УДК 004.7:519.2

О. Карпухін, Л. Кіріченко, Т. Радівілова
Харківський національний університет радіоелектроніки

ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ РОБОТИ TCP- ПРОТОКОЛУ ПРИ САМОПОДІБНОМУ ТРАФІКУ У ВУЗЬКОМУ МІСЦІ МЕРЕЖІ

Здійснено 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 цих протоколів. Показано, що найоптимальнішим є використання протоколу TCP Vegas.

Ключові слова – протоколи сімейства TCP, вузьке місце, самоподібний трафік, вікно перевантаження.

In this paper imitating modelling of a network work is spent at the self-similar traffic and bottleneck presence in system. The comparative analysis of reports versions of family TCP (NewReno, Reno, Tahoe, etc.) is carried out and various algorithms of given reports work are analysed. It is shown that the optimal is use of report TCP Vegas.

Keywords – TCP-protocol, , bottleneck, self-similar traffic, congestion window.

Вступ

Протокол TCP є основним протоколом роботи в Internet. Він здійснює доставку даних у вигляді потоків байтів зі встановленням з'єднання і застосовується у випадках, коли потрібна гарантована доставка повідомлень. Протокол TCP використовує контрольні суми пакетів для перевірки їх цілісності і звільняє прикладні процеси від необхідності таймаутів і повторних передач для забезпечення надійності.

Численні дослідження процесів в мережі Internet показали, що статистичні характеристики трафіка мають властивість часової масштабної інваріантності (самоподоби). Причина такого ефекту – в особливостях розподілу файлів по серверах, їх розмірах, а також в типовій поведінці користувачів. Виявилось, що потоки даних спочатку не проявляють властивості самоподоби, але, пройшовши обробку на вузлових серверах і активних мережевих елементах, починають набувати яскраво виражених ознак самоподібності. Через це можливе швидке перевантаження буферів пристроїв навіть при невеликих коефіцієнтах використання. Якщо не прийняти заходів з обмеження трафіка, що надходить, черги на найбільш навантажених лініях будуть необмежено рости і врешті-решт перевищать розміри буферів у відповідних вузлах. Це призводить до того, що пакети, які знов надходять у вузли, в яких відсутнє вільне місце в буфері, будуть скинуті і повинні будуть передаватися повторно, що призведе до нераціональної витрати ресурсів мережі [1–3].

Регулювання трафіка в протоколі TCP розуміють як існування двох незалежних процесів: контроль доставки, що керується одержувачем за допомогою параметра *Window*, і контроль перевантаження, що керується відправником за допомогою *вікна перевантаження CWND* (congestion window) і процедури *Повільного старту з параметром SSTHRETH* (slow start threshold). Перший процес відстежує заповнення вхідного буфера одержувача, другий реєструє перевантаження каналу, а також пов'язані з цим втрати і знижує рівень трафіку. *Вікно перевантаження CWND* і процедура *Повільного старту* дають змогу погоджувати повне завантаження віртуального з'єднання і поточні можливості каналу, мінімізуючи втрати пакетів при перевантаженні.

У протокол TCP постійно вносяться зміни і доповнення, за допомогою яких намагаються вирішити проблеми, що виявляються під час застосування протоколу, або поліпшити його характеристики для систем вузької спеціалізації.

Метою роботи є порівняльний аналіз різновидів протоколів сімейства TCP і алгоритмів їхньої роботи; виявлення недоліків і переваг кожного алгоритму при різних сценаріях роботи в мережі.

Протоколи сімейства TCP

Розглянемо основні різновиди протоколів сімейства TCP: Tahoe, Reno, NewReno, SACK, Vegas [4, 5].

Реалізація **TCP Tahoe** додала багато нових алгоритмів і удосконалень до наявних реалізацій. Нові алгоритми містять: *Повільний старт*, *Запобігання перевантаженню* і *Швидку повторну передачу*. Удосконалення полягають у модифікації оцінки часу проходження пакетів в каналі зв'язку до адресата і назад для встановлення значення часу очікування повторної передачі. Сенс алгоритму *Запобігання перевантаженню* полягає в утриманні значення *CWND* в області максимально можливих значень. По суті, ця оптимізація здійснюється за допомогою втрати пакетів. Якщо втрати пакетів не відбувається, значення *CWND* досягає значення *Window* за умовчанням TCP-драйвера, що задається при конфігурації.

