Bikash_Sah_002010501018_Computer Networks Lab Report_2

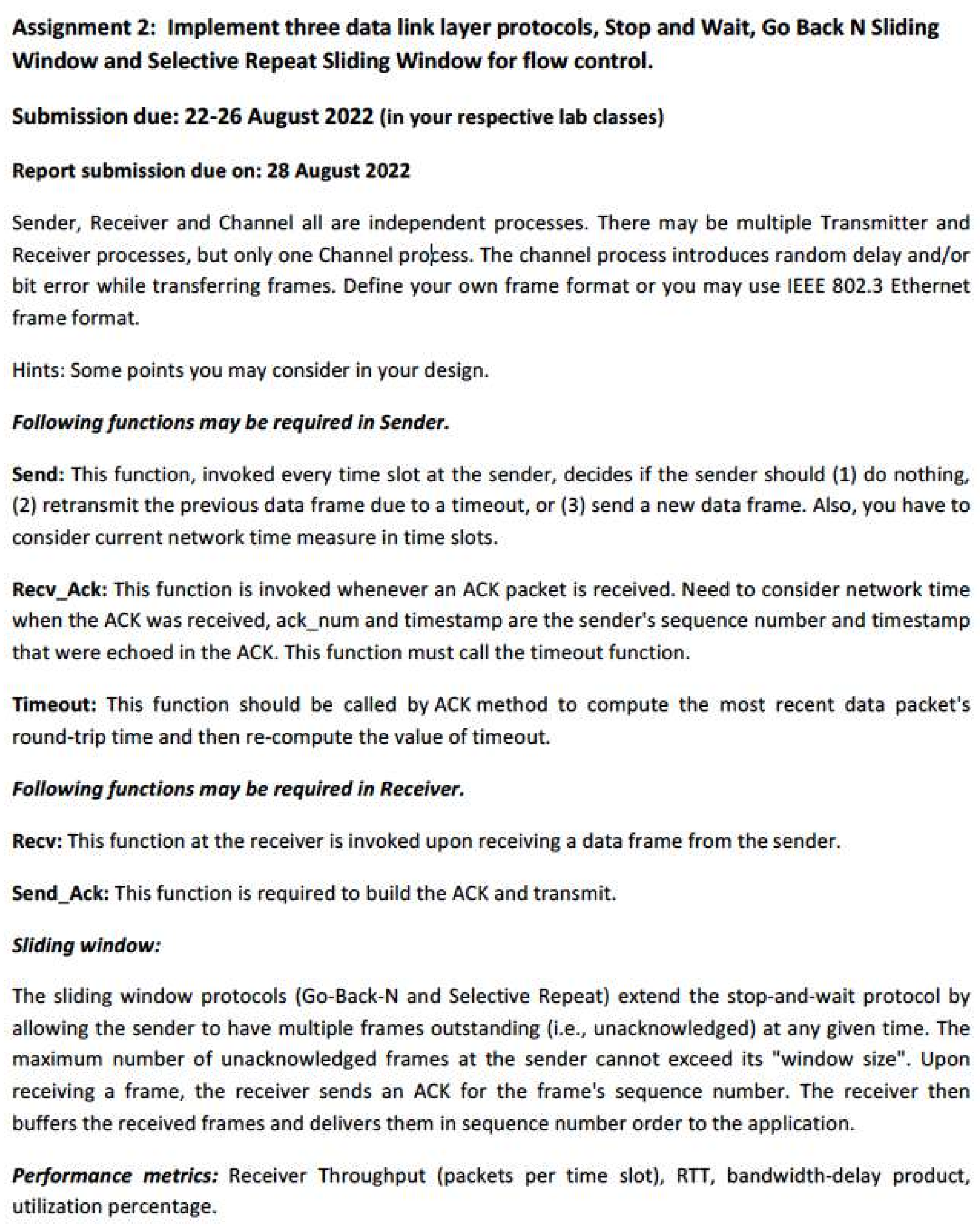
Name: Bikash Sah

Class: BCSE-3

Group: A1

Assignment Number: 2

Problem Statement: **Implement three data link layer protocols, `Stop and Wait`, `Go Back N Sliding Window` and `Selective Repeat Sliding Window` for flow control.**



Assignment 2: Implement three data link layer protocols, Stop and Wait, Go Back N Sliding Window and Selective Repeat Sliding Window for flow control.

