УДК 004.942
doi:10.15217/issn1684-8853.2017.4.44

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ НАСТРАИВАЕМОЙ ОЧЕРЕДЬЮ ИЗ ДВУХ ПОСЛЕДОВАТЕЛЬНЫХ ЦИКЛИЧЕСКИХ FIFO-ОЧЕРЕДЕЙ В ОБЩЕЙ ПАМЯТИ

А. М. Сазонов[а], *магистрант*
А. В. Соколов[б], *доктор физ.-мат. наук, профессор*
[а]*Петрозаводский государственный университет, Петрозаводск, РФ*
[б]*Институт прикладных математических исследований Карельского научного центра Российской академии наук, Петрозаводск, РФ*

Введение: *при разработке многих аппаратных и программных приложений применяют структуру данных «настраиваемая очередь» (Custom Queuing). В различных сетевых устройствах и встроенных операционных системах настраиваемая очередь представлена в виде нескольких последовательных циклических FIFO-очередей, расположенных в общем пространстве памяти.* ***Цель:*** *построить и проанализировать математическую модель процесса работы с настраиваемой очередью, представленной в виде двух последовательных циклических FIFO-очередей, в целях повышения стабильности работы системы.* ***Результаты:*** *построена математическая модель процесса работы с настраиваемой очередью, в которой на каждом шаге дискретного времени происходят операции включения и исключения элементов в одну из очередей. Математическая модель представлена в виде случайного блуждания по двухмерной целочисленной решетке, для которой строится соответствующая регулярная однородная цепь Маркова. Критерием оптимальности является минимальная средняя доля потерянных при переполнении элементов очередей. Проведены численные эксперименты, основывающиеся на теоретических данных.* ***Практическая значимость:*** *с помощью разработанной модели можно найти оптимальное назначение весов каждой FIFO-очереди для повышения стабильности работы системы. Предложенные модели, алгоритмы и разработанный программный комплекс могут применяться при проектировании сетевых устройств, например маршрутизаторов, где потери пакетов являются допустимой, но нежелательной ситуацией. Назначая веса для очередей оптимально, мы теряем меньше пакетов, и, как следствие, данные доставляются быстрее.*

Ключевые слова *— структуры данных, настраиваемая очередь (Custom Queue), случайные блуждания, регулярные цепи Маркова.*

Введение

Во многих приложениях, например при разработке различных сетевых устройств и встроенных операционных систем, требуется работа с несколькими очередями, расположенными в общем пространстве памяти. Механизм страничной виртуальной памяти здесь не используется, и вся работа происходит в нескольких пулах оперативной памяти. Существует несколько дисциплин обслуживания очередей в сетевых устройствах: FIFO, приоритетные, настраиваемые и др. Существуют и комбинированные схемы работы с очередями, например, одна очередь приоритетная, а остальные обслуживаются по схеме настраиваемых очередей.

Для представления FIFO-очередей применяют различные программные или аппаратные решения [1–5].

Модели последовательного, связанного и страничного способов представления нескольких FIFO-очередей в памяти одного уровня описаны в работах [6–12]; решаются задачи оптимального управления приоритетными очередями [13] и решается задача оптимального разбиения общей памяти для настраиваемой очереди, представленной в виде двух очередей в случае их последовательного циклического представления [14].

В этих моделях предполагается, что на каждом шаге дискретного времени происходят некоторые операции со структурами данных (с заданными вероятностями). Так как время выполнения операций не случайная величина, а константа, фиксированным является и шаг времени. Первоначально такие модели в виде случайного блуждания в треугольнике [15–19] были построены для решения задачи анализа процесса работы с двумя стеками, растущими навстречу друг другу [4].

Обзор некоторых методов работы с очередями в сетевых устройствах

Одна из распространенных стратегий обслуживания — это настраиваемые очереди (Custom Queuing [3]), или взвешенные очереди в другой терминологии [2]. Механизм настраиваемых очередей разработан для того, чтобы можно было предоставить всем классам трафика какой-то минимум пропускной способности. Под весом данного класса понимается процент предоставляемой классу трафика пропускной способности

от полной пропускной способности выходного интерфейса.

