Инфокоммуникационные технологии и системы

УДК 621.391 **DOI: 10.14529/ctcr190105**

АЛГОРИТМЫ КОНВЕЙЕРНОГО ИНТЕРВАЛЬНОГО АНАЛИЗА ТРАФИКА

В.И. Моисеев[1, 2], Б.Я. Лихтциндер[2]

[1] Пермский государственный национальный исследовательский университет, г. Пермь, Россия,
[2] Поволжский государственный университет телекоммуникаций и информатики, г. Самара, Россия

Рассматривается модель участка доступа мультисервисной сети оператора связи от магистральной линии до порта абонента. Рассматривается трафик мультисервисной сети на данном участке. Сформулированы проблемы, возникающие при анализе свойств очередей IP-TV видеотрафика стандарта H.264 в реальном времени. Приводится описание интервального метода анализа трафика и предлагается ряд алгоритмов, расширяющих и дополняющих этот метод. Интервальный метод дополняется возможностью анализа реальных потоков трафика заявок произвольной длины. Рассматривается вопрос выбора единицы квантизации длины заявки. За единицу времени обработки заявки в интервальном методе принимается время обработки исходящим каналом 1 КБ данных. Подтверждена справедливость формул интервального метода для расчета дифференциальной интенсивности событий и дифференциального коэффициента загрузки. Рассматривается работа интервального метода с поддержкой пакетов произвольной длины в режиме конвейера реального времени с непрерывно поступающими заявками. Предлагается алгоритм конвейеризации в виде «скользящего окна». Вводится понятие скользящего окна на пространстве временных меток заявок. Рассматривается возможность независимой эволюции границ окна. Алгоритм предусматривает расчет количества заявок на интервалах обслуживания и определение размеров очередей при заданных коэффициентах загрузки. Предложен способ получения указанных величин в аналогичном скользящем окне в зависимости от интервалов обслуживания. В результате приводится алгоритм обратного асинхронного расчета очереди для параллельной обработки скользящего окна с разными значениями коэффициента загрузки. Демонстрируются результаты реализации данных алгоритмов в рамках с работы системы конвейерного интервального анализа трафика на тестовом стенде.

Ключевые слова: мультисервисные сети связи, сеть доступа, обслуживание очередей, IP-телевидение, H.264, качество обслуживания, пакетный буфер, алгоритмы потоковой обработки.

Введение

Для изучения поведения очередей в реальном времени на данный момент не существует универсальных решений. При возникновении неисправностей операторам связи приходится довольствоваться лишь косвенными признаками и сигналами о фактах переполнения пакетного буфера уже по факту [1, 2]. При анализе потока трафика в коммутаторе последний рассматривают как систему массового обслуживания (СМО). Поток трафика принято аппроксимировать пуассоновским потоком и для расчета очередей использовать известную формулу Хинчина – Поллячека. Но, как показывают исследования, потоки в сетях доступа, а тем более видеотрафик, не демонстрируют свойств пуассоновских потоков [3–5].

Для исследования потоков общего вида был предложен интервальный метод анализа трафика [6], в рамках которого получена обобщенная формула Хинчина – Поллячека, применимая к трафику сетей доступа. Цель данной работы – предложить алгоритмы для конвейерной работы данного интервального метода. В начале работы представлен алгоритм работы в режиме сколь-

зящего окна с пакетами произвольной длины. Далее рассмотрен алгоритм обратного асинхронного расчета очереди для параллельной обработки скользящего окна с переменной длиной интервала обслуживания. В заключение мы демонстрируем результат работы реализации данных алгоритмов в рамках системы конвейерного интервального анализа с построением графиков эволюции очередей на реальном потоке H.264 видеотрафика.

1. Интервальный метод с учетом переменной длины пакетов

В работах [3, 6] разработан метод интервального анализа. Согласно этому методу среднюю длину очереди можно определить, исходя из наблюдаемого количества поступающих заявок, приходящихся на одну обслуживаемую заявку. Рассмотрим поток трафика одной из услуг мультисервисной сети. Заданный поток прибывает в коммутатор со стороны опорной сети и отправляется коммутатором с абонентского порта доступа. Работу коммутационной матрицы представим как одноприборную СМО с очередью и дисциплиной обслуживания FIFO. Время обслуживания заявки примем равным сериализационной задержке при пересылке пакета с исходящего порта [7, 8]. Интервальный метод предполагает время обслуживания заявки фиксированным и равным τ. Реальный поток видеотрафика, как и любой поток трафика общего вида, может содержать пакеты различной длины. Рассмотрим алгоритм перехода от пакетов произвольной длины к ряду заявок с фиксированным временем обслуживания.

Введем в рассмотрение длину обрабатываемого пакета или Ethernet-фрейма. Возможны несколько вариантов выбора единицы длины. При выборе в качестве единицы измерения трафика 1 бит или 1 байт возникает риск ухудшить производительность системы анализа в реальном времени [9]. Во-первых, Ethernet-фреймы всегда содержат не менее 64 байт информации [10]. Во-вторых, производить на каждый принятый пакет по нескольку сотен вычислений рекуррентных функций будет очень накладно с точки зрения производительности системы анализа. При выборе размера, равного максимальному размеру пакета на данном носителе (MTU) за единицу измерения, мы рискуем потерей точности, так как значительная часть пакетов видеотрафика имеет меньшие размеры. На наш взгляд, целесообразным выглядит вариант принять за единицу размера 1 Кбайт. Таким образом, мы незначительно теряем в производительности относительно учета пакетов без размера, но значительно выигрываем в точности расчета. В дальнейших рассуждениях условимся считать единицей измерения длины 1 Кбайт трафика. Назовем этот размер длиной базового пакета и обозначим s [Кбайт] (s = 1 Кбайт).

В реальной телекоммуникационной системе, например Ethernet-коммутаторе или участке сети доступа, существует конечная битовая скорость приема данных с входящего интерфейса физического уровня и конечная скорость отправки данных на физический уровень исходящего интерфейса (рис. 1).