Submission due: 22-26 August 2022 (in your respective lab classes)

Report submission due on: 28 August 2022

Sender, Receiver and Channel all are independent processes. There may be multiple Transmitter and Receiver processes, but only one Channel process. The channel process introduces random delay and/or bit error while transferring frames. Define your own frame format or you may use IEEE 802.3 Ethernet frame format.

Hints: Some points you may consider in your design.

Following functions may be required in Sender.

Send: This function, invoked every time slot at the sender, decides if the sender should (1) do nothing, (2) retransmit the previous data frame due to a timeout, or (3) send a new data frame. Also, you have to consider current network time measure in time slots.

Recv_Ack: This function is invoked whenever an ACK packet is received. Need to consider network time when the ACK was received, ack_num and timestamp are the sender's sequence number and timestamp that were echoed in the ACK. This function must call the timeout function.

Timeout: This function should be called by ACK method to compute the most recent data packet's round-trip time and then re-compute the value of timeout.

Following functions may be required in Receiver.

Recv: This function at the receiver is invoked upon receiving a data frame from the sender.

Send_Ack: This function is required to build the ACK and transmit.

Sliding window:

The sliding window protocols (Go-Back-N and Selective Repeat) extend the stop-and-wait protocol by allowing the sender to have multiple frames outstanding (i.e., unacknowledged) at any given time. The maximum number of unacknowledged frames at the sender cannot exceed its "window size". Upon receiving a frame, the receiver sends an ACK for the frame's sequence number. The receiver then buffers the received frames and delivers them in sequence number order to the application.

Performance metrics: Receiver Throughput (packets per time slot), RTT, bandwidth-delay product, utilization percentage.

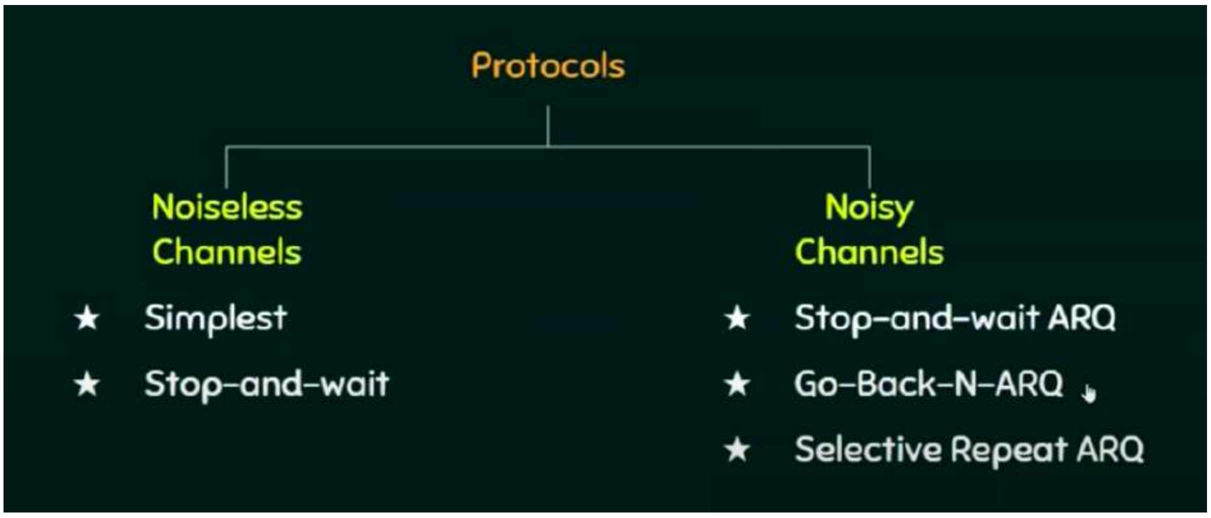
Submission Date: **23 August 2022**

Deadline: **28 August 2022**

Introduction

Flow control is a speed matching mechanism between the receiver and sender. Flow control coordinates the amount of data that can be send before receiving an acknowledgement.

Some flow control protocols are:



In this assignment, we shall discuss the following data link layer protocols in detail.

