УДК 681.3:519.2

**Т.А. РАДИВИЛОВА, Л.О. КИРИЧЕНКО, А.В. КАРПУХИН, А.В. БОРИСОВ, Э. КАЙАЛИ**

***Харьковский национальный университет радиоэлектроники***

# ИССЛЕДОВАНИЕ НЕЛИНЕЙНОЙ ДИНАМИКИ ИЗМЕНЕНИЯ ОКНА ПЕРЕГРУЗКИ TCP ПРОТОКОЛА

*В данной работе проведено имитационное моделирование работы сети при самоподобном трафике и наличии узкого места в системе. Проведен сравнительный анализ разновидностей протоколов семейства TCP (NewReno, Reno, Tahoe и др.) и проанализированы различные алгоритмы работы данных протоколов. Динамика изменения окна перегрузки представлена с помощью итерационного отображения. Показано, что при работе протокола TCP поступление самоподобного трафика вызывает хаотический режим в динамике окна перегрузки.*

***Ключевые слова:*** *протокол TCP, окно перегрузки, время задержки, нелинейное отображение, хаотический режим.*

## Введение

Многочисленные исследования процессов в сети Интернет показали, что статистические характеристики трафика обладают свойством временной масштабной инвариантности (самоподобием). Большинство исследователей считают, что основной причиной самоподобия трафика является протокол TCP. [1, 2]. Чтобы понять механизм воздействия протокола TCP на самоподобие сетевого трафика, необходимо описать в общих чертах алгоритм работы TCP. Протокол TCP является основным протоколом работы в Internet, осуществляет доставку данных в виде байтовых потоков с установлением соединения и применяется в тех случаях, когда требуется гарантированная доставка сообщений. Он включает механизм контроля потока, который гарантирует, что отправитель не переполняет буфер приемника, и механизм контроля заторов, который пробует препятствовать попаданию слишком большого объема данных в сеть (что приводит к потерям пакетов).

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 (SSTHRETH - slow start threshold). Процесс доставки 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 SSTHRETH позволяют согласовать полную загрузку виртуального соединения и текущие возможности канала, минимизируя потери пакетов при перегрузке.

В протокол TCP постоянно вносятся изменения и дополнения, которые пытаются решить проблемы, выявляющиеся по ходу применения протокола, либо улучшить его характеристики для систем узкой специализации.

## 1. Исследование механизма управления перегрузкой для протоколов TCP

Рассмотрим работу основных разновидностей протоколов семейства TCP: Tahoe, Reno, NewReno, SACK, Vegas [3, 4], особое внимание уделяя механизму управления размером окна CWND при перегрузке [5, 6].

Для исследования механизма управления перегрузкой было проведено имитационное моделирование в сетевом симуляторе Opnet и построена модельная сеть, состоящая из нескольких отправителей, получателя и установленного между ними маршрутизатора (рис. 1). Узким местом в моделируемой сети являлись маршрутизатор и выходящий канал. В ходе проведения численного эксперимента изменялись алгоритмы работы протокола TCP, размеры буфера маршрутизатора и получателя, пропускная полоса на выходе маршрутизатора. Трафик в рассматриваемой сети представляет собой самоподобный случайный процесс с задаваемыми пользователем параметрами.

Одним из параметров является показатель Херста, который характеризует долгосрочную зависимость процесса и лежит в диапазоне [0.5, 1].

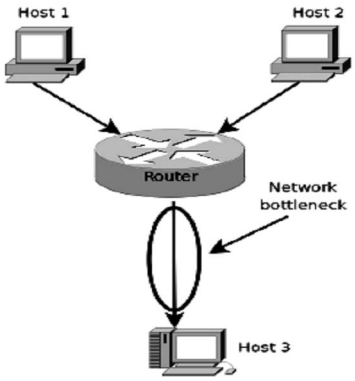



Рис. 1. Модельная сеть

Алгоритм **TCP-Tahoe** является наиболее старым и широко распространенным. Усовершенствования алгоритмов 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. Если потери пакетов не происходит, значение CWND достигает значения Window по умолчанию, задаваемого при конфигурации TCP-драйвера.

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.