У роботі особливу увагу приділено алгоритму *Швидкої повторної передачі*, тому що він модифікується в подальших версіях TCP. При роботі алгоритму *Швидкої повторної передачі* після отримання малої кількості подвійних підтверджень для одного пакета TCP (ACK), джерело даних укладає, що пакет було втрачено і повторно передає пакет і всі послані після нього пакети без очікування закінчення таймера повторної передачі. Це веде до зменшення пропускної спроможності та збільшення і без того високого завантаження каналу.

Отримання подвійного ACK не є надійним сигналом втрати пакета. Подвійні ACK виникають і при зміні маршруту обміну. З цієї причини сигналом втрати вважається отримання трьох ACK пакетів підряд.

TCP Reno зберігає розширення, які включені в Tahoe, але змінює операцію *Швидкої повторної передачі*, додаючи *Швидке відновлення*. Новий алгоритм запобігає знаходженню каналу в порожньому стані після *Швидкої повторної передачі* і не перемикається в режим *Повільного старту* для наповнення каналу після єдиної втрати пакета. *Швидке відновлення* припускає, що кожен отриманий подвійний ACK представляє один пакет, що залишив канал. Отже, протягом роботи в режимі *Швидкого відновлення* відправник TCP здатний підрахувати кількість відправлених даних. Замість *Повільного старту*, як в Tahoe TCP, відправник Reno використовує додатковий подвійний ACK, щоб синхронізувати подальші відправлені пакети.

У новому алгоритмі **NewReno** розробники змінили операцію *Швидкого відновлення*, яка значно відрізняється від тієї, що в базовому алгоритмі Reno. Запропонований алгоритм дає певні переваги порівняно з канонічним Reno при самих різних сценаріях роботи. Проте, при одному сценарії канонічний Reno перевершує NewReno – це відбувається при зміні порядку проходження пакетів.

Алгоритм NewReno використовує змінну *Recover* (відновлення), початкове значення якої дорівнює початковому порядковому номеру пакета, і алгоритми *Швидкої повторної пересилки* і *Швидкого відновлення*. У разі, коли доступна опція *Вибіркового підтвердження* (SACK), відправник знає, які пакети слід переслати повторно на фазі *Швидкого відновлення*.

TCP Vegas контролює розмір вікна шляхом моніторингу відправником RTT (часу проходження пакетів по каналу зв'язку до адресата і назад) для пакетів, надісланих раніше. Якщо виявляється збільшення RTT, система дізнається, що мережа наближається до перевантаження і скорочує ширину вікна. Якщо RTT зменшується, відправник визначить, що мережа подолала перевантаження, і збільшить розмір вікна. Отже, розмір вікна в ідеальній ситуації прагнутиме до необхідного значення. Ця модифікація TCP вимагає високої частоти таймера відправника.

Алгоритм **TCP SACK** використовує поле "Опції" заголовка кадру TCP для додаткової інформації про отримані одержувачем пакети. Якщо відбулася втрата, то кожен сегмент потрійного ACK, що відправляється станцією-одержувачем, містить інформацію про кадр, що викликав посилку цього сегмента. Отже, відправник, отримавши даний кадр, має інформацію не тільки про те, який кадр було втрачено, але й про те, які кадри успішно досягли одержувача. Завдяки цьому уникають зайвої повторної посилки сегментів, успішно буферованих на стороні одержувача.

Як і Reno, TCP SACK входить в режим *Швидкого відновлення* при отриманні 3-х ACK, що дублюються. Під час *Швидкого відновлення* відправник підтримує змінну *Pipe*, яка відображає число пакетів, що знаходяться в мережі. Ця змінна збільшується кожного разу, коли новий сегмент був відправлений і зменшується, якщо було отримано чергове підтвердження. Передачу нового пакета в мережу дозволено, якщо значення *Pipe* менше за вікно перевантаження.