Рис. 1. Пример топологии включения системы анализа трафика

За событие обработки одной заявки в такой СМО примем отправку с исходящего порта данных в количестве длины базового пакета s. Обозначим через τ [с] интервал времени обработки s Кбайт трафика. Интервал времени обработки на практике определяется, как

$$\tau = \frac{s}{C_{\text{вых}}},$$

где $C_{\text{вых}}$ – скорость передачи 1 Кбайт данных в секунду с исходящего порта коммутатора. Если обозначить битовую скорость исходящего канала $B_{\text{вых}}$, то $C_{\text{вых}} = \frac{B_{\text{вых}}}{8 \cdot 1024}$ [Кбайт/с]. Аналогично введем $C_{\text{вх}}$ – скорость приема 1 Кбайт данных в секунду на входящий порт коммутатора и соответствующую битовую скорость канала $B_{\text{вх}}$, бит/с. $C_{\text{вх}} = \frac{B_{\text{вх}}}{8 \cdot 1024}$ [Кбайт/с]. Таким образом, интервал обработки τ одного Килобайта данных на данном конкретном канале представляет собой фиксированную величину. Промежуток времени между моментами поступления двух соседних заявок ϑ может быть произвольным, но не менее $\vartheta_{\min}$, так как входящий канал имеет ограниченную емкость. Минимальный промежуток между соседними заявками для данного входного канала выражается через скорость этого канала и тоже является величиной фиксированной:

$$\vartheta_{\min} = \frac{s}{C_{\text{вх}}}. \quad (1)$$

Поток поступающих заявок или событий во времени будем считать стационарным и ординарным. Ординарность в нашем случае означает, что за элементарный промежуток времени в систему приходит не более одного события, что справедливо для каналов Ethernet с последовательной передачей кадров. Отметим, что при рассмотрении каналов с параллельной передачей символов для удовлетворения требования ординарности потребуется вводить несколько иные допущения. Требование стационарности в потоке реального трафика выполняется, если будет учтена сезонность в уровнях загрузки каналов доступа операторской сети и время наблюдения будет достаточно большим. Выберем достаточно большой промежуток времени T относительно произвольно выбранного момента начала наблюдений $t_0 = 0$. Среднюю интенсивность поступления заявок λ на рассматриваемом интервале T зададим, как

$$\lambda = \frac{K}{T} \left[\frac{\text{заявок}}{\text{с}}\right],$$

где K, как и прежде [6], есть число заявок, пришедших за указанный интервал времени. Но в данном случае под заявкой мы понимаем поступление данных в количестве s Кбайт. Каждый входящий пакет данных произвольного размера l мы будем рассматривать как целое число заявок объема s, следующих непосредственно одна за другой с интервалом $\vartheta_{\min}$ (рис. 2).

Разделим весь интервал T на $N(\tau)$ интервалов длительностью τ. В каждый из интервалов может попасть одна или несколько заявок, а может не попасть ни одной заявки. Каждая заявка имеет длину s, что эквивалентно, согласно формуле (1), временному промежутку $\vartheta_{\min}$. На рис. 2 представлен пример интерпретации потока пакетов (верхний график) в поток заявок (нижний график).

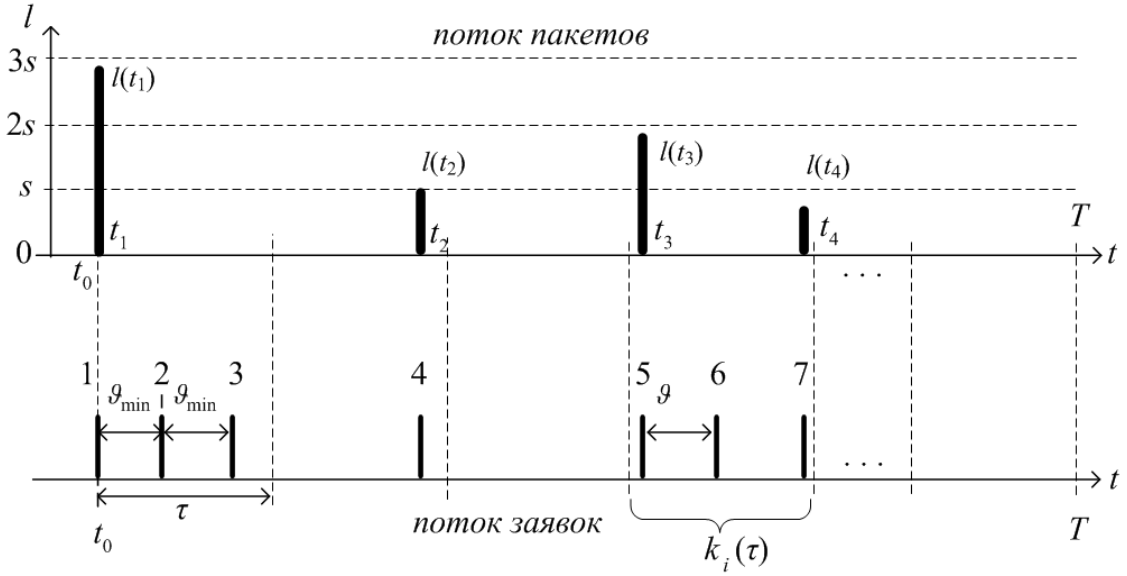

Рис. 2. Процесс формирования потока заявок из потока входящих пакетов

Пусть за весь интервал наблюдения системой зарегистрировано K_Π пакетов трафика в моменты времени t_j с длинами пакетов $l(t_j)$, где $j = \overline{1, K_\Pi}$. Суммарная длина пакетов, пришедших за i-й интервал времени τ:

$$L_i(\tau) = \sum_j l(t_j),$$

где суммирование производится по всем j, удовлетворяющим условию

$$i\tau \le t_j < (i+1)\tau.$$

Количество заявок объема s =1 Кбайт на i-м интервале τ, порожденных всей совокупностью попавших в заданный интервал пакетов с размерами $l(t_j)$, обозначим за $k_i(\tau)$ и выразим через функцию округления до ближайшего целого в большую сторону:

$$k_i(\tau) = \left\lceil \frac{L_i(\tau)}{s} \right\rceil. \quad (2)$$

Порожденные пакетом k_i штук заявок распределены по оси времени равномерно, начиная с момента времени $i\tau$, через интервалы времени $\vartheta_{\min}$. В представленном на рис. 2 примере первым пакетом порождено $k_1 = 3$ заявки. Для остальных пакетов соответственно: $k_2 = 1$; $k_3 = 2$; $k_4 = 1$.

Покажем, что полученные таким образом заявки всегда укладываются в тот интервал времени τ, на котором были получены исходные пакеты трафика. Рассчитаем максимальное число заявок, которые могут поступить в систему за интервал τ на данном потоке пакетов с входящей канальной скоростью $C_{\text{вх}}$. Максимально возможный объем пакетных данных за время τ на такой канальной скорости будет равен: $L_{\max}(\tau) = C_{\text{вх}}\tau$. Отсюда, с учетом (1), максимальное число заявок объема s

$$k_{\max}(\tau) = \left\lceil \frac{L_{\max}(\tau)}{s} \right\rceil = \left\lceil \frac{\tau}{\vartheta_{\min}} \right\rceil.$$

Выберем элементарный интервал времени Δt таким образом, что $r = \frac{\tau}{\Delta t} = \left\lceil \frac{\tau}{\vartheta_{\min}} \right\rceil$.

Максимальное число заявок на интервале образуется, если на всех элементарных интервалах Δt произойдет по одному событию $\Delta m_i = 1$. Просуммировав, получим

$$m_{\max}(\tau) = \sum_{z=0}^{r-1} \Delta m_{i+z} = \left\lceil \frac{\tau}{\vartheta_{\min}} \right\rceil.$$

Мы видим, что $k_{\max}(\tau) = m_{\max}(\tau)$. Значит с точностью до погрешности округления все заявки, порожденные пакетами произвольной длины при пиковой нагрузке входящего канала за время τ, полностью укладываются в этот интервал времени. Учитывая, что канальные скорости технологий Ethernet, как правило, кратны друг другу, можем предположить, что погрешность будет пренебрежимо мала.

