УДК 004.738.5.057.4

Ю.Ю. Завизиступ, А.А. Коваленко, А.С. Мохаммад

*Харьковский национальный университет радиоэлектроники*

# МОДИФИКАЦИЯ ПРОТОКОЛА TCP SACK

*Предложена модификация протокола TCP SACK, позволяющая сдерживать падение пропускной способности при потерях сегментов в соединении. Произведено моделирование TCP соединений в состоянии перегрузки на основе аппарата цветных сетей Петри, показана корректность работы модифицированного протокола.*

***протокол TCP, соединение, перегрузка, плавающее окно, сегмент, подтверждение, модель***

## Введение

**Постановка задачи и анализ литературы.** Протокол TCP разрабатывался с целью обеспечения надежности сквозного соединения между двумя узлами в сети. Со временем он стал фактически протоколом транспортного уровня в подавляющем большинстве компьютерных сетей, в том числе и Интернет, и используется повсеместно. Протокол TCP развивался в течение последних десятилетий, и до сих пор его основными версиями были Tahoe TCP, Reno TCP, NewReno TCP [1] и SACK TCP [2, 3]. Tahoe TCP – самая старая версия, содержащая только алгоритмы медленного старта, избегания перегрузки и быстрой повторной передачи. По своей сути, протокол Reno TCP является усовершенствованной версией Tahoe TCP, основным его недостатком является то, что при потерях многочисленных сегментов в пределах одного окна передаваемых данных происходит значительное снижение производительности. Протоколы NewReno TCP и SACK TCP являются основными решениями для проблемы потери множества сегментов.

**Цель статьи.** В данной статье описывается модернизация механизма SACK TCP, позволяющая ему достигать больших значений производительности за счет уменьшения времени, которое проводит протокол в фазе медленного старта.

**Результаты теоретических исследований и постановка проблемы.** Протокол SACK TCP является улучшенной версией Reno TCP. Совместная реализация механизмов выборочных подтверждений (selective acknowledgement – SACK) и выборочной повторной передачи способствуют повышению эффективности работы протокола TCP при потере множества сегментов из одного окна передаваемых данных. Механизм выборочных подтверждений реализован путем добавления к пакету подтверждения ряда непоследовательных блоков данных. Когда еще не подтвержденный сегмент попадает в буфер получателя, он отсылает отправителю пакет выборочных подтверждений.

Процесс управления перегрузками протокола SACK TCP состоит из следующих фаз: медленный старт, избегание перегрузки, быстрая повторная передача/быстрое восстановление. В этом процессе можно выделить четыре параметра:

– плавающее окно (congestion window, cwnd) – максимальное количество данных, одновременно отправляемых в сеть отправителем;

– заявленное окно получателя (rwnd) – максимально возможное количество данных для одновременного приема получателем;

– пороговое значение (ssthresh) – это величина, служащая порогом между фазами медленного старта и избегания перегрузки протокола TCP;

– величина *pipe* – отражает предполагаемое количество сегментов на маршруте соединения, ожидающих доставки.

Также имеется параметр scoreboard, которая отслеживает блоки последовательных данных, прибывающих получателю. Период таймаута повторной передачи (RTO) тоже является важным параметром для управления перегрузками протоколом SACK TCP [1].

Для количественного определения последствий сбоя используется такой параметр, как время пересылки данных (TTI):

$$TTI = ATT - NTT, \qquad (1)$$

где ATT (Actual Transfer Time) обозначает фактическое время пересылки данных в случае сбоев, а NTT (Normal Transfer Time) – расчетное время пересылки данных при нормальной работе сети.

Параметр dup-ACK является признаком наличия дублирующихся подтверждений, а частичный пакет ACK – подтверждение приема пакета, полученное во время фазы быстрого восстановления.

В момент открытия соединения протоколом TCP размер плавающего окна устанавливается в единицу и запускается фаза медленного старта. Размер cwnd экспоненциально увеличивается по мере получения отправителем подтверждений и этот процесс длится до тех пор, пока значение cwnd не сравняется с ssthresh. Затем TCP переходит в фазу избегания перегрузок, в течение которой cwnd увеличивается на единицу за одно время RTT (время кругового обращения пакета по маршруту). При нормальной работе сети, протокол TCP остается в фазе

 

избегания перегрузки до тех пор, данные полностью не передадутся и соединение не будет закрыто.

