RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT - 1

Course Code	CSC303A
Course Name	Computer Networks
Programme	B Tech
Department	CSE
Faculty	FET

Name of the Student	K Srikanth
Reg. No	17ETCS002124
Semester/Year	5th / 3rd Year
Course Leader/s	Dr. Rinki Sharma

Declaration Sheet				
Student Name	K Srikanth			
Reg. No	17ETCS002124			
Programme	B.Tech		Semester/Year	5th / 3rd Year
Course Code	CSC303A			
Course Title	Computer Networks			
Course Date	14/09/2020	to	16/02/2021	
Course Leader	Dr. Rinki Sharma			

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Faculty of Engineering & Technology			
Ramaiah University of Applied Sciences			
Department	Computer Science and Engineering	**Programme**	B. Tech.
Semester	5th		
Course Code	CSC303A	**Course Title**	Computer Networks
Course Leader	Dr. Rinki Sharma, Ms. Suvidha K S, Mr. Nithin Rao R		

Assignment - 1					
Register No.		Name of Student			
Sections		**Marking Scheme**	**Max Marks**	**First Examiner Marks**	**Second Examiner Marks**
Q.1	1.1	Disadvantages of the protocol	02		
	1.2	Modifications to overcome the disadvantages	03		
		Max Marks	**05**		
Q.2	2.1	Program to compute checksum at the transmitter	10		
	2.2	Program to check for error free data transmission at the receiver	10		
		Max Marks	**20**		
		Total Assignment Marks	**25**		

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Course Marks Tabulation				
Component- 1(B) Assignment	**First Examiner**	**Remarks**	**Second Examiner**	**Remarks**
Q 1				
Q 2				
Marks (Max 25)				
Signature of First Examiner			Signature of Second Examiner	

Please note:

1. Documental evidence for all the components/parts of the assessment such as the reports, photographs, laboratory exam / tool tests are required to be attached to the assignment report in a proper order.
2. The First Examiner is required to mark the comments in RED ink and the Second Examiner's comments should be in GREEN ink.
3. If the variation between the marks awarded by the first examiner and the second examiner lies within +/- 3 marks, then the marks allotted by the first examiner is considered to be final. If the variation is more than +/- 3 marks then both the examiners should resolve the issue in consultation with the Chairman BoE.

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Assignment

Instructions to students:

1. The assignment consists of **3** questions.
2. Maximum marks is **25.**
3. The assignment has to be neatly word processed as per the prescribed format.
4. The maximum number of pages should be restricted to **9**.
5. The printed assignment must be submitted to the course leader.
6. **Submission Date: December 5th 2020**
7. **Submission after the due date is not permitted.**
8. **IMPORTANT**: It is essential that all the sources used in preparation of the assignment must be suitably referenced in the text.
9. Marks will be awarded only to the sections and subsections clearly indicated as per the problem statement/exercise/question

Preamble

This course is intended to provide a thorough knowledge of the concepts of computer networks to students. It introduces the layered software hierarchy and the protocols that are applied at each layer. This course also touches on certain application areas of computer networks such as Local Area Networks and Mobile Ad-hoc Networks.

Question 1

1.1)

Introduction

Stop and wait protocol is the simplest flow control method used when a sender wants to sends the data to receiver and after sending data sender stops sending the data and waits for an acknowledgment from the receiver and this process repeats until the data flow is completed from both the parties.

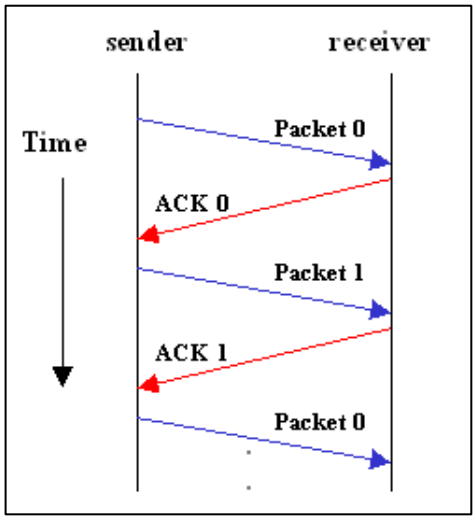


Figure 1 Stop and Wait Protocol

Looking at the image 1 we can see that sender is transmitting packet 0 and is waiting for acknowledgment from the receiver when he receives an acknowledgment that can be accepted then sender will send packet 1 and waits for the next acknowledgment and this process happens until the data is received completely, we can clearly see that it takes lot of time to send and receive data.