Заявка, застающая обслуживающий прибор свободным, сразу попадает в обслуживание и не создает очереди. Такая заявка обслуживается полностью в течение того интервала τ, в который она попала в систему, вне зависимости от конкретного времени прибытия внутри заданного интервала. Если заявка застает прибор занятым, она становится в очередь, увеличивая значение уже имеющейся очереди $q_i(\tau)$ на единицу. Общее число заявок, поступивших за i-й интервал времени τ, мы обозначаем $m_i(\tau)$. Время обслуживания очередной заявки всегда заканчивается на границе интервала обслуживания, в следующем интервале очередь будет меньше на одну заявку. Выпишем рекуррентную формулу для расчета длины такой очереди на интервале с номером i:

$$q_i(\tau) = q_{i-1}(\tau) + m_i(\tau) - \delta_i(\tau); \quad (3)$$

$$\delta_i(\tau) = \begin{cases} 0, \text{если } q_{i-1}(\tau) = m_i(\tau) = 0; \\ 1, \text{в противном случае.} \end{cases}$$

Отсчет начинается с $i = 1$, при этом мы считаем, что в исходный момент времени очередь отсутствовала и $q_0 = 0$. При переходе к расчету скользящего окна очередь в начальный момент времени будет задаваться с учетом очереди в последнем интервале предыдущего окна.

Пакеты трафика будут поступать на анализирующую систему непрерывным потоком переменной интенсивности. Для анализа и пересчета данных системе требуется работать в цикле, оперируя рядом рекуррентных формул. В операционных системах общего назначения отсутст-

вуют инструменты для вызова программных прерываний с достаточной точностью по времени [11]. С другой стороны, сетевым адаптером обычно выставляются метки времени на каждый полученный пакет трафика [12]. Поэтому анализ потока мы будем производить по меткам точного времени, выставляемым на пакетах сетевым адаптером.

Пусть на анализирующую систему поступает поток случайных величин, представляющих собой метки точного времени прихода пакетов $\{t_i\}$, и соответствующие размеры пакетов $l(t_i)$ (см. рис. 2). Пусть алгоритм конвейерной обработки имеет возможность запускаться циклически. Очевидно, что система должна успевать обработать все накопленные данные, начиная с прошлого момента обработки. Обозначим период времени, требуемый для анализа порции данных T_A. Поступающие данные t_i требуется хранить в буфере такого объема, чтобы он не переполнился за время анализа предыдущей порции данных. Максимально возможное число поступающих пакетов $K_{\text{П}}$ за время T_A ограничено скоростью входящего канала $C_{\text{вх}}$ [бит/с] и максимально допустимым размером пакета $l_{\max}$ [бит]:

$$K_{\text{П}} = \frac{C_{\text{вх}} T_A}{l_{\max}}.$$

Максимально допустимый размер пакета $l_{\max}$ является также характеристикой входного канала и должен задаваться заранее либо детектироваться по настройкам сетевого адаптера (MTU). Поскольку за время анализа первой части буфера во вторую часть буфера системы поступит такой же либо меньший объем данных, размер итогового буфера для хранения данной информации будет равен удвоенному значению $K_{\text{П}}$. Обозначим необходимое количество ячеек буфера для хранения меток времени через B, тогда

$$B = 2\frac{C_{\text{вх}} T_A}{l_{\max}}.$$

Такой же объем буфера понадобится для хранения размеров пакетов. При отсутствии входящих пакетов в интервале окна T_A в буфер не поступит новой информации.

Назовем часть буфера размера W скользящим окном. Система должна успевать проанализировать количество пакетов, равное $K_{\text{П}}$, за время T_A. Согласно алгоритму, система всегда выбирает и фиксирует для анализа наиболее актуальную часть буфера B со вновь пришедшими пакетами. Соответственно максимальное количество данных для анализа будет равно $K_{\text{П}}$, минимальное – 0. Таким образом, скользящее окно будет сдвигаться каждый раз на актуальную часть буфера.

Обозначим через W_A и W_B значения индексов i первого и последнего элемента скользящего окна для переменной $\{t_i\}$.

Общий размер окна тогда равен

$$W = W_B - W_A + 1, \text{ если } W_A \le W_B.$$

Левая и правая части окна сдвигаются независимо до начала процедуры анализа и в зависимости от текущего момента времени t вычисляются по следующему алгоритму.

В произвольный момент времени t значение текущего индекса i фиксируется как правая граница скользящего окна W_B. Само значение t фиксируется как метка времени окончания съема текущего окна – T_f. Обозначим метку времени по индексу W_B за t_{W_B}. Отметим, что в общем случае значения t_{W_B} и T_f будут отличаться, так как время фиксируется в произвольный момент. Метка времени принимается к анализу, только если она попадает во временное окно, т. е. $t_i \ge t - T_A$. Поскольку метка времени увеличивается монотонно, очевидно, что

$$t_{W_B} = \max(\{t_i : t_i \ge t - T_A\}).$$

Если временное условие не выполняется, то значение W_B остается неопределенным.

Последовательно просматривая все значения $\{t_i\}$ от правой границы W_B до начала отсчета, находим индекс левой границы W_A, такой, что

$$t_{W_A} = \min(\{t_i : t_i \ge t - T_A\}),$$

где $i = W_B' \ldots W_B$, а индексом W_B' обозначена правая граница окна на предыдущей итерации алгоритма. Данное ограничение вводится, чтобы избежать повторного пересчета интервального алгоритма по уже отработанным заявкам.

В том случае, если условие $t_i \geq t - T_A$ не выполняется ни для одной временной метки (т. е. если пакетов за текущий интервал анализа не поступило), значения W_A и W_B остаются неопределенными и окно имеет нулевую длину.

Проиллюстрируем алгоритм скользящего окна на примере диаграммы (рис. 3). Пусть максимально допустимое время анализа временного окна T_A = 0,5 с. Пусть в момент времени t = 4,500 с в буфере находится пять временных меток с номерами i от 1 до 5 соответственно (рис. 3а). Тогда согласно вышеописанному алгоритму границы временного окна будут находиться по индексам $W_A = 3$, $W_B = 5$ и общая длина буфера составит три ячейки. Далее, в момент времени t = 4,750 с, для меток на диаграмме (рис. 3б) окно составит тоже три ячейки. И, наконец, на момент t = 5,500 с (рис. 3в) в данном примере в буфер меток не поступало, обе границы окна не определены и окно «схлапывается».

