УДК 004.7:519.2

А.В. КАРПУХИН, канд. техн. наук, Л.О. КИРИЧЕНКО, канд. техн. наук,
Т.А. РАДИВИЛОВА, канд. техн. наук

ИССЛЕДОВАНИЕ РАБОТЫ ПРОТОКОЛОВ СЕМЕЙСТВА TCP ПРИ КРИТИЧЕСКИХ РЕЖИМАХ ФУНКЦИОНИРОВАНИЯ СЕТИ

Введение

Протокол TCP – это протокол обеспечения надежности прямых соединений, созданный для многоуровневой иерархии протоколов, поддерживающих межсетевые приложения. Протокол TCP обеспечивает надежность коммуникаций между парами процессов на хост-компьютерах, включенных в различные компьютерные коммуникационные сети, которые объединены в единую систему.

Многочисленные исследования процессов в сети Интернет показали, что статистические характеристики трафика обладают свойством временной масштабной инвариантности – самоподобием. Причина такого эффекта заключается в особенностях распределения файлов по серверам, их размерах, а также в типичном поведении пользователей. Оказалось, что изначально не проявляющие свойств самоподобия потоки данных, пройдя обработку на узловых серверах и активных сетевых элементах, начинают подавать ярко выраженные признаки самоподобия. Из-за этого возможно быстрое переполнение буферов устройств даже при небольших коэффициентах использования. Если не принять мер по ограничению поступающего трафика, очереди на наиболее нагруженных линиях будут неограниченно расти и, в конце концов, превысят размеры буферов в соответствующих узлах. Таким образом, пакеты, вновь поступающие на узлы, в которых нет свободного места в буфере, будут отброшены и должны будут передаваться повторно, что приводит к нерациональной трате ресурсов сети [1 – 3].

Регулирование трафика в протоколе TCP подразумевает существование двух независимых процессов: контроль доставки, управляемый получателем с помощью параметра Window, и контроль перегрузки, управляемый отправителем с помощью окна перегрузки (CWND – congestion window) и процедуры медленного старта (slow start). Первый процесс отслеживает заполнение входного буфера получателя, второй – регистрирует перегрузку канала, а также связанные с этим потери, и понижает уровень трафика. Окно перегрузки CWND и процедура медленного старта позволяют согласовать полную загрузку виртуального соединения и текущие возможности канала, минимизируя потери пакетов при перегрузке.

Потребности в быстрой передаче больших объемов данных постоянно растут. Однако основной транспортный протокол сетей TCP уже перестает удовлетворять потребностям. Долгое время ответа TCP в высокоскоростных сетях приводит к тому, что ощутимая часть пропускной способности такой сети остается неиспользованной.

Цель данной работы – выделение наиболее критичных параметров протоколов семейства TCP и определение степени их влияния на производительность TCP, выявление недостатков и преимуществ каждого алгоритма при различных сценариях работы в сети. Углубленное понимание работы механизмов протокола TCP и вопросов его производительности является важным условием успешного внедрения сетевых проектов.

Протоколы семейства TCP

Рассмотрим наиболее часто используемые в сетях связи современных информационных систем разновидности протоколов семейства TCP: Reno, HSTCP, Vegas, BIC, CUBIC [4 – 11].

Алгоритм **TCP Reno** является одним из наиболее старых. Реализация данного алгоритма добавила множество новых алгоритмов и усовершенствований к более ранним реализациям. Новые алгоритмы включают: медленный старт, предотвращение перегрузки (congestion avoidance) и быструю повторную передачу (fast retransmit). Усовершенствования состоят в модификации оценки времени прохождения пакетов по каналу связи до адресата и обратно

для установки значения времени ожидания повторной передачи. Смысл алгоритма предотвращения перегрузки заключается в удержании значения CWND в области максимально возможных значений. По существу эта оптимизация осуществляется с помощью потери пакетов.

Для соединений, в которых используются окна большого размера, применяется алгоритм временных меток (TCP Timestamps). Временные метки помогают проводить точное измерение времени обращения RTT для последующей корректировки значения таймера повторной передачи.

