УДК 004.056.2

О. А. Турдиев, В. В. Яковлев, С. В. Клименко
Петербургский государственный университет путей сообщения Императора Александра I

А. Х. Болтаев
Ташкентский институт инженеров железнодорожного транспорта

Исследование формирования блоковой контрольной суммы (BCC) передаваемых данных

Методы обнаружения ошибок предназначены для выявления искажений в сообщениях при их передаче по зашумленным каналам. Основная идея, заложенная в них, – передача избыточной служебной информации, по которой можно судить с некоторой степенью вероятности о достоверности принятых данных. В статье рассматривается применение метода формирования блочного контроля четности (Block Check Character, BCC) для блока данных. BCC используется при передаче блоков символов для увеличения вероятности обнаружения ошибок. Суть данного метода заключается в вычислении бита четности как для строки, так и для столбца. Для рассматриваемого способа генерации BCC было написано приложение на языке высокого уровня Java, позволяющее получить результаты и собрать статистику обнаружения/необнаружения ошибки передаваемых данных. Собранная статистика показывает вероятность необнаружения ошибки для передаваемого блока данных. Для оценки вероятности необнаружения искаженных бит был сымитирован процесс передачи данных с ошибочными битами. Для написания приложения использовано программное обеспечение Java Development Kit версии 1.8 и среда разработки Net Beans IDE 8.0.2.

BCC (Block Check Character), блочный контроль четности, логическая операция XOR, байт данных, битчетности, метод проверки

Надежную передачу информации обеспечивают различные методы. Основной принцип работы протоколов, которые обеспечивают надежность передачи информации, – повторная передача искаженных или потерянных пакетов. Такие протоколы основаны на том, что приемник в состоянии распознать факт искажения информации в принятом кадре.

Методы обнаружения ошибок основаны на передаче в составе блока данных избыточной служебной информации, по которой можно судить с некоторой степенью вероятности о достоверности принятых данных.

Избыточную служебную информацию принято называть контрольной суммой, или контрольной последовательностью кадра. Контрольная сумма вычисляется как функция от основной информации.

Принимающая сторона повторно вычисляет контрольную сумму кадра по известному алгоритму и в случае ее совпадения с контрольной суммой, вычисленной передающей стороной, делает вывод о том, что данные были переданы через сеть корректно [1]–[2].

Одним из таких алгоритмов является блочный контроль четности (Block Check Character, BCC) – результат последовательной проверки данных на четность (логическая операция XOR) посылаемого или принимаемого блока.

Краткое описание алгоритма подсчета BCC. При передаче блоков символов (или байт) в составе кадров увеличивается вероятность того, что символы, а следовательно, и блоки будут содержать множество битовых ошибок. При передаче блоков символов можно добиться увеличения вероятности обнаружения ошибок, используя дополнительный бит, называемый битом четности. С помощью этого метода каждому символу (байту) в кадре присваивается бит четности построчно. Кроме того, дополнительный бит вычисляется для каждой позиции бита по столбцам (продольная четность) в полном передаваемом кадре символов [1]–[3]. Полученный набор бит четности для каждого столбца называется блоковым символом проверки блока суммы (BCC), поскольку каждый бит, составляющий символ, является суммой по модулю 2 всех бит в соответствующем столбце.

Простой код с обнаружением ошибок основан на схеме проверки четности, применимой к сообщениям $B_0 \ldots B_m$ любой фиксированной длины m. Схема кодирования определяется следующими формулами:

$$P_r = \begin{cases} 0, \text{если} \sum_{i=0}^{m} B_i - \text{четна}; \\ 1, \text{если} \sum_{i=0}^{m} B_i - \text{нечетна}. \end{cases}$$

Соответствующая схема декодирования:

$$D(B_0 .. B_m P_r) = \begin{cases} B_0 \ldots B_m, \text{если} \sum_{i=0}^{m+1} B_i - \text{четна}; \\ \text{ошибка, если} \sum_{i=0}^{m+1} B_i - \text{нечетна}. \end{cases}$$