В случае, когда некоторые сегменты теряются во время сбоя, клиент получает лишь все последующие сегменты. В результате отправитель получает три дублирующих подтверждения, и протокол TCP переходит в фазу быстрой повторной передачи/быстрого восстановления. Он повторно отправляет первый неподтвержденный сегмент, и устанавливает значение pipe в количество ожидающих доставки сегментов (обычно число неподтвержденных сегментов равняется rwnd). В момент запуска фазы быстрой повторной передачи/быстрого восстановления отправитель подразумевает потерю сегмента и устанавливает значение pipe в rwnd-l, значение ssthresh в pipe/2, а cwnd увеличивает на величину ssthresh+3:

$$\text{cwnd} = [\text{pipe}/2] + 3. \quad (2)$$

В фазе быстрого восстановления значение pipe увеличивается на единицу при повторной отправке сегмента и уменьшается на единицу при получении дублирующегося подтверждения. Для каждого частичного ACK значение параметра pipe уменьшается на два, поскольку каждый частичный ACK фактически представляет собой два сегмента, покинувших канал связи: исходный сегмент, который считается потерянным, и повторно переданный сегмент. Когда значение pipe станет меньше cwnd, отправитель сверится со счетчиком, и либо заново отправит первый неподтвержденный сегмент, либо отправит новый сегмент, если более не останется неподтвержденных сегментов. Если обозначить количество дублирующихся пакетов ACK $n_D$, полученных отправителем до того, как значение pipe станет меньше cwnd, то

$$\text{pipe} - n_D = \text{cwnd} - 1; \quad (3)$$

$$n_D = \text{pipe} - \text{cwnd} + 1. \quad (4)$$

Учитывая (2), получается:

$$n_D = \text{pipe} - [\text{pipe}/2] - 2. \quad (5)$$

Если обозначить предельно допустимое количество потерянных сегментов с помощью $n_{c,s}$, то

$$n_{c,s} = \text{pipe} - n_D = [\text{pipe}/2] + 2. \quad (6)$$

В нормальном состоянии получатель отправляет подтверждение только для каждого второго полноразмерного сообщения в течение 200 миллисекунд после принятия первого неподтвержденного сегмента. Подтверждение получения внеочередных сегментов также должно происходить немедленно: когда первый внеочередной сегмент появляется в окне и у получателя нет неподтвержденных сегментов, он отправляет дублирующееся подтверждение. Такая ситуация называется *сбой первого типа*.

Если первый внеочередной сегмент прибывает в течение 200 миллисекунд после неподтвержденного сегмента, получатель не станет посылать дублирующее подтверждение, а отправит только подтверждение предыдущего неподтвержденного сегмента. Получение каждого сегмента, после первого внеочередного сегмента, приведет к генерации дублирующего подтверждения. Такая ситуация называется *сбоем второго типа*. Уравнение (6) относится к сбою первого типа. Для сбоя второго типа в уравнении (6) нужно уменьшить $n_{c,s}$ на единицу, для учета сегмента, запускающего пакет ACK для ранее неподтвержденного сегмента.

$$n_{c,s} = \begin{cases} [\text{pipe}/2] + 2 & \text{при сбое первого типа;} \\ [\text{pipe}/2] + 1 & \text{при сбое второго типа.} \end{cases} \quad (7)$$

Если в пределах одного окна теряется менее $n_{c,s}$ сегментов, то у получателя будет достаточное количество сегментов для запуска дублирующих подтверждений, которые в итоге сделают значение pipe меньшим, чем cwnd у отправителя. В таком случае отправитель может повторно отправлять другие потерянные сегменты после повторной передачи первого неподтвержденного сегмента. Когда прибывает недублирующийся пакет ACK, подтверждающий все данные, ожидающие доставки при запуске фазы быстрой повторной передачи/быстрого восстановления, TCP выходит из фазы быстрой повторной передачи/быстрого восстановления и переходит к фазе избегания перегрузок. Таким образом, протокол SACK TCP быстро восстанавливается после потери менее $n_{c,s}$ сегментов. Общее время передачи при этом увеличивается ненамного.

Если теряется более $n_{c,s}$ сегментов и остается еще три (при сбое первого типа) или четыре (при сбое второго типа) непереданных повторно сегмента, следующих за потерянными сегментами, отправитель получит достаточное количество дублирующих подтверждений для запуска фазы быстрой повторной передачи/быстрого восстановления. Эти дублирующие подтверждения не сделают значение pipe меньшим, чем cwnd и отправитель не станет повторно передавать остальные потерянные сегменты. После таймаута протокол TCP перейдет в фазу медленного старта и ему необходимо повторно передать первое неподтвержденное сообщение на текущий момент, которым будет являться второй потерянный сегмент.