Відправник також підтримує структуру даних, яка запам'ятовує підтвердження з опції SACK підтверджень, що прибувають. Якщо відправнику дозволено передачу, він передає наступний пакет із списку пакетів, які вважаються втраченими. Якщо такі пакети відсутні, то посилається новий пакет. Для з'єднань, в яких використовуються вікна великого розміру, використовується алгоритм *Часових міток*. *Часові мітки* допомагають точно вимірювати час звернення RTT для подальшого коригування значення таймера повторної передачі.

Результати імітаційного моделювання

Для імітаційного моделювання в мережевому симуляторі Opnet було побудовано модельну мережу, що складається з декількох відправників, одержувача і встановленого між ними маршрутизатора (рис. 1). Вузьким місцем у мережі є маршрутизатор і вихідний канал. Під час числового експерименту змінювали алгоритми роботи протоколу TCP, розміри буфера маршрутизатора і одержувача, пропускну смугу на виході маршрутизатора. Трафік у цій мережі є самоподібним випадковим процесом з параметрами, що задаються користувачем. Одним з параметрів є показник Херста, який характеризує довгострокову залежність процесу і лежить в діапазоні [0.5, 1].

Характеристиками, за якими проводили порівняльний аналіз роботи мережі, були: кількість втрачених даних, завантаження буфера маршрутизатора, утилізація каналу, продуктивність роботи мережі.

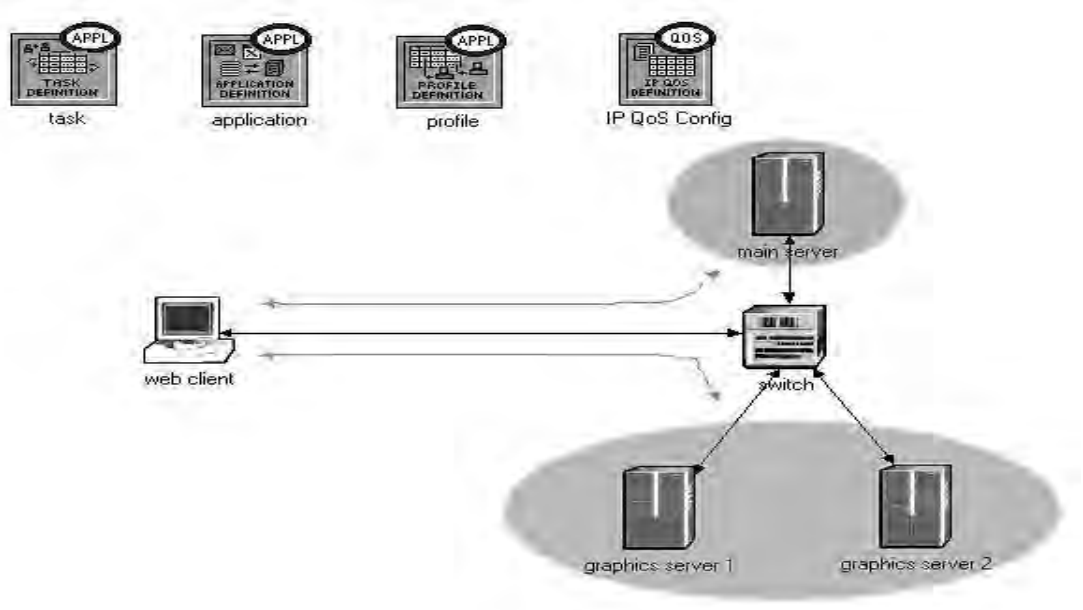

Рис. 1. Модельна мережа

Імітаційне моделювання показало, що для всіх алгоритмів протоколу TCP при роботі алгоритмів *Повільного старту* і *Уникнення перевантаження* розмір вікна *CWND* з часом росте лінійно (лінія 1 на рис. 2).