В работе особое внимание уделено алгоритму быстрой повторной передачи, так как он модифицируется в последующих версиях TCP. При работе алгоритма быстрой повторной передачи после получения малого числа двойных подтверждений для одного пакета TCP (ACK) источник данных заключает, что пакет был потерян и повторно передает пакет и все посланные после него пакеты без ожидания истечения таймера повторной передачи. Это ведет к уменьшению пропускной способности и увеличению и без того высокой загрузки канала. Алгоритм быстрой повторной передачи предотвращает канал от нахождения в пустом состоянии после быстрой повторной передачи и не переключается в режим медленного старта для наполнения канала после единственной потери пакета. Быстрое восстановление предполагает, что каждый полученный двойной ACK представляет один пакет, покинувший канал. Таким образом, в течение быстрого восстановления отправитель TCP способен подсчитать количество отправленных данных.

В TCP-Reno при нормальной ситуации размер окна меняется циклически. Размер окна увеличивается до тех пор, пока не произойдет потеря сегмента. TCP-Reno имеет две фазы изменения размера окна: фаза медленного старта и фаза избегания перегрузки.

Получение двойного ACK не является надежным сигналом потери пакета. Двойные ACK возникают и при смене маршрута обмена. По этой причине сигналом потери считается получение трех ACK пакетов подряд.

Если буфер переполнен, какое-то число сегментов будет потеряно. При этом может быть запущено несколько сценариев. Основной вариант – медленный старт, который запускается в рамках классического алгоритма TCP-Reno при потере сегмента и сопряженным с ним таймаутом (RTO) у отправителя, так как отправитель не получит сигнала подтверждения ACK для потерянного сегмента. Медленный старт предполагает установку окна перегрузки равным 1, а значение порога медленного старта равным половине значения CWND, при котором произошел таймаут. Сокращение CWND до единицы происходит потому, что отправитель не имеет никакой информации о состоянии сети. Далее, после каждого i-го подтверждения $CWND_{i+1} = CWND_i + 1$. Эта формула работает до тех пор, пока CWND не станет равным размеру окна window в режиме медленного старта. После этого рост CWND становится линейным. Если потери пакетов не происходит, значение CWND достигает значения window по умолчанию, задаваемого при конфигурации TCP-драйвера.

Реализация протокола **TCP Vegas** хорошо подходит для работы в сетях, когда необходимо определить доступную пропускную способность и динамически подстроить оптимальные параметры. Алгоритм Vegas оценивает буферизацию, происходящую в сети, и соответствующим образом управляет скоростью потока. Алгоритм в состоянии просчитать и уменьшить скорость потока прежде, чем произойдет потеря пакетов. Он контролирует размер окна путем мониторинга отправителем значения RTT (времени прохождения пакетов по каналу связи до адресата и обратно) для пакетов, посланных ранее. Если обнаруживается увеличение RTT, система узнает, что сеть приближается к перегрузке и сокращает ширину окна. Если RTT уменьшается, отправитель определит, что сеть преодолела перегрузку, и увеличит размер окна. Следовательно, размер окна в идеальной ситуации будет стремиться к требуемому значению. Эта модификация TCP требует высокого разрешения таймера отправителя.

HighSpeed TCP **(HSTCP)** является модификацией механизма управления перегрузкой в TCP, улучшает рабочие характеристики TCP в быстродействующих сетях с большой задерж-

кой. Когда величина CWND большая (больше чем 38 пакетов, что эквивалентно коэффициенту потерь 0,1%), данная версия TCP использует предварительно вычисленную таблицу для определения того, насколько должно быть увеличено окно перегрузки, когда получен ACK.

Медленная реакция TCP в быстродействующих сетях большой длины (fast long-distance networks) приводит к тому, что остается большой объем неиспользованной пропускной способности. **BIC TCP [6] и CUBIC TCP** [7,10 – 12]– это протоколы управления перегрузкой, разработанные для устранения этой проблемы. BIC TCP реализован и используется в ядре Linux версии 2.6.8 и выше. По умолчанию модель реализации протокола была заменена на CUBIC TCP в версии 2.6.19.

