МОДИФИЦИРОВАННЫЙ АЛГОРИТМ УПРАВЛЕНИЯ ПЕРЕГРУЗКОЙ ДЛЯ СОЕДИНЕНИЙ TCP-SACK

Цапенко И. В.
Каф. ЭВМ ДонНТУ
igortsapenko@mail.ru

Abstract

Tsapenko I., A modified congestion control algorithm for SACK-TCP. *A new congestion control algorithm using packet loss information obtained from Selective Acknowledgments (SACK) is proposed in order to improve fairness and stability of TCP best-effort service. The proposed mechanism adjusts the decrease in congestion window for Fast Recovery and the increase in congestion window per ACK during the congestion avoidance phase according to the number of lost packets. Simulation results show that fairness and stability are improved and that the utilization of router buffer is effectively stabilized. Also, it is shown that coexistence of the proposed scheme and conventional TCP on the same network does not introduce any problems as the characteristics of each scheme are preserved.*

Введение

Управление перегрузкой - один из ключевых механизмов для адаптации увеличивающегося разнообразия услуг и видов трафика в Internet. Первоначально, при разработке алгоритма управления перегрузкой TCP, сеть рассматривалась как "черный ящик". Это означает, что конечные узлы, осуществляя управление, не устанавливают непосредственно состояние маршрутизаторов и линии передачи, а скорее регулируют трафик, определяя сетевую загрузку косвенным образом от потери пакета и колебаний времени ответа. Таким образом, становится более трудным распределять сетевые ресурсы соответствующим образом, используя только управление перегрузкой на конечных узлах, и становится все более важной организация управления всей сетью связи, включающая взаимодействие между конечными узлами и маршрутизаторами.

При поддержке услуг максимально возможного качества, основным требованием является равномерность распределения ресурсов [1]. В этой работе, равномерность распределения подразумевает одинаковый раздел ширины полосы пропускания среди конкурирующих потоков, совместно использующих «узкое место» сети. В дополнение к равномерности распределения, существует еще один важный критерий - стабильность, которая направлена на минимизацию колебаний производительности соединения. Необходимо отметить, что, даже если средние значения производительности для двух различных соединений за длительный промежуток времени являются одинаковыми, для эффективного использования сетевых ресурсов является также важным обеспечить минимизацию колебаний производительности соединений.

В данной работе предлагается модификация к базовому алгоритму управления перегрузки TCP, которая использует информацию о потерянных пакетах, полученную из избирательных подтверждений (SACK) [2]. С помощью моделирования оценивается эффективность этого подхода. Предложенный алгоритм улучшает равномерность распределения и стабильность услуг TCP.

Использование избирательных подтверждений (SACK) рекомендуется в RFC 2018 как наиболее эффективный механизм восстановления от потери нескольких пакетов спутниковых и беспроводных соединений, в которых характерно наличие высоких коэффициентов ошибок. Хотя алгоритм SACK был, прежде всего, разработан для управления повторной передачей, он может также использоваться для осуществления

более эффективного управления перегрузкой, так как с его помощью можно получать наиболее точную информацию о потерянных пакетах.

Базовый алгоритм управления перегрузкой

У базового алгоритма есть две переменные: текущий размер окна перегрузки cwnd и пороговое значение ssthresh, которые модифицируются следующим образом:

(1) После каждого подтверждения нового пакета:
Если cwnd < ssthresh, установить cwnd = cwnd + 1; **Фаза «медленный старт»**
иначе установить cwnd = cwnd + 1/[cwnd]. **Фаза «избежание перегрузки»**

(2) При поступлении некоторого числа дублирующихся подтверждений (ACK пакетов), узел-отправитель делает вывод, что пакет с номером ACK был потерян, передает его повторно, после чего изменяет пороговое значение ssthresh и размер окна перегрузки cwnd следующим образом:

ssthresh = max {min(cwnd / 2, adwnd),2*MSS};
cwnd = ssthresh,

где adwnd – размер окна, предлагаемый приемником, MSS – максимальный размер сегмента. После этого алгоритм входит в **фазу «быстрого восстановления»**, где и находится до того, как повторная передача будет подтверждена.