При взвешенном обслуживании, так же, как и при приоритетном, трафик делится на несколько классов, и для каждого класса ведется отдельная очередь пакетов. Но каждой очереди назначается не приоритет, а доля пропускной способности ресурса, предоставляемая данному классу трафика при его перегрузках. Для входного потока таким ресурсом может быть процессор, а для выходного — выходной интерфейс. Custom Queuing обрабатывает трафик, ориентируясь на количество пакетов или байтов, которые надо отправить. Очереди обслуживаются по алгоритму round-robin — часть пакетов обрабатывается и пересылается для каждой очереди.

В данной работе мы предлагаем математическую модель и решаем задачу нахождения оптимального назначения весов каждой FIFO-очереди для повышения стабильности работы системы.

На каждом шаге дискретного времени происходят операции включения или исключения элементов в одну из очередей с заданными вероятностями. Исключение элементов происходит одновременно из обеих очередей во время так называемого цикла просмотра очередей. Количество исключаемых элементов из первой очереди — c_1, из второй — c_2. Заметим, что в этой задаче мы не можем выбирать параметры c_1 и c_2 произвольно, так как они зависят от цикла просмотра очередей в рассматриваемом сетевом устройстве t и скорости выходного трафика v ($c_1 + c_2 = vt$). Например, если $t = 2$ с, а $v = 100$ Мбит/с, то $c_1 + c_2 = 200$ Мбит. Это значит, что в зависимости от вероятностных характеристик очередей мы можем менять веса очередей, но их сумма определяется характеристиками сети и устройства. На практике в большинстве сетевых протоколов данные выбираются из очереди пакетами, а не битами, и реальное распределение ресурса между классами трафика несколько отличается от планируемого. Чем больше время цикла, тем точнее соблюдаются требуемые веса классов трафика. Но, с другой стороны, большой цикл приводит к большим задержкам времени нахождения пакетов в сетевом устройстве. Поэтому при выборе времени цикла нужно удерживать баланс между точностью соблюдения весов классов и стремлением к уменьшению задержки.

В данной статье мы рассматриваем модель настраиваемой очереди для двух очередей, входящих в настраиваемую очередь, и считаем, что пакеты (элементы очередей) имеют одинаковую длину. На практике существуют сетевые протоколы, например, многие годы широко использовалась в компьютерных сетях технология ATM, в которых длина пакета фиксирована. В ATM она была равна 53 Б. Такие размеры позволяли сетям ATM качественно передавать видео- и аудиотрафики. Причины снижения популярности этой технологии заключены в ее сложности, влиянии конкурирующих технологий и др. [2]. Можно привести много аргументов в пользу применения пакетов фиксированной или переменной длины в компьютерных сетях. Так как история развития компьютерных сетей имеет относительно небольшой период, трудно предсказать, какие длины пакетов будут использоваться в будущем, и сейчас представляется, что нужно анализировать оба этих подхода к выбору длин пакетов.

Построение моделей, аналогичных приведенной в данной статье, в случае пакетов разных длин будет существенно сложнее. В работах, где близкие к предлагаемой модели строятся на основе теории массового обслуживания, обычно также рассматриваются пакеты фиксированной длины [20–22]. Можно выделить работу [23], где строится модель для пакетов случайной длины, но задачи оптимизации не рассматриваются. Пакеты разных длин могут использоваться, например, в случае последовательного циклического представления FIFO-очереди, но потребуются некоторые изменения метода представления, так как при достижении конца буфера, выделенного для очереди, нужно будет предусмотреть механизм, который позволит выделить для нового пакета достаточный блок свободной памяти в начале буфера.

Модели систем с групповым поступлением заявок не являются моделями работы с очередями с пакетами разных длин, так как в этих моделях группы заявок, принятые в очередь для ожидания обслуживания, поступают на прибор в порядке поступления, а на обслуживание заявка из группы выбирается случайным образом. Модели работы с очередями с пакетами разных длин должны, в отличие от моделей систем с групповым поступлением заявок, удалять из очереди пакеты переменной длины целиком. В моделях процесса буферизации данных в потоковых одноранговых сетях под порцией данных понимается минимальный неделимый фрагмент потоковых данных фиксированного размера от 16 до 4096 кБ в зависимости от разновидности протокола потоковой сети P2P.

Отметим также, что случай двух очередей важен не только как первый шаг к построению общей модели, но у него есть и свое независимое значение.