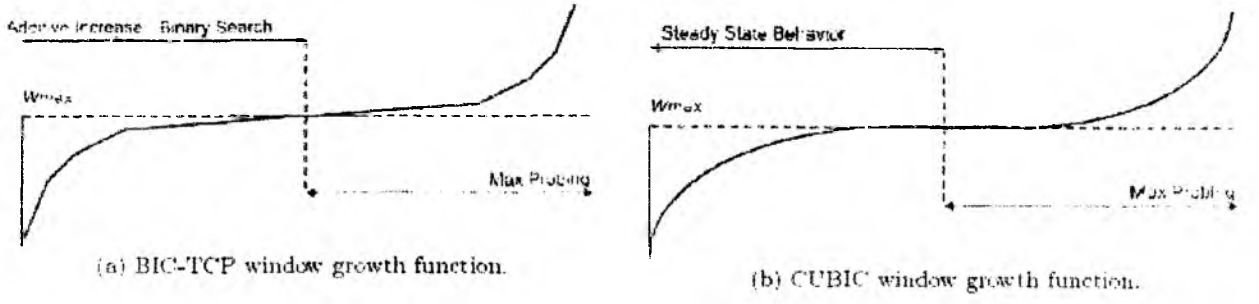
Главной особенностью **BIC** является уникальная функция роста окон (window growth function). На рис. 1, *а* изображен рост CWND для протокола BIC. Когда приходит информация о потере пакета, BIC уменьшает окно за счет мультипликативного фактора (или фактора роста). Размер окна непосредственно перед уменьшением устанавливается в качестве максимума, размер окна сразу после уменьшения устанавливается в качестве минимума. Затем BIC TCP выполняет двоичный поиск, используя эти два параметра и переходя к "середине" между максимумом (Wmax) и минимумом (Wmin).Так как потери пакетов происходят при размере окна, равном Wmax, то размер окна, при котором в сети пакеты будут проходить без потерь, должен находиться где-то между этими двумя значениями.

Однако переход к середине может быть слишком большим увеличением в пределах одного RTT. Так, если расстояние между серединой и текущим минимумом является большим, чем фиксированная постоянная (названная Smax), BIC TCP увеличивает текущий размер окна на Smax (линейное увеличение).Если BIC не получает уведомлений о потерях пакетов при измененном размере окна, то этот размер окна становится новым минимумом. Если происходят потери пакетов, то этот размер окна становится новым максимумом. Этот процесс продолжается, пока изменение окна меньше некоторой малой константы Smin, при достижении которой размер окна становится текущим максимумом. Поэтому растущая функция после уменьшения окна будет похожа на линейную, переходящую в логарифмическую.

Если окно увеличивается сверх максимума, среднее значение размера окна должно быть больше текущего максимума, а новый максимум должен быть установлен. BIC вводит новую фазу, именуемую "поиском максимума" ("max probing"). Рост функции во время поиска максимума противоположен росту во время двоичного поиска и пошагового прироста. Она растет экспоненциально (т.е. в начале – медленно), а затем – линейно. Поиск максимума использует функцию прироста окна симметрично используемой при двоичном увеличении (пошаговый прирост и двоичный поиск), только в другом порядке: он использует инверсию двоичного поиска (который логарифмичен; противоположностью ему является экспонента), а затем пошаговый прирост. Рис. 1, *а* демонстрирует рост функции при поиске максимума. Во время поиска максимума окно в начале возрастает медленно при поиске ближайшего максимума, а затем, после некоторого периода медленного роста, если новый максимум не найдется, т.е. будут наблюдаться потери пакетов, тогда принимается, что новый максимум находится гораздо дальше и происходит более быстрый рост при переходе на пошаговый прирост, где размер окна изменяется на большую фиксированную величину.