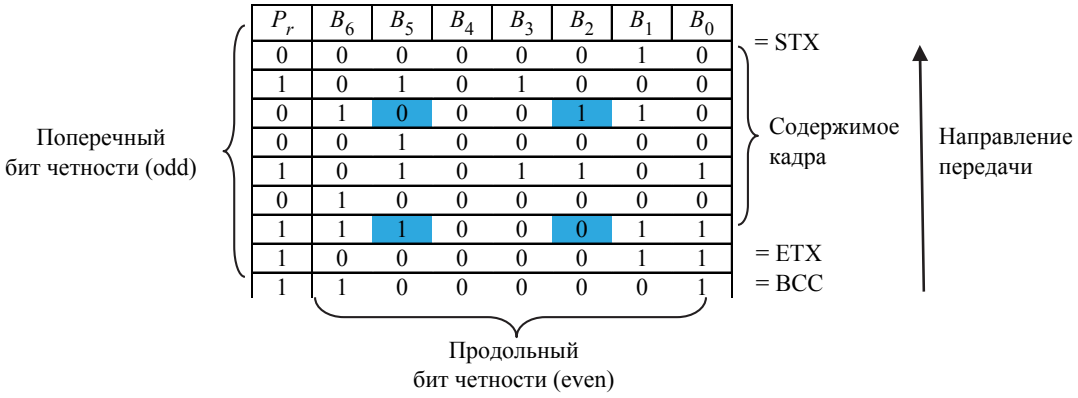
Рис. 1 иллюстрирует метод проверки на основе формирования BCC: бит четности строки и столбца. Пример на рис. 1 использует «even-odd» для бит четности строк и столбцов и предполагает, что кадр содержит только печатные символы [4].

Из этого примера можно сделать вывод, что хотя две битовые ошибки в символе не обнаруживаются проверкой четности строки, они будут обнаружены с помощью соответствующей проверки четности столбцов. Это верно только в том случае, если в одном столбце одновременно нет двух битовых ошибок [5].

Вариантом представленной схемы контроля является использование сумматоров в дополнительном коде в качестве основы проверки суммы блока вместо суммы по модулю 2. Принцип работы схемы показан на рис. 2 (метод проверки на основе формирования BCC: формирование дополнительного кода суммы) [6].

В этой схеме символы (или байты) в подлежащем передаче блоке рассматриваются как двоичные числа без знака. Они сначала суммируются используя двоичную арифметику. Все биты в полученной сумме инвертируются, а результат используется как символ проверки контрольной суммы блока (BCC). В приемнике вычисляется аналогичная сумма, включая символ проверки блока, и если ошибок нет, результат должен быть равен нулю.

Оценка эффективности обнаружения ошибок метода BCC. Под эффективностью понимается возможность выявить ошибки. Для рассматриваемого способа генерации BCC было написано приложение на языке высокого уровня Java, позволяющее получить результаты и статистику обнаружения ошибки передаваемых данных. Статистика показывает обнаружение и необнаруже-



P_r	B_6	B_5	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	1	0
1	0	1	0	1	0	0	0
0	1	0	0	0	1	1	0
0	0	1	0	0	0	0	0
1	0	1	0	1	1	0	1
0	1	0	0	0	0	0	0
1	1	1	0	0	0	1	1
1	0	0	0	0	0	1	1
1	1	0	0	0	0	0	1

Рис. 1

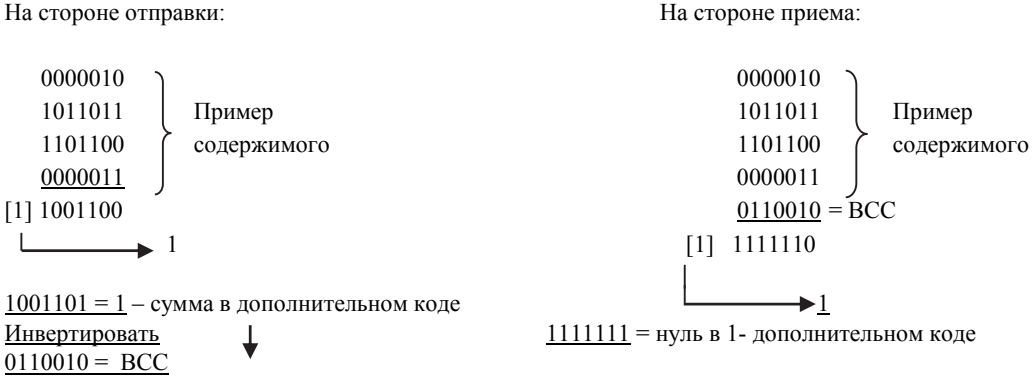


Рис. 2

ние ошибки. Для написания приложения использовано программное обеспечение Java Development Kit версии 1.8 и среда разработки NetBeans IDE 8.0.2. Эксперимент проводился с помощью следующих аппаратных и программных ресурсов:

– операционная система Windows 10 (64-разрядная);

– двухъядерный процессор IntelCore i3-3217U с тактовой частотой 1.8 Ггц;

– 16 Гбайт оперативной памяти;

– жесткий диск объемом 1 Тбайт.

Программа предназначена для определения целостности передаваемых данных с использованием BCC (рис. 2).