При одному втраченому сегменті для протоколу TCP Tahoe (лінія 2) і протоколу TCP Reno (лінія 3) *CWND* зменшується значно раніше, відповідно завантаження каналу і буфера є рівномірнішим і ресурси системи використовуються оптимальніше.

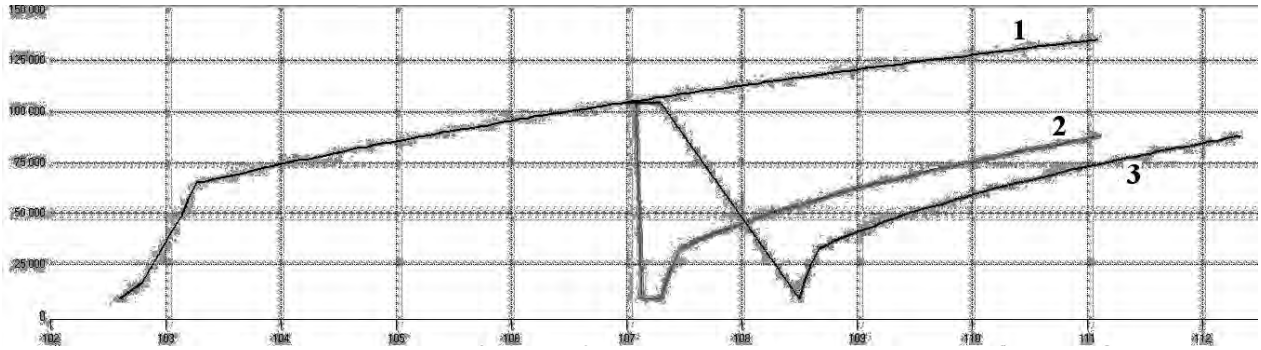

Рис. 2. Змінення CWND: лінія 1 – Повільний старт і Уникнення перевантаження; лінія 2 – Tahoe за однієї втрати сегмента; лінія 3 – Reno за однієї втрати сегмента

На рис.3 показано кількість відісланих і прийнятих сегментів і відповідну зміну розміру вікна. Приведена залежність є типовою для всіх протоколів сімейства TCP. З графіка видно, що кількість відісланих сегментів росте лінійно відповідно до збільшення розміру вікна. Проте, кількість втрачених даних у момент зменшення розміру вікна значно більше допустимих втрат, які визначено Qos (якість обслуговування мережі).