CUBIC TCP представляет собой реализацию протокола TCP с оптимизацией алгоритма управления перегрузкой для быстродействующих сетей с большими задержками. Версия CUBIC является менее агрессивной и более системной, чем BIC TCP, в которой значение ширины окна является кубической функцией времени с точкой перегиба, привязанной к окну, а не к событию как раньше. Рис. 1, *б* демонстрирует кубическую функцию, чей график похож на кривую окна BIC. Функция растет гораздо медленнее. Устанавливается начальное значение Wmax. После уменьшения окна оно растет очень быстро, но по мере приближения к Wmax, окно замедляет свой рост. При достижении Wmax прирост становится нулевым. Затем, окно начинает медленно расти, ускоряя рост по мере удаления от Wmax. На графике наблюдается такое же "плато" как и на кривой окна BIC, но уровень роста ускоряется намного

медленнее, чем у BIC. Такой медленный прирост свидетельствует об улучшенной совместимости протокола с TCP. Далее функция максимально упрощается в контроле окна, так как там используется уже только одна функция и по этой причине нет различных фаз.



(a) BIC-TCP window growth function. (b) CUBIC window growth function.

Рис. 1. Функция роста окна для BIC-TCP и CUBIC

Многообразие расширений протокола TCP позволяет администратору сети творчески подойти к построению и оптимизации своей сети. В системном реестре хранится большое количество параметров, которые позволяют усовершенствовать сеть. Одни расширения имеют два состояния – включено по умолчанию или выключено, для других предусмотрено несколько настраиваемых значений.

Сравнение работы алгоритмов при перегрузке сети

Одним из наиболее используемых симуляторов является ns2, который позволяет оценить производительность проектируемой или существующей сети, предоставляет данные, помогающие выработать рекомендации для повышения эффективности работы сети, выявить узкие места и спрогнозировать ее дальнейшее развитие. Доступность и гибкость позволяют успешно использовать ns2 для изучения механизма работы сетевых протоколов, протоколов маршрутизации и дисциплин обслуживания очередей, знакомства с основами моделирования вычислительных сетей, исследования и сравнения различных сетевых топологий. Была построена модельная сеть, состоящая из нескольких отправителей, получателя и установленных между ними маршрутизаторов (рис. 2). Узким местом в моделируемой сети являлся канал между маршрутизаторами. В ходе проведения численного эксперимента изменялись алгоритмы работы протокола TCP, размеры буфера маршрутизаторов, пропускная полоса на выходе маршрутизаторов. Трафик в рассматриваемой сети представляет собой самоподобный случайный процесс с задаваемыми пользователем параметрами, одним из которых является показатель Херста, который характеризует долгосрочную зависимость процесса.

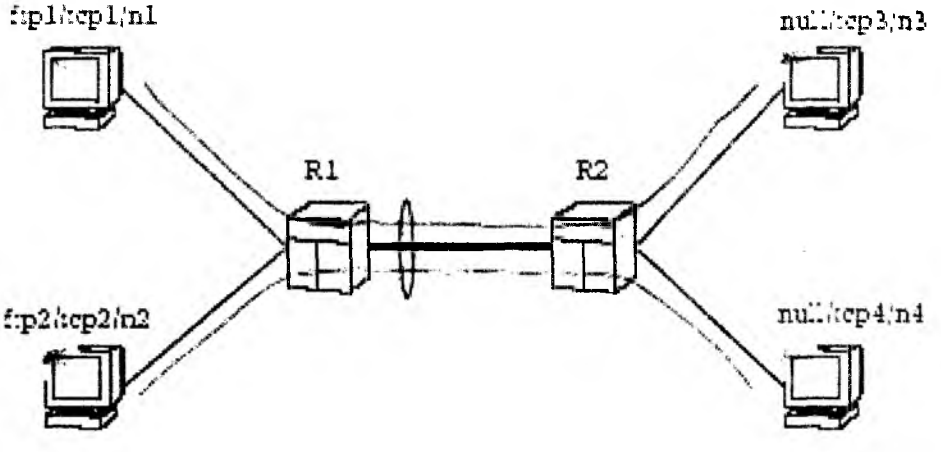


Рис. 2. Модельная сеть

Имитационное моделирование работы сети в симуляторе ns2 при использовании алгоритма Reno с опцией быстрого восстановления показало, что данный алгоритм оптимален лишь в случаях единичных потерь пакетов, т.е. когда отправитель Reno передает не более одного пакета за время прохождения одного пакета по каналу связи до адресата и обратно. Результаты исследования работы TCP-Reno показывают, что каждое соединение обычно теряет около двух пакетов в каждом эпизоде перегрузки. Потери случаются, когда буфер полон и одно соединение увеличивает размер окна на единицу. Когда ячейки из этого нового пакета приходят в буфер (например, в случае использования технологии ATM), они обычно вызывают потерю ячеек, принадлежащих двум пакетам (конец пакета, пришедшего из другого соединения, и начало следующего). Следовательно, в среднем следует ожидать потерю трех пакетов на один эпизод перегрузки.