1. **Stop and Wait protocol**
2. **Go Back N Sliding Window protocol**
3. **Selective Repeat Sliding Window protocol**

Design

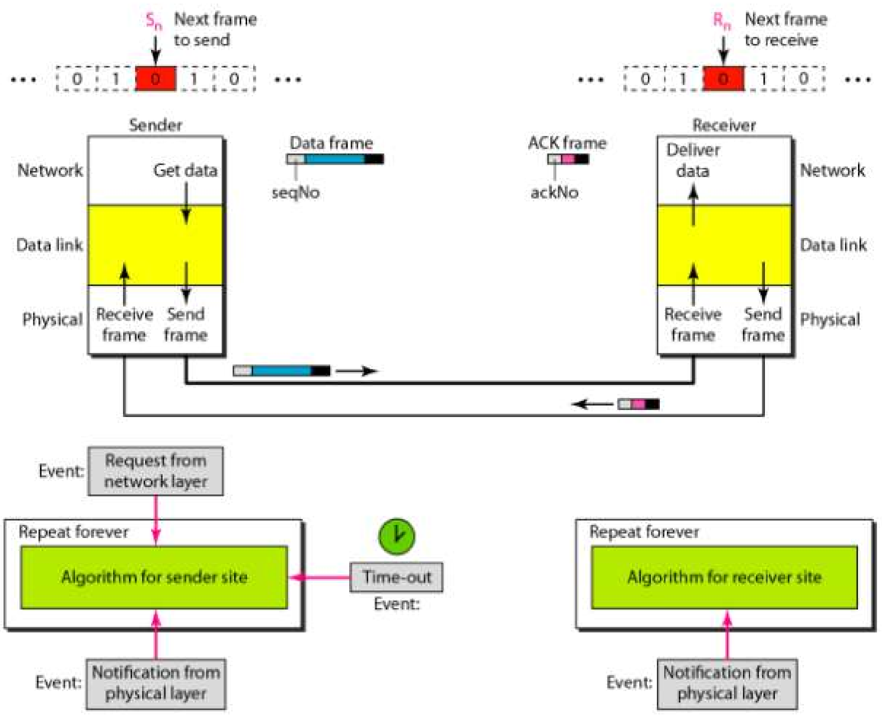
1. **Stop and Wait ARQ (Automatic Repeat Request Protocol)**

    - After sending one frame, the sender waits for an acknowledgement before sending the next frame.
    - If the acknowledgement doesn't arrive after a certain amount of time then the sender times out and retransmit the original frame.
    - Stop and Wait ARQ = Simple stop and wait + timeout timer + sequence number.

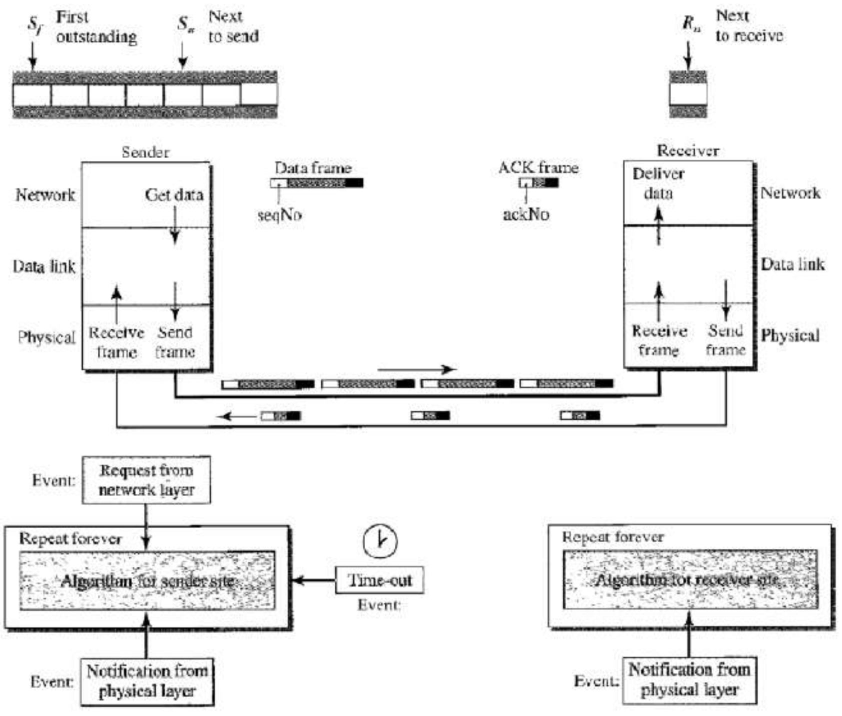
    4 Cases

    1. Frame and ACK are sent and received within time.
    2. Frame is lost.
    3. ACK is lost.
    4. ACK is received after timeout.