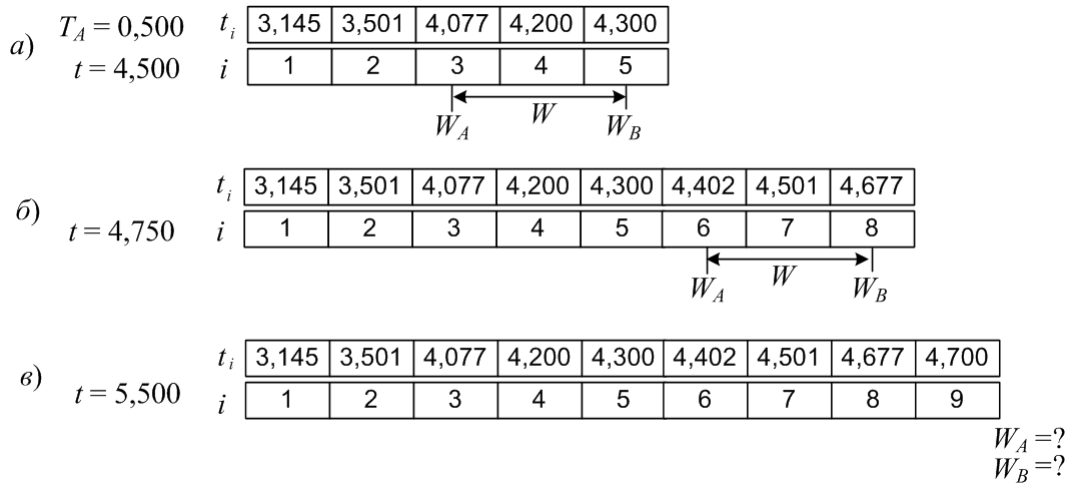

Рис. 3. Независимое перемещение границ скользящего окна

Временное окно будет постоянно флуктуировать по длине в зависимости от того, сколько пакетов поступит в буфер на предыдущем анализируемом интервале. Помимо представленного на диаграмме буфера с временными метками параллельно синхронно заполняется буфер с длинами соответствующих пакетов. Чтобы обеспечить непрерывность вывода результата (рассчитанного графика очереди и других параметров), скользящее окно всегда сдвигается на необработанную часть буфера. При этом значение очереди в новой итерации алгоритма будет отсчитываться не от нуля, а от значения очереди в последней ячейке предыдущего окна.

2. Алгоритм обратного асинхронного расчета очереди для параллельной обработки скользящего окна с разными коэффициентами загрузки

Одним из требований к системе конвейерного анализа является поддержка параллельного или одновременного расчета и отображения параметров очередей для нескольких значений коэффициента загрузки. Диапазон изменения коэффициента загрузки и шаг между значениями (гранулярность) задаются в системе конвейерного анализа.

Рассмотрим на следующем примере переход от буферов временных меток и размеров пакетов к ряду временных меток поступления заявок фиксированной длины. Пусть для данной итерации алгоритма скользящего окна длины пакетов в КБ соответствуют (рис. 4).

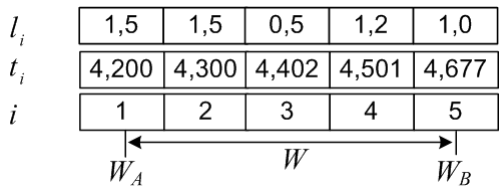

Рис. 4. Метки времени и длины пакетов в пределах скользящего окна

Пусть время обработки 1 КБ данных на выходном канале составляет 0,2 с. Количество заявок на интервалах τ с номерами j будет рассчитываться по формуле (2), и соответствующий массив чисел заявок m_j изображен на рис. 5 (интервалы пронумерованы независимо).

m_j	3	2	1
j	1	2	3

Рис. 5. Числа заявок на интервалах обслуживания: j – порядковый номер интервала обслуживания, m_j – число заявок на данном интервале

Как видно из формулы (2), для нахождения чисел заявок m_j требуется перебрать в цикле все возможные значения j. Таким образом, время выполнения алгоритма будет пропорционально количеству интервалов обслуживания в текущем временном окне. В реальных системах трафик поступает не непрерывно, а с промежутками и имеет пачечный характер. Для оптимизации времени работы системы предлагается усовершенствовать алгоритм для работы только с фактически поступившими заявками, игнорируя интервалы простоя. Это свойство может быть получено при обратном асинхронном расчете очереди. В качестве входных данных в алгоритме обратного асинхронного расчета очереди используется массив значений t_i в поддиапазоне значений i от W_A до W_B (скользящее окно). Результат работы алгоритма – заполненные массивы значений количеств заявок m_j и очередей q_j во все интервалы обслуживания j.

Вводим новый индекс a, обозначающий позицию текущей заявки внутри скользящего окна и начинающийся от 0. Таким образом, t_a – время поступления заявки с номером a внутри окна. Время поступления начальной заявки берется за начало отсчета. Увеличиваем индекс a на единицу и рассчитываем время, прошедшее с момента поступления начальной заявки $\Delta t = t_a - t_0$ (рис. 6).

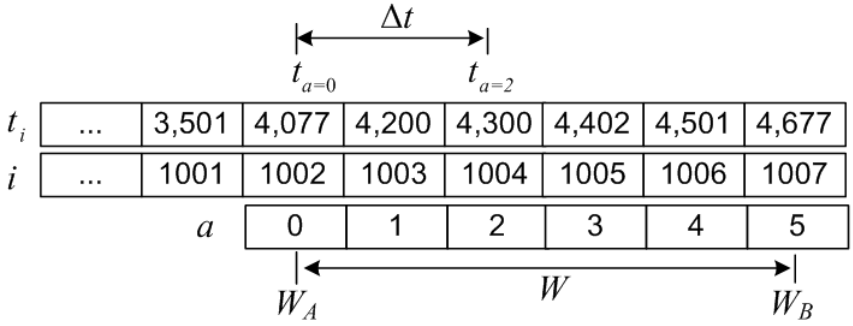

Рис. 6. Анализ скользящего окна

Определим порядковый номер интервала обслуживания, в котором находится текущая заявка:

$$j_a(\tau) = \frac{\Delta t}{\tau}. \quad (5)$$

Пусть $\Delta m_{j_a}(t_a, \tau)$ – элементарное приращение количества заявок в интервале обслуживания с номером j_a заявкой со временем прибытия t_a (для заданного τ). Полное количество заявок, поступивших за интервал обслуживания с номером j_a, будет складываться из соответствующих элементарных приращений.

Сравниваем текущий номер интервала $j_a(\tau)$ с номером для предыдущей заявки, т. е. номером интервала из предыдущей итерации алгоритма $j_{a-1}(\tau)$. Если текущий номер совпадает с предыдущим, делаем вывод, что текущая заявка обслуживается на данном интервале, следовательно, элементарное приращение количества заявок будет равно 1. Увеличиваем значения количества заявок и длины очереди на данном интервале обслуживания согласно формулам (3). На этом учет текущей заявки t_a заканчивается. Таким образом:

$$\Delta m_{j_a}(t_a, \tau) = 1; \text{ если } j_a(\tau) = j_{a-1}(\tau);$$

$$m_{j_a}(\tau) = \sum_{\forall t_a:\, j_a(\tau)=j_{a-1}(\tau)} \Delta m_{j_a}(t_a, \tau).$$

Если при вычислении (5) мы получаем $j_a(\tau) \neq j_{a-1}(\tau)$, это означает, что текущая заявка t_a будет обслужена на интервале с номером, отличным от предыдущего. Поскольку все метки времени из скользящего окна мы просматриваем строго последовательно, мы делаем вывод, что в интервалы обслуживания от $j_{a-1}(\tau)$ до $j_a(\tau)$ заявок не поступало. Нам предстоит «задним» числом рассчитать значения для пропущенных интервалов. Количество «пустых» интервалов:

$z(\tau) = j_a(\tau) - j_{a-1}(\tau)$.