Performance Issue and Disadvantages with stop and wait protocol

- "Time" is the major performance and disadvantage issue with this protocol as the complete data takes lot of time to reach the receiver
- If there is no acknowledgment from the receiver it stops sending the data and it doesn't retry and the whole data transfer fails
- If the data is lost from the sender side then the receiver doesn't retry for the data and the whole data transfer fails
- If there is a delay of acknowledgment then it is considered as an incorrect acknowledgment

1.2)

The Modification of the above protocol can be done and improved by adding an **Automatic Repeat Request** so that the if there is a delay or error in transmitting or receiving the data then the system would send the request to server for sending or receiving data depending on the error.

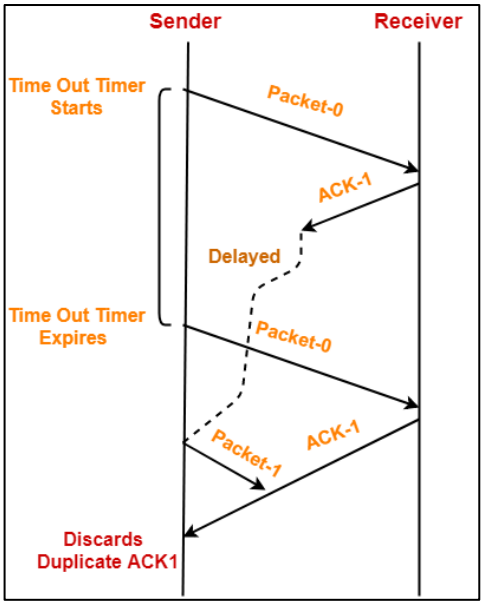


Figure 2 Stop and Wait Protocol with ARQ

The modifications are been made by adding **Automatic Repeat Request** to Stop and wait protocol

Scenario 1

Sender transmits the data packet 0 to receiver it is received at receiver side now the receiver has to send an acknowledgment to sender but it fails while transmitting back so what ARQ does is it waits for a certain amount of time and if there is no acknowledgment then it ARQ sends the data packet 0 to receiver and now the acknowledgment is received if by any chance that the previous acknowledgment was interfered then it will discard the previous acknowledgment. we can still see the delay of receiving the full data at that receiver side but compared to Stop and wait protocol this protocol performed better and all the disadvantages were resolved by adding an **Automatic Repeat Request to the protocol**.

Question 2

Binary Addition Function

```
Binary Addition Function Algorithm withs args x, y
1. Start
2. length = max(len(x), len(y))
3. x= fill a with zeros with length of length
4. y= fill b with zeros with length of length
5. initialize result
6. initialize carry
7. for loop begins
     a. for i in range of length
     b. reminder = carry
     c. reminder = reminder + 1
          i. if x[i] == "1" else 0
     d. reminder = reminder + 1
          i. if y[i] == "1" else 0
     e. result = ("1" if reminder = 1 else 0) + result
     f. if reminder<2 then carry = 0 else 1
     g. Exit
8.if carry! = 0 then result = Binary Addition function
(result,'1')
9. return result. zfill(length)
10. Stop
```

One's Compliment Function

```
One's compliment function algorithm with args a
1. Start
2. initialize data
3. for loop beings
       a. for in range of length (a)
       b. if a[i] = 0
          i. data = data + 1
       c. else
          i. data = data + 0
       d. exit
4.return data
5.Stop
```

Check Sum Function

```
Check Sum Function algorithm with args x,k
1. Start
2. blocks = split x with k for i in range of 0, length(x), k
3. result = zero fill with length of k
4. for loop begins
       i. result = binary_addition(blocks[i],result)
```

```
       ii.exit
5. result = onescompliment(result)
6. return result
7. Stop
```

Main Function

```
Main Function algorithm
1. Start
2. while loop begins
     i. input choice
     ii. if choice = 1
          a. checksum at the transmitter (Question 2.1).
     iii. if choice = 2
          a. checksum at the receiver (Question 2.2).
     iv. if choice = 3
          a. break
     v. exit
3.Stop
```

2.1)

Algorithm for checksum at the transmitter.

```
1. Start
2. Input 32-bit string into an array = “Data”
3. Input number of segments
(only with powers of 2^n where n>=0) = “Segmented”
4. Display Function call checksumcalculator (data, segmented)
5. Stop
```

2.2)

Algorithm for checksum at the receiver with error free data transmission.

```
1. Start
2. Input 32-bit string with check sum into an array = “Data”
3. Input number of segments
(only with powers of 2^n where n>=0) = “Segmented”
4. Display Function call checksumcalculator (data, segmented)
5. Stop
```

Code (Python)