Если же менее трех (сбой первого типа) или четырех (сбой второго типа) сегментов идут вслед за потерянными сегментами, запуск фазы быстрой повторной передачи/быстрого восстановления не произойдет, так как дублирующего подтверждения будет недостаточно и запустится таймаут. Когда закончится время повторной передачи сообщения, протокол TCP перейдет в фазу медленного старта и повторно передаст первый потерянный сегмент.

Несмотря на то, что в этих двух случаях описаны разные фазы передач сегментов, общее время передачи изменяется ненамного, так как основным фактором здесь выступает таймаут.

Когда перегрузка длится достаточно долго и сегмент, повторно переданный по причине таймаута, тоже теряется, снова происходят изменения общего времени передачи, что обусловлено тем, что

при отправке повторно передаваемого сегмента таймер повторной передачи увеличивается вдвое. Если повторная передача не удается, отправитель будет ждать вдвое дольше предыдущего RTO до истечения таймаута и повторной передачи самого первого неподтвержденного сегмента, и так далее. Процесс будет продолжаться до тех пор, пока TCP не закроет это соединение, вызывая катастрофическую потерю производительность.

## Основной материал исследований

**Описание разработанного алгоритма.** Если теряется более $n_{c,s}$ сегментов, и остается еще три (в сбое первого типа) или четыре (в сбое второго типа) непереданных повторно сегмента, идущих вслед за потерянными сегментами, прибывающими к получателю, отправитель перейдет в фазу быстрой повторной передачи/быстрого восстановления, и отправит первый неподтвержденный сегмент. Повторно переданный сегмент приводит к частичному ACK, но в данном случае этот единственный частичный ACK, возможно, вместе с дублирующими подтверждениями, предшествующими ему, не сделает значение pipe меньше, чем cwnd. Отправитель запустит таймаут, что приведет к значительному увеличению TTI.

Когда частичный ACK прибывает к отправителю, это однозначно указывает на то, что после того, как отправитель переходит в фазу быстрой повторной передачи/быстрого восстановления, канал связи продолжает работать, хотя, возможно, и с очень низкой производительностью. В такой ситуации отправитель должен отправить несколько пробных пустых сообщений, вместо ожидания таймаута. Данная «проба» канала, с одной стороны, помогает отправителю попытаться передать некоторые сегменты в течение того времени, которое он просто потратит на ожидание; с другой стороны, может породить другие частичные пакеты ACK, которые сделают значение pipe станет меньше cwnd и удастся избежать таймаута.

Используется следующий механизм для улучшения свойств протокола SACK TCP в области управления перегрузками. Когда протокол TCP находится в фазе быстрой повторной передачи/быстрого восстановления и прибывает частичный ACK, TCP проверяет сделает ли этот ACK значение pipe меньшим, чем cwnd. Если да, то поведение TCP не изменяется; иначе TCP либо повторно передаст самый первый неподтвержденный сегмент, либо передаст новый сегмент, если уже не будет неподтвержденных сегментов.

На рис. 1 показана схема моделируемого соединения, состоящего из отправителя пакетов, оптического канала связи OC-3 (со скоростью передачи сигнала 155.52 Mbps), каналов связи OC-192 (со скоростью передачи сигнала 9953,28 Mbps), канала связи DS1 (со скоростью 1,544 Mbps), локальных маршрутизаторов Cisco 12008 (ЛМ), участка глобальной сети (NSFN) и получателя пакетов.

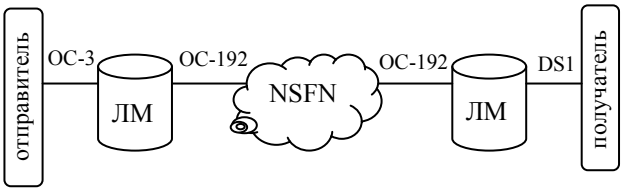


Рис. 1. Схема моделируемого соединения

**Формальное доказательство.** В работе использовался пакет Design/CPN для расчета графа присутствия (occurrence graph), анализа ограниченности и достижимости модели [4]. Построены различные графы присутствия для трех типичных сценариев: 0 потерянных сегментов, 19 потерянных сегментов и 20 потерянных сегментов. Свойства этих графов приведены в табл. 1.