При этом уменьшение ssthresh и cwnd – стандартны и никак не зависят от количества потерянных пакетов. Изменение размера окна перегрузки, согласно данному алгоритму представлено на рис.1.

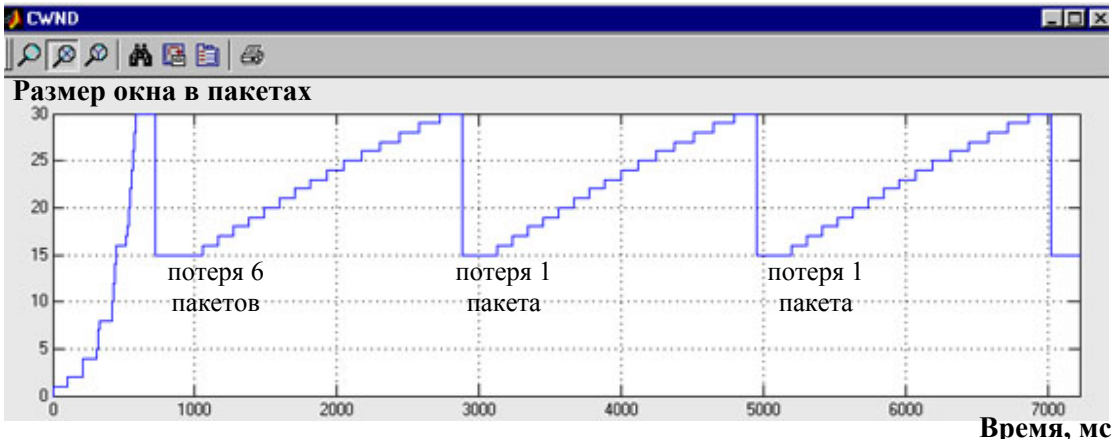


Рис.1 Пример изменения окна (cwnd) базового TCP SACK

Как показано на рис.1 даже при потере одного пакета окно перегрузки, уменьшается вдвое, что приводит к использованию лишь половины максимально возможной полосы пропускания канала. Модификация существующего алгоритма управления перегрузки направлена на уменьшение неэффективности использования полосы пропускания канала.

Модифицированный алгоритм

В данном алгоритме предлагается использовать информацию из избирательных подтверждений (SACK) для регулировки потока трафика. Поток трафика регулируется уменьшением cwnd при инициализации «быстрого восстановления» и увеличением роста cwnd на каждый ACK пакет во время фазы «предотвращения перегрузки».

- Если потерянно небольшое количество пакетов: предполагается, что используемая полоса пропускания показывает высокую степень равномерности распределения, поэтому величина, на которую сокращается cwnd, уменьшается. Кроме

того, для более гладкого увеличения cwnd, замедляется его рост при поступлении АСК пакетов.

- Если потеряно большое количество пакетов, предполагается, что используется большая часть полосы пропускания канала и степень равномерности ее распределения низка. Поэтому cwnd уменьшается на большую величину. Однако cwnd быстрей увеличивается с поступлением каждого АСК пакета, что бы быстрей достичь оптимального значения.

Предложенный метод сохраняет все базовые алгоритмы TCP Reno, включая «медленный старт», «предотвращение перегрузки», «быстрое восстановление» и «быструю повторную передачу» [4].

Вначале, алгоритм подсчитывает количество потерянных пакетов. Интервал, в течении которого происходит подсчет потерянных пакетов, начинается с момента обнаружения первой потери пакета (при этом в переменную send_high заносится номер последнего посланного пакета) и заканчивается моментом, когда будет получен АСК пакет с номером send_high.