Медленный старт предполагает установку окна перегрузки равным 1, а порога медленного старта равным половине значения CWND, при котором произошел таймаут (RTO). Сокращение CWND до единицы происходит потому, что отправитель не имеет никакой информации о состоянии сети. Далее после каждого i-ого подтверждения (ACK) $CWND_{i+1} = CWND_i + 1$. Эта формула работает до тех пор, пока CWND не станет равным ssthresh. После этого рост CWND становится линейным.

**TCP Reno** сохраняет расширения, включенные в Tahoe, но изменяет операцию Быстрой повторной передачи, добавляя Быстрое Восстановление. Новый алгоритм 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.

Вход в Быстрое Восстановление отправителем TCP происходит после получения начального порога двойного ACK. Этот порог обычно устанавливается равным трем. Как только порог двойного ACK получен, отправитель повторно передает один пакет и уменьшает его окно перегрузки на половину. Вместо Медленного старта, как в Tahoe TCP, отправитель Reno использует дополнительный приход двойного ACK, чтобы синхронизировать последующие уходящие пакеты.

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 избежания перегрузки.

**NewReno** значительно отличается от базового алгоритма Reno и дает определенные преимущества по сравнению с каноническим Reno при самых разных сценариях. Однако, при одном сценарии канонический Reno превосходит NewReno - это происходит при изменении порядка следования пакетов.

Алгоритм **TCP SACK** использует поле "Опции" заголовка кадра TCP для дополнительной информации о полученных пакетах получателем. Если произошла потеря, то каждый сегмент тройного ACK, отправляемый станцией-получателем, содержит информацию о кадре вызвавшем посылку данного сегмента. Таким образом, отправитель, получив данный кадр, имеет информацию не только о том, какой кадр был потерян, но также и о том, какие кадры успешно достигли получателя. Благодаря этому избегается ненужная повторная посылка сегментов, успешно буферированных на стороне получателя.

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 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 идеаль-

ной ситуации будет стремиться к требуемому значению. Чем больше размер этого окна, тем выше пропускная способность.

Максимально возможная пропускная способность TCP-соединения зависит от размера окна CWND и, задержки и скорости передачи данных. Нормированная пропускная способность S может быть выражена как

$$S = \begin{cases} 1, & \text{если } CWND \geq R * RTT / 4; \\ 4CWND / R * RTT, & \text{если } CWND < R * RTT / 4, \end{cases}$$

где R - скорость передачи данных (бит/с) по определенному соединению, доступная на стороне отправителя.

Анализируя взаимное влияние рассмотренных параметров, можно получить представление о потенциале протокола TCP в плане повышения производительности. При этом следует принять во внимание множество факторов, которые заметно усложняют описанную выше ситуацию.

Объекты, вовлеченные в соединения, оказываются в определенной мере синхронизованными. Это связано с тем, что когда происходит любое столкновение, 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.

Имитационное моделирование показало, что для протокола TCP-Vegas при одном потерянном сегменте уменьшение CWND не происходит (линия 1 на рис. 2). При одном потерянном сегменте для протокола TCP-Tahoe (линия 2) и протокола TCP Reno (линия 3) уменьшение CWND происходит значительно раньше, соответственно ресурсы канала используются не рационально.

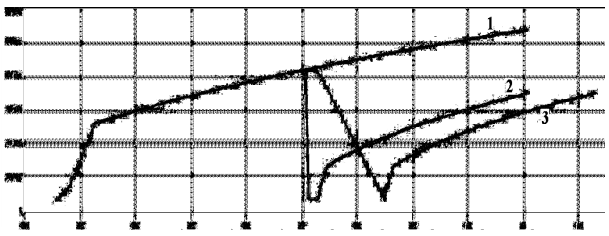



Рис. 2. Изменение CWND:
линия 1 – Tahoe при одной потере сегмента,
линия 2 – Tahoe при одной потере сегмента;
линия 3 – TCP Reno при одной потере сегмента

В случае работы алгоритма NewReno уменьшается количество потерянных данных и загруженность каналов, чем при использовании алгоритма Reno. Это происходит благодаря опции быстрой повторной пересылки и быстрого восстановления. Однако, при включенной опции выборочного подтверждения (SACK) при работе алгоритма Reno количество потерянных данных лишь незначительно больше, чем при NewReno. Из графика, показанного на рис. 3, видно, что протокол TCP-SACK менее интенсивно изменяет размер CWND. Загрузка буфера будет более равномерная и использование ресурсов канала также будет лучше, чем при использовании алгоритма Reno.

