Computer Networks Lab Report – Assignment 3

# TITLE

**Name –** Sourav Dutta

**Roll –** 001610501076

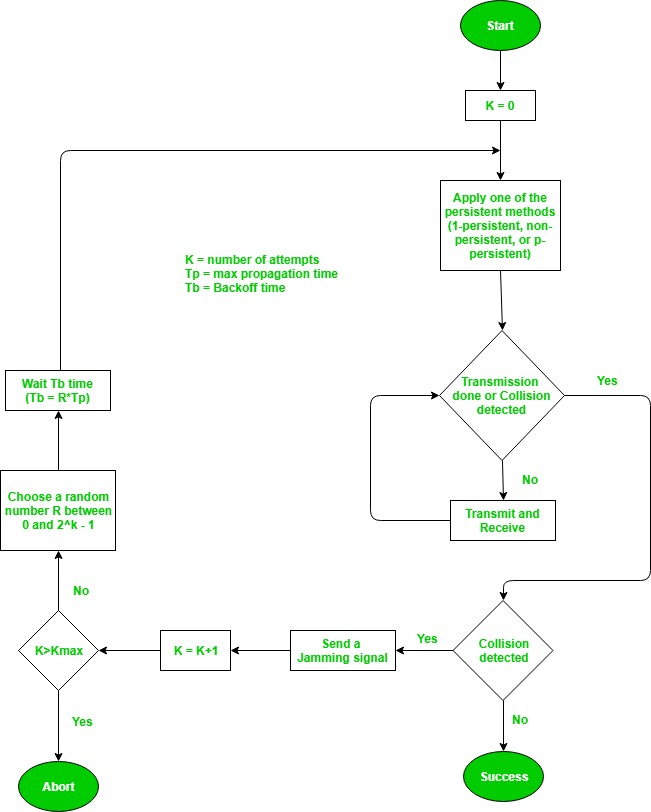
**Class –** BCSE 3rd year **Group –** A3 **Assignment Number – 3**

## Problem Statement – Implement p-persistent CSMA and CSMA/CD.

In this assignment, you have to implement p-persistent CSMA with exponential backoff and additive backoff. Measure the performance parameters like throughput (i.e., average amount of data bits successfully transmitted per unit time) and forwarding delay (i.e., average end-to--end delay, including the queuing delay and the transmission delay) experienced by the CSMA frames (IEEE 802.3). Plot the comparison graphs for throughput and forwarding delay by varying p. State your observations on the impact of different data rates for exponential/additive backoff along with p-persistent CSMA.

Evaluation date – 18/03/2019 Submission date – 25/03/2019

# DESIGN



I have implemented the error detection module in three program files.

* **sender.py** (Sender program)
* **receiver.py** (Receiver program)
* **channel.py** (Channel program)

The individual files fulfils different assignment purposes, following which have been explained in details :

1. **sender.py** – The following are the tasks performed in this Sender program :
   1. The data is entered by the user.
   2. It follows the above design to transmit the data.
2. **receiver.py** – The following are the tasks performed in this Receiver program :
   1. The data is received from one of the sender processes.
3. **channel.py** – The following are the tasks performed in this Channel program :
   1. Asks for number of sender processes and receiver processes.
   2. Initiates all sender and receiver processes.
   3. Receives data from one sender at a time.
   4. When the channel receives data from a sender, it changes its state to Busy. (The duration is set by the sender process).
   5. Sends the received data to one of the receiver (chosen randomly).

