УДК 621.313.684

К.В. Шинкаренко, А.М. Кориков

Исследование эффективности помехоустойчивых кодов Лаби

Проведен обзор существующих подходов к решению актуальной проблемы потери пакетов данных в пакетных сетях передачи. Установлено, что способностью исправлять потери пакетов целиком обладают стирающие помехоустойчивые коды. Представлены результаты экспериментального исследования эффективности и производительности стирающих кодов Лаби.

Ключевые слова: помехоустойчивое кодирование, восстановление потерь, стирающие коды, коды Лаби.

В настоящее время происходит бурное развитие компьютерных сетей и рост числа сетевых приложений. Как и в любой системе передачи данных, в компьютерных сетях существуют помехи, шумы, ведущие к потерям и искажениям данных. Для борьбы с помехами в системах передачи данных применяется помехоустойчивое кодирование. Помехоустойчивое кодирование выполняет важнейшую роль в цифровых системах передачи данных, так как повышает достоверность передаваемых данных путем внесения систематической избыточности согласно некоторому математическому закону (коду).

Теория помехоустойчивого кодирования была основана в середине XX в. и с тех пор превратилась в хорошо проработанную, математически стройную отрасль науки. Теория помехоустойчивого кодирования тесно переплетена с теорией информации. Основоположниками теории помехоустойчивого кодирования стали выдающиеся ученые В.А. Котельников, К. Шеннон, У. Питерсон, П. Элиас. Большинство ранних научных работ в области теории помехоустойчивого кодирования были направлены на поиск «оптимального» кода. «Оптимальный» код с точки зрения теории кодирования должен быть длинным и обладать достаточной глубиной для эффективного исправления ошибок [1]. Этот поиск до сих пор не увенчался успехом, хотя изобретенные несколько десятилетий назад Турбо-коды являются квазиоптимальными [2].

Одной из наиболее острых актуальных проблем в современных компьютерных сетях является проблема потерь пакетов. В качестве примеров приложений, подверженных риску потерь пакетов при передаче, можно назвать цифровое телевещание через Интернет (IPTV), Интернет-телефония (VoIP), распределенные сетевые игры и т.д. Потери данных ведут к деградации качества сервисов: потере голосовых сообщений, ухудшению качества восприятия видеоизображения, рассинхронизации звука и видео, и т.д. Вышеперечисленные приложения имеют одно важное общее свойство: жесткие ограничения на максимально допустимую задержку данных в цепи «передатчик – приемник». Сетевые протоколы с гарантированной доставкой данных (TCP) не применяются в подобных приложениях, так как используют повторную пересылку данных в случае потерь, что ведет к увеличению задержек. По этой причине для приложений, работающих в режиме реального времени, используются протоколы *IP* с негарантированной доставкой пакетов, такие как UDP (User Datagram Protocol), RTP (Real Time Protocol).

Классические помехоустойчивые коды не способны решить названную проблему. Сверточные коды исправляют отдельные битовые ошибки. Блоковые коды, в том числе коды Рид–Соломона, способны исправлять пачки ошибок в отдельно взятом пакете [1]. Но в случае потери пакета целиком они бессильны. Способностью исправлять потери целых пакетов обладают стирающие коды.

Стирающие коды принципиально отличаются от рассмотренных выше классических блоковых и сверточных кодов тем, что позволяют закодировать исходное сообщение конечной длины потенциально неограниченным потоком кодовых символов. В случае необходимости стирающий кодер генерирует кодовые символы до тех пор, пока исходное сообщение не будет восстановлено получателем. Стирающие коды обладают свойством *фонтанности* [3,

4]. Свойство фонтанности состоит в том, что для стирающего кода неважен *порядок* передачи кодовых символов, и, соответственно, неважен порядок их приема в рамках одного блока кодовых символов.

1. Коды Лаби

Стирающие коды были изобретены М. Лаби в 1998, однако опубликованы впервые в 2002 году [5]. Изобретенные коды были названы в его честь «Лаби Трансформ» (ЛТ), или «преобразование Лаби». Ключом к пониманию ЛТ-кодов является описание процесса кодирования.

Допустим, что мы имеем исходное сообщение, состоящее из K исходных символов, или пакетов. Длины всех символов эквивалентны и равны L.