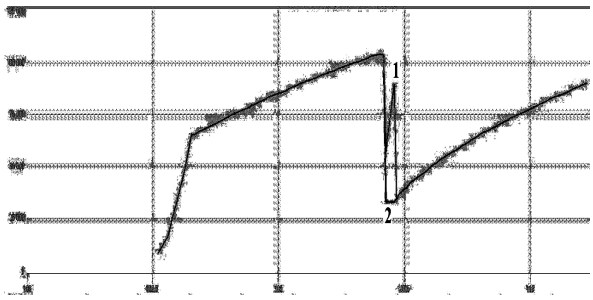



Рис. 3. Изменение CWND:
для протоколов TCP Reno (линия 1)
и TCP SACK (линия 2) при потере одного сегмента.

Использование групповых подтверждений в TCP мотивирует в TCP-Tahoe сокращение ширины окна до единицы после потери, чтобы избежать всплеска пакетов из-за повторной их передачи. Это в свою очередь приводит к экспоненциальному увеличению размера окна при медленном старте, необходимому для сетей с большим произведением полосы на задержку. С другой стороны, это экспоненциальное увеличение вызывает серьезные флуктуации трафика, которые, если размер буфера меньше чем 1/3 от произведения полосы на задержку, вызывает переполнение буфера и второй медленный старт, понижая пропускную способность. TCP-Reno пытается избежать этого явления путем сокращения окна вдвое, при обнаружении потери. В то время как это при идеальных условиях обеспечивает улучшенную пропускную способность, TCP-Reno в его нынешнем виде слишком уязвим для фазовых эффектов и множественной потери пакетов, чтобы стать заменителем для TCP-Tahoe. 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 способно-

сти, если используется таймеры с низким разрешением).

## 2. Представление динамики изменения CWND с помощью итерационного отображения

В случае, когда две TCP сессии одновременно работают по одному и тому же соединению, можно рассматривать временные реализации изменения cwnd1 и cwnd2 как орбиту некоторой нелинейной дискретной системы, описываемой итерационным отображением

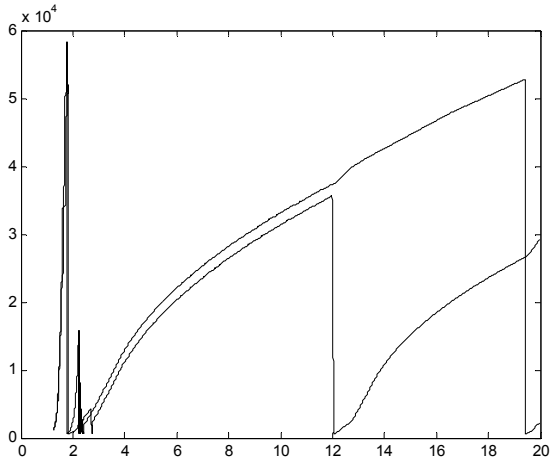
$$\begin{cases} \text{cwnd1}(i+1) = f(\text{cwnd1}(i), \text{cwnd2}(i), C), \\ \text{cwnd2}(i+1) = g(\text{cwnd1}(i), \text{cwnd2}(i), C) \end{cases}, \quad (1)$$

где $\text{cwnd1}_i$ и $\text{cwnd2}_i$ - размеры соответствующих окон в момент времени $t_i$, C – множество управляющих параметров.