# IMPLEMENTATION

## Code Snippet of channel.py:

import socket import time import subprocess import random import os

class Channel():

def init (self, totalsender, totalreceiver): self.totalsender = totalsender self.senderhost = '127.0.0.1'

self.senderport = 8080 self.senderconn = []

self.totalreceiver = totalreceiver self.receiverhost = '127.0.0.2'

self.receiverport = 9090 self.receiverconn = []

def initSenders(self):

senderSocket = socket.socket() senderSocket.bind((self.senderhost, self.senderport)) senderSocket.listen(self.totalsender)

for i in range(1, self.totalsender+1): conn = senderSocket.accept() self.senderconn.append(conn)

print('Initiated all sender connections')

def closeSenders(self):

for conn in self.senderconn: conn[0].close()

print('Closed all sender connections')

def initReceivers(self):

receiverSocket = socket.socket() receiverSocket.bind((self.receiverhost, self.receiverport)) receiverSocket.listen(self.totalreceiver)

for i in range(1, self.totalreceiver+1): conn = receiverSocket.accept() self.receiverconn.append(conn)

print('Initiated all receiver connections')

def closeReceivers(self):

for conn in self.receiverconn: conn[0].close()

print('Closed all receiver connections')

def processData(self):

fileout = open('status.txt',"w") fileout.write(str(0)) fileout.close()

while True:

for i in range(len(self.senderconn)): print()

conn = self.senderconn[i] fileout = open('status.txt',"w") fileout.write(str(0)) fileout.close()

data = conn[0].recv(1024).decode() fileout = open('status.txt',"w") fileout.write(str(1))

fileout.close()

if not data:

break if data == 'q0':

break

print('Received from Sender',i+1,':',str(data))

recvno = random.randint(0,len(self.receiverconn)-1) print('Sending to Receiver',recvno+1)

rconn = self.receiverconn[recvno] rconn[0].sendto(data.encode(), rconn[1])

if data == 'q0':

break

return

if name == ' main ':

totalsen = int(input('Enter number of senders: '))

totalrecv = int(input('Enter number of receivers: '))

ch = Channel(totalsen, totalrecv) ch.initSenders() ch.initReceivers() ch.processData() ch.closeSenders() ch.closeReceivers()

## Code Snippet of sender.py:

import socket import sys import time import random

def Main(senderno):

print('Initiating Receiver #',senderno) host = '127.0.0.2'

port = 9090

mySocket = socket.socket() mySocket.connect((host, port))

while True:

print()

data = mySocket.recv(1024).decode() if not data:

break if data == 'q':

break

print('Received from channel :', str(data)) mySocket.close()

if name == ' main ': if len(sys.argv) > 1:

senderno = int(sys.argv[1])

else:

senderno = 1

Main(senderno)

## Code Snippet of receiver.py:

import socket import sys import time import random

def Main(senderno):

print('Initiating Sender #',senderno)

host = '127.0.0.1'

port = 8080

mySocket = socket.socket() mySocket.connect((host, port)) prevtime = time.time()

success = 0 while True:

print()

data = input("Enter $ ") #prevtime = time.time() k = 0

kmax = 15 while True:

print('ATTEMPT NUMBER',str(k)) print('Checking channel status ...') filein = open('status.txt',"r") status = int(filein.read())

if status == 0:

print("Channel is IDLE!")

seconds to send!')

prob = random.uniform(0,1) print('probability value is :',str(prob)) print()

if prob <= 0.5:

fileout = open('status.txt',"w") fileout.write(str(1)) fileout.close()

waittime = random.randint(3,7)

print('Channel has been captured. It will take '+str(waittime)+'

else:

print('Sending to channel :',str(data)) time.sleep(waittime) mySocket.send(data.encode()) success += 1

break

print('Waiting for time-slot 2 seconds') time.sleep(2)

filein = open('status.txt',"r") status = int(filein.read())

print('After waiting for 2 second, the channel is',end=' ') if status == 0:

print('IDLE')

else:

k += 1

print('BUSY')

if k > kmax:

print("Transmission aborted!") break

if status == 0:

continue

else:

r = random.randint(0,pow(2,k)-1) print("waiting for back off period")

elif status == 1:

time.sleep(0.1) print("Channel is BUSY!")

print() if not data:

break

if data == 'q':

break

print(" ") curtime = time.time() totaltime = curtime - prevtime

throughput = success/totaltime print("Throughput :",str(throughput))

mySocket.close()

if name == ' main ': if len(sys.argv) > 1:

senderno = int(sys.argv[1])

else:

senderno = 1

Main(senderno)