Так, в работе [2, с. 476] приводятся следующие пояснения по поводу возможности разбиения всего трафика на две очереди: «В приложении G стандарта 802.1 D-2004 даются рекомендации по разделению всего трафика локальных сетей на семь классов. <...>

Коммутатор обычно поддерживает некоторое максимальное количество очередей, которое мо-

жет оказаться меньше, чем требуемое число классов трафика. В этой ситуации несколько классов будут обслуживаться одной очередью, то есть фактически сольются в один класс. Стандарт 802.1D-2004 дает рекомендации в отношении того, какие классы трафика нужно реализовывать в сети в условиях ограниченного количества очередей в коммутаторах.

При существовании только одной очереди в сети все классы трафика обслуживаются этой очередью. <...>

Две очереди дают возможность дифференцированно обслуживать группы классов трафика — менее требовательные классы Bk, BE и EE в одной очереди, а более требовательные классы VO, CL, VI, NC — в другой.

Дальнейшее увеличение количества очередей позволяет более дифференцированно обслуживать трафик, вплоть до рекомендуемых семи классов».

Здесь также можно сослаться на опыт в области построения многоядерных процессоров. Среди многоядерных архитектур есть такие, в которых отсутствует кэш-память. Для примера, в архитектуре AsAP-II каждое ядро имеет два FIFO-буфера, а в архитектуре SEAforth — два стека (для хранения данных и адресов возврата) [5]. Очереди и стеки реализованы циклически и отделены друг от друга, также допускается потеря элементов при переполнении.

Эти процессоры используются в системах, где потеря элементов очередей, как и в сетевых приложениях, является допустимой, но нежелательной ситуацией (например, цифровая обработка сигналов, работа с мультимедийными приложениями и др.). Настраиваемые очереди могут быть реализованы аппаратно, по такому же принципу. Тогда важной задачей является исследование оптимального управления двумя очередями. В случае произвольного количества очередей можно конструировать требуемые микросхемы из микросхем, состоящих из двух очередей.

В качестве критерия оптимальности рассмотрена минимальная средняя доля потерянных при переполнении элементов на бесконечном времени работы. Этот критерий оптимальности стоит рассматривать, если переполнение очереди является стандартной ситуацией. Когда размер очереди превышает размер предоставленной ей памяти, все поступающие в нее элементы отбрасываются до тех пор, пока не появится свободный участок памяти (этот участок появится только после исключения элемента из очереди). Сетевые маршрутизаторы [2, 3] работают по этой схеме. Такие потери элементов приводят к нежелательному результату. Например, в некоторых сетевых протоколах отброшенные пакеты будут посылаться снова, что приведет к замедлению работы системы. Поэтому число таких ситуаций необходимо свести к минимуму.

В данной работе целью ставится минимизация вероятности потерь в системе в целом. Если же нужно оптимизировать показатели каждого класса трафика в отдельности с учетом требований QoS, то критерий оптимальности в модели следует изменить (мы сейчас предполагаем, что требования QoS касаются только потерь данных). Нужно будет учитывать средние доли потерянных пакетов при переполнении размера очереди каждого класса трафика и минимизировать потери с учетом весов, заявленных требованиями QoS.

Например, во второй строке таблицы (см. ниже) $P_{loss} = 0{,}0171815474618332$. Это означает, что из тысячи пакетов в среднем будет теряться 17 пакетов, и нам не важно, пакеты какого класса трафика теряются. Другими словами, мы считаем, что требования QoS установлены таким образом, что показатели важности всех классов трафика равны. Если же важность первого класса трафика в два раза больше, то мы должны оптимизировать обработку пакетов так, чтобы из потерянных 17 пакетов шесть пакетов были первого класса, а 11 пакетов — второго. То есть пакеты первого класса будут теряться примерно в два раза реже. Модель и алгоритм для решения задачи с таким критерием останутся такими же. Также нужно будет решать задачу целочисленного нелинейного программирования, где функция критерия оптимальности задается алгоритмически. Эта задача в наших работах решается численно, а аналитические результаты удалось получить только в некоторых случаях задач для FIFO-очередей [7, 10].