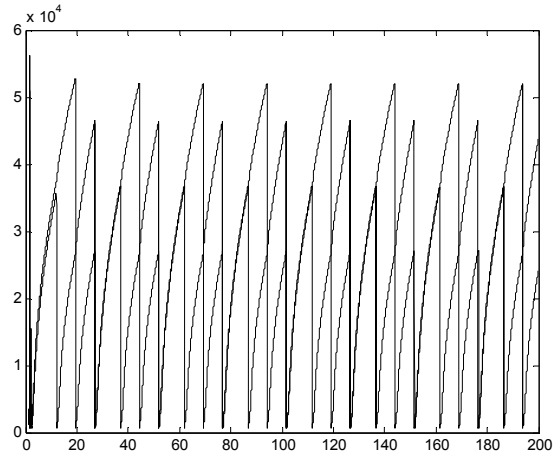
Данное отображение в зависимости от значений управляющих параметров может иметь различную динамику: как периодическую, так и хаотическую. Хаос представляет собой нерегулярное поведение детерминированной нелинейной системы, чувствительно зависящее от начальных условий. Поведение такой системы является предсказуемым лишь на ограниченном интервале времени. В работе [7] в качестве управляющих параметров предложено выбрать следующие: скорость R; задержку RTT; размер буфера B.

В модельном эксперименте при симулировании работы вышеописанной сети было получено простое периодическое поведение для отображения (1) для определенного диапазона параметров соединения. В этом случае обе сессии запускаются одновременно и после короткого переходного процесса устанавливаются в периодические колебания, одно из которых всегда впереди другого, но оба следуют одним и тем же фазам: медленному старту, избежанию перегрузки и возврату. Данный процесс представлен на рис. 4. Здесь показано изменение окна перегрузки двух TCP сессий, одновременно работающих по одному соединению.Этот график показывает, что обе сессии синхронизированы. Они увеличивают свои CWND один за другим, на вершине обнаруживаются потери и обе сессии уменьшают свои окна до одного пакета, далее этот процесс повторяется.

Можно наблюдать эволюцию динамической системы не только как функцию времени, но и как траекторию системы в фазовом пространстве. В данном случае фазовое пространство - двумерное пространство, где фазовые переменные равны значениям соответствующих окон перегрузки в одинаковые моменты времени. Изменению состояния системы во времени отвечает движение фазовой точки вдоль фазовой траекторией.




a – фаза переходного процесса




b – установившийся режим

Рис. 4. Именение окна перегрузки двух TCP сессий, одновременно работающих по одному соединению

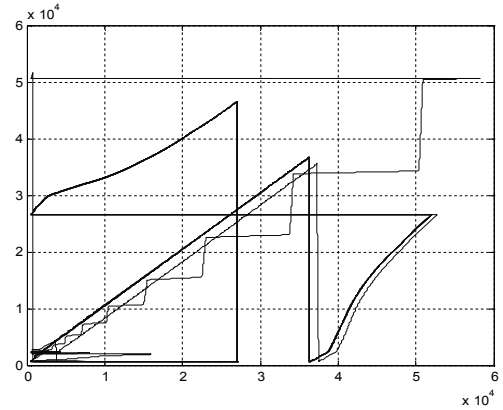
Динамическая система (1) является автоколебательной: основные характеристики колебаний (амплитуда, частота, форма колебаний и т.д.) определяются параметрами системы и в определенных пределах не зависят от выбора исходного начального состояния. Т.е. не имеет значения, каким образом мы возбуждаем систему (например, случайными потерями пакетов или, изменяя начальные условия, например, запускаем вторую сессию TCP на несколько секунд позже) в конце концов, система вновь вернется к тому предельному циклу, являющемуся аттрактором системы. Аттрактор - предельное множество траекторий в фазовом пространстве системы, к которому стремятся все траектории из некоторой окрестности этого множества. Если это предельное множество есть устойчивое состояние равновесия — аттрактор системы будет просто неподвижной точкой, если это устойчивое периодическое движение - аттрактором будет замкнутая кривая, называемая также предельным циклом.