Для определения целостности передаваемых данных выполняются следующие действия:

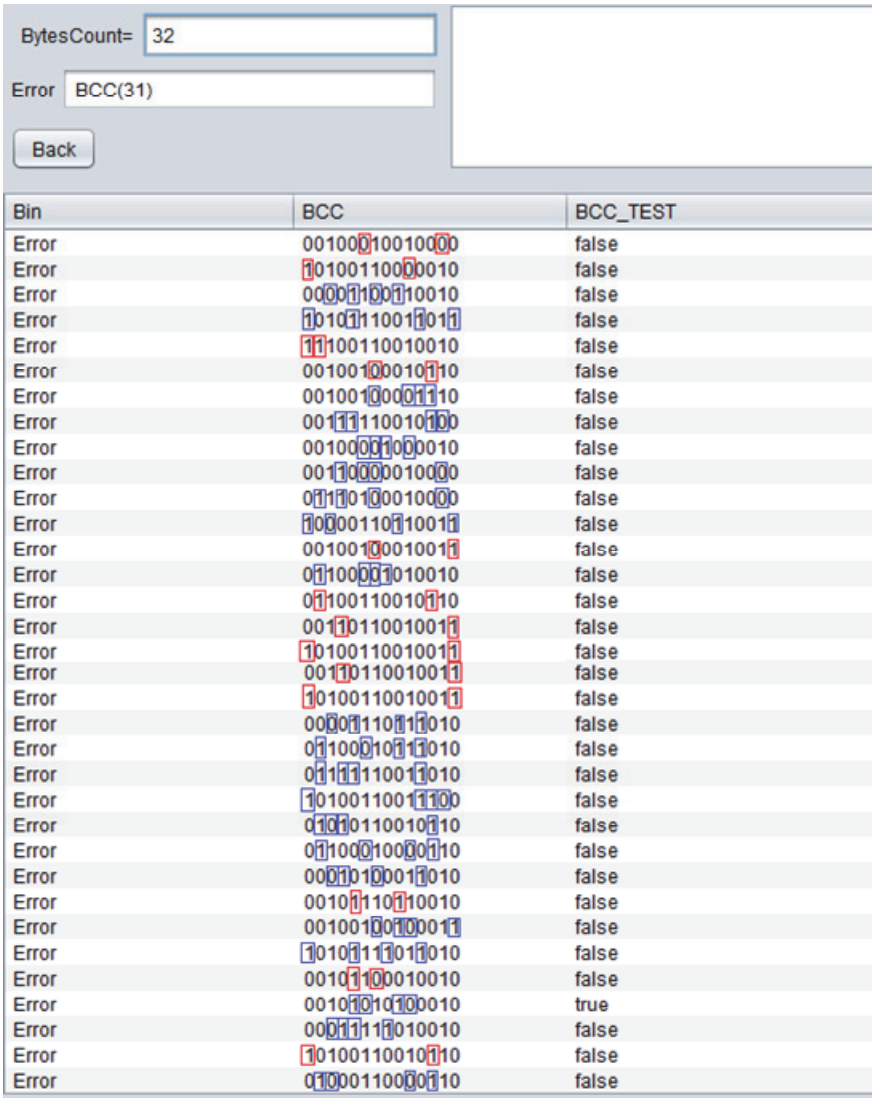
1. Задание исходных данных по символу (к примеру, символ – &).

2. Конвертация исходных данных (&) в двоичную систему вычисления равно 0100110 (двоичное представление символа &).

3. По полученным исходным данным производится подсчет блоковой контрольной суммы бит четности BCC0100110**010010.**

4. В программе в исходные данные можно внести искусственные ошибки (для рассматриваемого примера были искажены случайные значения четырех бит).

5. После имитации процесса передачи данных заново подсчитывается контрольная сумма блока бит четности BCC для выявления факта искажения передаваемых данных (рис. 2), поле «BCC_TEST»



Bin	BCC	BCC_TEST
Error	00100010010000	false
Error	10100110000010	false
Error	00001100110010	false
Error	10101110011011	false
Error	11100110010010	false
Error	00100100010110	false
Error	00100100011110	false
Error	00111100101000	false
Error	00100001000010	false
Error	00110000010000	false
Error	01110100010000	false
Error	10000110110011	false
Error	00100100010011	false
Error	01100001010010	false
Error	01100110010110	false
Error	00110110010011	false
Error	10100110010011	false
Error	00110110010011	false
Error	10100110010011	false
Error	00001110111010	false
Error	01100010111010	false
Error	01111110011010	false
Error	10100110011100	false
Error	01011011001010	false
Error	01100010001110	false
Error	00011010001110	false
Error	00101110110010	false
Error	00100100100011	false
Error	10101111011010	false
Error	00101100010010	false
Error	00101010100010	true
Error	00011110010010	false
Error	10100110010110	false
Error	01000110001110	false

4- битное изменение

2- битное изменение

Рис. 3

указывает на обнаруженную ошибку посредством флагов «true» («истина», данные верны) и «false» («ложь», данные содержат ошибку).