В TCP-Vegas подтверждение ACK объединяется с пакетом данных (который называется *piggybacking*) вместо независимой передачи, как в других алгоритмах. Это экономит пятьдесят процентов времени, в отличие от нормального выполнения передачи TCP подтверждения ACK, и не тратит дополнительного времени. Поэтому TCP-Vegas может передавать большее количество данных.

Недостатком модификации HSTCP является то, что при нескольких потоках с разными RTT они некорректно распределяют полосу. В этих условиях заметные трудности создает синхронизация потерь для конкурирующих потоков.

Исследование изменения CWND.

Вовлеченные соединения оказываются в определенной мере синхронизованными. Это связано с тем, что когда происходит любое столкновение пакетов в канале, сопряженное с увеличением ширины окна, когда буфер полон, все приходящие ячейки, принадлежащие пакетам, отбрасываются. В предположении о постоянной готовности отправителя к передаче и о том, что временной разброс ячеек не превосходит времени пересылки пакета во входном канале, все соединения будут передавать ячейки в течение времени транспортировки пакетов, вовлеченных в столкновение. Следовательно, все соединения теряют пакеты и сокращают вдвое ширину окна в пределах RTT.

На рис. 3 показано изменение количества отосланных и принятых сегментов и соответствующее изменение размера окна. Приведенная зависимость типична для всех протоколов семейства TCP. Из графика видно, что количество отосланных сегментов растет линейно, в соответствии с увеличением размера окна. Однако количество потерянных данных в момент уменьшения размера окна значительно больше допустимых потерь, определяемых QoS (качество обслуживания сети).

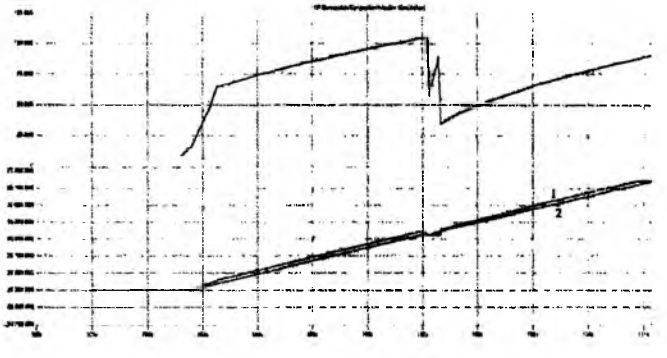


Рис. 3. Изменение CWND (вверху) и количества отосланных сегментов (линия 1) и принятых (линия 2) с течением времени (внизу)

TCP-Reno при обнаружении потери сокращает окно вдвое. Основной проблемой с TCP-Reno является то, что могут быть множественные ограничения на окно, связанные с одним эпизодом перегрузки, и что множественные потери могут приводить к таймауту (который на

практике вызывает значительное снижение пропускной способности, если используется таймеры с низким разрешением).

Протокол TCP-Vegas пытается реализовать ряд усовершенствований, таких, как более изощренная обработка и оценка RTT. Изменение значения RTT происходит по многим причинам. Для того чтобы существенно улучшить существующие версии TCP, необходимо избегать резких сокращений размеров окна (например, в TCP-Vegas при оценке RTT).

При анализе протоколов BIC-TCP и CUBIC следует обратить внимание на два момента: RTT-доступность и внутрипротокольная доступность. Стратегия в контроле этих свойств заключается в том, чтобы позволить окну расти на уровне, зависящем от прошедшего времени после последней потери пакета. Окно размером W определяется следующей функцией:

$$W = C(T - K)^3 + W_{\max},$$

где $W_{\max}$ – это размер окна непосредственно перед предыдущим уменьшением, T – прошедшее время после уменьшения окна, K – растущий коэффициент уменьшения после потери пакета, C – константа, зависящая от выбора операционной системы.