На рис. 5, а представлены траектории на фазовой плоскости для фазовых переменных ( $cwnd1_i$ и $cwnd2_i$ ). Для лучшей визуализации динамики отображения и выявления скрытых тенденций в [7] предложено использовать усредненные по n значениям фазовые переменные W1 и W2 :

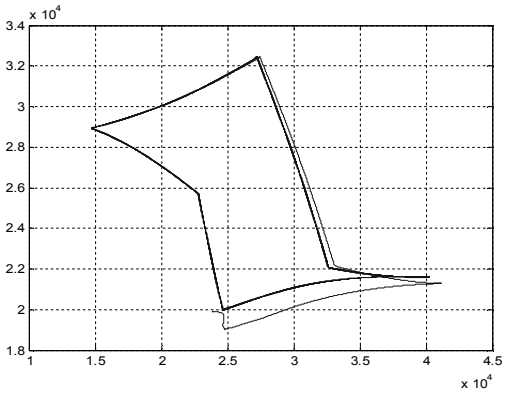
$$W1(i)=\frac{1}{n}\sum_{j=1}^{n} cwnd1(i-j);$$

$$W2(i)=\frac{1}{n}\sum_{j=1}^{n} cwnd2(i-j).$$

Очевидно (рис. 5, б), что фазовые траектории образуют предельный цикл.



а

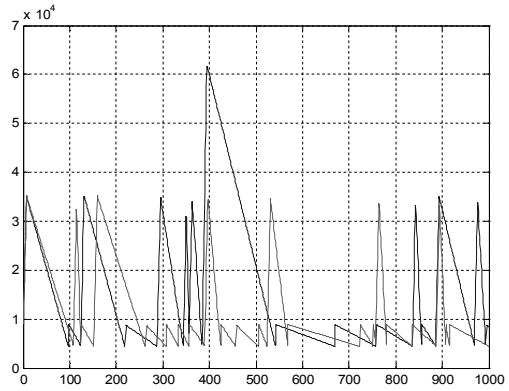


б

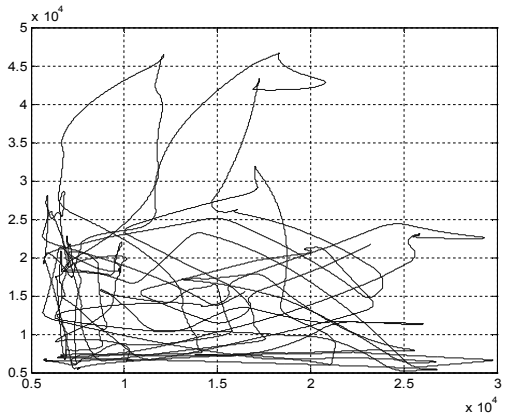
Рис.5. Периодический режим (синхронизированные сессии)

При определенных значениях управляющих параметров нелинейная детерминированная система может демонстрировать хаотическое поведение. Режим детерминированного хаоса также характеризуется аттрактором (так называемым странным аттрактором), но траектория такого аттрактора непериодическая (не замыкается) и режим функционирования неустойчив (малые отклонения от начальных условий нарастают экспоненциально).

При определенном наборе значений параметров, например, количество конкурирующих TCP сессий, скорость, размеры буферов, задержки передачи, поведение системы является нерегулярным. На рис. 6, а показано изменение окна перегрузки двух одновременно работающих TCP сессий в хаотическом режиме. Соответствующий аттрактор усредненных фазовых переменных представлен н на рис 6, б.



а



б

Рис.6. Хаотическая динамика

## Заключение

В настоящее время механизм возникновения самоподобия информационных потоков является слабо изученным. Однако очевидно, что одной из причин является алгоритм изменения окна CWND TCP протокола. В ходе проведенных экспериментов и анализа результатов, выявлено, что при работе TCP поступление самоподобного трафика вызывает хаотический режим в динамике окна перегрузки, что в свою очередь вызывает усиление самоподобных свойств (долгосрочной зависимости и всплесков) выходного трафика.

Дальнейшие исследования предполагают более детальный анализ динамики изменения CWND для различных протоколов TCP и установление зависи-

мости между параметрами самоподобия входного и выходного трафика и параметрами нелинейного отображения, описывающего динамику CWND.

