Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

**Отчет**

**по лабораторной работе №5**

**по дисциплине «Основы систем мобильной связи»**

**Тема: «Циклический избыточный код. CRC»**

**Вариант 11**

Выполнил:

студент гр. ИА-232

Московских Дмитрий Петрович

GitHub: <https://github.com/DmitriyMosk/osms-labs>



Новосибирск 2024

**Содержание**

Цель 3

задачи 3

теоретические сведения 3

исходные данные 5

этапы выполнения работы 5

контрольные вопросы 13

вывод 13

**Цель работы**

Получить представление о том, как осуществляется проверка на наличие ошибок в пакетах с данными в современных системах связи (Error detection) посредством использования циклического избыточного кода CRC (Cyclic Redundancy Check).

**Краткие теоретические сведения**

Псевдослучайные двоичные последовательности

CRC — циклический избыточный код, иногда называемый также

контрольным кодом или контрольной суммой. CRC – это добавочная порция

избыточных бит, вычисляемых по заранее известному алгоритму на основе

исходного передаваемого пакета данных (информационной битовой

последовательности), которое передаётся вместе с самим пакетом по каналам

связи (добавляется после информационных битов) и служит для контроля его

безошибочной передачи.

Простыми словами, CRC – это остаток от двоичного деления

оригинального пакета с данными на какое-то двоичное n-разрядное число

(порождающий полином), и его длина будет равна n-1 бит. Рассмотрим пример,

где имеется 7 бит данных: 100100 и 4-битный порождающий полином 1101.

Требуется определить CRC. Для того, чтобы выполнить деление этих битовых

последовательностей нужно в конце последовательности с данными добавить

n-1 нулей, как показано ниже, где n=4, для нашего случая.

Делитель - 1 1 0 1 | 1 0 0 1 0 0 0 0 0 - Делимое (данные+n-1 нулей).

Основной операцией, используемой при делении бинарных чисел,

является исключающее ИЛИ (XOR). Ниже показана таблица истинности для

данной операции.

Пошаговое вычисление CRC (на стороне передатчика):

1) 1 1 0 1 | 1 0 0 1 0 0 0 0 0

1 1 0 1 (операция XOR)

--------------

1 0 0 0

2) 1 1 0 1 | 1 0 0 1 0 0 0 0 0

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

3) 1 1 0 1 | 1 0 0 1 0 0 0 0 0

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

1 1 0 1

------------

1 1 1 0

4) 1 1 0 1 | 1 0 0 1 0 0 0 0 0

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

1 1 0 1

------------

1 1 1 0

1 1 0 1

--------------

0 1 1 0

При появлении 0, на следующем шаге делим на 0000.

3

5) 1 1 0 1 | 1 0 0 1 0 0 0 0 0

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

1 1 0 1

------------

1 1 1 0

1 1 0 1

--------------

0 1 1 0

0 0 0 0

-------------

1 1 0 0

6) 1 1 0 1 | 1 0 0 1 0 0 0 0 0

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

1 1 0 1

------------

1 1 1 0

1 1 0 1

--------------

0 1 1 0

0 0 0 0

-------------

1 1 0 0

1 1 0 1

---------------

0 0 1 – это и есть CRC, остаток от деления.

Делитель принято записывать в виде полинома. Если считать, что каждый

разряд делителя — это коэффициент полинома, то этот полином будет иметь

вид:

xn-1+xn-2+…+x2+x1+x0

Таким образом, делитель из примера выше можно записать в виде

полинома как: 1\*x3+1\*x2+0\*x1+1\*х0, или сокращенно как: x3+x2+1 = 1101.

4

Полученный остаток от деления CRC добавляется на передающей

стороне к исходным данным и уже эта битовая последовательность,

преобразованная в радиосигнал, передается в канал связи: 1 0 0 1 0 0 0 0 1.

На приемной стороне для обнаружения ошибки (или ее отсутствия) с

полученным пакетом осуществляется ровно такая же процедура – деление на

порождающий CRC полином. Если полученный в результате данного деления

остаток будет ненулевым, то фиксируется факт наличия ошибки.