В будущем будет интересно рассмотреть задачи оптимального управления памятью для настраиваемой очереди для такого критерия, а также задачи, когда в качестве управляемого параметра выступают не только доли пропускной способности ресурса, предоставляемого каждому классу трафика при его перегрузках, но и размеры очередей каждого класса трафика, как это было в задачах оптимального управления несколькими FIFO-очередями в общей памяти [6–12].

Математическая модель

Пусть в памяти размера n единиц мы работаем с двумя настраиваемыми очередями и каждой очереди выделен размер памяти $k = n/2$. Будем искать такие c_1, c_2, чтобы вероятность потери в стационарном режиме в обеих очередях P_{loss} была минимальной.

Пусть p_1 и p_2 — вероятности включения элемента в первую и вторую очереди соответственно, q — вероятность исключения элементов из первой и второй очередей.

Поскольку построенная на основе такой постановки задачи марковская цепь не будет являться регулярной и однородной, шаги цикла просмотра очередей, во время которого происходят исключения элементов, объединяем в один, а также вводим вероятность операции, не изменяющей длины очередей (например, чтение), r. Соответственно: $p_1 + p_2 + q + r = 1$.

Тогда состояние на каждом шаге определяется наступлением одного из следующих событий:

— включения в первую очередь с вероятностью p_1;

— включения во вторую очередь с вероятностью p_2;

— исключения из первой очереди c_1 и из второй очереди c_2 элементов с вероятностью q;

— операции, не изменяющей длину очереди, с вероятностью r,

— где сумма всех вероятностей равна 1.

Предполагается, что в очередях хранятся данные фиксированного размера. При исключении информации из пустой очереди не происходит завершения работы.

Обозначим через x и y текущие длины очередей. Пусть (x, y) — текущее состояние процесса, тогда блуждание по прямоугольной решетке можно описать следующим образом:

$$(x, y) \xrightarrow[r]{} (x, y);$$

$$(x, y) \xrightarrow[p_1]{} (x', y') = \begin{cases} (x+1, y), & x < k \\ (x, y), & x = k \end{cases};$$

$$(x, y) \xrightarrow[p_2]{} (x', y') = \begin{cases} (x, y+1), & y < k \\ (x, y), & y = k \end{cases};$$

$$(x, y) \xrightarrow[q]{} (x', y') = \begin{cases} (x - c_1, y - c_2), \\ c_1 + 1 \le x \le k,\ c_2 + 1 \le y \le k \\ (0, y - c_2 - (c_1 - x)), \\ x \le c_1,\ c_1 + c_2 - x + 1 \le y \le k. \\ (x - c_1 - (c_2 - y), 0), \\ c_1 + c_2 - y + 1 \le x \le k,\ y \le c_2 \\ (0, 0), \text{ иначе} \end{cases}$$

Случайное блуждание рассматриваем в виде регулярной конечной цепи Маркова с переходной матрицей **P**.

Нумерацию начинаем с нуля, общее количество состояний в цепи будет

$$N = (k+1)(k+1).$$

Далее, согласно введенной нумерации состояний и вышеуказанной схеме переходов, составляем матрицу переходных вероятностей **P**. В данном исследовании был установлен вид матрицы **P** для произвольных значений параметра ОВ C_1, C_2. При составлении матрицы для каждого конкретного состояния определяем те состояния, в которые процесс переходит при выполнении допустимых операций, и вычисляем соответствующие вероятности переходов. Данный процесс был автоматизирован с помощью программы на языке С++.

Следующим шагом является решение уравнения $\boldsymbol{\alpha}\mathbf{P} = \boldsymbol{\alpha}$, где $\boldsymbol{\alpha}$ — предельный вектор для полученной марковской цепи.

Элемент вектора $\boldsymbol{\alpha}_i$ — это средняя доля времени, которое процесс проводит в состоянии i [24]. Поскольку моменты прихода не зависят от числа элементов в очереди, для вычисления предельной вероятности потери нужно просуммировать элементы вектора $\boldsymbol{\alpha}$, соответствующие состояниям, когда первая очередь максимально заполнена, умножив их на p_1, и элементы вектора $\boldsymbol{\alpha}$, соответствующие состояниям, когда вторая очередь максимально заполнена, умножив их на p_2. При введенной нумерации это будут элементы с номерами $k + (k + 1)j$, $j = 0, ..., k$ и последние k элементов вектора.