## Литература

*1. Leland W.E. On the self-similarnature of ethernet traffic / W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson // IEEE/ACM Transactions of Networking. – 1994. – Vol. 2(1) – P. 1-15.*

*2. Kirichenko L. Improvement quality of network service under selfsimilar loading / L. Kirichenko, T. Radivilova, O. Karpukhin // Стратегия качества в промышленности и образовании: 4-я междунар. конф.: мат. конф. – Варна, 2008. – P. 612–615.*

*3. Floyd S. Random Early Detection Gateways for Congestion Avoidance / S. Floyd, V. Jacobson // IEEE/ACM Transactions on Networking. – 1993. – Vol. 1(4) – P. 397-413.*

*4. Fall K. Simulation-based comparison of tahoe, reno, and sack tcp / K. Fall, S. Floyd. // Computer Communication Review. – 2002. – Vol. 26 – P. 5-21.*

*5. Stochastic Ordering for Internet Congestion Control and its Applications / H. Cai, D. Eun, S. Ha, I. Rhee, L. Xu // IEEE INFOCOM. – 2007. – P. 45-51.*

*6. Towards a Common TCP Evaluation Suite / L. Andrew, C. Marcondes, S. Floyd, L. Dunn, R. Guillier, W. Gang, L. Eggert, S. Ha, I. Rhee // PFLDnet, Manchester – UK, 2008. – P. 67-74.*

*7. Veres A. The chaotic nature of TCP congestion control / A. Veres, M. Boda // Computer and Communications Societies. Proceedings. IEEE. – Vol. 3 – 2000. – P. 1715-1723.*

*Поступила в редакцию 15.01.2010*

**Рецензент:** д-р техн. наук, проф., проф. кафедры С.Г. Удовенко, Харьковский национальный университет радиоэлектроники, Харьков.

## ДОСЛІДЖЕННЯ НЕЛІНІЙНОЇ ДИНАМІКИ ЗМІНИ ВІКНА ПЕРЕВАНТАЖЕННЯ ТСР ПРОТОКОЛУ

***Т.А. Радівілова, Л.О. Кіріченко, О.В. Карпухін, О.В. Борісов, Е. Кайалі***

В даній роботі проведено імітаційне моделювання роботи мережі при самоподібному трафіку і наявності вузького місця в системі. Проведено порівняльний аналіз різновидів протоколів сімейства TCP (NewReno, Reno, Tahoe і ін.) і проаналізовано різні алгоритми роботи даних протоколів. Динаміка зміни вікна перевантаження представлена за допомогою ітераційного відображення. Показано, що при роботі протоколу TCP надходження самоподібного трафіку викликає хаотичний режим в динаміці вікна перевантаження.

**Ключові слова:** протокол TCP, вікно перевантаження, час затримки, нелінійне відображення, хаотичний режим.

## INVESTIGATION OF NONLINEAR DYNAMICS TCP PROTOCOL'S CONGESTION WINDOW

***T.A. Radivilova, L.O. Kirichenko, O.V. Karpukhin, O.V. Borisov, E. Kayali***

In this work the simulation of network's work is carried out at a selfsimilar traffic and presence of bottleneck in the system. The comparative analysis of varieties of TCP protocols (NewReno, Reno, Tahoe and other) is fulfiled out and the different algorithms of these protocols work are analysed. The dynamics of congestion windows change is presented by an iterative reflection. It is shown that during work of TCP protocol the receipt of selfsimilar traffic is caused by the chaotic mode in the dynamics of congestion window.

**Key words:** protocol of TCP, congestion window, time of delay, nonlinear reflection, chaotic mode.

**Радивилова Тамара Анатольевна** – канд. техн. наук, ассистент кафедры электронных вычислительных машин Харьковского национального университета радиоэлектроники, Харьков, Украина, e-mail: lmd@kture.kharkov.ua.

**Кириченко Людмила Олеговна** – канд. техн. наук, доцент, доцент кафедры прикладной математики Харьковского национального университета радиоэлектроники, Харьков, Украина, e-mail: ludmila.kirichenko@gmail.com.

**Карпухин Александр Владимирович** – канд. техн. наук, доцент, доцент кафедры информатики Харьковского национального университета радиоэлектроники, Харьков, Украина, e-mail: kav@kture.kharkov.ua.

**Борисов Александр Викторович** – студент 5 курса физико-энергетического факультета Харьковского национального университета, Харьков, Украина.

**Эйас Кайали** – аспирант кафедры прикладной математики Харьковского национального университета радиоэлектроники, Харьков, Украина.