В модели каждое из мест "Sender.RCV ACK", "Sender.CWND", "Sender.SSTHRESH", "Receiver.Q LENGTH" в любом случае должны иметь маркер. «Предельные значения целых чисел» (Integer Bounds) в таблице подтверждают это, так как в них верхние и нижние предельные значения целых чисел всегда равны 1. Место "Receiver.SNDACK" в любом случае должно иметь либо один, либо ни одного маркера, что подтверждается тем, что его верхнее предельное значение целого числа равно 1, а нижнее равно 0. Также, каждый из "Sender.SNDSEG" и "Sender.RCVSEG" в любом случае должны иметь ноль, один или два маркера; верхнее предельное значение 2 и нижнее предельное значение 0 подтверждают ограниченность этих двух мест.

В случае потери 0 сегментов, верхний предел мультинабора "Sender.SSTHRESH" состоит только из 45 меток, так как по данному сценарию сбой не происходит. В случае потери 19 или 20 сегментов, предел содержит и 11 и 45 меток соответственно, поскольку параметр ssthresh после сбоя устанавливается в значение pipe/2, и в данном случае ssthresh может быть 11 или 45. Во всех трех случаях предел верхнего мультинабора "Receiver.Q_LENGTH" имеет 12 меток, самый большой из которых 18 меток. Это показывает, что максимум 18 сегментов находятся в ожидании обслуживания получателем, что вполне обоснованно, так как значение rwnd равно 23 в экспериментальном сценарии, и количество сегментов, ожидающих обслуживания, должно быть менее 23 сегментов. Пределы нижнего мультинабора для двух мест достаточно просты: у них либо есть один маркер, либо его нет, поскольку каждое из этих двух мест в любом случае должно иметь по одному маркеру. В данном случае, если предел верхнего мультинабора мест имеет только один маркер, тогда соответствующий предел нижнего мультинабора также должен содержать один маркер, если верхний предел мультинабора мест име-

ет множество маркеров, соответствующий нижний предел мультинабора должен быть пустым. Таким образом, данные по ограниченности, представленные графами присутствия показывают, что места ограничены, как того требует протокол SACK TCP. Из этого следует корректность модели SACK TCP.

Таблица 1

Графы присутствия для сценариев при 0, 19 и 20 потерянных сегментах (стандартный протокол SACK TCP)

| | | 0 потерь | | 19 потерь | | 20 потерь | |
|---|---|---|---|---|---|---|---|
| Целочисленные границы | места | верхняя | нижняя | верхняя | нижняя | верхняя | нижняя |
| | Sender.PROBE | 1 | 1 | 1 | 1 | 1 | 1 |
| | Sender.SND SEG | 2 | 0 | 2 | 0 | 2 | 0 |
| | Sender. RCV_ACK | 1 | 1 | 1 | 1 | 1 | 1 |
| | Receiver.SND ACK | 1 | 0 | 1 | 0 | 1 | 0 |
| | Sender.SSTHRESH | 1 | 1 | 1 | 1 | 1 | 1 |
| Верхние пределы мультинабора | Sender.PROBE` | 1`0 | | 1`0+1`1 | | 1`0 | |
| | Sender.SSTHRESH | 1`45 | | 1`11+1`45 | | 1`11+1`45 | |
| | Receiver.Q_LENGTH | 1`0+1`1+1`2+ 1`3+1`4+1`5+ 1`6+1`7+1`8+ 1`9+1`10+1`11+ 1`12+1`13+ 1`14+ 1`15+ 1`16+ 1`17+1`18 | | 1`0+1`1+1`2+ 1`3+1`4+1`5+ 1`6+1`7+1`8+ 1`9+1`10+1`11+ 1`12+1`13+ 1`14+ 1`15+ 1`16+ 1`17+1`18 | | 1`0+1`1+1`2+ 1`3+1`4+1`5+ 1`6+1`7+1`8+ 1`9+1`10+1`11+ 1`12+1`13+ 1`14+ 1`15+ 1`16+ 1`17+1`18 | |
| Нижние пределы мультинабора | Sender.SSTHRESH | 1`45 | | пусто | | пусто | |
| Свойства живучести | Заблокированные переходы | Sender.RTX SEG1 Sender.RTX SEG2 Sender.TIMEOUT | | Sender.TIMEOUT | | | |