Оценка сложности алгоритма:

$$O\left(((k+1)(k+1))^3\right).$$

Некоторые примеры численного анализа. В таблице приведены оптимальные значения c_1, c_2 для $n = 40$, $\upsilon = 8$, $t = 1$ при различных вероятностных характеристиках.

Исходя из численного анализа, можно отметить, что из очереди с большим значением вероятности прихода p_i во время цикла просмотра оп-

■ Оптимальные значения c_1, c_2

■ Optimal values c_1, c_2

Вероятностные характеристики				c_1	c_2	P_{loss}
q	p_1	p_2	r			
0,1	0,2	0,1	0,6	7	1	0,0000845502396969
0,1	0,2	0,4	0,3	2	6	0,0171815474618332
0,1	0,2	0,5	0,2	1	7	0,0463256286690630
0,1	0,3	0,2	0,4	6	2	0,0041560682040080
0,1	0,3	0,3	0,3	4	4	0,0144698437336146
0,1	0,3	0,4	0,2	3	5	0,0415428932630922
0,1	0,3	0,5	0,1	3	5	0,0895292282853496
0,1	0,4	0,1	0,4	7	1	0,0095622291753297
0,1	0,4	0,3	0,2	5	3	0,0415428932630900
0,2	0,2	0,4	0,2	1	7	0,0001691004793931
0,2	0,3	0,4	0,1	3	5	0,0003374480688416
0,3	0,1	0,2	0,4	1	7	0,0000000022263685
0,3	0,3	0,2	0,2	7	1	0,0000003189925640
0,3	0,3	0,3	0,1	4	4	0,0000009211925787

тимально исключать больше элементов (т. е. соответствующее значение c_i большее). Кроме того, с увеличением вероятности исключения увеличивается разница между значениями c_1 и c_2, поскольку вероятность потери в очереди с меньшей вероятностью прихода снизится за счет более частого исключения элементов. Данные выводы совпадают с интуитивными предположениями.

Заключение

В статье предложена и проанализирована математическая модель, которая описывает способ управления работой двумя параллельными настраиваемыми очередями в общей памяти. Данная модель может быть полезна при разработке различных технических систем, в архитектуре которых необходим именно такой способ управления.

Несмотря на то, что в настоящее время память становится все больше и дешевле, есть устройства, где ее размер ограничен архитектурой, и имеющийся ресурс необходимо использовать эффективно. Примером могут служить различные сетевые устройства, такие как маршрутизаторы. В них размер памяти ограничен, поскольку при больших размерах очередей значительно увеличивается время обработки элементов. Вследствие этого возникает проблема: сетевые протоколы устроены так, что при большом времени обработки пакеты считаются потерянными, и их посылают снова, тем самым увеличивая загрузку сети [25]. Механизм настраиваемых очередей используется в маршрутизаторах системы Cisco [3].

Кроме того, к устройствам с ограниченным размером памяти относятся микросхемы и микропроцессоры, которые используются в системах, где потеря элементов очередей является допустимой, но нежелательной ситуацией (например, цифровая обработка сигналов или работа с мультимедийными приложениями [5]).

Приносим благодарность рецензентам статьи за полезные критические замечания, которые, как мы надеемся, приведут к решению новых задач оптимизации показателей каждого класса трафика в отдельности с учетом требований QoS.

Работа выполнена при финансовой поддержке РФФИ, грант 15-01-03404-а.