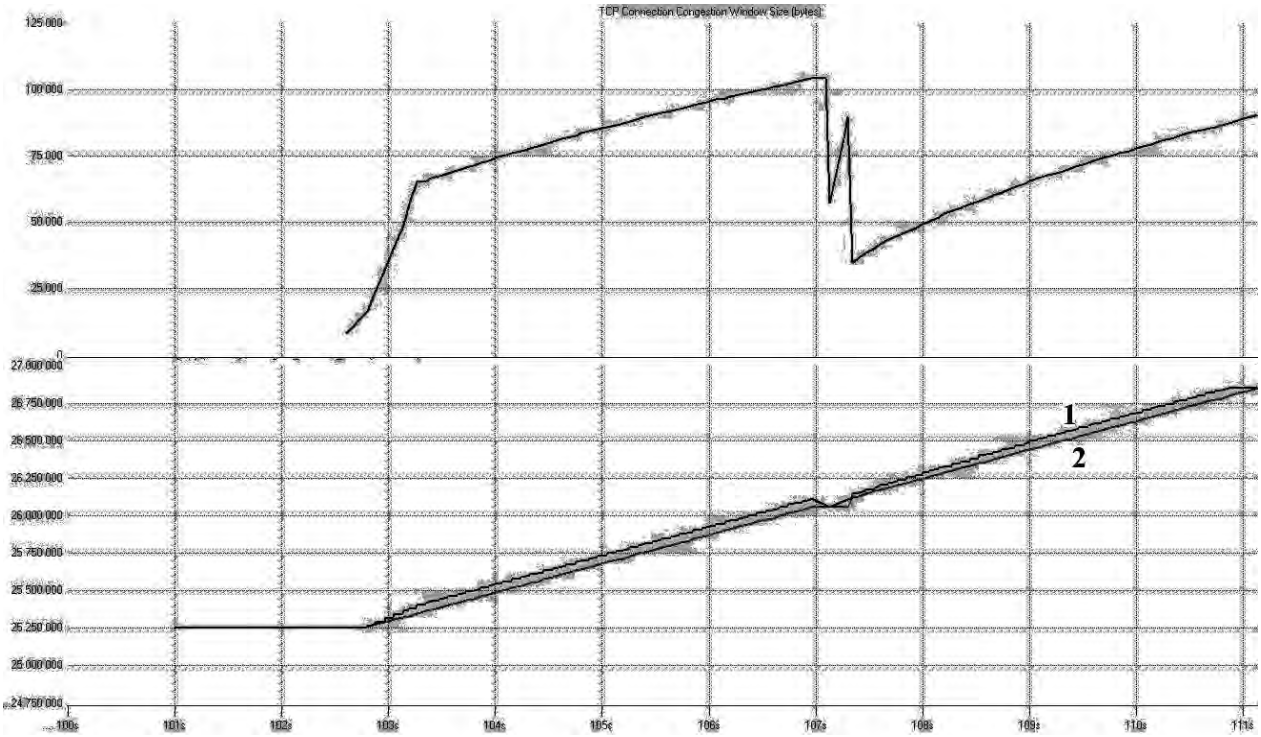

Рис.3. Змінення CWND (вгорі) і кількості відісланих сегментів (лінія 1) і прийнятих (лінія 2) з часом (внизу)

Результати дослідження роботи TCP Reno показують, що кожне з'єднання зазвичай втрачає близько двох пакетів у кожному епізоді перевантаження. Втрати виникають, коли буфер повний та одне з'єднання збільшує розмір вікна на одиницю. Коли дані з цього нового пакета приходять у буфер, вони зазвичай викликають втрату даних, що належать двом пакетам (кінець пакета, що прийшов з іншого з'єднання, і початок наступного). Отже, в середньому слід чекати втрати трьох пакетів на один епізод перевантаження.

Імітаційне моделювання роботи мережі при використанні алгоритму Reno з опцією *Швидке відновлення* показало, що цей алгоритм є оптимальним лише у випадках одиничних втрат пакетів, тобто коли відправник Reno передає не більш ніж один пакет за час проходження одного пакета по

каналу зв'язку до адресата і назад. Коли втрачено один пакет, алгоритм Reno значно кращий порівняно з Tahoe TCP, проте Reno значно погіршує характеристики мережі, якщо втрачено декілька пакетів у межах одного вікна *Window*. При роботі алгоритму Tahoe, який не використовує опцію *Швидкого відновлення*, кількість втрачених даних приблизно однакова з випадком роботи алгоритму NewReno, проте завантаженість каналу значно менша.

Проблеми оптимальності використання ресурсів мережі для алгоритмів *Швидкого відновлення* і *Швидкої повторної пересилки* в алгоритмі TCP Reno, які пов'язані з багатьма швидкими повторними пересилками, відносно менше порівняно з аналогічними проблемами, що виникають у разі роботи алгоритму Tahoe TCP, який не використовує *Швидкого відновлення*. Проте, якщо не використовувати додаткових механізмів, які позв'язані із застосуванням змінної *Recover*, при роботі Reno TCP можуть відбутися зайві повторні пересилки.

У разі роботи алгоритму NewReno зменшується завантаженість каналів; кількість втрачених даних менша, ніж при використанні алгоритму Reno. Це відбувається завдяки опції *Швидкої повторної пересилки* і *Швидкого відновлення*. Проте з опцією *Вибіркового підтвердження* (SACK) при роботі алгоритму Reno кількість втрачених даних лише трохи більша, ніж при NewReno. З графіка, показаного на рис.4, видно, що протокол TCP SACK менш інтенсивно змінює розмір *CWND*. Завантаження буфера буде більш рівномірним, і використання ресурсів каналу також буде кращим, ніж при використанні алгоритму Reno.