Для корректировки скорости передачи, используются два коэффициента: коэффициент сокращения cwnd – f1 и коэффициент расширения cwnd – f2, которые используются после возврата в фазу «предотвращения перегрузки» из фазы «быстрого восстановления».

Уменьшение ssthresh и cwnd после обнаружения потери пакетов происходит следующим образом:

ssthresh = cwnd *f1;

cwnd = ssthresh.

Увеличение cwnd при получении АСК во время фазы «предотвращения перегрузки» происходит следующим образом:

cwnd = cwnd + f2/cwnd.

Во время фазы восстановления cwnd увеличивается с поступлением каждого АСК пакета, как и базовом TCP Reno. В конце фазы восстановления устанавливается cwnd = ssthresh и происходит переход в фазу «избежания перегрузки».

Расчет коэффициентов f1 и f2

Можно привести множество методов для вычисления коэффициента сокращения f1 и коэффициента расширения f2. В предлагаемом алгоритме будем выбирать их значения в зависимости от того, является ли число потерянных пакетов (LostDT) равным или большим 1. То есть

если LostDT = 1: то f1 = 4/5, f2 = 2/5;

если LostDT > 1: то f1 = 1/2, f2 = 1.

Обоснование выбора этих параметров следующее. Прежде всего, в TCP- Reno используется коэффициент сокращения f1 =1/2, независимого от LostDT . В более ранних версиях типа TCP-Tahoe, cwnd устанавливалось равным размеру одного пакета (то есть приблизительно f1 = 1/cwnd). Хотя установка f1 = 1/2 является значительным усовершенствованием относительно более ранних версий, она все еще неоптимальна из-за недостатка информации о точном числе потерянных пакетов. Это означает, что установка cwnd равным 1/2 от его текущего значения (f1 = 1/2) не всегда является наиболее подходящим вариантом. Это слишком обобщающее решение для всех случаев.

С другой стороны, так как при использовании TCP-SACK доступна информация о количестве потерянных пакетов, она может быть отражена на выборе коэффициента сокращения f1. Например, рассмотрим потерю одного пакета.

Используя дублирующиеся подтверждения ACK, можно определить, что число потерянных пакетов (LostDT) равно 1, количество подтвержденных пакетов данных, соответствующих дублирующимся ACK равно 3, и количество пакетов данных перед

потерянным равно 1, так что общее число пакетов равно 5, из которых 4 пакета успешно переданы. Если количество потерянных пакетов за один интервал кругового обращения (RTT) известно, то нет необходимости устанавливать f1 = 1/2. Поскольку информация относительно LostDT получена от TCP SACK, то можно установить f1 = 4/5. Те же аргументы можно привести, если LostDT > 1. Например, для LostDT = 2, число успешно полученных пакетов может быть определенно как 4 из 6 (f1 = 2/3), или 6 из 8 (f1 = 3/4), в зависимости от конкретного расположения потерянных пакетов.

Аналогично для LostDT = 3, это может быть 4 из 7 (f1 = 4/7), или 6 из 9 (f1 = 2/3), опять же, в зависимости от конкретного расположения потерянных пакетов. Очевидно, что трудно выбрать подходящее значение f1, когда LostDT >= 2. С другой стороны, среднее число пакетов, потерянных в течение одного интервала RTT, может быть рассчитано по следующей формуле:

Ср. число потерянных пакетов = (скорость потерь) *

(полоса пропускания) * RTT/MTU

В [7] сообщается, что в установившемся состоянии скорость потерь пакетов в Internet для одного потока составляет - 2.7 % и 5.2 %, в зависимости от его местоположения. Типичная ширина полосы пропускания потока может находиться в пределах от 10 Kbit/s до 1 Mbit/s. Значение RTT может измениться от 10 msec и 100 msec, а типовое значения MTU - 576 и 1500 байтов. Подставляя эти значения в вышеприведенное выражение, получаем диапазон для среднего числа потерянных пакетов между 0.0002 и 1.13. Возвращаясь к вышеупомянутым аргументам, получаем, что уместно установить пороговое значение LostDT равным 1 пакету, и отдельно не рассматривать случаи для потерь более 1 пакета (то есть устанавливает f1 =1/2 для всех LostDT > 1).