Литература

1. **Sedgewick R.** Algorithms in C++. Parts 1–4. — Addison-Wesley Professional, 1998. — 752 p.
2. **Олифер В. Г., Олифер Н. А.** Компьютерные сети. Принципы, технологии, протоколы. — СПб.: Питер, 2010. — 944 с.
3. **Bollapragada V., Murphy C., White R.** Inside Cisco IOS Software Architecture. — Cisco Press, 2000. — 240 p.
4. **Knuth D.** The Art of Computer Programming. Vol. 1. — Addison-Wesley Professional, 1997. — 672 p.
5. **Калачев А. В.** Многоядерные процессоры. — М.: БИНОМ, 2014. — 247 с.
6. **Соколов А. В.** Математические модели и алгоритмы оптимального управления динамическими структурами данных. — Петрозаводск: ПетрГУ, 2002. — 216 с.
7. **Аксенова Е. А., Драц А. В., Соколов А. В.** Оптимальное управление n FIFO-очередями на бесконечном времени // Информационно-управляющие системы. 2009. № 6. С. 46–54.
8. **Aksenova E. A., Sokolov A. V.** The Optimal Implementation of Two FIFO-Queues in Single-Level Memory // Applied Mathematics. 2011. Vol. 2. P. 1297–1302.
9. **Sokolov A. V., Drac A. V.** The Linked List Representation of n LIFO-Stacks and/or FIFO-Queues in the Single-Level Memory // Information Processing Letters. 2013. Vol. 13. P. 832–835.
10. **Соколов А. В., Сазонов А. М., Морозов Е. В., Некрасова Р. С., Разумчик Р. В.** Математические модели и алгоритмы оптимального управления FIFO-очередями в общей памяти// Тр. КарНЦ РАН. Сер. Математическое моделирование и информационные технологии. 2016. № 8. С. 98–107. doi:10.17076/mat396
11. **Соколов А. В., Барковский Е. А.** Оптимальное управление двумя параллельными FIFO-очередями на бесконечном времени // Информационно-управляющие системы. 2015. № 5. С. 65–71. doi:10.15217/issn1684-8853.2015.5.65
12. **Sokolov A. V., Barkovsky E. A.** Some Problems of Optimal Control of Two Parallel FIFO-queues // International Conference on Numerical Analysis and Applied Mathematics: Proc. 12th Intern. Conf., Rhodes, 18–22 Sept. 2014. AIP Publishing, 2015. Vol. 1648. P. 520003.
13. **Драц А. В., Соколов А. В.** Оптимальное управление приоритетной очередью в памяти одного уровня// Тр. КарНЦ РАН. 2011. № 2. С. 103–110.
14. **Барковский Е. А.** Оптимальное управление двумя очередями, работающими по принципу настраиваемых очередей// Стохастическая оптимизация в информатике. 2016. Т. 12. Вып. 2. С. 1–14. http://www.math.spbu.ru/user/gran/optstoch.htm
15. **Соколов А. В.** О распределении памяти для двух стеков // Автоматизация эксперимента и обработки данных. — Петрозаводск: Изд-во Карельского филиала АН СССР, 1980. — С. 65–71.
16. **Yao A. C.** An Analysis of a Memory Allocation Scheme for Implementing Stacks // SIAM Journal on Computing. 1981. Vol. 10. P. 398–403.
17. **Flajolet P.** The Evolution of Two Stacks in Bounded Space and Random Walks in a Triangle // Lecture Notes in Computer Science. 1986. Vol. 223. P. 325–340.

18. **Louchard G., Schott R., Tolley M., Zimmermann P.** Random Walks, Heat Equation and Distributed Algorithms // Journal of Computational and Applied Mathematics. 1994. N 53. P. 243–274.
19. **Maier R. S.** Colliding Stacks: A Large Deviations Analysis // Random Structures and Algorithms. 1991. N 2. P. 379–421.
20. **Ozel O., Uysal-Biyikoglu E., and Girici T.** Optimal Buffer Partitioning on a Multiuser Wireless Link// Proc. of Information Theory and Applications Workshop, UCSD, San Diego, CA, Jan. 31–Feb. 5, 2010. doi:10.1109/ITA.2010.5454079
21. **Гайдамака Ю. В., Самуйлов А. К.** Анализ стратегий заполнения буфера оборудования пользователя при предоставлении услуги потокового видео в одноранговой сети // T-Comm — Телекоммуникации и Транспорт. 2013. № 11. С. 77–81.
22. **Gaidamaka Yu., Shorgin S., Samouylov K., Etezov Sh.** Polling System with Threshold Control for Modeling of SIP Server under Overload / J. Swiatek et al. (Eds.) // Advances in Systems Science, AISC 240. 2014. P. 97–107. doi:10.1007/978–3–319–01857–7_10
23. **Morozov E., Nekrasova R., Potakhina L., Tikhonenko O.** Asymptotic Analysis of Queueing Systems with Finite Buffer Space // Communications in Computer and Information Science: Proc. 21st Intern. Conf., CN 2014, Brunów, Poland, June 23–27, 2014. Vol. 431. P. 223–232.
24. **Кемени Дж., Снелл Дж.** Конечные цепи Маркова. — М.: Наука, 1970. — 272 с.
25. **Танненбаум Э., Уэзеролл В.** Компьютерные сети. 5-е изд. — СПб.: Питер, 2012. — 960 с.