По определению, за эти интервалы заявок не поступало, следовательно

$m_j(\tau) = 0$, где $j = \overline{j(\tau)_{a-1} + 1, j(\tau)_a - 1}$.

Здесь мы полагаем, что $j_{a-1}(\tau)$ интервал уже учтен полностью, а в текущем интервале $\Delta m_{j_a}(t_a, \tau) = 1$, поскольку это первая заявка текущего интервала. Аналогично для длины очереди: анализируются пропущенные пустые интервалы обслуживания и пересчитываются очереди на них по формулам (3):

$q_j(\tau) = q_{j-1}(\tau) - 1$, при $q_{j-1}(\tau) > 0$, где $j = \overline{j(\tau)_{a-1} + 1, j(\tau)_a}$.

Рекуррентная формула в данном случае работает до индекса *a*, после чего к последнему значению очереди добавляется текущая заявка.

Когда все заявки скользящего окна обслужены, приравниваются к нулю количество заявок и очереди за все оставшиеся интервалы обслуживания, начиная с текущего *j* вплоть до времени T_f:

$m_j(\tau) = 0, q_j(\tau) = 0$, для $\forall j: j_a(\tau) < j < \frac{T_f - T_A}{\tau}$.

Отличительной особенностью данного алгоритма является возможность рассчитывать очереди параллельно для нескольких значений величины интервала обслуживания. Это позволит системе конвейерного анализа строить графики всех рассчитываемых величин в зависимости от коэффициента загрузки для любого момента времени.

Пусть задана сетка из $T+1$ значений интервала обслуживания от некоторого начального значения $\tau_{\min}$ до максимального $\tau_{\max}$:

$\tau_x = \tau_{\min} + x\frac{\tau_{\max} - \tau_{\min}}{T}, x = \overline{0, T}$.

Будем интерпретировать любую заявку, как принадлежащую одновременно *T* различным потокам с различными временами обслуживания τ_x (рис. 7).

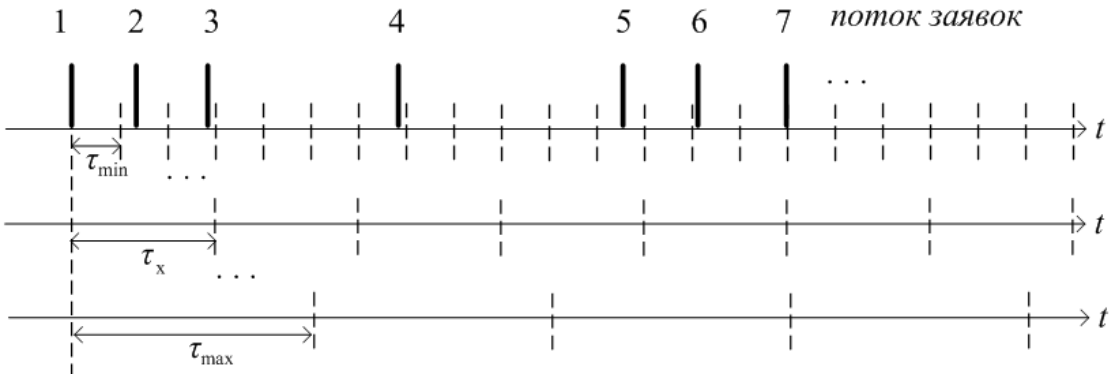

Рис. 7. Интерпретация заявок как принадлежащих нескольким потокам с разными временами обслуживания

Тогда алгоритм можно переписать с учетом прохода по всей сетке значений τ_x следующим образом.

Для всех *a*, таких, что $t_a \in [t_{W_A}, t_{W_B}]$ (для всего скользящего окна временных меток), вычисляем позицию текущей заявки на сетке интервалов обслуживания:

$j_{ax}(\tau_x) = \frac{t_a - t_0}{\tau_x}$.

Учитываем все заявки, принадлежащие текущему интервалу обслуживания:

$$\Delta m_{j_{ax}}(t_a, \tau_x) = \begin{cases} 1; \text{ если } j_{ax}(\tau_x) = j_{a-1,x}(\tau_x); \\ 0; \text{ если } j_{ax}(\tau_x) \neq j_{a-1,x}(\tau_x); \end{cases}$$

$$m_{j_{ax}}(\tau_x) = \sum_{\forall t_a:\, j_{ax}(\tau_x)=j_{a-1,x}(\tau_x)} \Delta m_{j_{ax}}(t_a, \tau_x);$$

$$q_j(\tau_x) = \begin{cases} q_{j-1}(\tau_x) + m_{j_{ax}}(\tau_x) - 1, \text{при } q_{j-1}(\tau_x) > 0; \\ m_{j_{ax}}(\tau_x) - 1, \text{при } q_{j-1}(\tau_x) = 0, m_{j_{ax}}(\tau_x) > 0; \\ 0, \text{при } q_{j-1}(\tau_x) = m_{j_{ax}}(\tau_x) = 0. \end{cases}$$

Учитываем эволюцию очереди в период отсутствия заявок:

$m_j(\tau_x) = 0$, где $j = \overline{j(\tau_x)_{a-1} + 1, j(\tau_x)_a - 1}$;

$q_j(\tau_x) = q_{j-1}(\tau_x) - 1$, при $q_{j-1}(\tau_x) > 0$, где $j = \overline{j(\tau_x)_{a-1} + 1, j(\tau_x)_a}$.

Для всех оставшихся $m_{j_{ax}}(\tau_x)$, т. е. для оставшихся интервалов обслуживания на скользящем окне:

$m_j(\tau_x) = 0$, $q_j(\tau_x) = 0$, для $\forall j: j_a(\tau_x) < j < \frac{T_f - T_A}{\tau_x}$.

Схематически работа алгоритма и порядок заполнения скользящего окна очередей для диапазона τ_x представлены на рис. 8. Ячейки массивов $m_j(\tau_x)$ заполняются в порядке, указанном стрелками. В периоды отсутствия заявок ячейки $m_j(\tau_x)$ заполняются по алгоритму обратного расчета. Ячейки, соответствующие разным длинам интервала обслуживания, заполняются параллельно, вне зависимости от наличия или отсутствия заявок.