Что касается коэффициента расширения f2, cwnd увеличивается с каждым полученным ACK таким образом, чтобы быстрей достичь значения, при котором еще не происходят потери пакетов. Это соответствует установке f2= 2/5 для LostDT = 1, и f2 =1 для LostDT > 1.

Процесс изменения cwnd, с учетом вышеописанного алгоритма, иллюстрирует рис.2.

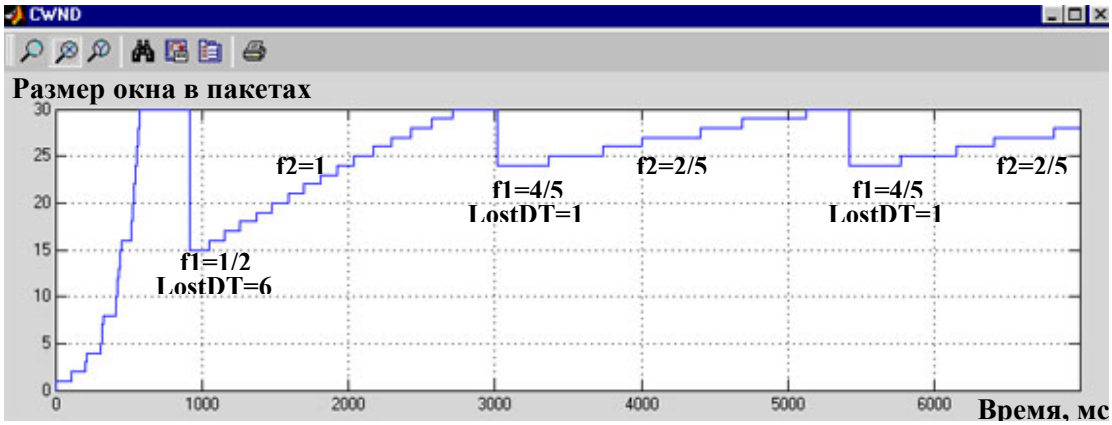


Рис.2 Пример изменения окна перегрузки (cwnd)

Моделирование

Оценим поведение предложенного алгоритма с помощью простой сетевой топологии (рис. 3), состоящий из десяти соединений, совместно использующих один канал. Моделирования будут выполнены для предложенного алгоритма (далее – Sack-new), и базового TCP SACK (далее Sack-orig).

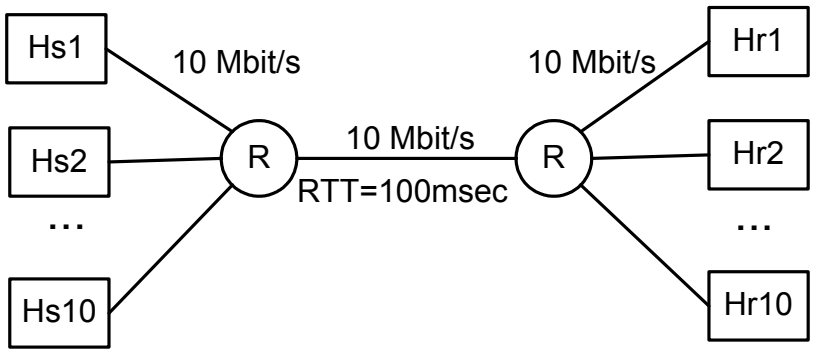


Рис. 3 Топология моделируемой сети

Каждый передающий узел (Hs1-Hs10) передает файл узлу-приемнику (Hr1-Hr10) через канал, являющийся «узким местом» сети. Каждый отправитель имеет достаточно данных для непрерывной посылки. Отношение общей пропускной способности каналов всех узлов-передатчиков к пропускной способности общего канала - 10 к 1, так образом общий канал и буфера маршрутизаторов R всегда утилизированы. Время кругового обращения (RTT) установлено в 100 msec.

