VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Final Assessment Test, NOVEMBER 2020

B.Tech., Fall-2020-2021

NAME	PRIYAL BHARDWAJ
REG. NO.	18BIT0272
COURSE CODE	ITE3001
COURSE NAME	DATA COMMUNICATION & COMPUTER NETWORKS
SLOT	L15+L16
FACULTY	Prof. DINAKARAN MURUGANANDAM

44	**18BIT0272**	**PRIYAL BHARDWAJ**	**A**	**E**

1. Company ANN has its branches in Chennai, Bangalore (B1), Hyderabad (B2) and Thiruvananthapuram (B3). Chennai is the headquarters (HQ), and other branches are supporting branches. All official activities are controlled by HQ. Hence HQ is supporting the company activities as Server. B1, B2 and B3 are clients.

a. How you will check the availability of one node in HQ from B1. Implement it with suitable network command. Demonstrate the options of the same.

e. Given the dataword 1011100110 and the divisor 00101 in HQ, it is transferring the data to B1

i. Generate the codeword at the HQ (using binary division).

ii. Check the codeword at the B1 (assume no error).

Q.a. How you will check the availability of one node in HQ from B1. Implement it with suitable network command. Demonstrate the options of the same.

Ans.

DCCN LAB FAT LIS-16 PRIYAL BHARDWAJ 18BIT0272

Q. a

AIM: To check availability of one node in HQ from B1, by implementing suitable network command.

Algorithm:

The suitable network command for our aim is "Ping".
It is used to test the ability of one network host to communicate with another.
We can simply enter the "Ping" command followed by the name or the IP address of the destination network host.

Ping test & traceroute tools to ~~step~~ test network connectivity between 2 hosts. It performs basic test to determine if a remote host is available, while traceroute test the complete route network packets take from one host to another. Traceroute is especially helpful for diagnosing where network slows down & congestion occurs

Ping is used diagnostically that lets you notify that a particular IP address/host computer exists in network & can accept requests. It is used to send a very small packet to a TCP/IP device to determine if working connection between hosts exists.

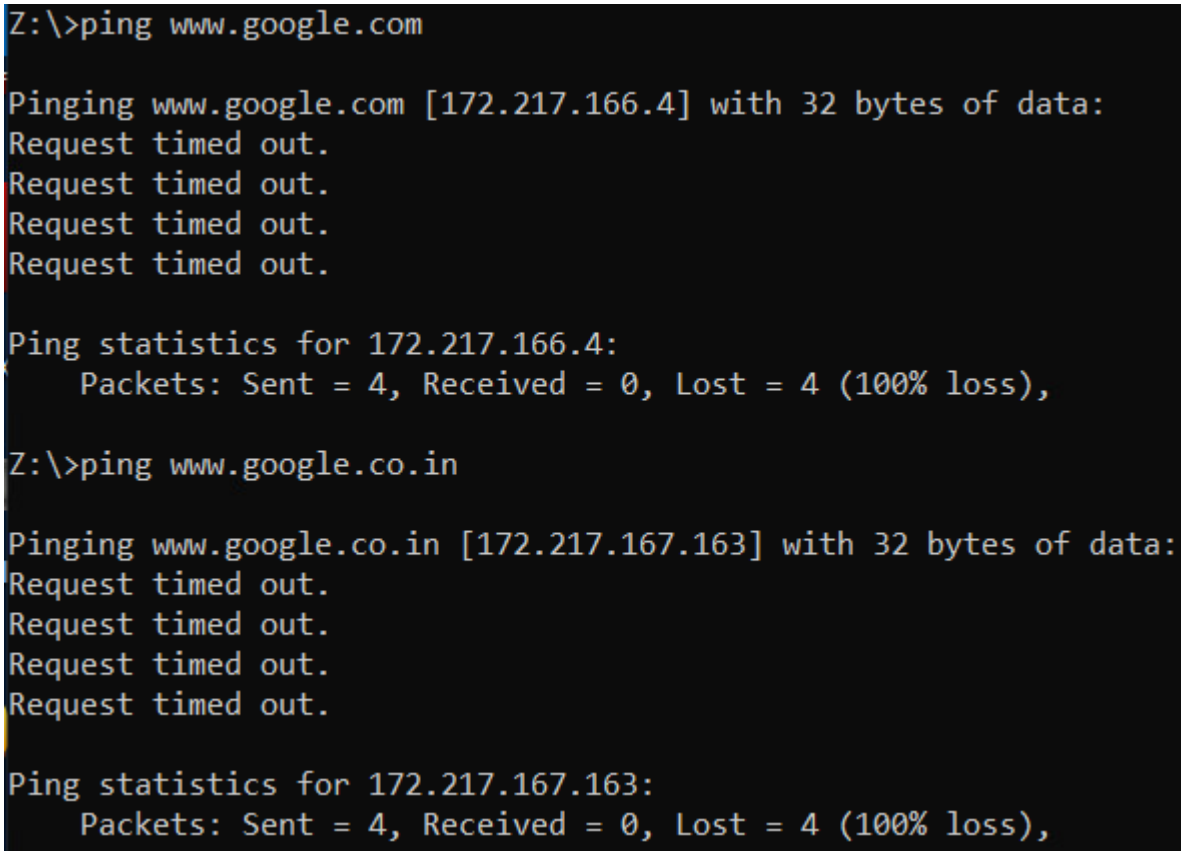
Various options of ping command
-t Ping the specified host until stopped.
To see statistics and continue - type Control-Break;
To stop - type Control-C.
-a Resolve addresses to hostnames.
-n count Number of echo requests to send.
-l size Send buffer size.
-f Set Don't Fragment flag in packet.