# TEST CASES

## Sender.py (1st sender):

**C:\Users\SOURAV\Desktop\comp-networks-lab\ass3\csma>python sender.py 1 Initiating Sender # 1**

**Enter $ 1001**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is IDLE!**

**probability value is : 0.14882951956621937**

**Channel has been captured. It will take 3 seconds to send! Sending to channel : 1001**

## Sender.py (2nd sender):

**C:\Users\SOURAV\Desktop\comp-networks-lab\ass3\csma>python sender.py 2 Initiating Sender # 2**

**Enter $ 0001**

**ATTEMPT NUMBER 0**

**Checking channel status ...**

**Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is BUSY!**

**ATTEMPT NUMBER 0**

**Checking channel status ... Channel is IDLE!**

**probability value is : 0.736050029321153**

**Waiting for time-slot 2 seconds**

**After waiting for 2 second, the channel is IDLE ATTEMPT NUMBER 1**

**Checking channel status ... Channel is IDLE!**

**probability value is : 0.19798104388936366**

**Channel has been captured. It will take 3 seconds to send! Sending to channel : 0001**

## Receiver.py (1st receiver):

**C:\Users\SOURAV\Desktop\comp-networks-lab\ass3\csma>python receiver.py 1 Initiating Receiver # 1**

**Received from channel : 1001**

## Receiver.py (2nd receiver):

**C:\Users\SOURAV\Desktop\comp-networks-lab\ass3\csma>python receiver.py 2 Initiating Receiver # 2**

**Received from channel : 0001**

## Channel.py:

**C:\Users\SOURAV\Desktop\comp-networks-lab\ass3\csma>python channel.py Enter number of senders: 2**

**Enter number of receivers: 2 Initiated all sender connections Initiated all receiver connections**

**Received from Sender 1 : 1001 Sending to Receiver 1**

**Received from Sender 2 : 0001 Sending to Receiver 2**

# RESULTS & ANALYSIS

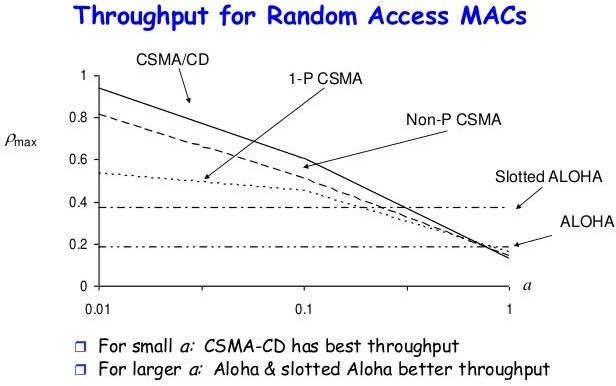
## p-persistent CSMA :

* This method is used when channel has time slots such that the time slot duration is equal to or greater than the maximum propagation delay time.
* Whenever a station becomes ready to send, it senses the channel.
* If channel is busy, station waits until next slot.
* If channel is idle, it transmits with a probability p.
* With the probability q=l-p, the station then waits for the beginning of the next time slot.
* If the next slot is also idle, it either transmits or waits again with probabilities p and q.
* This process is repeated till either frame has been transmitted or another station has begun transmitting.
* In case of the transmission by another station, the station acts as though a collision has occurred and it waits a random amount of time and starts again.

**Advantage of p-persistent CSMA:**

* It reduces the chance of collision and improves the efficiency of the network.

## CSMA/CD :



* The station that has a ready frame sets the back off parameter to zero.
* Then it senses the line using one of the persistent strategies.
* If then sends the frame. If there is no collision for a period corresponding to one complete frame, then the transmission is successful.
* Otherwise the station sends the jam signal to inform the other stations about the collision.
* The station then increments the back off time and waits for a random back off time and sends the frame again.
* If the back off has reached its limit then the station aborts the transmission.

# COMMENTS

This assignment has helped me in understanding the CSMA and CSMA/CD, on implementing them.