Среда имитационного моделирования:

Модули Simulink и Stateflow (входящие в состав пакета Matlab)

- Поддерживают направляемую событиями передачу, позволяя задавать состояние приемника для конкретного события.
- Разрешает программирование обычных языковых структур, таких как циклы, и операторы if-then-else на потоковых диаграммах.
- С помощью Stateflow можно быстро разрабатывать графические модели управляемых событиями систем, используя теорию автоматов конечного состояния, формализм диаграмм состояния и нотацию потоковых диаграмм.
- Поддерживает иерархию, параллелизм, переходы и историю.

Анализ результатов моделирования

Для анализа результатов будем использовать характеристику распределения производительности (гистограмму). Гистограммы в этом разделе показывают вероятность достижения некоторого значения производительности всеми соединениями за интервал времени равный 1 сек - интервал, за который пользователь может заметить ухудшение функционирования.

Мы также исследуем влияние предложенного алгоритма на утилизацию буфера маршрутизатора. Кроме того, для оценки возможного эффекта при внедрении предложенного алгоритма в Интернет, были выполнены моделирования для ситуаций, когда в одной сети сосуществуют соединения обоих типов (Sack - new и Sack-orig).

Характеристики производительности

Рассмотрим результаты моделирования для производительности типичного соединения с алгоритмами Sack- new и Sack-orig (рис.4). Согласно данному рисунку, при использовании предложенного алгоритма, равномерность распределения ширины полосы пропускания канала заметно улучшается, а колебание производительности понижается.

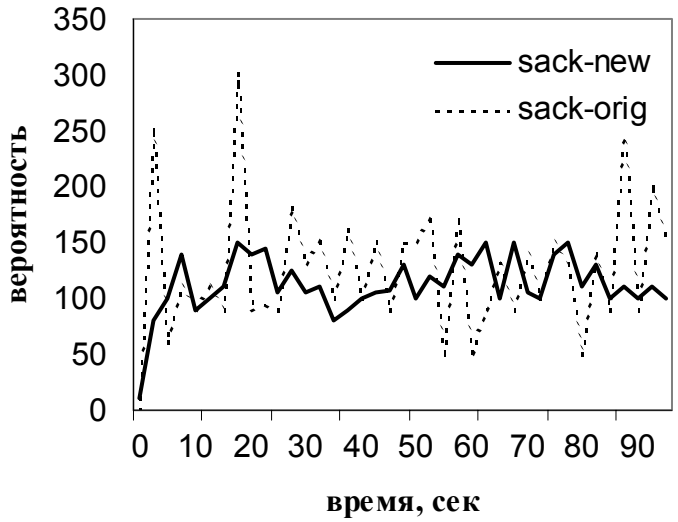


Рис. 4 Производительность соединений

Что касается равномерности распределения, была сравнена вероятность достижения некоторого значения производительности всеми соединениями. Результаты представлены на рис.5. На этом рисунке горизонтальная ось представляет отношение фактической производительности к равномерности распределения. Так как имеются 10 соединений, совместно использующие канал, являющийся «узким местом» сети, 1 представляет десятую часть 10 Mbit/s, то есть равномерное распределение - это 1 Mbit/s на каждое соединение. Наблюдая концентрацию вокруг 1, можно отметить, что для данной конфигурации, соединения Sack -new достигают более равномерного распределения полосы пропускания. Таблица 1 суммирует эти результаты.