-i TTL Time To Live. (Max -255)
-v TOS Type Of Service.
-r count Record route for count hops.(min 1 & max 9)
-s count Timestamp for count hops.(min 1 & max 4)
-j host-list Loose source route along host-list.
-k host-list Strict source route along host-list.
-w timeout Timeout in milliseconds to wait for each reply

```
Z:\>ping www.vit.ac.in

Pinging vit.ac.in [10.10.1.75] with 32 bytes of data:
Reply from 10.10.1.75: bytes=32 time<1ms TTL=61
Reply from 10.10.1.75: bytes=32 time<1ms TTL=61
Reply from 10.10.1.75: bytes=32 time<1ms TTL=61
Reply from 10.10.1.75: bytes=32 time<1ms TTL=61

Ping statistics for 10.10.1.75:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

Z:\>ping intranet.vit.ac.in

Pinging intranet.vit.ac.in [10.10.1.61] with 32 bytes of data:
Reply from 10.10.1.61: bytes=32 time<1ms TTL=61
Reply from 10.10.1.61: bytes=32 time<1ms TTL=61
Reply from 10.10.1.61: bytes=32 time<1ms TTL=61
Reply from 10.10.1.61: bytes=32 time<1ms TTL=61

Ping statistics for 10.10.1.61:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

www.vit.ac.in and intranet.vit.ac.in are available since we get a reply.

PTO



www.google.com and www.google.co.in are not available since we do not get a reply.

PTO

Q.e. Given the data-word 1011100110 and the divisor 00101 in HQ, it is transferring the data to B1
i. Generate the codeword at the HQ (using binary division).
ii. Check the codeword at the B1 (assume no error)

Ans.

Q.e

AIM: To generate & check codeword from dataword using CRC algorithm.

Algorithm (for encoding)

- Sender & receiver agree on size of message (M(x)) & generator polynomial (G(x)).
- If r is the order of G(x), r bits are appended to the low order end of M(x). This makes block size bits, the value of which is $x^r M(x)$.
- The block $x^r M(x)$ is divided by G(x) using %. 2 division.
- The remainder is added to $x^r M(x)$ using Modulo 2 addition.
- Result is framed to be transmitted
- The encoding procedure makes exactly divisible by G(x).

Algorithm (for decoding)

- Receiver divides the incoming data frame T(x) unit by G(x) using %.2 - division.
- If no remainder then data frame is accepted.
- If remainder is there then presence of error is confirmed & data frame is rejected.

Server Code – CRCServer.py

```
import socket
s = socket.socket()
host = socket.gethostname()
port = 8888
s.bind((host, port))
s.listen(5)
```

```
c, addr = s.accept()
print('Connection received from '+str(addr))
dataword = c.recv(1024).decode()
print('Dataword received:'+dataword)
divisor = c.recv(1024).decode()
#Division
divident = dataword
main_divident = []
main_divisor = []
divident1 = int(divident)
divisor1 = int(divisor)
l = 0
while divident1 > 0:
    x = int(divident1 % 10)
    main_divident.append(x)
    divident1 = int(divident1/10)
    l = l+1
while divisor1 > 0:
    x = int(divisor1 % 10)
    main_divisor.append(x)
    divisor1 = int(divisor1/10)
main_divident.reverse()
main_divident1 = main_divident
main_divisor.reverse()
#print(main_divident)
#print(main_divisor)
lm = len(main_divisor)
l = l-3
for i in range(l):
    if(main_divident[i] == 1):
        k = i
        for j in range(lm):
            main_divident[k+j] = main_divident[k+j] ^ main_divisor[j]
syndrome = []
for i in range(3):
    syndrome.append(main_divident[l+i])
print(syndrome)
```

Client Code – CRCClient.py

```
import socket
s = socket.socket()
host = socket.gethostname()
port = 8888
s.connect((host, port))
dataword = int(input('Dataword:'))
divisor = int(input('Divisor:'))
divident = int(dataword*1000)
```

```
main_divident = []
main_divisor = []
divident1 = int(divident)
divisor1 = int(divisor)
l = 0
while divident1 > 0:
    x = int(divident1 % 10)
    main_divident.append(x)
    divident1 = int(divident1/10)
    l = l+1
while divisor1 > 0:
    x = int(divisor1 % 10)
    main_divisor.append(x)
    divisor1 = int(divisor1/10)
main_divident.reverse()
main_divident1 = main_divident
main_divisor.reverse()
print(main_divident)
print(main_divisor)
lm = len(main_divisor)
l = l-3
for i in range(l):
    if(main_divident[i] == 1):
        k = i
        for j in range(lm):
            main_divident[k+j] = main_divident[k+j] ^ main_divisor[j]
for i in range(3):
 dataword = dataword*10 + main_divident[l+i]
print('Dataword to be Sent:' + str(dataword))
s.send(str(dataword).encode())
s.send(str(divisor).encode())
```

```
C:\Users\PRIYAL BHARDWAJ\Desktop\DA's\DCCN>python CRCServer.py
Connection received from ('192.168.1.12', 53734)
Dataword received:1011100110000
[0, 0, 0]
```

```
C:\Users\PRIYAL BHARDWAJ\Desktop\DA's\DCCN>python CRCClient.py
Dataword:1011100110
Divisor:00101
[1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0]
[1, 0, 1]
Dataword to be Sent:1011100110000
```

****