Пошаговое вычисление CRC (на стороне приемника):

1) 1 1 0 1 | 1 0 0 1 0 0 0 0 1

1 1 0 1 (операция XOR)

--------------

1 0 0 0

2) 1 1 0 1 | 1 0 0 1 0 0 0 0 1

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

3) 1 1 0 1 | 1 0 0 1 0 0 0 0 1

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

1 1 0 1

------------

1 1 1 0

4) 1 1 0 1 | 1 0 0 1 0 0 0 0 1

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

1 1 0 1

------------

1 1 1 0

1 1 0 1

--------------

0 1 1 0

При появлении 0, на следующем шаге делим на 0000.

5

5) 1 1 0 1 | 1 0 0 1 0 0 0 0 1

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

1 1 0 1

------------

1 1 1 0

1 1 0 1

--------------

0 1 1 0

0 0 0 0

-------------

0 1 0 1

6) 1 1 0 1 | 1 0 0 1 0 0 0 0 1

1 1 0 1

-------------

1 0 0 0

1 1 0 1

-------------

1 0 1 0

1 1 0 1

------------

1 1 1 0

1 1 0 1

--------------

0 1 1 0

0 0 0 0

-------------

1 1 0 1

1 1 0 1

---------------

0 0 0 – то есть, пакет передан без ошибок.

**Задание для выполнения практической работы**

В рамках данной работы студенты должны научиться вычислять CRC

последовательности, а также на их основании детектировать ошибки.

**Порядок выполнения работы:**

1) Напишите программу на языке С/С++ для вычисления CRC для пакета

данных длиной N бит (N= 20 + порядковый номер в журнале) и

определения факта наличия ошибки при передаче пакета по каналу связи.

2) Порождающий полином G для делителя выберите в соответствии с

вариантом. Номер варианта – порядковый номер в журнале группы.



3) Добавьте полученный остаток от деления на G к пакету исходными данными и на приемной стороне вычислите повторно остаток от деления пакета с данными+CRC на полином G. Определите есть ли ошибка в принятом пакете. Выведите в окно терминала полученное значение CRC и отчет об ошибках в принятом пакете.

vector<uint8\_t> initialData = {1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0};

    Packet packet31(initialData);

    packet31.CRCWrite();

    packet31.CRCValidate();

4. Возьмите N, равное 250 битам. Проделайте п.1-3

5. Сделайте цикл из 250+CRC length итераций и в этом цикле по очереди искажайте по одному биту – с 0-го до 250+CRC-1, проверьте в соответствии с п.3 обнаружена ли ошибка на приемной стороне и выполните подсчет того сколько раз за этот цикл приемник обнаружил и не обнаружил ошибки. Результат выведите в окно терминала.

vector<uint8\_t> randomData = Packet::generateData(250);

    Packet largePacket(randomData);

    largePacket.CRCWrite();

    int errorsDetected = 0, errorsMissed = 0;

    size\_t totalBits = randomData.size() + 8;

    for (size\_t i = 0; i < totalBits; ++i) {

        largePacket.toggleBit(i);

        if (!largePacket.CRCValidate()) {

            errorsDetected++;

        } else {

            errorsMissed++;

        }

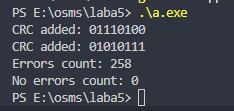
        largePacket.toggleBit(i);

    }

    cout << "Errors count: " << errorsDetected << endl;

    cout << "No errors count: " << errorsMissed << endl;

Результат работы программы



**Контрольные вопросы**

**Для чего в мобильных сетях используются CRC-проверки?** CRC (Cyclic Redundancy Check) — это метод обнаружения ошибок, используемый для проверки целостности данных в мобильных сетях (и не только). При передаче данных через сеть они могут быть искажены из-за помех или сбоев. CRC позволяет выявить эти ошибки, обеспечивая тем самым более надежную передачу данных.

**Что такое порождающий полином?** Порождающий полином — это полином, используемый для генерации CRC-кода. Он определяет набор операций, которые необходимо выполнить для вычисления значения CRC. Полином может быть задан различными способами, но обычно он представлен в виде бинарного числа.

**Как вычислить CRC для пакета с данными?** Процесс вычисления CRC для пакета с данными включает следующие шаги:

* Добавьте нули в конец данных. Количество нулей должно быть равно степени порождающего полинома.
* Выполните деление полученной последовательности данных на порождающий полином с использованием бинарной арифметики (операция XOR).
* Остаток от деления и будет значением CRC.