Такая функция обеспечивает внутрипротокольную доступность для соревнующихся потоков одного протокола. Два потока разделят между собой всю доступную им пропускную способность, так как у них одинаковый растущий коэффициент, поэтому поток с большим $W_{\max}$ значительнее уменьшится, т.е. K больше, если $W_{\max}$ больше.

Функция также обеспечивает лучшее значение RTT доступности в силу того, что уровень роста окна определяется временем T. Это гарантирует линейную RTT доступность за счет того, что любые соревнующиеся потоки с различным RTT будут иметь одинаковое время T после синхронизации потери пакета (TCP и BIC предлагают квадратную величину RTT доступности в переводе на коэффициент пропускной способности).

BIC увеличивает окно пошагово, когда разница окна при RTT становится больше некоторой величины. Напротив, CUBIC увеличивает окно в зависимости от текущего момента; при коротких RTT линейное увеличение RTT меньше, хотя и остается постоянным в реальном времени.

Увеличение окна в реальном времени чрезвычайно способствует улучшению совместимости CUBIC с другими протоколами TCP. Функция роста окна другого протокола, зависящего от RTT, растет пропорционально быстрее в реальном времени в сетях с коротким RTT, тогда как CUBIC будет расти независимо от RTT.

Также нужно отметить, что полная функция роста окна описывается лишь одной функцией: в CUBIC не требуются различные фазы контроля окна (как, например, пошаговое увеличение, двоичный поиск и поиск максимума в BIC). Это упрощает анализ CUBIC.

Хорошая производительность протокола BIC: достигается за счет медленного увеличения $W_{\max}$ и линейный рост во время пошагового прироста и поиска максимума; дает совместимость с алгоритмами TCP. Функция изменения окна: дает время для конкурирующих TCP-потоков увеличить их окна; улучшает стабильность протокола и сети; при приближении окна к максимуму, при достижении максимума продолжает удерживать окно длительное время, в результате этого уменьшаются колебания окна. BIC стремится нагрузить сеть более плавно, чем другие протоколы. Это делается с целью снизить «ударную» нагрузку для остальных конкурирующих потоков. Линейное увеличение при пошаговом приросте и поиске максимума также способствует улучшению пропускной способности протокола. Помимо этого, также достигается предельная пропускная доступность, разделяемая между конкурирующими BIC-потоками с совпадающимм временем RTT.

Большинство высокоскоростных вариантов протокола TCP имеют тот или иной вид "режима TCP", при котором протокол ведет себя так же, как TCP. HSTCP и Vegas переходят в свои режимы TCP, когда размер окна меньше некоторой небольшой константы (обычно, это где-то около 30 пакетов).

Фактически условия, в которых TCP работает успешно, зависят от продолжительности периода загрузки, т.е. периода между двумя последовательными потерями. Например, если RTT равен 1 *мс*, то TCP может увеличивать свое окно по 1000 пакетов в секунду (при отложенном подтверждении – по 500), а такая скорость должна быть достаточно быстрой для полного использования емкости в большинстве широкополосных сетей (если там достаточно места в буфере в "узком" соединении). С другой стороны, если RTT равно 200 *мс*, то TCP может увеличивать окно на 5 пакетов в секунду. Контроль режима TCP при помощи лишь размера окна может оказаться подходящим далеко не для всех ситуаций.