Введем понятие степени i-го кодового символа d_i, $i = 0, \ldots N$, где N – число кодовых символов, сгенерированных енкодером. N является потенциально бесконечной величиной.

Степень d кодового символа есть *количество* исходных символов, вовлеченных в операцию кодирования для генерации i-го кодового символа. Исходные символы, использованные для генерации некоторого кодового символа i, будем называть «соседями» [4, 5].

Введем также плотность распределения $f(d)$: для всех d, $f(d)$ есть вероятность того, что кодовый символ имеет степень d. Отметим, что плотность распределения степеней кодовых пакетов есть дискретная функция и может быть задана как аналитически, так и в виде набора значений $f(1)$, $f(2)$, ..., $f(M-1)$, $f(M)$, где M – максимальная степень кодовых символов.

Ниже приведено описание процесса генерации кодовых символов, предложенное М. Лаби [5]:

1. Случайным образом выбрать степень d кодового символа из плотности распределения степеней символов $f(d)$;
2. Выбрать случайным образом d различных исходных символов в качестве соседей кодового символа;
3. Значение кодового символа задать равным результату операции «исключающее или» (XOR) над d выбранными соседями.

Процесс генерации кодовых символов повторяется до тех пор, пока исходное сообщение не будет получателем декодировано полностью, либо на некотором шаге дальнейшее декодирование станет невозможным из-за отсутствия кодовых символов со степенью 1 [4, 5].

Ключевым аспектом разработки кодов ЛТ является разработка плотности распределения вероятности $f(d)$. Зарубежными авторами были предложены различные способы построения плотности распределения вероятности [4, 5].

Идеальным распределением называется распределение, плотность распределения вероятности которого задается согласно следующей формуле:

$$p(d) = \begin{cases} 1/K, d = 1, \\ \dfrac{1}{d(d-1)}, d = 2, \ldots, K, \end{cases} \tag{1}$$

Для обеспечения стабильного декодирования было разработано робастное распределение [5]. Плотность робастного распределения $\mu(d)$ задается формулой:

$$\mu(d) = \frac{p(d) + \tau(d)}{\sum_d p(d) + \tau(d)}, \tag{2}$$

где $p(d)$ определяется согласно (1),

$$\tau(d) = \begin{cases} \dfrac{S}{K}\dfrac{1}{d}, d = 1, 2, \ldots, (K/S) - 1, \\ \dfrac{S}{K}\log(S/\delta), d = K/S, \\ 0, d > K/S. \end{cases} \tag{3}$$

В формуле (3) S определяется выражением

$$S(K,\delta) = c\ln(K/\delta)\sqrt{K}. \quad (4)$$

Параметрами робастного распределения являются: c – параметр распределения; задается как некоторое положительное число, рекомендуется задавать $0<c<1$; δ – параметр, определяющий вероятность успешного декодирования; процесс декодирования будет успешно завершен с вероятностью $(1-\delta)$ [5].

Величина $K \cdot \sum_d p(d) + \tau(d)$ определяет количество кодовых символов, необходимых для успешного декодирования последовательности с вероятностью $(1-\delta)$.

Робастное распределение разработано для решения следующих задач: определения количества пакетов со степенью 1, необходимого для успешного полного декодирования последовательности, а также, за счет пика в точке K/S, повышения вероятности того, что каждый из исходных символов будет взят в качестве соседа хотя бы один раз.

2. Экспериментальное исследование эффективности кодов Лаби

Важнейшими характеристиками помехоустойчивого кода являются способность восстанавливать данные от ошибок и вносимый кодированием объем избыточности. Не менее важным свойством кода является *реализуемость*, т.е. возможность реализации алгоритмов кодирования и декодирования с помощью вычислительных ресурсов современных ЭВМ. В применении к помехоустойчивому кодированию в системах реального времени к реализуемости кода предъявляются повышенные требования, так как кодирование и декодирование производятся в режиме реального времени.

Известные на данный момент исследования кодов Лаби, как правило, посвящены теоретическим вопросам разработки распределений кодовых символов [4–6]. В этой связи авторами было проведено экспериментальное исследование эффективности восстановления пакетов от потерь с помощью кодов Лаби, а также произведена оценка сложности алгоритмов кодирования и декодирования.