Binary Addition Function

```python
# K Srikanth 17ETCS002124
def binary_addition(x,y): # Perform Binary Addition with x,y
    max_len = max(len(x), len(y)) # length of x,y
    x = x.zfill(max_len)  #fill x with zeros for max length
    y = y.zfill(max_len) #fill y with zeros for max length
    result = '' #initialize result
    carry = 0 # initialize carray
    for i in range(max_len-1, -1, -1): # for i in range of the length of a and b
        r = carry # reminder = carry
        r += 1 if x[i] == '1' else 0 #if x[i] is 1 then reminder = reminder + 1 else 0
        r += 1 if y[i] == '1' else 0 #if y[i] is 1 then reminder = reminder + 1 else 0
        result = ('1' if r % 2 == 1 else '0') + result # Result = if remindner is 1 then 1 else 0 + result
        carry = 0 if r < 2 else 1   # if reminder < 2 carry = 0 else 1 and exit the loop
    if carry !=0 : result = binary_addition(result,'1') # if carrt is != 0 then result = binary_addition (result,1)
    return result.zfill(max_len) # return result
```

Figure 3 Binary Addition Function for given problem statement in python

One's Compliment Function

```python
# K Srikanth 17ETCS002124
def onescompliment(a): # Perform one's compliment with a
    new_data = '' # initialize new_data
    for i in range (len(a)): # for i in range of a
        if a[i] == '0': # if a[i] = 0 then new_data = new_data + 1 else new_data = new_data + 0
            new_data += "1"
        else:
            new_data +="0"
    return new_data # return new_data
```

Figure 4 One's Compliment Function for given problem statement in python

Check Sum Function

```python
# K Srikanth 17ETCS002124
def checksumcalculator(x,k): # Perform checksum of x,k
    blocks = [x[i:i+k] for i in range(0, len(x), k)] # split x with k for i in range of 0, length(x), k
    result = ''.zfill(k) # fill result with for length of k
    for i in range (len(blocks)): # for i in range of length of blocks
        result = binary_addition(blocks[i],result) # Binary Addition of (blocks[i],result)
    result = onescompliment(result)  # Compliment the result onescompliment(result
    return result # return result
```

Figure 5 Check Sum Calculator Function for given problem statement in python

Main Function

```python
# K Srikanth 17ETCS002124
if __name__ == "__main__": # Main function
    print("**************** K Srikanth 17ETCS002124 ****************")
    print("")
    while (1): #while loop begins for the menu
        print("******** Checksum ********")
        print("Press 1 to compute checksum at the transmitter.")
        print("Press 2 to receive data transmission from transmitter")
        choice = int(input()) # Choice of input from the menu
        if choice==1: # if choice is 1 then Tramisit the given data
            print("****** Transmitter Console ******")
            data = input("Enter the 32 bit data to be trasmitted :") # enter the 32bit long data
            # enter the number that data has to be segmented wrt 2^n
            segmented = int(input("Enter the number for which the data has to be segmented "))
            # checksum function with data and segmented as args
            print ("The Checksum at transmitter is ",checksumcalculator(data,segmented))
        elif choice == 2: # if choice is 2 then recevie the given data
            print("****** Receiver Console ******")
            data = input("Enter recevied data from trasmitter :") # enter the 32bit long data with checksum
            # enter the numb str at data has to be segmented wrt 2^n
            segmented = int(input("Enter the number for which the data has to be segmented "))
            # checksum function with data and segmented as args
            print ("The Checksum at transmitter is ",checksumcalculator(data,segmented))
        elif choice == 3: # if choice is 3 then exit the loop and break
            break
```

Figure 6 Main Function for given problem statement in python

Link for my code
Check Sum Python

To run this python code

Python3 filename.py **<- Unix Distribution**
Filename.py **<- Windows Distribution**

Result

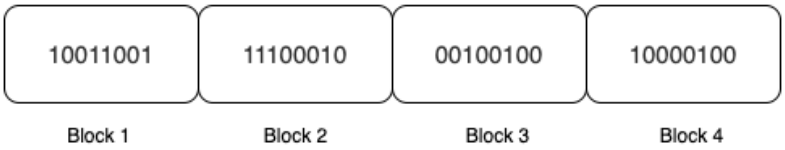
Test Case 1

Sender Side

Let's take input 32-bit stream that has to be transmitted

```
10011001111000100010010010000100
```

Let's divide the input but stream into segments of 8 bits



Now the Number of Blocks is 4 why so 32 / 8 = 4 Blocks

Now we perform binary addition of all the four blocks

10011001
11100010
00100100
10000100
+10

00100101 <- Addition Result

Since there are two extra bits, we add them at last to make it 8 bits

Now we perform 1's compliment on addition result 00100101 and it would be 11011010.

Check Sum = 11011010.