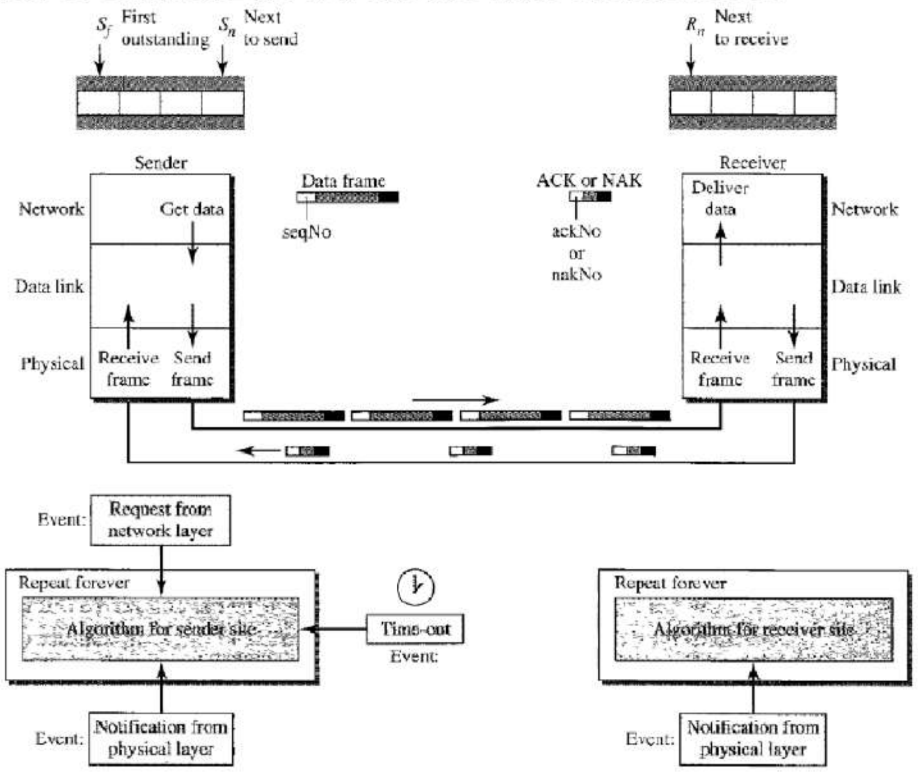
Design of the Stop-and-Wait ARQ Protocol



DESIGN OF GO BACK N SLIDING WINDOW PROTOCOL:



DESIGN OF SELECTIVE REPEAT SLIDING WINDOW PROTOCOL:



Implementation

Stop and Wait:

Sender:

```
import socket
import sys
import time
def createFrame(data):
  countOnes = 0
  for ch in data:
    if ch == '1':
      countOnes += 1
  data += str(countOnes%2)
  return data

def Main(senderno):
  print('Initiating Sender #',senderno)
  host = '127.0.0.1'
  port = 8080

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

  while True:
    print()
    data = input("Enter $ ")
    prevtime = time.time()
    data = createFrame(data)
    print('Sending to channel :',str(data))
    mySocket.send(data.encode())
    if not data:
      break
    if data == 'q0':
      break
    rdata = mySocket.recv(1024).decode()
    print('Received from channel :',str(rdata))
    curtime = time.time()
    print('Round trip time: ',str(curtime-prevtime))
    if curtime-prevtime > 2:
      timeout = 1
    else:
      timeout = 0
    fileout = open('checktime.txt', "w")
    fileout.write(str(timeout))
    fileout.close()
    while timeout==1:
      print()
      prevtime = time.time()
      if timeout == 1:
        print('TIMEOUT of 2s EXPIRED !!')
      else:
        print('THE FRAME GOT CORRUPTED !!!')
      print('Again Sending to channel :',str(data))
      mySocket.send(data.encode())
      rdata = mySocket.recv(1024).decode()
      print('Again Received from channel :',str(rdata))
      curtime = time.time()
      print('Round trip time:',str(curtime-prevtime),'seconds')
      #print('Bandwidth-delay product:',str((curtime-prevtime)*8),'bits/seconds')
      if curtime-prevtime > 2:
        timeout = 1
      else:
        timeout = 0
      fileout = open('checktime.txt', "w")
      fileout.write(str(timeout))
      fileout.close()

  mySocket.close()

if __name__ == '__main__':
  if len(sys.argv) > 1:
    senderno = int(sys.argv[1])
  else:
    senderno = 1
  Main(senderno)
```