В пакете Design/CPN пассивное маркирование - это маркирование без активированной передачи сообщения. При подсчете графа присутствия маркировки пересчитываются последовательно. Исходная маркировка нумеруется первой, а каждая следующая считаемая маркировка нумеруется последовательно. В графе «Свойства живучести» табл. 1 каждый из трех сценариев имеет пассивную маркировку. Зная номера пассивных маркировок, можно их проанализировать и убедиться в том, что они являются маркировками, соответствующими последним шагам передачи данных, при которой сегменты прибыли адресату. Сервисной программой запросов в Design/CPN эти маркировки проверены на предмет достижимости от изначальных маркировок, где все сегменты находятся у источника, указывая на то, что все сегменты отправителя могут быть успешно переданы тремя сценариями.

В модели на сети Петри множественные переходы могут быть активированы одновременно, что является обычным для параллельных систем. При имитационном запуске, Design/CPN заказывает все активированные передачи наугад. Поэтому успешное моделирование указывает на корректность только одной из многих возможных последовательностей выполнения. Но, разрабатывая графы присутствия для модели на основе сети Петри и анализируя ее достижимость, можно сделать вывод, что для всех последовательностей передача данных всегда выполняется успешно, поскольку каждый граф присутствия имеет только одну пассивную маркировку, которая соответствует последнему шагу передачи соответствующих данных, и такая пассивная маркировка всегда доступна от первоначальной маркировки. Информация о достижимости в табл. 1 показывает корректность модели SACK TCP.

В Design/CPN заблокированный переход определяется как никогда не активируемый в достижимой маркировке переход. В представленной модели "Sender.RTX SEG1" и "Sender.RTX SEG2" относятся к фазе быстрой повторной передачи/быстрого восстановления, а "Sender.TIMEOUT" - к таймауту. При отсутствии потерь TCP не переходит к фазе быстрой повторной передачи/быстрого восстановления и таймауту, в итоге четыре ранее упомянутых перехода не активируются. В случае потери 12 сегментов происходит быстрая повторная передача/быстрое восстановление, так что переходы, относящиеся к таймауту, оказываются заблокированными. При потере 19 сегментов TCP не переходит в фазу быстрой повторной передачи/быстрого восстановления по причине недостаточного количества пакетов ACK и наступает тайм-аут. В результате, два перехода, относящиеся к фазе быстрой повторной передачи/быстрого восстановления становятся заблокированными. Информация о заблокированном переходе в табл. 1 подтверждает корректность представленной модели.

Как уже было отмечено, если в окне передачи три (в сбое первого типа) или четыре (в сбое второго типа) сегмента, отправитель может перейти в фазу быстрой повторной передачи/быстрого восстановления. $n'_{c,s}$ обозначает предельное количество поте-

рянных сегментов, если из окна теряется три или четыре сегмента, следующая формула приводит отношения между $n'_{c,s}$ и rwnd:

$$n'_{c,s} = \begin{cases} \text{rwnd} - 3 \text{ при сбое первого типа}; \\ \text{rwnd} - 4 \text{ при сбое второго типа}, \end{cases} \quad (8)$$

иллюстрируя сбои второго типа, $n'_{c,s}$ будет равен 23 – 4 = 19.

Четыре кривые на рис. 2 иллюстрируют изменения, происходящие в плавающем окне отправителя в трех случаях. Если в пределах одного окна теряется менее $n_{c,s}$ сегментов, TCP восстанавливает потери и общее время передачи увеличивается ненамного. Если количество потерянных сегментов находится в пределах от $n_{c,s}$ до $n'_{c,s}$, то первые несколько частичных пакетов ACK уменьшат значение pipe и отправитель передаст нескольких пробных сообщений, что позволит избежать таймаута, и, следовательно, сократить общее время передачи. Таким образом, в случае потери от 1 до $n'_{c,s}$ сегментов, у протокола TCP наблюдаются практически одинаковые изменения cwnd и задержек в цепи передачи сегмента. Если теряется более $n'_{c,s}$ сегментов, за потерянными сегментами последуют менее трех (сбои первого типа) или четырех (сбои второго типа) сегментов. В этом случае вход в фазу быстрой повторной передачи/быстрого восстановлении не произойдет по причине недостаточного количества дублирующихся пакетов ACK, что приведет к таймауту.