Now the 32-bit string along with checksum

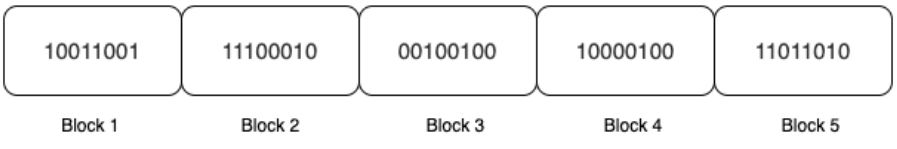
```
10011001111000100010010010000100110110 10
```

Receiver Side

Let's take input 40-bit stream that has been received with check sum added

```
10011001111000100010010010000100110110 10
```

Let's divide the input but stream into segments of 8 bits



Now the Number of Blocks is 5. why so 40 / 8 = 5 Blocks

Now we perform binary addition of all the five blocks with check sum

00100101 <- Addition Result from sender side
11011010

11111111 <- Addition result on receiver side

Now we perform 1's compliment on addition result on receiver side **11111111** and it would be **0000000**. Which is an error free data

Python Result

```
********** K Srikanth 17ETCS002124 **********

******* Checksum *******
Press 1 to compute checksum at the transmitter.
Press 2 to receive data transmission from transmitter
1
****** Transmitter Console ******
Enter the 32 bit data to be trasmitted :10011001111000100010010010000100
Enter the number for which the data has to be segmented 8
The Checksum at transmitter is  11011010
******* Checksum *******
Press 1 to compute checksum at the transmitter.
Press 2 to receive data transmission from transmitter
2
****** Receiver Console ******
Enter recevied data from trasmitter :10011001111000100010010010000010011011010
Enter the number for which the data has to be segmented 8
The Checksum at transmitter is  00000000
```

Figure 7 Python output console for Test case 1

Test Case 2

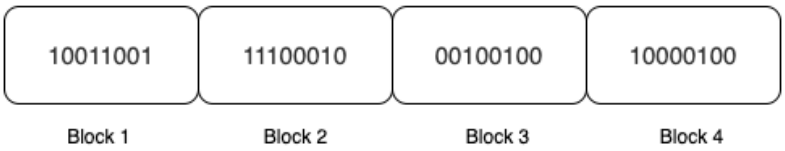
Now let's flip a bit and check for the error

<u>Sender Side</u>

Let's take input 32-bit stream that has to be transmitted

```
10011001111000100010010010000100
```

Let's divide the input but stream into segments of 8 bits



Now the Number of Blocks is 4 why so 32 / 8 = 4 Blocks

Now we perform binary addition of all the four blocks

10011001
11100010
00100100
10000100
+10

00100101 <- Addition Result

Since there are two extra bits, we add them at last to make it 8 bits

Now we perform 1's compliment on addition result 00100101 and it would be 11011010.

Check Sum = 11011010.

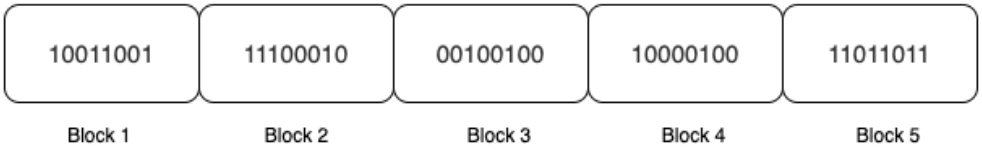
So, I am going to change the check sum last bit to 1 and that would be 11011011.

Receiver Side

Let's take input 40-bit stream that has been received with check sum added

```
1001100111100010001001001000010011011011
```

Let's divide the input but stream into segments of 8 bits



Now the Number of Blocks is 5. why so 40 / 8 = 5 Blocks

Now we perform binary addition of all the five blocks with check sum

00100101 <- Addition Result from sender side
11011011 <- Check sum

00000001 <- Addition result on receiver side

Now we perform 1's compliment on addition result on receiver side **00000001** and it would be **11111110** Which is an error data for a successful error free data when you do check sum the data bits have to be 0's

Python Result

```
********** K Srikanth 17ETCS002124 **********

******** Checksum ********
Press 1 to compute checksum at the transmitter.
Press 2 to receive data transmission from transmitter
1
****** Transmitter Console ******
Enter the 32 bit data to be trasmitted :10011001111000100010010010000100
Enter the number for which the data has to be segmented 8
The Checksum at transmitter is  11011010
******** Checksum ********
Press 1 to compute checksum at the transmitter.
Press 2 to receive data transmission from transmitter
2
****** Receiver Console ******
Enter recevied data from trasmitter :10011001111000100010010010000100111011011
Enter the number for which the data has to be segmented 8
The Checksum at transmitter is  11111110
******** Checksum ********
Press 1 to compute checksum at the transmitter.
Press 2 to receive data transmission from transmitter
```

Figure 8 Python output console for Test case 2