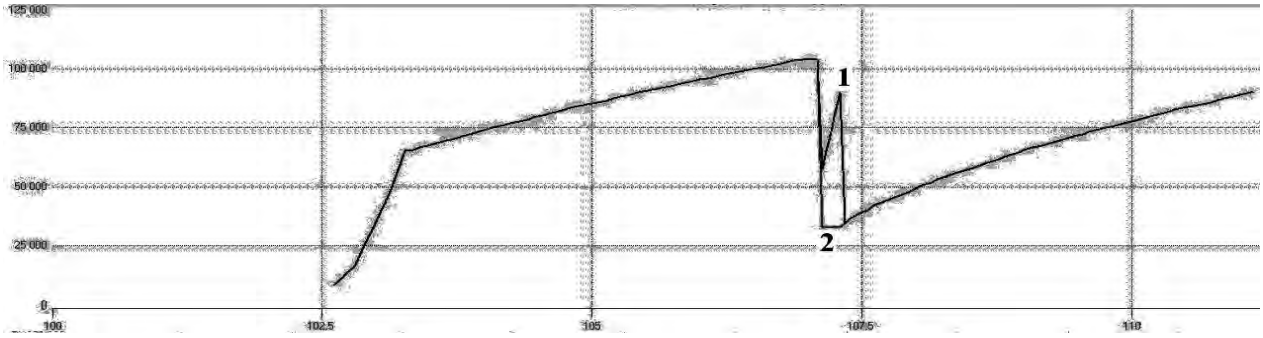

Рис.4. Змінення CWND для протоколів Reno (лінія 1) і SACK (лінія 2) при втраті одного сегмента

Результати дослідження роботи протоколів з *Часовою міткою* показали, що розмір вікна для алгоритму TCP з *Часовою міткою* зростає лінійно із самого початку, а без *Часової мітки* спочатку встановлюється номінальне вікно, а тільки потім це вікно лінійно зростає (рис. 5). Проте розмір вікна для алгоритму TCP з *Часовою міткою* швидше приходить до нормального стану.

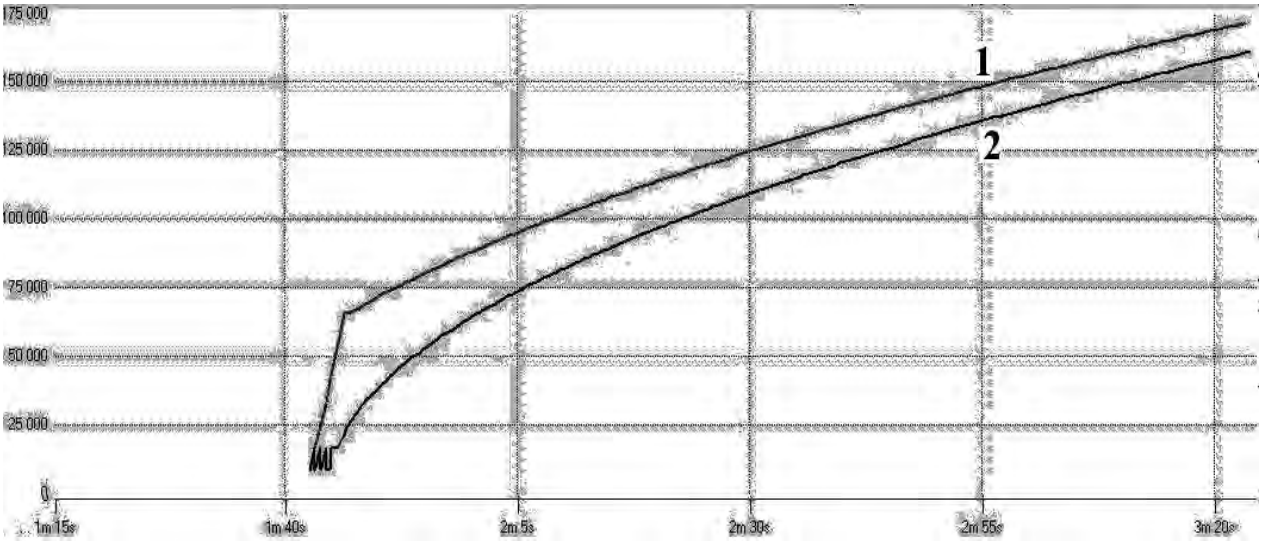

Рис.5. Змінення CWND для TCP з Часовою міткою (лінія 1) і без неї (лінія 2)

