AIM: implement CRC Error detection method.

Theory:

CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in the communication channel.

CRC uses Generator Polynomial which is available on both sender and receiver side. An example generator polynomial is of the form like x3 + x + 1. This generator polynomial represents key 1011. Another example is x2 + 1 that represents key 101.

Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

The binary data is first augmented by adding k-1 zeros in the end of the data

Use modulo-2 binary division to divide binary data by the key and store remainder of division.

Append the remainder at the end of the data to form the encoded data and send the same

Receiver Side (Check if there are errors introduced in transmission)

Perform modulo-2 division again and if the remainder is 0, then there are no errors.

INPUT:

def xor(a, b):

result = []

for i in range(1, len(b)):

if a[i] == b[i]:

result.append('0')

else:

result.append('1')

return ''.join(result)

def mod2div(dividend, divisor):

pick = len(divisor)

tmp = dividend[0: pick]

while pick < len(dividend):

if tmp[0] == '1':

tmp = xor(divisor, tmp) + dividend[pick]

else:

tmp = xor('0' \* pick, tmp) + dividend[pick]

pick += 1

if tmp[0] == '1':

tmp = xor(divisor, tmp)

else:

tmp = xor('0' \* pick, tmp)

checkword = tmp

return checkword

class sender:

def encodedata(data, key):

l\_key = len(key)

appended\_data = data + '0' \* (l\_key - 1)

remainder = mod2div(appended\_data, key)

codeword = data + remainder

print("Remainder : ", remainder)

print("Encoded Data (Data + Remainder) : ",

codeword)

dataword = input("enter the data-word\n") # data=input('enter your data') # key=input('enter your 2 data')

gen = input("enter the generator\n")

# removing x and + and appending the numbers in a list

gen2 = ""

for i in gen:

if i == 'x':

pass

else:

gen2 += i

gen2 = gen2.split("+") # here gen2 becomes a list

print(gen2)

# replacing the str numbers with int numbers and '' with 0

for idx, i in enumerate(gen2):

if i == '':

gen2[idx] = 0

else:

gen2[idx] = int(gen2[idx])

# generating the divisor

generator = ""

for i in range(int(gen2[0]) + 1):

if i in gen2:

generator = '1' + generator

else:

generator = '0' + generator

print("generator", generator)

print("frame", dataword)

encodedata(dataword, generator)

# receiver side:1

# #data=input('enter your data') #key=input('enter your 2 data')

#

data2=input('enter the frame received')

key=input("enter the generated output")

def xor(a, b):

result = []

for i in range(1, len(b)):

if a[i] == b[i]:

result.append('0')

else:

result.append('1')

return ''.join(result)

def mod2div(dividend, divisor):

pick = len(divisor)

tmp = dividend[0: pick]

while pick < len(dividend):

if tmp[0] == '1':

tmp = xor(divisor, tmp) + dividend[pick]

else:

tmp = xor('0' \* pick, tmp) + dividend[pick]

pick += 1

if tmp[0] == '1':

tmp = xor(divisor, tmp)

else:

tmp = xor('0' \* pick, tmp)

checkword = tmp

return checkword

def encodedata(data, key):

l\_key = len(key)

appended\_data = data + '0' \* (l\_key - 1)

remainder = mod2div(appended\_data, key)

codeword = data + remainder

print("Remainder : ", remainder)

print("Encoded Data (Data + Remainder) : ",

codeword)

if remainder==0:

print('there is an error')

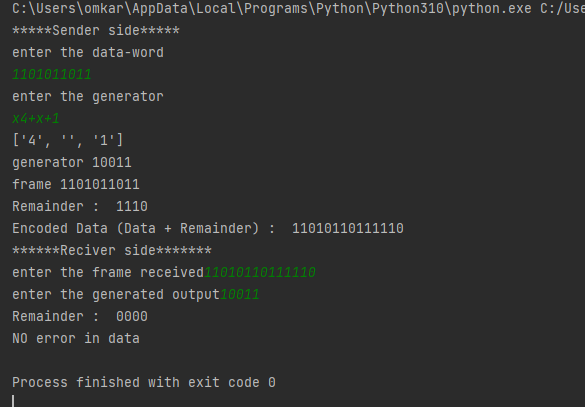
else:

print('received frame is successfull')

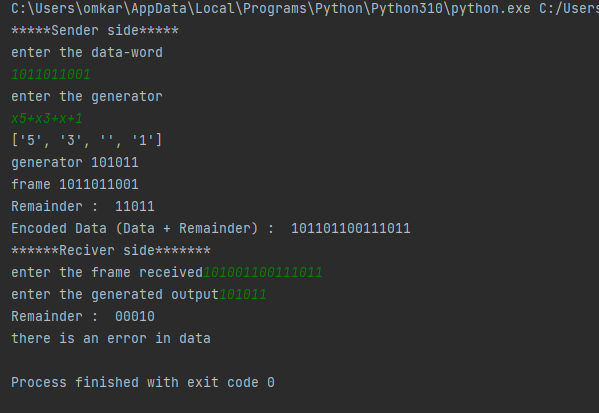
encodedata(data2,key)

OUTPUT:

Example1:



Example2:



Conclusion: The following experiment for implementing CRC detection method is successfully implemented.