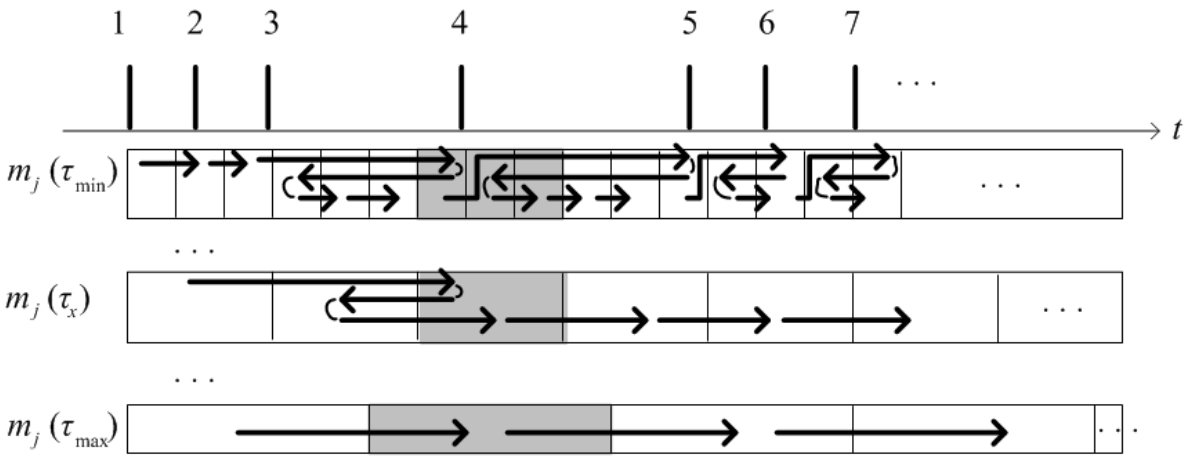

Рис. 8. Пример схемы обратного асинхронного расчета количества заявок с параллельной обработкой интервалов обслуживания разной длины

Алгоритм пригоден для расчета очередей для данных, поступающих в реальном времени. Входными данными являются только времена поступления пакетов и их размеры. Скорость прохода системы с данным алгоритмом по скользящему окну пропорциональна размеру этого окна, а значит, количеству фактически поступивших заявок, и лишь в пределе – количеству интервалов обслуживания. Данное свойство дает значительный прирост производительности системы на разреженных потоках. Из алгоритма обратного асинхронного расчета также следует возможность реализации в режиме реального времени без скользящего окна, т. е. с минимальным расходом буферной памяти. В таком случае будет анализироваться на каждой итерации лишь отрезок времени от очередного пакета трафика до последующего, сохранять же времена прихода пакетов не потребуется.

3. Результаты работы алгоритмов в рамках системы конвейерной обработки видеотрафика

На рис. 9 приведен пример результатов параллельного расчета очередей системой конвейерного анализа с учетом длин пакетов для трех значений коэффициента загрузки (ρ = 0,1; 0,7 и 0,9).

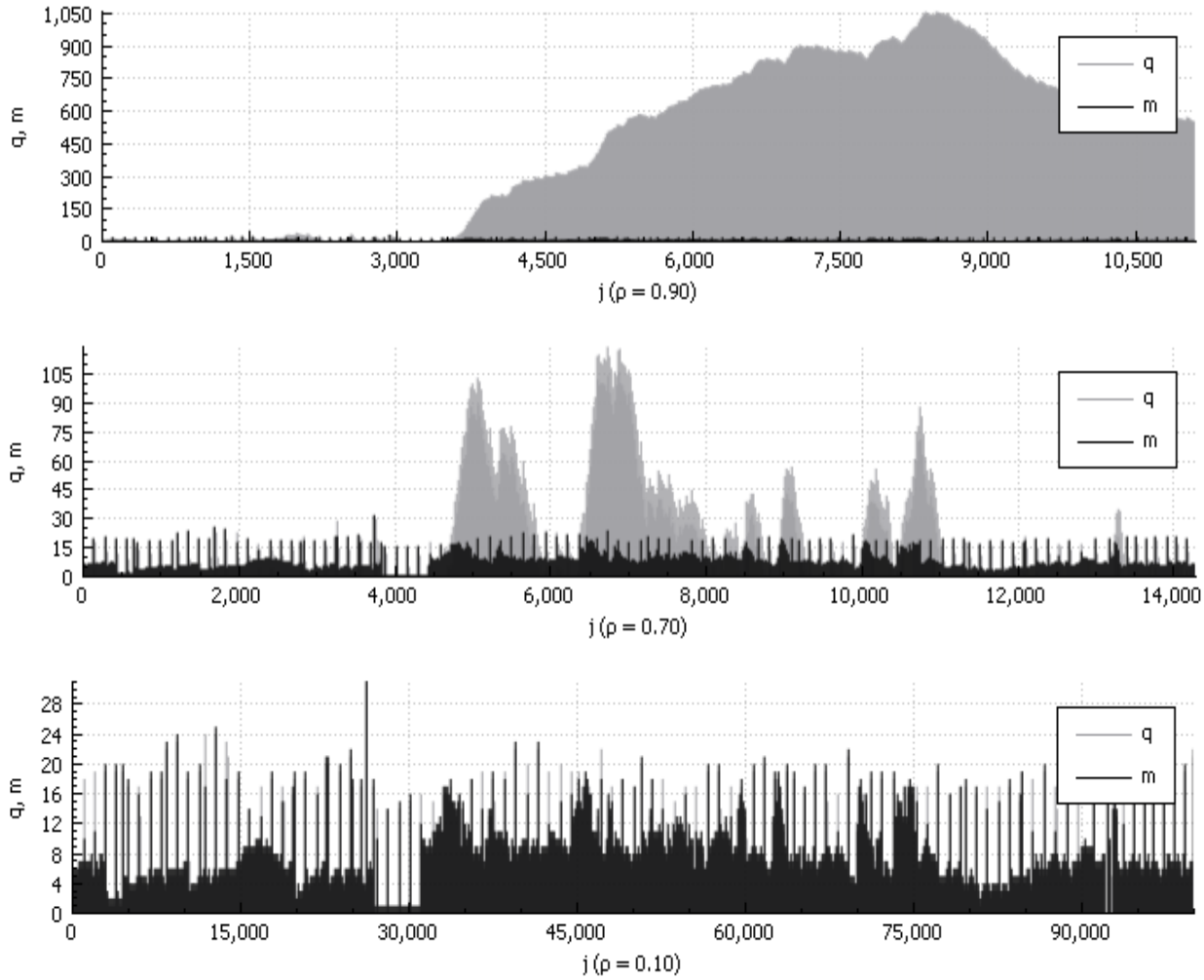

Рис. 9. Результаты расчетов количества заявок на интервалах обслуживания *m* и очередей *q* в зависимости от номера интервала *j*

Графики расчетов для всех ρ выводились на экран один раз в секунду. Наблюдение произведено для трафика IP-TV DVB вещания по технологии MPEG-2 TS мультикаст [13]. Трафик представлял собой тестовый видеофрагмент, закодированный кодеком H.264 с функциональным профилем «high» и уровнем ограничений 4.1 [14, 15]. Вещание производилось утилитой ffmpeg 3.0.1 с сервера под управлением ОС Ubuntu 14.04. Система конвейерного анализа трафика реализована на языке C++ в среде Qt 5.5 и была запущена на ПК с конфигурацией: Intel X5670, 4 ядра по 2.933 ГГц, 4ГБ ОЗУ. Минимальный размер скользящего окна в целях демонстрации в данном эксперименте был ограничен 50 тыс. временных меток пакетов. Таким образом, скользящие окна на каждой итерации перекрывали часть предыдущих данных.