Імітаційне моделювання показало, що час розповсюдження пакета туди і назад (рис.6) в TCP з *Часовою міткою* є значно меншим, ніж для TCP без *Часової мітки.*

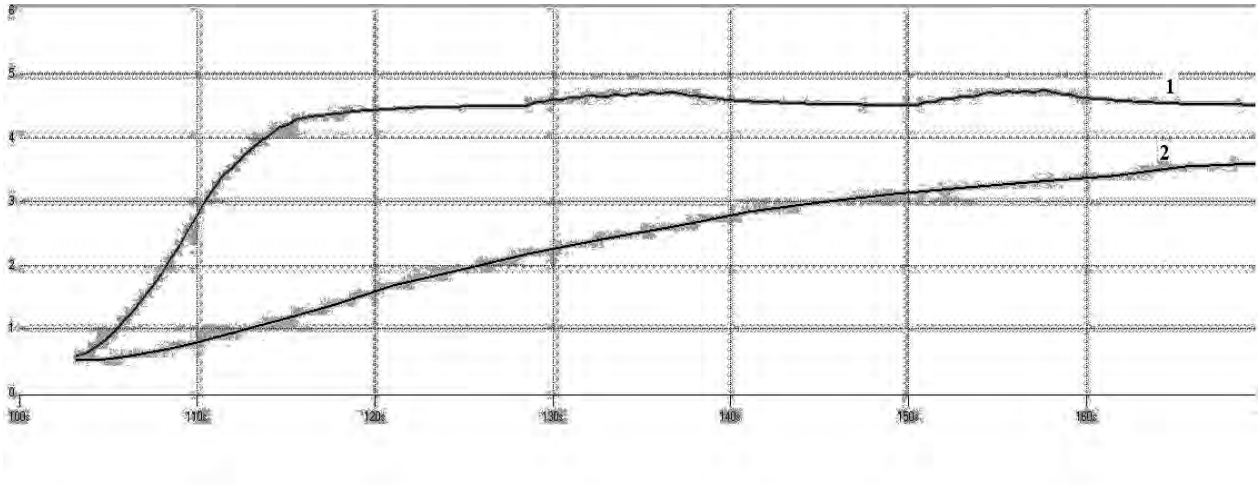

Рис.6. Змінення часу проходження пакетів по каналу зв'язку для TCP з Часовою міткою (лінія 1) і без неї (лінія 2)

Під час роботи здійснено порівняльний аналіз утворення черг у буфері для різних алгоритмів протоколу TCP із збільшенням кількості потоків вхідних даних (табл. 1). Це має особливе значення, наприклад, для мереж з декількома каналами VPN і багатопортовими маршрутизаторами.

Із збільшенням довжини черги зростання часу RTT є неминучим. При досягненні нижнього порогу черги починаються втрати пакетів. Коли ж буде перевищено верхній поріг, починають відкидатися вхідні пакети з низьким пріоритетом. Якби відкидання пакетів не починалося до переповнювання буфера, можна було б чекати лінійного зростання RTT. Досягши граничного значення черги, кількість втрачених пакетів зростала б практично стрибкоподібно.

Таблиця 1

Значення средньої довжини черги (мб) при различної кількості потоків

	1	2	3	4	5	6	7
Tahoe	9900	11700	11900	15000	15300	15600	16000
Reno	9800	11500	11500	14700	15800	16000	16100
NewReno	9100	9500	9700	10600	12000	13800	15200
Sack	5300	5600	6900	7200	8600	8800	9100
Vegas	5200	5600	7000	7000	7700	8500	8900

Одним з найважливіших показників якості обслуговування мережі є коефіцієнт використання каналу. У табл. 2 наведено значення цього показника при еволюції розміру вікна з часом.