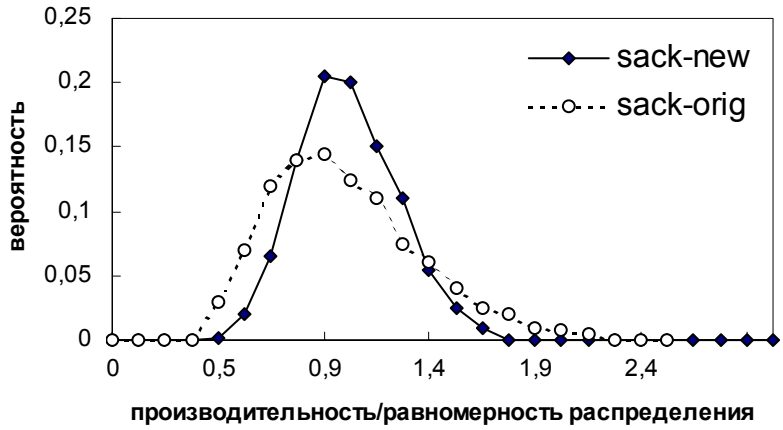


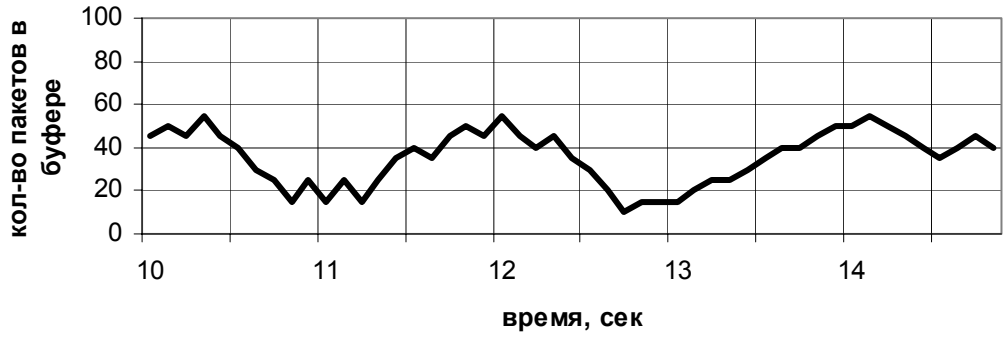
Рис. 5 Гистограмма равномерности распределения полосы пропускания канала

Таблица 1 Сравнение равномерности распределения

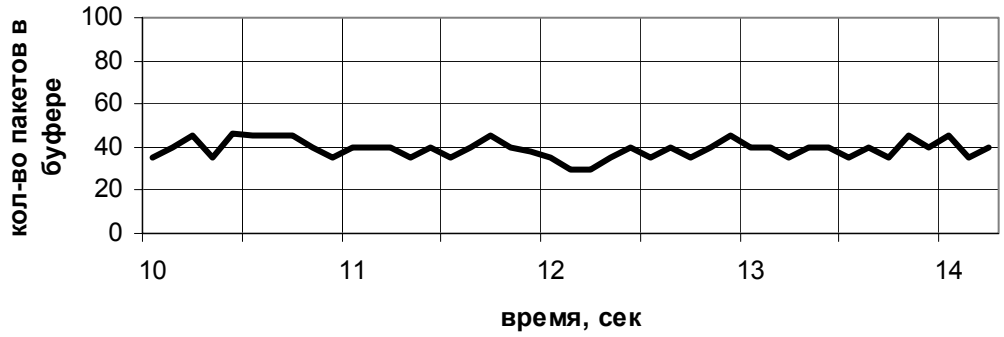
	Sack-orig	Sack-new
Вероятность равномерности распределения	0.13	0.20

По табл. 1 видно, вероятность равномерного распределения полосы пропускания канала между соединениями Sack-new выше на 54%.

Использование буфера (утилизация) показана на рис. 6. Можно отметить, что sack - new значительно стабилизирует утилизацию буфера по сравнению с sack-orig.



а) Sack-orig



б) Sack-new

Рис.6 Сравнение утилизации буфера маршрутизатора

Поведение при сосуществовании двух типов соединений

Моделирование было выполнено для 5 соединений Sack-orig и 5 соединений Sack - new. На рис.7 показана вероятность достижения некоторого значения производительности для каждого типа алгоритмов.

