Assignment 6

Chaiwat Plongkaew 2021326660023

P3. UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error?

Ans   
 Calculate the sum of the given 3 bytes Add first two 8-bit bytes:

0 1 0 1 0 0 1 1

0 1 1 0 0 1 1 0

1 0 1 1 1 0 0 1

Now add the result with the 3rd byte.

1 0 1 1 1 0 0 1

0 1 1 1 0 1 0 0

1 0 0 1 0 1 1 0 1

Wrap around the extra bit.

0 0 1 0 1 1 0 1

1

0 0 1 0 1 1 1 0

The sum three 8-bit bytes is 00101110. Invert all the bits to get the checksum. So, the checksum is 11010001.

**The 1’s complement of (sum) 001011110 is 11010001.**

**UDP uses the 1s complement as it is the same as the checksum of the sum. The checksum is used by the receiver (host) to identify the errors in the segment.**

**The receiver will add all the bytes including the checksum. Then observe the sum, if it contains all 1s then the segment has errors, or if it contains 1 or more 0s then the segment contains errors.**

**Using 1’s complement method, it is possible to detect all the 1-bit errors.**

**Using 1’s complement method, there is a possibility that some 2-bit errors are left undetected.**

P4. a. Suppose you have the following 2 bytes: 01011100 and 01100101. What is the 1s complement of the sum of these 2 bytes?

Ans 0 1 0 1 1 1 0 0

0 1 1 0 0 1 0 1

1 1 0 0 0 0 0 1 **=> 00111110 is the 1s complement of those two bytes.**  
  
b. Suppose you have the following 2 bytes: 11011010 and 01100101. What is the 1s complement of the sum of these 2 bytes?

Ans 1 1 0 1 1 0 1 0

0 1 1 0 0 1 0 1

0 0 1 1 1 1 1 1 **=> 11000000 is the 1s complement of those two bytes.**

c. For the bytes in part (a), give an example where one bit is flipped in each of the 2 bytes and yet the 1s complement doesn’t change.

Ans 0 1 0 1 1 1 0 1

0 1 1 0 0 1 0 0

**1 1 0 0 0 0 0 1 => 00111110 is the 1s complement.**

**If we need to keep the 1s complement of the 2 bytes, under the condition of changing 1 bit in each byte, we need to change the bytes in the same position, and they are not supposed to be the same bit.**

Use python to implement the Internet checksum, try to employ matplotlib for data visualization.

Client.py

import struct  
from socket import \*  
import zlib  
  
# sender  
  
  
# Function to find the Checksum of Sent Message  
def findChecksum(data, k):  
 SentMessage = zlib.crc32(data)  
 # print(SentMessage) # integer  
 SentMessage = format(SentMessage, 'b')  
 # print(SentMessage) # binary  
  
 # Dividing sent message in packets of k bits.  
 c1 = SentMessage[0:k]  
 c2 = SentMessage[k:2 \* k]  
 c3 = SentMessage[2 \* k:3 \* k]  
 c4 = SentMessage[3 \* k:4 \* k]  
  
 # Calculating the binary sum of packets  
 Sum = bin(int(c1, 2) + int(c2, 2) + int(c3, 2) + int(c4, 2))[2:]  
  
 # Adding the overflow bits  
 if (len(Sum) > k):  
 x = len(Sum) - k  
 Sum = bin(int(Sum[0:x], 2) + int(Sum[x:], 2))[2:]  
 if (len(Sum) < k):  
 Sum = '0' \* (k - len(Sum)) + Sum  
  
 # Calculating the complement of sum  
 Checksum = ''  
 for i in Sum:  
 if (i == '1'):  
 Checksum += '0'  
 else:  
 Checksum += '1'  
 return Checksum  
  
  
# driver  
clientSocket = socket(AF\_INET, SOCK\_DGRAM)  
data = "Hello world"  
packet = data.encode()  
receiver\_addr = ("10.131.194.15", 1111)  
checksum = findChecksum(packet, 8)  
print(checksum)  
  
checksum = struct.pack("11s 8s", packet, checksum.encode())  
  
clientSocket.sendto(checksum, receiver\_addr)

Server.py

import struct  
import zlib  
import matplotlib.pyplot as plt  
from socket import \*  
  
  
  
  
# receiver  
  
# Function to find the Complement of binary addition of  
# k bit packets of the Received Message + Checksum  
def checkReceiverChecksum(ReceivedMessage, k, Checksum):  
 ReceivedMessage = zlib.crc32(ReceivedMessage)  
 ReceivedMessage = format(ReceivedMessage, 'b')  
  
 # Dividing sent message in packets of k bits.  
 c1 = ReceivedMessage[0:k]  
 c2 = ReceivedMessage[k:2 \* k]  
 c3 = ReceivedMessage[2 \* k:3 \* k]  
 c4 = ReceivedMessage[3 \* k:4 \* k]  
  
 # Calculating the binary sum of packets + checksum  
 ReceiverSum = bin(int(c1, 2) + int(c2, 2) + int(Checksum, 2) +  
 int(c3, 2) + int(c4, 2) + int(Checksum, 2))[2:]  
  
 # Adding the overflow bits  
 if (len(ReceiverSum) > k):  
 x = len(ReceiverSum) - k  
 ReceiverSum = bin(int(ReceiverSum[0:x], 2) + int(ReceiverSum[x:], 2))[2:]  
  
 # Calculating the complement of sum  
 ReceiverChecksum = ''  
 for i in ReceiverSum:  
 if (i == '1'):  
 ReceiverChecksum += '0'  
 else:  
 ReceiverChecksum += '1'  
 return ReceiverChecksum  
  
  
# Driver  
serverSocket = socket(AF\_INET, SOCK\_DGRAM)  
serverSocket.bind(('10.131.194.15', 1111))  
  
print("The server is ready to receive")  
  
while True:  
 message, clientAddress = serverSocket.recvfrom(2048)  
  
 packet, checksum = struct.unpack("11s 8s", message)  
 checksum = checksum.decode()  
  
 # testing error condition  
 # packet = "hello World"  
 # packet = packet.encode()  
  
 ReceiverChecksum = checkReceiverChecksum(packet, 8, checksum)  
  
 # Printing Checksum  
 print("SENDER SIDE CHECKSUM: ", checksum)  
 print("RECEIVER SIDE CHECKSUM: ", ReceiverChecksum)  
 finalsum = bin(int(checksum, 2) + int(ReceiverChecksum, 2))[2:]  
  
 # Finding the sum of checksum and received checksum  
 finalcomp = ''  
 for i in finalsum:  
 if (i == '1'):  
 finalcomp += '0'  
 else:  
 finalcomp += '1'  
  
 # If sum = 0, No error is detected  
 if (int(finalcomp, 2) == 0):  
 print("Receiver Checksum is equal to 0. Therefore,")  
 print("STATUS: ACCEPTED")  
 # Otherwise, Error is detected  
 else:  
 print("Receiver Checksum is not equal to 0. Therefore,")  
 print("STATUS: ERROR DETECTED")  
  
 sender\_side\_checksum = [int(x) for x in list(checksum)]  
 receiver\_side\_checksum = [int(x) for x in list(ReceiverChecksum)]  
  
 plt.plot(sender\_side\_checksum, label="Sender Side Checksum")  
 plt.plot(receiver\_side\_checksum, label="Receiver Side Checksum")  
 plt.title("Difference between Sender Side and Receiver Side Checksum")  
 plt.xlabel("Bits")  
 plt.ylabel("Value")  
 plt.legend()  
 plt.show()