Таблиця 2

Коефіцієнт використання каналу при еволюції размеру вікна з часом (мс)

	100	200	300	400	500	600	700
Tahoe	1	0,8	0,72	0,72	0,65	0,6	0,56
Reno	1	0,82	0,75	0,78	0,62	0,54	0,4
NewReno	1	0,87	0,79	0,85	0,74	0,68	0,6
Sack	1	0,93	0,88	0,99	0,82	0,87	0,99
Vegas	1	0,94	0,88	1	0,85	0,9	0,98

Висновки

Отже, у результаті проведених експериментів і аналізу результатів, виявлено, що при роботі TCP Vegas якість обслуговування мережі значно краща і кількість втрачених пакетів значно менша, ніж при роботі інших протоколів TCP. Протокол TCP Vegas: є стійкішим при втратах пакетів і може

набагато швидше виявити і повторно передати втрачений пакет, ніж Tahoe; не повинен чекати 3-х подвійних АСК і тому він може швидше повторно передати втрачений пакет, ніж Reno і NewReno; завдяки зміненим алгоритмам *Запобігання перевантаженню* і *Повільного старту* здійснює менше повторних передач, що призводить до ефективнішого використання ресурсів мережі, ніж NewReno і Tahoe; при оцінюванні перевантаження вимірює пропускну здатність і змінює її замість втрати пакета, що дає краще використання смуги пропускання і менше перевантажень, чим в Tahoe і SACK; вирівнює свою норму посилки пакетів до одержувача в оптимальній смузі пропускання, тобто спричиняє стабільність, на відміну від SACK.

1. Leland W.E., Taqqu M.S., Willinger W., Wilson D.V. On the self-similarnature of ethernet traffic, IEEE/ACM Transactions of Networking, vol. 2(1). – pp. 1–15, 1994. 2. Столлингс 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 // Мат-лы 4-й междунар. конф. – Варна, 2008. – С. 612–615. 4. Floyd S., Jacobson V. Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, August 1993. – Vol. 1(4). – pp. 397–413, 5. Fall K., Floyd S. Simulation-based comparison of tahoe, reno, and sack tcp, Computer Communication Review, 2002. – Vol. 26. – pp. 5–21.

УДК 621.395

В. Хома
Національний університет “Львівська політехніка”,
кафедра захисту інформації,
Політехніка Опольська, інститут автоматики і інформатики

ОПИС І ХАРАКТЕРИСТИКА КРИПТОЛОГІЧНОГО ПАКЕТА CRYPTOOL ЯК НАВЧАЛЬНОГО ІНСТРУМЕНТА

Репрезентовано вільно поширюваний криптологічний пакет CrypTool. Описано функціональні можливості цього програмного пакета, набір алгоритмів шифрування та цифрового підпису, а також інструментів їхнього криптоаналізу. Наведено оцінку застосування криптологічного пакета CrypTool у навчльному процесі.

Ключові слова: крипто логічний, програмний, крипто аналіз, цифровий підпис.

The free distribute cryptology package of CrypTool is presented in the article. Functional possibilities of this programm package, set of encryption and digital signature algorithms and also instruments of their cryptoanalysis are described. Application of cryptology package of CrypTool in an educational process is estimated.

Keywords:

1. Вступ. Історія створення та розвитку криптопакета

CrypTool – це вільно поширюваний криптологічний програмний пакет. У ньому заімплементовано багато криптографічних алгоритмів та протоколів як класичної, так і асиметричної криптографії, а також засоби криптоаналізу [1–3]. Пакет CrypTool є надзвичайно потрібним і корисним інструментом для вивчення й аналізу криптографічних перетворень, а також висвітлення реальних загроз під час практичного застосування криптографічних засобів у сучасних інформаційних системах.

Проект CrypTool започаткував професор Бернхард Есслінгер (Esslinger) у 1998 році, тоді ж розвивається на засадах відкритого програмного забезпечення такими німецькими університетами,