1,c2,c3,i;

cout<<"Enter 4 bits of data one by one\n";

cin>>data[0];

cin>>data[1];

cin>>data[2];

cin>>data[4];

//Calculation of even parity

data[6]=data[0]^data[2]^data[4];

data[5]=data[0]^data[1]^data[4];

data[3]=data[0]^data[1]^data[2];

cout<<"\nEncoded data is\n";

for(i=0;i<7;i++)

cout<<data[i];

cout<<"\n\nEnter received data bits one by one\n";

for(i=0;i<7;i++)

cin>>dataatrec[i];

c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];

c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];

c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];

c=c3\*4+c2\*2+c1 ;

if(c==0) {

cout<<"\nNo error while transmission of data\n";

}

else {

cout<<"\nError on position "<<c;

cout<<"\nData sent : ";

for(i=0;i<7;i++)

cout<<data[i];

cout<<"\nData received : ";

for(i=0;i<7;i++)

cout<<dataatrec[i];

cout<<"\nCorrect message is\n";

//if errorneous bit is 0 we complement it else vice versa

if(dataatrec[7-c]==0)

dataatrec[7-c]=1;

else

dataatrec[7-c]=0;

for (i=0;i<7;i++) {

cout<<dataatrec[i];

}

}

return 0;

}

* In output box:

**Enter:**

1

0

1

0

* When Prompted again,  
  **Enter:**

1

0

1

0

0

1

0

### 

### Exp 5: IP forwarding using ip tables in Linux

* Open terminal
* **Enter**: sudo ip addr
* Look for ens33/enp0s3 and ip address “inet 192.168.24.(whatever is over here)/24 (eg:inet 192.168.29.26/24)
* Adds an ip address **Enter**: sudo ip addr add 192.168.24.(whatever is here)+1/24 (eg: 192.168.29.27/24) dev ens33/enp0s3
* Ping above ip address **Enter:** sudo ping 192.168.24.(whatever is here)+1/24 (eg:192.168.29.27/24) **Enter**: ctrl+c to interrupt pinging.
* **Enter:** netstat -rn
* Add a new route. **Enter:** sudo route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.(check this address from ip addr)[eg:29].1 dev ens33
* Delete new route. Enter: sudo route del -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.(check this address from ip addr)[eg:29].1 dev ens33
* Add Default route. **Enter:** route add default gw 192.168.(check this address from ip addr)[eg:29].1
* Delete Default route. **Enter:** route del default gw 192.168.(check this address from ip addr)[eg:29].1

### Exp 6: Socket Programming

* Check Java Version: java -version
* Install java package. **Enter:** sudo apt install default-jdk
* Create a text file.  
  **Enter in text file:**

import java.io.\*;

import java.net.\*;

public class MyServer {

public static void main(String[] args) {

try {

ServerSocket ss = new ServerSocket(6666); // Create server socket

System.out.println("Server is running and waiting for a client...");

Socket s = ss.accept(); // Wait for client connection

System.out.println("Client connected!");

DataInputStream dis = new DataInputStream(s.getInputStream());

String str = (String) dis.readUTF(); // Read message from client

System.out.println("Message received from client: " + str);

ss.close(); // Close connection

} catch(Exception e) {

System.out.println(e);

}

}

}

**Save text file as:** MyServer.java

* Create a new text file:  
  **Enter in text file:**

import java.io.\*;

import java.net.\*;

public class MyClient {

public static void main(String[] args) {

try {

Socket s = new Socket("localhost", 6666); // Connect to server

DataOutputStream dout = new DataOutputStream(s.getOutputStream());

dout.writeUTF("Hello Server"); // Send message to server

dout.flush();

dout.close();

s.close(); // Close connection

} catch(Exception e) {

System.out.println(e);

}

}

}

**Save text file as:** MyClient.java

* **Create Class for each**.  
  In terminal. **Enter**:  
  javac MyServer.java  
  javac MyClient.java
* **Make two terminals.**
* First terminal **Enter:** java MyServer
* Second terminal **Enter:** java MyClient
* First Terminal should output:  
  Client connected!

Message received from client: Hello Server

### Exp 7: File Transfer Protocol

* Sudo Command.

**Enter:** sudo apt-get -y install lftp  
sudo apt update

sudo apt install proftpd

* **Enter:** ip addr

Note Ip address (Eg: inet 192.168.29.26/24)

* **Enter:** lftp -u (username) ip address (eg: 192.168.29.26)
* **Enter:** pwd
* **Enter:** ls
* Create following text files in your system:

ubuntu.txt

test.txt

test2.txt

test.py

* Upload single file.  
  **Enter:** put -a ubuntu.txt
* Upload multiple files  
  **Enter:** mput -a test.txt test2.txt
* Download single file  
  **Enter:** get -a test.py
* Download Multiple files

Enter: mget -a test.txt test2.txt

* Create directory

**Enter:** mkdir testdir

**Enter:** ls

* Delete directory

**Enter:** rmdir testdir

**Enter:** ls

* Delete Single file

**Enter:** rm test2.txt

* Delete Multiple files

**Enter:** mrm ubuntu.txt test.txt

* **Enter:** ls
* **Enter:** quit

### Exp 8: Telnet

* Sudo commands.

**Enter:**

sudo apt-get install telnetd

sudo apt-get install xinetd

* Restart xinetd  
  **Enter:**

sudo systemctl restart xinetd

* **Enter:** ps aux | grep telnetd
* **Enter:** ip a
* Find ip address of ens33/enp0s3 from above ip address
* **Enter:** telnet (ip address)