Моделирование работы кодов Лаби осуществлялось с помощью программы *ErasureSimulator* [7]. Реализация алгоритмов кодирования и декодирования осуществлена на языке программирования *C++*.

Эксперимент строился следующим образом. Кодирование осуществлялось согласно алгоритму ЛТ с использованием робастного распределения (2). Значения параметров K, c и δ варьировались в интервалах:

$K = \{100, 500, 1000, 5000, 10000\}$,
$c = \{0.05, 0.1, 0.2, 0.5\}$,
$\delta = \{0.05, 0.1, 0.2, 0.5\}$.

Кодирование производилось для всех сочетаний триады параметров K, c и δ. При этом для каждого запуска варьировалось количество выходных символов N, генерируемых кодером ЛТ. Значения N были выбраны как $N=\{1.05\cdot K, 1.1\cdot K, 1.2\cdot K, 1.4\cdot K\}$ для каждого K. В каждой кодовой последовательности, полученной путем кодирования с параметрами $\{K, c, \delta, N\}$ производилась симуляция потерь пакетов. Процент потерь пакетов (PLR – Packet Loss Ratio) варьировался, принимая значения {0, 3, 5, 8, 15}.

В качестве критерия эффективности кода ЛТ в эксперименте использован процент успешно декодированных пакетов от общего количества исходных пакетов. Результаты эксперимента представляют собой усредненный процент успешно декодированных пакетов по $R = 1000$ запускам для каждого набора варьируемых параметров кода. Результаты эксперимента приведены в таблицах 1–3. Ввиду ограниченного объема статьи приведены лишь три таблицы для следующих комбинаций значений параметров c и δ: {0.05; 0.05}, {0.2; 0.05}, {0.2; 0.2}. Проведенный авторами анализ экспериментальных результатов, полученных для остальных сочетаний параметров, использованных в эксперименте, показал, что дальнейшее увеличение значений c и δ ведет к ухудшению эффективности кода в смысле процента восстановленных пакетов.

Таблица 1

Процент восстановлений пакетов для ЛТ: $c = 0.05$, $\delta = 0.05$

Избыток пакетов N/K, %	PLR, %	Длина исходной последовательности K, в пакетах			
		100	500	1000	5000
5	0	20	14.4	20	28.94
	3	3.8	9.88	5.47	7.36
	5	5.64	6.68	14.67	12.89
	8	18.28	16.4	20.97	4.95
	15	7.88	9.24	6.8	1.16
10	0	32	23.8	8.8	30.68
	3	21.24	18.68	6.06	2.17
	5	9.56	9.74	4.4	12.45
	8	3.68	5.58	6.27	15.4
	15	20.46	2.18	10.83	6.39
20	0	99	100	98.2	100
	3	2.88	90.69	99.02	100
	5	1	75.62	53.52	100
	8	42.98	23.96	27.89	100
	15	22.4	12.91	12.11	8.56
30	0	100	100	100	100
	3	62.52	100	100	100
	5	28.64	100	99.99	100
	8	59.58	86.78	96.74	100
	15	84.28	98.14	38.06	100

Таблица 2

Процент восстановлений пакетов для ЛТ: $c = 0.2$, $\delta = 0.05$

Избыток пакетов N/K, %	PLR, %	Длина исходной последовательности K, в пакетах			
		100	500	1000	5000
5	0	100	100	100	99.04
	3	97.6	98.08	98.12	98.78
	5	93.66	95.99	96.41	98.67
	8	89.18	89.13	88.07	97.32
	15	77.7	74.53	77.41	83.16
10	0	100	100	100	99.36
	3	98.54	98.85	98.8	98.3
	5	97.36	97.72	98.79	98.01
	8	95.1	96.53	98.52	98.16
	15	82.96	82.65	81.06	81.9
20	0	100	100	100	99.84
	3	99.46	99.25	99.3	99.14
	5	99.08	98.82	98.96	99.01
	8	98.16	98.62	99.38	98.8
	15	89.74	98.12	98.6	98.41
30	0	100	100	100	100
	3	99.78	99.9	99.92	99.61
	5	99.06	99.8	99.83	99.14
	8	97.44	99.64	99.76	98.87
	15	94.76	99.78	99.91	98.46

