Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Основы компьютерных сетей

ОТЧЕТ
по лабораторной работе №3
на тему
МЕТОДЫ ЗАЩИТЫ ОТ ОШИБОК, ИСПОЛЬЗУЕМЫЕ В СЕТЯХ

Студент Е.С. Мелюх

Преподаватель В.А. Марцинкевич

МИНСК 2024

1 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1.1 Исходные данные к работе

Взять за основу программу, относящуюся к лабораторной работе
№2.

Четный вариант:

Реализовать поддержку поля FCS в структуре кадра -- для проверки кадра с помощью кода Хэмминга. При этом код Хэмминга применять только к полю Data. Длину (длины) поля FCS необходимо рассчитать с учетом исходных требований и поля.

Исходные требования к коду Хэмминга: код должен обнаруживать двойные ошибки и исправлять одиночные.

В рамках кодирования и декодирования кадров, программно реализовать алгоритм вычисления проверочных битов (как «на бумаге»).

Предусмотреть возможность случайного искажения случайных битов (байтов) в поле Data каждого кадра перед передачей. Вероятность искажения одного бита должна составлять 60 %, а вероятность искажения двух битов должна составлять 25 %.

Модифицировать окно состояния. По-прежнему периодически выводить структуру текущего кадра после приема, но немного по-другому (до декодирования). Один кадр по-прежнему должен соответствовать одной строке. При этом выделять (подчеркиванием либо другим цветом) поле FCS вместо принятого значения поля FCS выводить вычисленное.

2 ОБЗОР ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

2.1 Теоретическая часть

Код Хэмминга – это циклический код, используемый для обнаружения и исправления ошибок в передаваемых данных. Он основан на принципе добавления к исходным данным дополнительных проверочных битов, позволяющих обнаруживать и исправлять ошибки определенного типа.

В коде Хэмминга биты исходных данных и проверочные биты объединяются в кодовое слово. Позиции проверочных битов в кодовом слове имеют степени двойки (1, 2, 4, 8 и т.д.). Значения проверочных битов вычисляются таким образом, чтобы сумма всех битов в кодовом слове, включая проверочные биты, была четной для всех позиций, являющихся степенями двойки.

Код Хэмминга должен обнаруживать двойные ошибки и исправлять
одиночные ошибки в поле данных (Data) кадра.

Длина поля FCS (Frame Check Sequence): Поскольку длина поля данных (Data) фиксирована и равна 15 байтам (120 бит), то для определения длины поля FCS необходимо найти минимальное значение r, удовлетворяющее условию $2^r \geq 120 + r + 1$. Минимальное значение r, удовлетворяющее этому условию, равно 8. Следовательно, длина поля FCS должна быть равна 8 битам.

Вероятность искажения битов: Вероятность искажения одного бита в поле данных должна составлять 60%, а вероятность искажения двух битов – 25%. Эти значения вероятностей выбраны для моделирования реальных условий передачи данных, где одиночные ошибки более вероятны, чем двойные, но двойные ошибки также могут возникать.

3 ВЫПОЛНЕНИЕ РАБОТЫ

3.1 Разработка функции, вычисляющей значение поля FCS

Функция generate_hamming_code предназначена для генерации кода Хэмминга, который используется для обнаружения и исправления ошибок в передаваемых данных.Принятие данных:

Функция принимает строку data, представляющую двоичные данные. Данные преобразуются в список битов (data_bits) с помощью map(int, data).Функция возвращает строку, представляющую код Хэмминга, сформированный из списка hamming_code. Также выводится отладочное сообщение с сгенерированным кодом Хэмминга для проверки.

```
@staticmethod
  def calculate_hamming_code(data_bits):
    m = len(data_bits)
    r = 0
    while (2 ** r) < (m + r + 1):
      r += 1

    code_bits = ['0'] * (m + r)
    j = 0
    for i in range(1, len(code_bits) + 1):
      if (i & (i - 1)) != 0:
        code_bits[i - 1] = data_bits[j]
        j += 1

    for i in range(r):
      pos = 2 ** i
      parity = 0
      for j in range(1, len(code_bits) + 1):
        if j & pos:
          parity ^= int(code_bits[j - 1])
      code_bits[pos - 1] = str(parity)

    return ''.join(code_bits)
```