Выводы

Таким образом, в ходе проведенных экспериментов и анализа результатов выявлены следующие преимущества и недостатки работы алгоритмов TCP:

– TCP Reno является оптимальным только в случае одиночных потерь пакетов, т.е. когда отправитель передает не более одного пакета за время прохождения его по каналу связи до получателя и назад. Поэтому его используют в локальных незагруженных сетях;

– HSTCP является оптимальным для высокоскоростных сетевых соединений, так как он оптимизирует производительности системы контроля насыщения TCP. Этот механизм использует переключатель режима для изменения параметров стека TCP Reno на основе существующих в сети условий с учетом интересов других потоков данных Reno и HSTCP;

– TCP Vegas целесообразно использовать для сетей с перегруженными каналами передачи, так как он предотвращает перегрузку за счет оценки полосы пропускания. TCP Vegas изменяет скорость передачи данных за счет управления размером окна насыщения TCP. При использовании алгоритма TCP Vegas следует ожидать снижения числа теряемых пакетов, но этот механизм обеспечивает менее агрессивный контроль насыщения, нежели TCP Reno;

– BIC-TCP является оптимальным для скоростных сетей при работе нескольких протоколов TCP. Он обеспечивает хорошую масштабируемость для скоростных сетей, эквивалентность для конкурирующих потоков и стабильность с низким уровнем осцилляций размера окна. Однако функция роста окна BIC-TCP может быть слишком агрессивной для различных алгоритмов TCP, особенно при малых значениях RTT или для низкоскоростных сетей. Более того, несколько фаз управления окном добавляет излишнюю сложность в реализацию протокола и анализ его рабочих характеристик;

– CUBIC TCP предназначен для использования в современных операционных системах. Он обеспечивает хорошее масштабирование для скоростных сетей, хотя и не отличается беспристрастностью и лишен многих недостатков BIC TCP, однако также обладает некоторыми недостатками и несовместимостью с классическими TCP проколами. Именно поэтому исследование и усовершенствование протокола TCP не прекращается до сих пор.

Список литературы: 1. *Leland W.E., M.S. Taqqu, Willinger W., and Wilson D.V.* "On the self-similarnature of ethernet traffic", *IEEE/ACM Transactions of Networking,* vol. 2(1), 1994 pp., 1-15. 2. *Столингс В.* Современные компьютерные сети. СПб.: Питер, 2003, с.783. 3. *Kirichenko L., Radivilova T., Karpukhin,O.* "Improvement quality of network service under selfsimilar loading // Стратегия качества в промышленности и образовании : 4-я междунар. конф.: материалы конф., Варна, 2008. С. 612–615. 4. *Floyd S. and Jacobson V.* Random Early Detection Gateways for Congestion Avoidance // IEEE/ACM Transactions on Networking, vol. 1(4), August 1993, pp. 397-413. 5. *Fall K. and Floyd S.* "Simulation-based comparison of tahoe, reno, and sack TCP // Computer Communication Review, vol. 26, 2002, pp. 5-21. 6. *Harfoush L. Xu, K., and Rhee I.*. Binary increase congestion Control (BIC) for Fast Long-Distance Networks. Proceedings of IEEE INFOCOM 2004, Hong Kong, March 2004.7. *Ha S., Rhee I. and Xu L.*, CUBIC: A New TCP-Friendly High-Speed TCP Variant // ACM SIGOPS Operating System Review, Volume 42, Issue 5, July 2008, Page(s):64-74, 2008.8. *Ha S., Le L., Rhee I., and Xu L.* Impact of background traffic on performance of high-speed TCP variant protocols // Computer Networks. 2007, p.142. 7. *Cai H., Eun D., Ha S., Rhee I., and Xu L.* Stochastic Ordering for Internet Congestion Control and its Applications // IEEE INFOCOM. 2007, p.145-168. 9. *Lachlan Andrew, Cesar Marcondes, Sally Floyd, Law-*

rence Dunn, Romaric Guillier, Wang Gang, Lars Eggert, Sangtae Ha and Injong Rhee. Towards a Common TCP Evaluation Suite // PFLDnet, Manchester, UK, 2008, p.436. 10. *Rhee I., and Xu L.* CUBIC: A New TCP-Friendly High-Speed TCP Variants // PFLDnet, Lyon, France, 2005, p.38-87.11. *Карпухин А.В.* Особенности реализации протокола TCP в современных компьютерных сетях // Системи обробки інформації. Х.: ХУПС, 2009. №6(80). С.49-53. 12. *Семенов Ю.А.* Модели реализации протокола TCP и его перспективы [Электронный ресурс].Режим доступа:http://book.itep.ru/4/44/tcp.htm.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 17.05.2010