Были произведены замеры суммарного времени работы конвейерных алгоритмов для случая расчета очередей по одному коэффициенту загрузки и случая с 10 значениями коэффициента загрузки одновременно (рис. 10 и 11 соответственно) в зависимости от интенсивности поступления пакетов трафика.

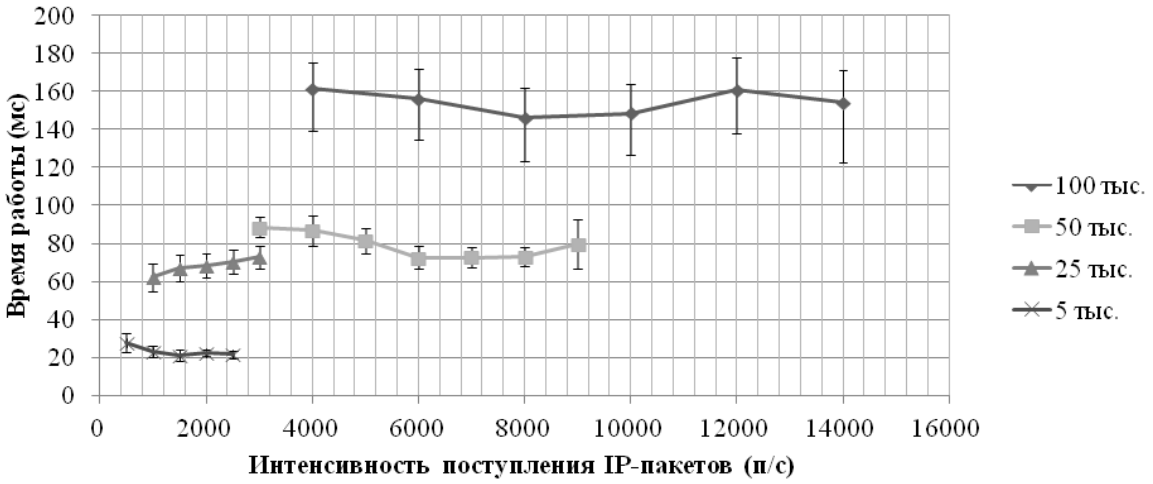

Рис. 10. Зависимость суммарного времени работы конвейерных алгоритмов для фиксированного коэффициента загрузки

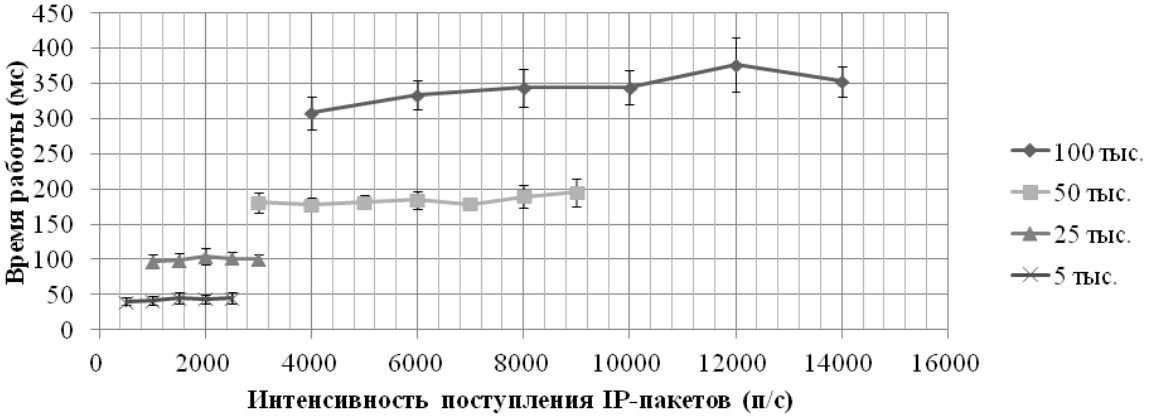

Рис. 11. Зависимость суммарного времени работы конвейерных алгоритмов с параллельным расчетом для 10 коэффициентов загрузки

На обоих графиках представлены результаты замеров времени выполнения всех расчетов, включая время вывода графиков очередей на экран для 4 значений максимального размера скользящего окна временных меток (5, 25, 50 и 100 тыс.). Как видно из графиков, время расчета очередей для 10 коэффициентов загрузки всего в 2 раза превышает время расчета для одного коэффициента. Также наблюдается весьма слабая зависимость времени от интенсивности трафика, что ожидаемо для ограничения минимальной длины окна. Зависимость времени выполнения от фактической длины скользящего окна на разных типах трафика мультисервисных сетей требует дальнейшего изучения.

Литература

1. Arista LANZ Overview. – https:// people.ucsc.edu/~warner/Bufs/Arista_LANZ_Overview_TechBulletin_0213.pdf (дата обращения: 26.10.2018).

2. Building an Open Source Data Center Monitoring Tool Using Broadcom BroadView™ Instrumentation Software. – https:// people.ucsc.edu/~warner/Bufs/BroadView-TB201-RDS.pdf (дата обращения: 26.10.2018).

3. Лихтциндер, Б.Я. Интервальный метод анализа трафика мультисервисных сетей доступа / Б.Я. Лихтциндер. – Самара: ПГУТИ, 2015. – 121 с.

4. Paxson, V. Why we don't know how to simulate the Internet / V. Paxson, S. Floyd. – http://www.cs.ucsb.edu/~almeroth/classes/F02.276/papers/paxson-97.pdf (дата обращения: 26.10.2018).

5. Paxson, V. Wide area traffic: the failure of poisson modeling / V. Paxson, S. Floyd // IEEE/ACM Trans. Netw. – 1995. – 3 (3). – P. 226–244. DOI: 10.1109/90.392383

6. Лихтциндер, Б.Я. Интервальный метода анализа очередей в системах массового обслуживания с пачечными потоками заявок / Б.Я. Лихтциндер // T-Comm: Телекоммуникации и транспорт. – 2017. – Т. II, № 3. – С. 17–23.

7. End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks / T. Szigeti, C. Hattingh, R. Barton, K. Briley. – 2nd Edition. – Cisco Press, 2012. – 1040 p.

8. Bradner, S. RFC-2544. Benchmarking Methodology for Network Interconnect Devices / S. Bradner, J. McQuaid. –https:// www.ietf.org/rfc/rfc2544.txt (дата обращения: 26.10.2018).

9. Intel Ethernet Switch Family Memory Efficiency Non-blocking Fabric Architecture. – https://people.ucsc.edu/~warner/Bufs/intel-memory-efficiency-paper.pdf (дата обращения: 26.10.2018).