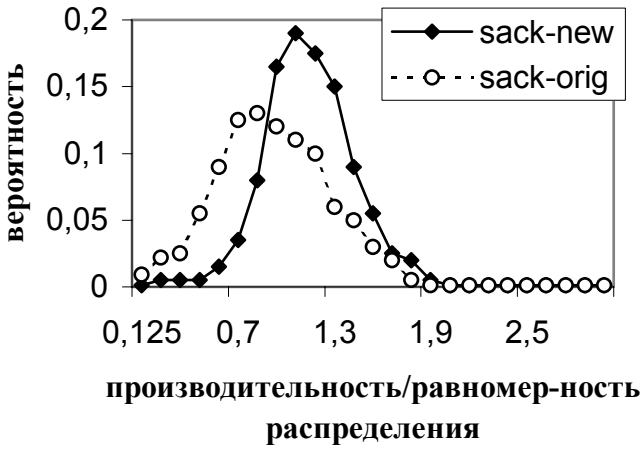


Рис.7 Гистограмма равномерности распределения полосы пропускания канала

Вероятность достижения равномерного использования полосы пропускания канала показана в табл.2. Сравнение табл. 2 с табл.1 показывает, что сосуществование соединений Sack- new с Sack-orig не ухудшает общее функционирование, показываемое каждым из соединений в отдельности.

Таблица 1 Сравнение равномерности распределения

	Sack-orig	Sack-new
Вероятность равномерности распределения	0.11	0.19

Выводы

В данной работе был предложен модифицированный алгоритм TCP управления перегрузкой, использующий информацию о потерянных пакетах, полученную из избирательных подтверждений (SACK), для регулирования потока трафика. Эффективность этого подхода была оценена при моделировании. Предложенная схема заметно улучшает равномерность распределения производительности и стабильность услуг TCP. Кроме того, значительно стабилизируется утилизация буфера маршрутизатора. Моделирование было выполнено для простой сетевой топологии, состоящей из десяти соединений, совместно использующих один канал с задержкой распространения в один конец- 50 msec, и пропускной способностью канала - 10 Mbit/s. Для данной конфигурации, равномерность распределения, которая отображается как вероятность достижения равномерного распределения, улучшается приблизительно на 50 % по сравнению с Sack-orig. Уменьшение колебания производительности не только уменьшает воздействие на другой трафик, который несет сеть, но также позволяет более эффективно управлять сетевыми ресурсами. Кроме того, показано, что сосуществование предложенного алгоритма и базового TCP Sack в одной и той же сети не представляет никаких проблем, поскольку характеристики каждого алгоритма сохраняются.

Литература

[1] B.Braden, et.al., "Recommendations on queue management and congestion avoidance in the Internet," *Internet draft draft-irtf-e2e-queue-mgt-01.tx*t, Feb.17, 1998.

[2] M. Mathis, J. Mahdavi, S. Floyd, and A.Romanow, "TCP selective acknowledgment option*s," RFC 201*8, Oct. 1996.

[3] V. Jacobson, "Congestion avoidance and control," *ACM CC*R, Vol.18, No.4, pp. 314-329, 1988

[4] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *RFC 200*1, Jan. 1997.

[5] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and SACK TCP," *ACM CC*R, vol. 26 no. 3, pp. 5-21, July 1996.

[6]M. Mathis and J.Mahdavi, "Forward acknowledgement: refining TCP congestion control," *Proc. ACM SIGCOMM '9*6, California, Aug. 1996.

[7] V. Paxson, "End-to-end Internet packet dynamics," *Proc. ACM SIGCOMM '9*7, Cannes, France, Sep. 1997.

[8] R. Morris, "TCP behavior with many flows," *Proc. IEEE ICNP'9*7, Atlanta,Oct. 1997.

Поступила в редакційну колегію 02.07.2002