UDC 004.942
doi:10.15217/issn1684-8853.2017.4.44

Mathematical Model of Optimal Memory Management for Custom Queue of Two Consecutive Cyclic FIFOs in Shared Memory

Sazonov A. M.[a], Master Student, sazon-tb@mail.ru
Sokolov A. V.[b], Dr. Sc., Phys.-Math., Professor, avs@krc.karelia.ru
[a]Petrozavodsk State University, 33, Lenin St., 185910, Petrozavodsk, Russian Federation
[b]Institute of Applied Mathematical Research of Karelian Research Centre Russian Academy of Sciences, 11, Pushkinskaya St., 185910, Petrozavodsk, Russian Federation

Introduction: Custom queue is a data structure used in many hardware and software applications. In various network devices and embedded operating systems, a custom queue is implemented as several successive cyclic FIFO queues located in a shared memory space. **Purpose:** In order to increase the system stability, we need to build and analyze a mathematical model of the custom queue operation process, represented as two consecutive cyclic FIFO queues. **Results:** A mathematical model has been built for a custom queue. At each step of discrete time, it performs operations of element insertion into and deletion from one of the queues. The mathematical model is a random walk on a two-dimensional integer lattice with reflecting screens, i.e. we deal with a regular uniform Markov chain. The optimality criterion is the lowest average fraction of the queue elements lost due to overflow. Numerical experiments based on theoretical data have been performed. **Practical relevance:** Using the developed model, you can find the optimal assignment of weights for each FIFO queue and increase the system stability. The proposed models, algorithms and software can be used in the design of network devices like routers where a packet loss is enabled but unwanted. When weights are assigned to the queues in the optimal way, you lose fewer packets, and, consequently, data is delivered quicker.
Keywords — Data Structures, Custom Queue, Random Walks, Regular Markov Chain.

References

1. Sedgewick R. *Algorithms in C++*. Parts 1–4. Addison-Wesley Professional, 1998. 752 p.
2. Olifer V. G., Olifer N. A. *Komp'iuternye seti. Printsipy, tekhnologii, protokoly* [Networks. Principles, Technologies, Protocols]. 4th ed. Saint-Petersburg, Piter Publ., 2010. 944 p.
3. Bollapragada V., Murphy C., White R. *Inside Cisco IOS Software Architecture*. Cisco Press, 2000. 240 p.
4. Knuth D. *The Art of Computer Programming*. Vol. 1. Addison-Wesley Professional, 1997. 672 p.
5. Kalachev A. V. *Mnogoiadernye protsessory* [Multicore Architectures]. Moscow, BINOM Publ., 2014. 247 p. (In Russian).
6. Sokolov A. V. *Matematicheskie modeli i algoritmy optimal'nogo upravleniia dinamicheskimi strukturami dannykh* [Mathematical Models and Algorithms of Optimal Control of Dynamic Data Structures]. Petrozavodsk, PetrGU Publ., 2002. 216 p. (In Russian).
7. Aksenova E. A., Drac A. V., Sokolov A. V. Optimal Control of the N FIFO-Queues for Infinity Time. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2009, no. 6, pp. 46–54 (In Russian).
8. Aksenova E. A., Sokolov A. V. The Optimal Implementation of Two FIFO-Queues in Single-Level Memory. *Applied Mathematics*, 2011, vol. 2, pp. 1297–1302.
9. Sokolov A. V., Drac A. V. The Linked List Representation of n LIFO-Stacks and/or FIFO-Queues in the Single-Level Memory. *Information Processing Letters*, 2013, vol. 13, pp. 832–835.
10. Sokolov A. V., Sazonov A. M., Morozov E. V., Nekrasova R. S., Razumchik R. V. Mathematical Models and Algorithms for the Optimal Control of FIFO-Queues in Shared Memory. *Trudy KarNTs RAN. Ser. Matematicheskoe modelirovanie i informatsionnye tekhnologii* [Proc. of Karelian Research Centre of Russian Academy of Sciences. Ser. Mathematical Modeling and Information Technology], 2016, no. 8, pp. 98–107 (In Russian). doi:10.17076 / mat396
11. Barkovsky E. A., Sokolov A. V. Optimal Control of Two Parallel FIFO Queues on an Infinite Time. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2015, no. 5, pp. 65–71 (In Russian). doi:10.15217/issn1684-8853.2015.5.65
12. Sokolov A. V., Barkovsky E. A. Some Problems of Optimal Control of Two Parallel FIFO-queues. *Proc. 12th Intern.*