10. Understanding Data Center Traffic Characteristics / T. Benson, A. Anand, A. Akella, and M. Zhang // ACM SIGCOMM Computer Communication Review. – 2010. – 40 (1). – P. 92–99.

11. Timekeeping in VMware Virtual Machines. – https://www.vmware.com/files/pdf/techpaper/Timekeeping-In-VirtualMachines.pdf (дата обращения: 26.10.2018). DOI: 10.1145/1672308.1672325

12. PCAP-TSTAMP – packet time stamps in libpcap. – https://www.tcpdump.org/manpages/pcap-tstamp.7.txt (дата обращения: 26.10.2018).

13. Transport of MPEG-2 TS Based DVB Services over IP Based Networks ETSI TS 102 034 V2.1.1. – European Broadcasting Union, France, 2016. – 331 p.

14. Advanced video coding for generic audiovisual services, Recommendation ITU-T H.264. – International Telecommunication Union, Geneve, Switzerland, 2016 – 807 p.

15. Jae-Beom Lee, Hari Kalva The VC-1 and H.264 Video Compression Standards for Broadband Video Services // Springer Science+Business Media, LLC. – New York, USA, 2008. – 515 p. DOI: 10.1007/978-0-387-71043-3

Моисеев Виктор Игоревич, ведущий программист отдела информационно-вычислительных сетей Университетского центра «Интернет», старший преподаватель кафедры радиоэлектроники и защиты информации, Пермский государственный национальный исследовательский университет, г. Пермь; аспирант кафедры мультисервисных сетей и информационной безопасности, Поволжский государственный университет телекоммуникаций и информатики, г. Самара; vim@psu.ru.

Лихтциндер Борис Яковлевич, д-р техн. наук, профессор, профессор кафедры мультисервисных сетей и информационной безопасности, Поволжский государственный университет телекоммуникаций и информатики, г. Самара; lixt@psati.ru.

Поступила в редакцию 24 ноября 2018 г.

DOI: 10.14529/ctcr190105

ALGORITHMS FOR PIPELINE INTERVAL ANALYSIS OF TRAFFIC

V.I. Moiseev*[1, 2]*, vim@psu.ru,
B.Ya. Lihtsinder*[2]*, lixt@psati.ru
[1] *Perm State University, Perm, Russian Federation,*
[2] *Povolzhskiy State University of Telecommunications and Informatics, Samara, Russian Federation*

The problem of real-time queue analysis of production IP traffic is stated. The method of interval analysis is described in context of real-time constraints. Drawbacks of such method are stated and solutions proposed. First, we propose an algorithm to enhance interval analysis support for variable length analysis which is crucial for real IP-TV traffic. We consider different units to measure packet to customer in the queue mapping. We estimate the 1KB worth of packet data to be a viable fit to calculate accurate queue sizes. Then we propose pipeline extension for interval analisys as a sliding window. We introduce sliding window on timestamp scale and describe evolution models

of left and right window edge separately. The algorithm proposed to map timestamps to number of customer in the queue. All proposed techniques then combined into algorithm of reverse queue calculation with parallel calculation of several service intervals simultaneously. We demonstrate and measure a performance of reference implementation of said algorithms in the lab under H.264 IP-TV traffic of various rates.

Keywords: multiservice communication networks, access network, queue service, IP television, H.264, service quality, package buffer, algorithms of stream processing.

References

1. *Arista LANZ Overview*. Available at: https://people.ucsc.edu/~warner/Bufs/Arista_LANZ_Overview_TechBulletin_0213.pdf (accessed 26.10.2018).

2. *Building an Open Source Data Center Monitoring Tool Using Broadcom BroadView™ Instrumentation Software*. Available at: https:/people.ucsc.edu/~warner/Bufs/BroadView-TB201-RDS.pdf (accessed 26.10.2018)

3. Lihtczinder B.Ya. *Intervalniy metod analiza trafika multiservisnykh setey dostupa* [Interval Analysis Method of Access Networks] [. Samara, PSUTI Publ., 2015. 121 p.

4. Paxson V., Floyd S. *Why We Don't Know How to Simulate the Internet.* Available at: http://www.cs.ucsb.edu/~almeroth/classes/F02. 276/ papers /paxson-97.pdf (accessed 26.10.2018).

5. Paxson V. and Floyd S. Wide Area Traffic: the Failure of Poisson Modeling. *IEEE/ACM Trans. Netw*., 1995, 3 (3), pp. 226–244. DOI: 10.1109/90.392383

6. Lihtczinder B.Ya. [Interval Queue Analysis Method for Queuing Systems with Batch Arrivals]. *T-Comm: Telecommunications and Transport*, 2017, vol. II, no. 3, pp. 17–23. (in Russ.)

7. Szigeti T., Hattingh C., Barton R., Briley K. *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks*. Cisco Press, 2012. 1040 p.

8. Bradner S., McQuaid J. *RFC-2544. Benchmarking Methodology for Network Interconnect Devices*. Available at: https://www.ietf.org/rfc/rfc2544.txt (accessed 26.10.2018).

9. *Intel Ethernet Switch Family Memory Efficiency Non-blocking Fabric Architecture*. Available at: https://people.ucsc.edu/~warner/Bufs/intel-memory-efficiency-paper.pdf (accessed 26.10.2018).

10. Benson T., Anand A., Akella A., Zhang M. Understanding Data Center Traffic Characteristics. *ACM SIGCOMM Computer Communication Review*, 2010, 40 (1), pp. 92–99. DOI: 10.1145/1672308.1672325

11. *Timekeeping in VMware Virtual Machines*. Available at: https://www.vmware.com/files/pdf/techpaper/Timekeeping-In-VirtualMachines.pdf (accessed 26.10.2018).

12. *PCAP-TSTAMP – Packet Time Stamps in Libpcap*. Available at: https://www.tcpdump.org/manpages/pcap-tstamp.7.txt (accessed 26.10.2018).

13. *Transport of MPEG-2 TS Based DVB Services over IP Based Networks ETSI TS 102 034 V2.1.1*. European Broadcasting Union, France, 2016. 331 p.

14. Advanced Video Coding for Generic Audiovisual Services, Recommendation ITU-T H.264. International Telecommunication Union, Geneve, Switzerland, 2016. 807 p.

15. Jae-Beom Lee, Hari Kalva The VC-1 and H.264 Video Compression Standards for Broadband Video Services. *Springer Science+Business Media, LLC*, New York, USA, 2008. 515 p. DOI: 10.1007/978-0-387-71043-3

Received 24 November 2018

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Моисеев, В.И. Алгоритмы конвейерного интервального анализа трафика / В.И. Моисеев, Б.Я. Лихтциндер // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». – 2019. – Т. 19, № 1. – С. 57–69. DOI: 10.14529/ctcr190105

FOR CITATION

Moiseev V.I., Lihtsinder B.Ya. Algorithms for Pipeline Interval Analysis of Traffic. *Bulletin of the South Ural State University. Ser. Computer Technologies, Automatic Control, Radio Electronics*, 2019, vol. 19, no. 1, pp. 57–69. (in Russ.) DOI: 10.14529/ctcr190105