3.2 Разработка функции, проверяющей правильность переданных данных

Функция decode_hamming_code предназначена для декодирования данных, полученных с использованием кода Хэмминга, а также для обнаружения и исправления ошибок. Функция принимает строку received_data, представляющую двоичные данные (код Хэмминга). Данные преобразуются в список битов (received_bits) с помощью map(int, received_data).

```
@staticmethod
    def hamming_decode(code_bits):
        r = 0
        while (2 ** r) < len(code_bits):
            r += 1

        error_pos = 0
        for i in range(r):
            pos = 2 ** i
            parity = 0
            for j in range(1, len(code_bits) + 1):
                if j & pos:
                    parity ^= int(code_bits[j - 1])
            if parity != 0:
                error_pos += pos

        if error_pos:
            print(f"Ошибка в позиции {error_pos}. Исправляем...")
            code_bits = list(code_bits)
            code_bits[error_pos - 1] = '0' if code_bits[error_pos - 1] == '1' else '1'

        return ''.join(code_bits)
```

3.3 Разработка программного обеспечения

Полный код программы, включающий реализацию функций для вычисления FCS и проверки целостности данных, представлен в ПРИЛОЖЕНИИ А.

4 РЕЗУЛЬТАТЫ РАБОТЫ

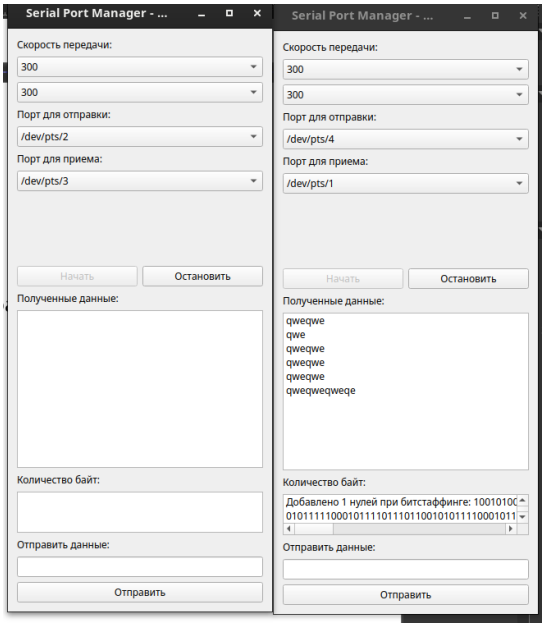

Рисунок 4.1 – Пример работы программы

5 ВЫВОДЫ

В рамках данной лабораторной работы был разработан модуль для кодирования и декодирования данных с использованием кода Хэмминга. Этот код обеспечивает обнаружение ошибок и исправление одиночных ошибок в передаваемых данных, что делает его полезным инструментом в системах передачи информации.

В программе реализованы две основные функции: hamming_decode и hamming_code.

Полученные в ходе выполнения лабораторной работы навыки реализации алгоритмов кодирования и декодирования данных, а также практический опыт работы с механизмами обнаружения и исправления ошибок будут полезны для дальнейшей разработки систем передачи данных и сетевых приложений, где требуется высокая надежность и целостность информации. Программа продемонстрировала свою эффективность и готовность к использованию в реальных сценариях, что открывает новые возможности для ее применения в различных областях, связанных с обработкой и передачей данных.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1] pySerial [Электронный ресурс]. Режим доступа: https://pyserial.readthedocs.io/en/latest/pyserial.html. – Дата доступа: 20.09.2024.

[2] Вычислительные комплексы, системы и сети: лабораторный практикум для студентов специальности I-40 02 01 «Вычислительные машины, системы и сети» / И. И. Глецевич, Д. В. Ламовский, Д. А.Пынькин. – Минск : БГУИР, 2010. – 36 с. : ил.

[3] Код Хэмминга. Пример работы алгоритма [Электронный ресурс]. Режим доступа: https://habr.com/ru/articles/140611/– Дата доступа: 20.09.2024.

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный текст программы