Conf. Numerical Analysis and Applied Mathematics", AIP Publishing, 2015, vol. 1648, pp. 520003.
13. Drac A. V., Sokolov A. V. Optimum Control of Queue with Priorityes in one Level Memory. *Trudy KarNC RAN*, 2011, no. 2, pp. 103–110 (In Russian).
14. Barkovsky E. A. Optimal Control of two Custom Queues. *Stokhasticheskaia optimizatsiia v informatike*, 2016, vol. 12, iss. 2, pp. 1–14 (In Russian).
15. Sokolov A. V. About Memory Distribution for Two Stacks. In: *Avtomatizatsiia eksperimenta i obrabotki dannykh*. Petrozavodsk, Karelskii filial AN SSSR Publ., 1980. Pp. 65–71 (In Russian).
16. Yao A. C. An Analysis of a Memory Allocation Scheme for Implementing Stacks. *SIAM Journal on Computing*, 1981, vol. 10, pp. 398–403.
17. Flajolet P. The Evolution of Two Stacks in Bounded Space and Random Walks in a Triangle. *Lecture Notes in Computer Science*, 1986, vol. 223, pp. 325–340.
18. Louchard G., Schott R., Tolley M., Zimmermann P. Random Walks, Heat Equation and Distributed Algorithms. *Journal of Computational and Applied Mathematics*, 1994, no. 53, pp. 243–274.
19. Maier R. S. Colliding Stacks: A Large Deviations Analysis. *Random Structures and Algorithms*, 1991, no. 2, pp. 379–421.
20. Ozel O., Uysal-Biyikoglu E., and Girici T. Optimal Buffer Partitioning on a Multiuser Wireless Link. *Proc. of Information Theory and Applications Workshop*, UCSD, San Diego, CA, Jan. 31–Feb. 5, 2010. doi:10.1109/ITA.2010.5454079
21. Gaidamaka Yu., Samouylov K. Analysis of Strategies to Fill the user Equipment Buffer when Providing a Streaming Video Service in a Peer-To-Peer Network. *T-Comm — Telekommunikatsii i Transport*, 2013, no. 11, pp. 77–81 (In Russian).
22. Gaidamaka Yu., Shorgin S., Samouylov K., Etezov Sh. Polling System with Threshold Control for Modeling of SIP Server under Overload. J. Swiatek et al. (Eds.). *Advances in Systems Science*, AISC 240, Springer Int. Publishing Switzerland, 2014, pp. 97–107. doi:10.1007/978–3–319–01857–7_10
23. Morozov E., Nekrasova R., Potakhina L., Tikhonenko O. Asymptotic Analysis of Queueing Systems with Finite Buffer Space. *Proc. of 21st Intern. Conf. "Communications in Computer and Information Science", CN 2014*, Brunów, Poland, June 23–27, 2014, vol. 431, pp. 223–232.
24. Kemeny J. G., Snell J. L. *Finite Markov Chains*. Van Nostrand, 1969. 210 p.
25. Tanenbaum A. S., Wetherall D. J. *Computer Networks*. 5th ed. Pearson Education, 2011. 933 p.

ПАМЯТКА ДЛЯ АВТОРОВ

Поступающие в редакцию статьи проходят обязательное рецензирование.

При наличии положительной рецензии статья рассматривается редакционной коллегией. Принятая в печать статья направляется автору для согласования редакторских правок. После согласования автор представляет в редакцию окончательный вариант текста статьи.

Процедуры согласования текста статьи могут осуществляться как непосредственно в редакции, так и по e-mail (ius.spb@gmail.com).

При отклонении статьи редакция представляет автору мотивированное заключение и рецензию, при необходимости доработать статью — рецензию. Рукописи не возвращаются.

Редакция журнала напоминает, что ответственность за достоверность и точность рекламных материалов несут рекламодатели.