Таблица 3

Процент восстановлений пакетов для ЛТ: $c = 0.2$, $\delta = 0.2$

Избыток пакетов N/K, %	PLR, %	Длина исходной последовательности K, в пакетах			
		100	500	1000	5000
5	0	100	100	100	100
	3	97.7	97.78	97.65	97.32
	5	95.04	95.69	96.31	96.15
	8	92.32	89.98	90.7	85.34
	15	80.76	80.4	80.97	75.7
10	0	100	100	100	100
	3	98.12	98.97	98.78	98.43
	5	97.1	97.28	98.26	97.12
	8	95.76	96.42	96.81	92.92
	15	84.64	78.86	84.56	82.44
20	0	100	100	100	99.99
	3	99.22	99.72	99.69	98.13
	5	98.52	99.47	99.29	93.64
	8	97.98	98.9	99.21	97.88
	15	92.7	96.08	98.36	95.82
30	0	100	100	100	99.9
	3	99.86	99.61	99.55	85.58
	5	99.5	99.47	99.48	86.77
	8	99.66	99.27	99.34	85.86
	15	98.1	99.64	99.7	90.61

Для таблицы 1 ($c = 0{,}05$, $\delta = 0{,}05$) характерным является то, что кодовую последовательность практически невозможно декодировать при малом значении избытка кодовых пакетов исходных пакетов (N/K). При величине избытка, меньшем 20%, удается декодировать не более 30% исходной информации. При повышении избытка до 20% для $K = 5000$ удается полностью восстановить исходное сообщение при уровне PLR вплоть до 8%. При дальнейшем увеличении PLR происходит резкий срыв способности восстановления пакетов кодом: для PLR = 15% наблюдаем, что восстановлено лишь 8% исходных пакетов, т.е. последовательность практически не подлежит восстановлению. Дальнейшее увеличение избытка кодовых пакетов позволяет достичь улучшения результата для меньшего количества исходных пакетов: для $K = 100$ пакетов появление любых потерь все еще критично, для $K = 500$ и $K = 1000$ удается полностью (или почти полностью – 99,99%) восстановить данные вплоть до уровня PLR = 5%. При дальнейшем повышении уровня потерь декодирование становится неуверенным: 96,74% для $K = 1000$, 86,78% для $K = 500$ при PLR = 8%. При этом для $K = 5000$ удается целиком восстановить исходное сообщение даже в случае 15% потерь.

Для комбинаций $c = 0.2$, $\delta = 0.05$ (табл. 2) и $c = 0.2$, $\delta = 0.2$ (табл. 3) отметим смещение эффективности в сторону малых значений K. При отсутствии потерь последовательность полностью декодируема для $K = 100$ ... 1000, что обусловлено достаточно большим числом кодовых символов со степенью 1 в кодовой последовательности. При этом в таблице 2 для $K = 5000$ полное восстановление достигается лишь при $N/K = 30\%$. Для рассматриваемых комбинаций параметров, при одинаковых условиях передачи и равном объеме избытка, эффективность восстановления незначительно изменяется в зависимости от значения K. При этом следует отметить, что для таблицы 3 для $K = 5000$ характерен худший результат при равных PLR и N/K по сравнению с $K = 100$...1000. Очевидно, это объясняется тем, что пик функции плотности робастного распределения соответствует значению аргумента (степени кодовых пакетов d), равному отношению K/S, где S вычисляется согласно (4). Позиция пика прямо пропорциональна корню квадратному из K, и в связи с этим увеличение K ведет к

сдвигу пика вправо по оси аргумента d. При этом количество кодовых символов, обладающих данной «пиковой» степенью, снижается согласно (3). Совокупность этих факторов ведет к снижению эффективности кода.

Естественно, что эффективность восстановления повышается при увеличении избытка кодовых символов, однако для таблиц 2 и 3 это не имеет столь резкого эффекта, в отличие от таблицы 1.

Анализ таблиц 2 и 3 показывает, что кодирование Лаби имеет смысл при объеме избытка не меньшем 20%, так как при меньших значениях выигрыш, достигаемый с помощью кодирования, минимален. Проиллюстрируем вышесказанное на примере таблицы 2. Рассмотрим случай PLR = 3%, избыток 5%. Процент восстановлений составляет от 97,6 до 98,78%, что дает выигрыш от 0,6 до 1,78% от общего объема данных по сравнению со случаем, если бы кодирование не использовалось.