Пример необнаружения ошибки в передаваемом символе & и ВСС приведен в таблице.

Передаваемый символ	00100110
ВСС для передаваемого символа	010010
Полученный символ	00101010
ВСС для полученного символа	100010

Поскольку ошибки могут возникать как в передаваемом символе, так и в поле ВСС, есть вероятность принятия полученных данных за достоверные. Факт такой ошибки был продемонстрирован в таблице и на рис. 3 (пример работы программы).

Поскольку в эксперименте участвует 14 бит (8 бит под исходный символ и 6 бит ВСС), то вероятность искажения отдельного бита определяется по формуле

$$p = \frac{1}{n} = 0.07,$$

где n – число бит.

Рис. 3 также демонстрирует, что 2 из четырех ошибок возникли в одном и том же бите, тем самым восстановив его. Вероятность такого исхода определяется по формуле

$$P = \left(\frac{1}{n}\right)^2 = 0.0049\,.$$

Вероятность безошибочной передачи всех n бит:

$$P = (1-p)^n = 0.36\,.$$

Для нахождения вероятности ошибки с заданной кратностью r (в данном случае $r = 4$) следует воспользоваться формулой Бернулли:

$$P_n(p,r) = C_n^r p^r (1-p)^{n-r} =$$

$$= \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r} = 0.012.$$

Результат обнаружения ошибок методом ВСС. Для оценки вероятности необнаружения искаженных бит был сымитирован процесс передачи данных с ошибочными битами. Результаты имитации приведены на рис. 4.

Рис. 4

Как видно из рисунка, метод позволяет обнаружить большую часть четырех битовых ошибок. Этот результат достигается за счет построчных и продольных бит четности.

В статье рассмотрен способ формирования блочного контроля четности как один из способов контроля целостности передаваемых данных. Как показала собранная статистика, рассматриваемый метод позволяет обнаружить большую часть четырех битовых ошибок. Для сбора статистики было создано программное обеспечение на языке Java.

СПИСОК ЛИТЕРАТУРЫ

1. Halsall F. Computer networks and the Internet. Fifth edition. Boston: Addison-Wesley: Pearson Education, 2005. 803 p.

2. Lin S., Costello D. J. Jr. Error Control Coding: Fundamentals and Applications. Upper Saddle River: Prentice-Hall, 1983.

3. Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы. СПб.: Питер, 2008. 958 с.

4. Halsall F. Data communications, computer networks and open systems. Addison-Wesley: Pearson Education, 1996. 907 p.

5. Ромащенко А. Е., Румянцев А. Ю., Шень А. Заметки по теории кодирования. 2-е изд., испр. и доп. / МЦНМО. М., 2017. 88 с.

6. Яковлев В. В., Кушназаров Ф. И. Оценка влияния помех на производительность протоколов канального уровня // Изв. Петерб. гос. ун-та путей сообщения. 2015. Вып. 1 (42). С. 133–138.

O. A. Turdiev, V. V. Yakovlev, S. V. Klimenko
Emperor Alexander I Petersburg State Transport University

A. X. Boltaev
Tashkent Institute of Railway Engineers

STUDY OF THE FORMATION OF BLOCK CHECKSUM (BCC) OF THE TRANSMITTED DATA

Error detection methods are designed to detect distortions in messages when they are transmitted through noisy channels. The main idea embedded in them is the transfer of redundant service information, which can be used to judge with some degree of probability the reliability of the data received. Error detection methods are designed to detect distortions in messages when they are transmitted through noisy channels. The main idea embedded in them is the transfer of redundant service information, which can be used to judge with some degree of probability the reliability of the data received. The article discusses the use of the method of forming a block parity (BlockCheckCharacter, BCC) for the data block. BCC is used when transmitting blocks of characters to increase the likelihood of error detection. The essence of this method is to calculate the parity bit for both the row and the column. For the BCC generation method in question, an application was written in a high-level Java language, which allows obtaining results and collecting statistics on the detection / non-detection of transmitted data errors. The collected statistics show the probability of not detecting an error for the transmitted data block. To assess the likelihood of undetected distorted bits, the process of transferring data with erroneous bits was simulated. JavaDevelopmentKit software version 1.8 and the NetBeans IDE 8.0.2 development environment are used to write the application.

BCC (Block Check Character), block checksum, logical XOR operation, data byte, parity bit, check method