Receiver:

```
import socket
import sys
import time
import random

def waitRandomTime():
  x = random.randint(0,5)
  if x <= 1:
    time.sleep(2)

def checkError(frame):
  countOnes = 0
  for ch in frame:
    if ch == '1':
      countOnes += 1
  return countOnes%2

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 == 'q0':
      break

    print('Received from channel :', str(data))
    waitRandomTime()
    if checkError(data) == 0:
      rdata = 'ACK'
    else:
      time.sleep(2)
      rdata = 'TIMEOUT'

    print('Sending to channel :',str(rdata))
    mySocket.send(rdata.encode())



  mySocket.close()

if __name__ == '__main__':
  if len(sys.argv) > 1:
    senderno = int(sys.argv[1])
  else:
    senderno = 1
  Main(senderno)
```

Interface:

```
import socket
import time
import subprocess
import random
import os

def injectRandomError(frame):
  pos = random.randint(0, len(frame)-1)
  frame = frame[:pos]+'1'+frame[pos+1:]
  return frame

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):
      while True:
        for i in range(len(self.senderconn)):
          print()
          conn = self.senderconn[i]
          data = conn[0].recv(1024).decode()
          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]
          data = injectRandomError(data)
          rconn[0].sendto(data.encode(), rconn[1])


          rdata = rconn[0].recv(1024).decode()
          print('Received from Receiver',recvno+1,':', str(rdata))

          print('Sending to Sender',i+1)
          conn[0].send(rdata.encode())

          time.sleep(0.002)
          filein = open('checktime.txt',"r")
          timeout = int(filein.read())
          filein.close()
          os.remove('checktime.txt')
          print(timeout)
          while timeout==1:
            print()
            data = conn[0].recv(1024).decode()
            print('Again Received from Sender',i+1,':',str(data))
            data = injectRandomError(data)
            print('Again Sending to Receiver',recvno+1)
            rconn[0].sendto(data.encode(), rconn[1])
            rdata = rconn[0].recv(1024).decode()
            print('Again Received from Receiver',recvno+1,':', str(rdata))
            print('Again Sending to Sender',i+1)
            conn[0].send(rdata.encode())

            time.sleep(0.002)
            filein = open('checktime.txt',"r")
            timeout = int(filein.read())
            filein.close()
            os.remove('checktime.txt')
            print(timeout)



        if data == 'q0':
          break
      return

if __name__ == '__main__':
  #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()
```

Go Back N:

Sender:

```
import socket
import sys
import time
def createFrame(data):
  countOnes = 0
  for ch in data:
    if ch == '1':
      countOnes += 1
  data += str(countOnes%2)
  return data

def extractMessage(frame):
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/' and endidx == -1:
      endidx = i
      break
  return frame[:endidx]

def extractCount(frame):
  startidx = -1
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      if startidx == -1:
        startidx = i+1
      else:
        endidx = i
  cnt = frame[startidx:endidx]
  return int(cnt)

def extractStatus(frame):
  count = 0
  startidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      count += 1
    if count == 2 and startidx == -1:
      startidx = i+1
      break
  return frame[startidx:]

def Main(senderno):
  count = 0
  sentframes = []
  print('Initiating Sender #',senderno)
  host = '127.0.0.1'
  port = 8080

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

  while True:
    print()
    data = input("Enter $ ")
    #prevtime = time.time()
    data = createFrame(data) + '/' + str(count) + '/'
    msg = extractMessage(data)
    print('Sending to channel :',str(msg))
    mySocket.send(data.encode())
    sentframes.append(data)
    count += 1

    if not msg:
      break
    if msg == 'q0':
      break
    '''rdata = mySocket.recv(1024).decode()
    print('Received from channel :',str(rdata))
    curtime = time.time()
    print('Round trip time: ',str(curtime-prevtime))
    '''
    '''
    time.sleep(0.005)
    filein = open('flag.txt',"r")
    flag = int(filein.read())
    filein.close()
    while flag == 1:
      rdata = mySocket.recv(1024).decode()
      print(rdata)

      cnt = extractCount(rdata)
      msg = extractMessage(sentframes[cnt])
      stat = extractStatus(rdata)
      print(msg,cnt,stat)
      print('Again Sending to channel :',str(msg))
      data = msg +'/'+cnt+'/'
      mySocket.send(data.encode())

      time.sleep(0.005)
      filein = open('flag.txt',"r")
      flag = int(filein.read())
      filein.close()

    '''
  mySocket.close()

if __name__ == '__main__':
  if len(sys.argv) > 1:
    senderno = int(sys.argv[1])
  else:
    senderno = 1
  Main(senderno)
```