Полученные экспериментальные результаты позволяют сделать следующие выводы:

- использование значений $c = 0.05$, $\delta = 0.05$ предоставляет возможность достичь максимальной степени защищенности данных, однако лишь при больших значениях количества исходных пакетов K в кодируемой последовательности (от 5000 пакетов в данном эксперименте);
- увеличение значений c и δ позволяет добиться более уверенного результата для меньших значений K, а также меньшего объема избыточных пакетов, но при этом даже увеличение избытка не дает гарантии полного восстановления исходной последовательности в случае потерь кодовых пакетов;
- минимальный объем избыточной информации, необходимый для осуществления эффективного кодирования Лаби, составляет порядка 20%.

3. Оценка быстродействия кодов Лаби

Наиболее ресурсоемкой операцией в алгоритмах кодирования и декодирования стирающих кодов является операция «исключающее или», поскольку требуется выполнение данной операции над большим объемом данных. Поэтому достоверной оценкой сложности кодирования-декодирования может считаться количество вызовов операции *XOR*. Общее количество вызовов операции *XOR* определяется разрядностью регистров процессора и длиной пакета. Целесообразно использовать меру количества вызовов *XOR* на пакет, а не на байт данных, чтобы абстрагироваться от конкретного приложения с заданной длиной пакета.

Операция декодирования требует аналогичное кодированию количество вызовов операции *XOR*, поэтому для проведения оценок допустимо ограничиться вычислительной сложностью кодирования, полагая кодирование и декодирование одинаково сложными.

Экспериментальная оценка вычислительной сложности кодов ЛТ была проведена с помощью программы ErasureSimulator [7]. Оценка проводилась для значений количества входных символов $K = 100, 500, 1000, 5000$. Значения параметров функции плотности робастного распределения вероятности степеней кодовых символов (2) были заданы как $c = 0.05$, $\delta = 0.05$. Выбор значений обусловлен тем, что при данных значениях параметров коды Лаби позволяют добиться наивысшей эффективности восстановление данных от потерь. Результаты оценки вычислительной сложности кодирования ЛТ приведены в таблице 4.

Оценка C в таблице 4 представляет собой количество вызовов операции *XOR* на пакет.

Таблица 4

Оценки вычислительной сложности кодирования ЛТ

	ЛТ	
K	C	CPU_Time, сек
100	857,198	0
500	5873,002	4
1000	12787,626	10
5000	62920,497	67

Оценка CPU_Time есть количество секунд процессорного времени, потраченного 1000 запусков кодирования. Для того, чтобы получить значение оценки CPU_Time, необходимо проводить кодирование над пакетами заданной длины. В проведенном эксперименте длина пакета была задана равной 188 байт, исходя из соображений практического характера, – данная величина является длиной пакета транспортного потока MPEG-2, широко используемого в системах цифрового телевещания для передачи данных. Эксперимент проводился на компьютере, оснащенном процессором Mobile Intel Pentium с тактовой частотой 2.2 ГГц. Поскольку оценка CPU_time представляет собой усреднение по 1000 запусков, для того, чтобы определить процессорное время, требуемое для одного запуска кодирования, необходимо разделить CPU_time на 1000. Для максимального значения $K = 5000$ получим значение 0,067 секунды, т.е. порядка 7% процессорных ресурсов при тактовой частоте процессора 2,2 ГГц, что является вполне приемлемым результатом.

На основе полученных оценок быстродействия кодов ЛТ при конкретной заданной длине пакета можно ориентировочно определить предельные способности кода ЛТ к кодированию потока данных в конкретном приложении. Поскольку оценка CPU_time получена для пакетов длиной 188 байт, рассмотрим в качестве приложения систему цифрового телевещания на основе транспортного потока MPEG-2. Исходя из ограничений на задержку в цепи «передатчик – приемник», практический интерес представляют значения $K = 100 \div 500$. Для этих значений параметра K время кодирования возрастает линейно, поэтому можно воспользоваться любым из имеющихся значений оценок. Рассмотрим случай $K = 500$. Для подсчета скорости передачи бит исходного мультимедиа потока следует умножить длину транспортного пакета (188x8 бит = 1504 бит) на количество пакетов $K = 500$. Далее требуется разделить полученное произведение на значение CPU_time. Поскольку CPU_time является суммой длительностей 1000 запусков кодирования, требуется умножить скорость передачи бит потока на 1000, чтобы получить предельное значение скорости передачи бит, для которого возможно кодирование ЛТ в режиме реального времени. Производя вычисления, получим результат 188000000 бит, или ~180 мегабит в секунду. Этот результат имеет основания считаться очень хорошим, так как скорость передачи бит в современных системах цифрового телевещания, как правило, не превышает 160 Мб/с. Следует также принять во внимание, что кодирование выполнялось на отнюдь не самом мощном современном процессоре. Полученный результат позволяет сделать вывод о том, что применение кодов ЛТ в системах цифрового телевещания с точки зрения производительности алгоритмов кодирования и декодирования является возможным.

