Федеральное агентство связи

СибГУТИ

Кафедра телекоммуникационных сетей и

вычислительных средств (ТС и ВС)

Дисциплина

Сети ЭВМ и телекоммуникации 2.0

Лабораторная работа №5

Контроль и исправление ошибок.

Выполнил: студент группы ИА-832

Тиванов.Д.Е

Проверил: преподаватель

Ахпашев Р.В

Новосибирск 2021

Задание

- Поверх CRC добавить код Хэмминга.
- Если присутствуют ошибки, реализовать ARQ процесс.
- Построить график зависимости среднего количества переотправок пакета от значения SNR, где SNR = [start=-5; end=3; step=1]
- Оценить % избыточности.

В данном разделе рассмотрим один из простейших алгоритмов поиска и исправления ошибок, код Хэмминга.

Bit Position		1	2	3	4	5	6	7	8	9	10	11
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7
Parity bit coverage	p1	x		x		x		x		x		x
	p2		x	x			x	x			x	x
	p4				x	x	x	x				
	p8								x	x	x	x

Ход работы:

Считывание файла и определение битовой разрядности

```
import math
from random import randint

handle = open("binary_input.txt", "r")
binaryInput = list(handle.read())

if (len(binaryInput) == 1):
    inputSize = 2
elif ((len(binaryInput) >= 2) & (len(binaryInput) <= 4)):
    inputSize = 3  # int(math.log(len(binaryInput), 2)) # size of input
elif ((len(binaryInput) >= 5) & (len(binaryInput) <= 11)):
    inputSize = 4
else:  # 12-26 = 5; 27-57 = 6...
    inputSize = 5
```

Добавление битовой ошибки

```
if wrongCode[wrongRandom] == '1':
    wrongCode[wrongRandom] = '0'
else:
    wrongCode[wrongRandom] = '1'
```

```
print("random wrong byte #: {}\nbinaryInput:
{}\nbinaryOutput:{}".format(wrongRandom + 1, binaryInput, wrongCode))
print(" ".rjust(14 + 5 * wrongRandom), "^\n", " ".rjust(13 + 5 *
wrongRandom), "|")
```

реализация кода

```
for i in range(inputSize):  # xor right and wrong code
    if i > 1:
        z += 1
for j in range(2 ** i - 1, len(binaryInput), 2 ** i + i + 1 + z):
    for k in range(2 ** i):
        if j + k < len(binaryInput):
            # print(j + k + 1)
            if binaryInput[j + k] == '1':
                xorArr[0][i] += 1
            if wrongCode[j + k] == '1':
                xorArr[1][i] += 1

for i in range(2):  # mod 2 to get 0, 1 (even or odd)
    for j in range(inputSize):
        xorArr[i][j] = xorArr[i][j] % 2

    for j in range(inputSize):  # before finel sum xor
        xorArr[2][j] = (xorArr[0][j] + xorArr[1][j]) % 2
```

Результат программы:

```
C:\Users\danil\PycharmProjects\client1\venv\Scripts\python.exe C:/Users/danil/PycharmProjects/client1/main.py
fileInput: ['1', '1', '0', '1', '1', '0', '1', '0']
random wrong byte #: 11
binaryInput: ['0', '0', '1', '0', '1', '0', '1', '0', '1', '0', '1', '0', '0']
binaryOutput:['0', '0', '1', '0', '1', '0', '1', '0', '1', '0', '0', '0', '0']
                                                          ^
                                                          |
xor's: [[0, 0, 0, 0], [0, 0, 0, 1], [0, 0, 0, 1]]
Random wrong byte: 11
Hamming code: 8

Process finished with exit code 0
```

Код программы:

```
import math
from random import randint

handle = open("binary_input.txt", "r")
binaryInput = list(handle.read())

if (len(binaryInput) == 1):
    inputSize = 2
elif ((len(binaryInput) >= 2) & (len(binaryInput) <= 4)):
    inputSize = 3  # int(math.log(len(binaryInput), 2)) # size of input
elif ((len(binaryInput) >= 5) & (len(binaryInput) <= 11)):
    inputSize = 4
```

```
else:  # 12-26 = 5; 27-57 = 6...
    inputSize = 5

print("fileInput: {}".format(binaryInput))

for i in range(inputSize + 1):
    binaryInput.insert(2 ** i - 1, '0')  # num of zeros

wrongCode = list(binaryInput)
wrongRandom = randint(0, len(binaryInput) - 1)

if wrongCode[wrongRandom] == '1':
    wrongCode[wrongRandom] = '0'
else:
    wrongCode[wrongRandom] = '1'

print("random wrong byte #: {}\nbinaryInput:
{}\nbinaryOutput:{}".format(wrongRandom + 1, binaryInput, wrongCode))
print(" ".rjust(14 + 5 * wrongRandom), "^\n", " ".rjust(13 + 5 *
wrongRandom), "|")

xorArr = [[0] * inputSize for i in range(3)]

z = 0

for i in range(inputSize):  # xor right and wrong code
    if i > 1:
        z += 1
for j in range(2 ** i - 1, len(binaryInput), 2 ** i + i + 1 + z):
    for k in range(2 ** i):
        if j + k < len(binaryInput):
            # print(j + k + 1)
            if binaryInput[j + k] == '1':
                xorArr[0][i] += 1
            if wrongCode[j + k] == '1':
                xorArr[1][i] += 1

for i in range(2):  # mod 2 to get 0, 1 (even or odd)
    for j in range(inputSize):
        xorArr[i][j] = xorArr[i][j] % 2

    for j in range(inputSize):  # before finel sum xor
        xorArr[2][j] = (xorArr[0][j] + xorArr[1][j]) % 2

wrongBytePos = 0

for j in range(inputSize):  # final sum and results
    if (xorArr[2][j] == 1):
        wrongBytePos += 2 ** j

print("xor's:", xorArr)
print("Random wrong byte: {}\nHamming code: {}".format(wrongRandom + 1,
wrongBytePos))
```