Receiver:

```
import socket
import sys
import time
import random

def waitRandomtime():
  x = random.randint(0,5)
  if x <= 1:
    time.sleep(2)

def checkError(frame):
  countOnes = 0
  for ch in frame:
    if ch == '1':
      countOnes += 1
  return countOnes%2

def extractMessage(frame):
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/' and endidx == -1:
      endidx = i
      break
  return frame[:endidx]

def extractCount(frame):
  startidx = -1
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      if startidx == -1:
        startidx = i+1
      else:
        endidx = i
  cnt = frame[startidx:endidx]
  return int(cnt)

def extractStatus(frame):
  count = 0
  startidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      count += 1
    if count == 2 and startidx == -1:
      startidx = i+1
      break
  return frame[startidx:]

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()
    data = str(data)
    msg = extractMessage(data)
    if not msg:
      break
    if msg == 'q0':
      break

    print('Received from channel :', str(data))
    waitRandomtime()
    if checkError(msg) == 0:
      rdata = 'ACK'
    else:
      rdata = 'NAK'

    print('Sending to channel :',str(rdata))
    mySocket.send(rdata.encode())



  mySocket.close()

if __name__ == '__main__':
  if len(sys.argv) > 1:
    senderno = int(sys.argv[1])
  else:
    senderno = 1
  Main(senderno)
```

Interface:

```
import socket
import time
import subprocess
import random
import os

def injectRandomError(frame):
  pos = random.randint(0, len(frame)-1)
  frame = frame[:pos]+'1'+frame[pos+1:]
  return frame

def extractMessage(frame):
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/' and endidx == -1:
      endidx = i
      break
  return frame[:endidx]

def extractCount(frame):
  startidx = -1
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      if startidx == -1:
        startidx = i+1
      else:
        endidx = i
  cnt = frame[startidx:endidx]
  return int(cnt)

def extractStatus(frame):
  count = 0
  startidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      count += 1
    if count == 2 and startidx == -1:
      startidx = i+1
      break
  return frame[startidx:]

class Channel():

  def __init__(self, totalsender, totalreceiver, windowsize):
    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 = []

    self.windowsize = windowsize
    self.slidingwindow = []
    self.currentcount = 0
    #self.statuswindow = []

  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('flag.txt', "w")
    fileout.write(str(0))
    fileout.close()'''
    while True:
      for i in range(len(self.senderconn)):
        print()

        conn = self.senderconn[i]
        data = conn[0].recv(1024).decode()
        prevtime = time.time()
        data = str(data)
        origmsg = extractMessage(data)
        if not origmsg:
          break
        if origmsg == '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]
          cnt = extractCount(data)
          msg = injectRandomError(origmsg)
          newdata = msg + '/' + str(cnt) + '/'
          rconn[0].sendto(newdata.encode(), rconn[1])

          #received from receiver
          rdata = rconn[0].recv(1024).decode()
          rdata = str(rdata)
          time.sleep(0.5)
          curtime = time.time()
          if curtime-prevtime > 2:
            timeout = 1
            newdata += 'TIMEOUT'
          else:
            timeout = 0
            newdata += rdata

          self.slidingwindow.append([data, newdata, i, recvno])


          msg = extractMessage(newdata)
          cnt = extractCount(newdata)
          status = extractStatus(newdata)
          print(msg,str(cnt),status)
          print('Round trip time: ',str(curtime-prevtime))
          print('Current frame no:',str((self.currentcount % windowsize)+1))
          if (self.currentcount % windowsize)+1 == self.windowsize:
            idx = 0
            flag = 1

            while flag == 1:
              idx = 0
              flag = 0
              while idx < self.windowsize:
                currframe = self.slidingwindow[idx][1]
                msg = extractMessage(currframe)
                cnt = extractCount(currframe)
                status = extractStatus(currframe)

                if status == 'NAK' or status == 'TIMEOUT':
                  flag = 1
                  break
                idx += 1
              print(' ------------------------------ ')
              if flag==1:
                print('RESEND FROM FRAME NO:',str(idx+1))
              else:
                print('BLOCK OF WINDOW SIZE',self.windowsize,'SUCCESSFULLY SENT')
              print(' ------------------------------ ')
              '''fileout = open('flag.txt', "w")
              fileout.write(str(flag))
              fileout.close()'''

              while flag == 1 and idx < self.windowsize:
                print()
                prevtime = time.time()
                prevframe = self.slidingwindow[idx][0]
                currframe = self.slidingwindow[idx][1]
                sendno = self.slidingwindow[idx][2]
                recvno = self.slidingwindow[idx][3]
                conn = self.senderconn[sendno]
                rconn = self.receiverconn[recvno]

                #sending all frames to its sender from first NAK
                #conn[0].send(currframe.encode())

                #data = conn[0].recv(1024).decode()
                print('Current frame no:',str(idx+1))
                print('Again Sending to Receiver',recvno+1)

                msg = extractMessage(prevframe)
                msg = injectRandomError(msg)
                data = msg + '/' + str(cnt) + '/'
                rconn[0].sendto(data.encode(), rconn[1])

                # receiving ACK or NAK from receiver
                rdata = rconn[0].recv(1024).decode()
                rdata = str(rdata)
                data += rdata

                msg = extractMessage(data)
                cnt = extractCount(data)
                stat = extractStatus(data)
                curtime = time.time()
                print(msg,str(cnt),stat)
                print('Round trip time: ',str(curtime-prevtime))
                self.slidingwindow[idx][1] = data
                idx += 1

          self.currentcount += 1
        if origmsg == 'q0':
          break
      return

  if __name__ == '__main__':
    #Main()

    totalsen = int(input('Enter number of senders: '))
    totalrecv = int(input('Enter number of receivers: '))
    windowsize =int(input('Enter window size: '))

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

Selective repeat:

Sender:

```
import socket
import sys
import time
def createFrame(data):
  countOnes = 0
  for ch in data:
    if ch == '1':
      countOnes += 1
  data += str(countOnes%2)
  return data

def extractMessage(frame):
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/' and endidx == -1:
      endidx = i
      break
  return frame[:endidx]

def extractCount(frame):
  startidx = -1
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      if startidx == -1:
        startidx = i+1
      else:
        endidx = i
  cnt = frame[startidx:endidx]
  return int(cnt)

def extractStatus(frame):
  count = 0
  startidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      count += 1
    if count == 2 and startidx == -1:
      startidx = i+1
      break
  return frame[startidx:]

def Main(senderno):
  count = 0
  sentframes = []
  print('Initiating Sender #',senderno)
  host = '127.0.0.1'
  port = 8080

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

  while True:
    print()
    data = input("Enter $ ")
    #prevtime = time.time()
    data = createFrame(data) + '/' + str(count) + '/'
    msg = extractMessage(data)
    print('Sending to channel :',str(msg))
    mySocket.send(data.encode())
    sentframes.append(data)
    count += 1

    if not msg:
      break
    if msg == 'q0':
      break

  mySocket.close()

if __name__ == '__main__':
  if len(sys.argv) > 1:
    senderno = int(sys.argv[1])
  else:
    senderno = 1
  Main(senderno)
```

Receiver:

```
import socket
import sys
import time
import random

def waitRandomTime():
  x = random.randint(0,5)
  if x <= 1:
    time.sleep(2)

def checkError(frame):
  countOnes = 0
  for ch in frame:
    if ch == '1':
      countOnes += 1
  return countOnes%2

def extractMessage(frame):
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/' and endidx == -1:
      endidx = i
      break
  return frame[:endidx]

def extractCount(frame):
  startidx = -1
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      if startidx == -1:
        startidx = i+1
      else:
        endidx = i
  cnt = frame[startidx:endidx]
  return int(cnt)

def extractStatus(frame):
  count = 0
  startidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      count += 1
    if count == 2 and startidx == -1:
      startidx = i+1
      break
  return frame[startidx:]

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()
    data = str(data)
    msg = extractMessage(data)
    if not msg:
      break
    if msg == 'q0':
      break

    print('Received from channel :', str(data))
    waitRandomTime()
    if checkError(msg) == 0:
      rdata = 'ACK'
    else:
      rdata = 'NAK'

    print('Sending to channel :',str(rdata))
    mySocket.send(rdata.encode())
  mySocket.close()

if __name__ == '__main__':
  if len(sys.argv) > 1:
    senderno = int(sys.argv[1])
  else:
    senderno = 1
  Main(senderno)
```

Interface:

```
import socket
import time
import subprocess
import random
import os

def injectRandomError(frame):
  pos = random.randint(0, len(frame)-1)
  frame = frame[:pos]+'1'+frame[pos+1:]
  return frame

def extractMessage(frame):
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/' and endidx == -1:
      endidx = i
      break
  return frame[:endidx]

def extractCount(frame):
  startidx = -1
  endidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      if startidx == -1:
        startidx = i+1
      else:
        endidx = i
  cnt = frame[startidx:endidx]
  return int(cnt)

def extractStatus(frame):
  count = 0
  startidx = -1
  for i in range(len(frame)-1):
    if frame[i] == '/':
      count += 1
    if count == 2 and startidx == -1:
      startidx = i+1
      break
  return frame[startidx:]

class Channel():

  def __init__(self, totalsender, totalreceiver, windowsize):
    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 = []
    self.windowsize = windowsize
    self.slidingwindow = []
    self.currentcount = 0
    #self.statuswindow = []

  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('flag.txt', "w")
    fileout.write(str(0))
    fileout.close()'''
    while True:
      for i in range(len(self.senderconn)):
        print()

        conn = self.senderconn[i]
        data = conn[0].recv(1024).decode()
        prevtime = time.time()
        data = str(data)
        origmsg = extractMessage(data)
        if not origmsg:
          break
        if origmsg == '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]
        cnt = extractCount(data)
        msg = injectRandomError(origmsg)
        newdata = msg + '/' + str(cnt) + '/'
        rconn[0].sendto(newdata.encode(), rconn[1])

        #received from receiver
        rdata = rconn[0].recv(1024).decode()
        rdata = str(rdata)
        time.sleep(0.5)
        curtime = time.time()
        if curtime-prevtime > 2:
          timeout = 1
          newdata += 'TIMEOUT'
        else:
          timeout = 0
          newdata += rdata

        self.slidingwindow.append([data, newdata, i, recvno])


        msg = extractMessage(newdata)
        cnt = extractCount(newdata)
        status = extractStatus(newdata)
        print(msg,str(cnt),status)
        print('Round trip time: ',str(curtime-prevtime))
        print('Current frame no:',str((self.currentcount % windowsize)+1))
        if (self.currentcount % windowsize)+1 == self.windowsize:
          idx = 0
          flag = 1

          while flag == 1:
            idx = 0
            flag = 0
            nakframes = []
            indices = []
            while idx < self.windowsize:
              currframe = self.slidingwindow[idx][1]
              msg = extractMessage(currframe)
              cnt = extractCount(currframe)
              status = extractStatus(currframe)

              if status == 'NAK' or status == 'TIMEOUT':
                flag = 1
                nakframes.append(self.slidingwindow[idx])
                indices.append(idx+1)
                #break
              idx += 1

            if flag==0:
              print(' ----------------------------- ')
              print('BLOCK OF WINDOW SIZE',self.windowsize,'SUCCESSFULLY SENT')
              print(' ----------------------------- ')
            '''fileout = open('flag.txt', "w")
            fileout.write(str(flag))
            fileout.close()'''
            idx = 0

            while flag == 1 and idx < len(nakframes):

              print()
              prevtime = time.time()
              prevframe = nakframes[idx][0]
              currframe = nakframes[idx][1]
              sendno = nakframes[idx][2]
              recvno = nakframes[idx][3]
              conn = self.senderconn[sendno]
              rconn = self.receiverconn[recvno]
              stat = extractStatus(currframe)

              print(' ----------------------------- ')
              print('RESENDING FRAME NO:',str(indices[idx]))
              print(' ----------------------------- ')
              #sending all frames to its sender from first NAK
              #conn[0].send(currframe.encode())

              #data = conn[0].recv(1024).decode()
              print('Current frame no:',str(indices[idx]))
              print('Again Sending to Receiver',recvno+1)

              msg = extractMessage(prevframe)
              msg = injectRandomError(msg)
```

```
                data = msg + '/' + str(cnt) + '/'
                rconn[0].sendto(data.encode(), rconn[1])

                # receiving ACK or NAK from receiver
                rdata = rconn[0].recv(1024).decode()
                rdata = str(rdata)
                data += rdata

                msg = extractMessage(data)
                cnt = extractCount(data)
                stat = extractStatus(data)
                curtime = time.time()
                print(msg,str(cnt),stat)
                print('Round trip time: ',str(curtime-prevtime))
                self.slidingwindow[indices[idx]-1][1] = data
                idx += 1


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


          self.currentcount += 1
        if origmsg == 'q0':
          break
      return

  if __name__ == '__main__':
    #Main()

    totalsen = int(input('Enter number of senders: '))
    totalrecv = int(input('Enter number of receivers: '))
    windowsize =int(input('Enter window size: '))


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