Требования к оперативной памяти определяются следующим образом: необходимо умножить количество исходных пакетов K на длину пакета L. Тем самым, минимизация требуемого объема памяти может быть достигнута путем уменьшения длины пакетов.

Рассмотрим кодирование транспортного потока MPEG-2 (длина пакета 188 байт). Для $K = 500$ требуемый объем памяти для буферизации исходной последовательности составит всего лишь 94000 байт, что не является критичным для современных ЭВМ.

Таким образом, можно сделать вывод о том, что требования кодов Лаби к объему оперативной памяти не являются препятствием для практической реализации кода.

Заключение

Результаты эксперимента по оценке производительности кодов Лаби позволяют сделать положительный вывод о возможности практической реализации алгоритмов кодирования и декодирования с помощью современных вычислительных устройств. Экспериментально подтверждено, что наивысшая эффективность восстановления данных с помощью кодов ЛТ на основе робастного распределения степеней кодовых пакетов достигается при больших значениях количества исходных пакетов (порядка 5000 и выше). Варьирование параметров робастного распределения позволяет добиться высокой эффективности восстановления данных от потерь (порядка 98–100%) при меньших значениях количества исходных пакетов. В качестве платы за эффективное восстановление пакетов от потерь коды Лаби требуют существенного объема избыточной информации, вносимой кодированием (не менее 20–30%).

Литература

1. Питерсон У. Коды, исправляющие ошибки / пер. с англ. под ред. Р. Добрушина – М. : Мир, 1972. – 338 с.

2. Варгаузин В.А. Турбо-коды и итеративное декодирование: принципы, свойства, применение / В.А. Варгаузин, Л.Н. Протопопов // ТелеМультиМедиа. – 2000. – №34. – С. 33–38.

3. Варгаузин В. Помехоустойчивое кодирование в пакетных сетях // ТелеМультиМедиа. – 2005. – №3. – С.10–16.

4. MacKay D.J.C. Fountain codes // IEE Proc.-Commun., 2005. – Vol.152. – №6 (December). – P. 1062–1068.

5. Luby M. LT Codes // Proc. of the 43rd Annual IEEE Symp. on Foundations of Computer Science (FOCS). – 2002. – P. 271–282.

6. Byers J. A digital foundation approach to asynchronous reliable multicast / J. Byers, M. Luby, M. Mitzenmacher // IEEE J. Selected Areas Comm. – 2002. – Vol.20. – P. 1528–1540.

7. Шинкаренко К.В. Программа моделирования помехоустойчивого кодирования стирающими кодами ErasureSimulator // Отраслевой фонд алгоритмов и программ. Свидетельство об отраслевой регистрации разработки №11720. Номер государственной регистрации : 50200802200. – 2008.

Шинкаренко Константин Всеволодович
М.н.с. кафедры автоматизированных систем управления (АСУ) ТУСУРа
Тел.: (3822) 41-42-79
Эл. почта: shinkarenko_k@mail.ru

Кориков Анатолий Михайлович
Доктор техн. наук, профессор, заведующий кафедрой АСУ ТУСУРа
Тел.: (3822) 41-42-79
Эл. почта: korikov@asu.tusur.ru

K.V. Shinkarenko, A.M. Korikov

Luby Transform error-correcting codes effectiveness estimation

The existing approaches for packet-loss recovery in packet networks are reviewed. It is determined, that ensure error-correcting codes are able to recover complete packet losses. The experimental results of Luby Transform codes effectiveness and performance estimation are presented.

Keywords: error-correcting coding, loss recovery, erasure codes, Luby Transform codes.