Оценив эффективность модифицированной версии SACK TCP, проанализирована ее ограниченность и достижимость. В усовершенствованной версии SACK TCP, случаи с потерей 0, 19 и 20 сегментов соответствуют случаям с потерей 0, 12 и 13 сегментов в стандартном SACK TCP, соответственно; а сравнение зависимостей полного времени передачи пакета от количества потерянных пакетов для стандартного и модернизированного вариантов протокола SACK TCP приведено на рис. 3.

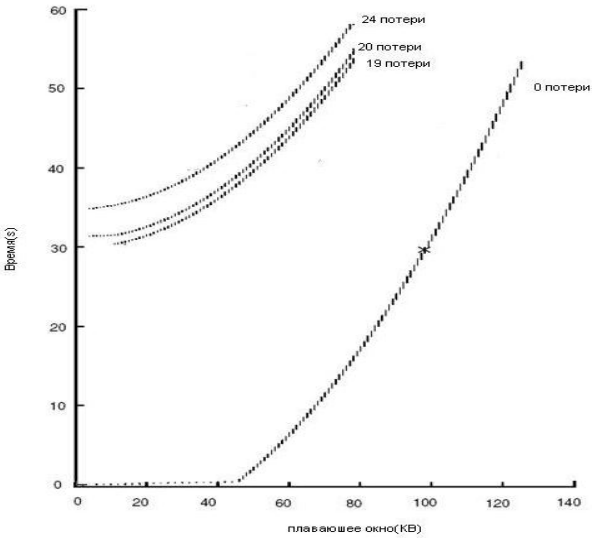


Рис. 2. Зависимость времени передачи от размера плавающего окна

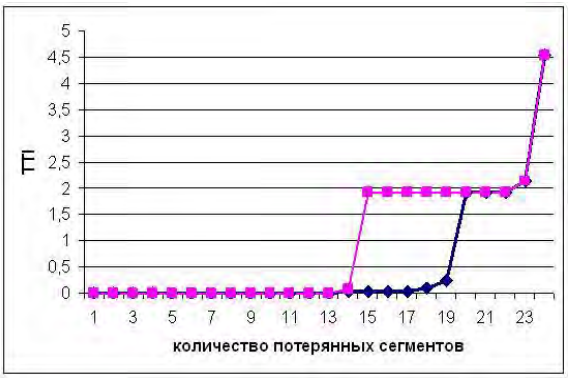


Рис. 3. Зависимость полного времени передачи пакета от количества потерянных пакетов для стандартного протокола SACK TCP (верхняя кривая) и его модернизированного варианта (нижняя кривая)

## Выводы

Предложена модифицированная версия протокола SACK TCP, позволяющая ему сохранять характерные показатели производительности при значительно большем количестве потерь (19-20 потерянных сегментах из одного окна данных против 12-13 в оригинальном варианте). На основе аппарата цветных сетей Петри разработаны модели соединения в состоянии перегрузки как для протокола SACK TCP, так и модифицированной его версии. Также проведено моделирование процессов перегрузки в TCP соединениях с использованием пакета Design/CPN. Установлено, что при числе потерянных сегментов, заданных значениями $n_{c,s}$ и $n'_{c,s}$ (критические значения потерь сегментов) существующая реализация SACK TCP в условиях перегрузок имеет низкую эффективность восстановления потерь. Модифицированная версия протокола в ответ на частичные пакеты ACK, при числе пакетов в очереди меньшем значения cwnd посылает пробные сегменты, что снижает время, необходимое для передачи, а применение изложенной методики позволяет повысить производительность соединения в среднем до 40%. Также показано, что данные изменения не влияют на корректность работы протокола TCP в том смысле, что все состояния сохраняют свои предельные значения, а желаемое конечное состояние всегда достижимо от начального состояния.

## Список литературы

1. *Fall K., Floyd S. Simulation-based Comparisons of Tahoe, Reno and SACK TCP // Computer Communication Review. – Jul. 1996. – V. 26 No. 3. – P. 5-21.*

2. *Mathis M., Mahdavi J. et al. RFC 2018: TCP Selective Acknowledgement Options. – Oct. 1996.*

3. *Ye Q., MacGregor M. On the Resilience of SACK and Reno TCP // Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003), Banff, Alberta, Canada, Oct. 2003. – P. 236-243.*

4. *http://www.daimi.aau.dk/designCPN/.*

*Поступила в редколлегию 6.09.2006*

**Рецензент:** д-р техн. наук, проф. С.В. Козелков, Национальная академия обороны Украины, Киев.