УДК 004.5

ЕКСПЕРИМЕНТАЛЬНА РЕАЛІЗАЦІЯ TCP-ПРОТОКОЛУ НА РІВНІ ДОСТУПУ ДО КАНАЛУ ЗВ'ЯЗКУ

Гученко М.І., д.т.н., доц., Позняк Є.В., асп., Іванова М.М., асис.
Кременчуцький державний політехнічний університет
імені Михайла Остроградського
м. Кременчук, вул. Першотравнева 20
E-mail: mig@ktp.polytech.poltava.ua

Представлена экспериментальная реализация TCP-подобного протокола, разработанная при помощи архитектуры промежуточных NDIS-драйверов и работает на уровне доступа к каналу. Разработаны програмные средства, которые предоставляют возможность быстрого создания TCP-подобных протоколов с использованием технологии TCP Offloading и способны работать на базе всех современных версий ОС Windows.
Ключові слова: NDIS-драйверы, TCP-подобные протоколы, TCP Offloading, канальный уровень

This work introduced experimental approach of TCP-friendly protocol developed by means of intermediate NDIS-drivers architecture and works at the data-link layer. Researched programming interfaces get possibility of TCP-friendly protocols rapid development on a base of TCP Offloading technology and can work with all latest versions of OS Windows.
Key words: NDIS-drivers, TCP-friendly protocols, TCP Offloading, data-link layer

Вступ. Головними цілями керування трафіком у сучасних мережах є, з одного боку, ефективне планування потоків трафіку згідно вимог до якості їх обслуговування (QoS) та наявних мережних ресурсів, а з іншого – запобігання перевантажень та втрат пакетів, що порушують QoS. Указані задачі керування покладаються, в першу чергу, на транспортні протоколи та проміжне обладнання.

У [1] показано, що для TCP-подібних протоколів необхідна:

- розробка нових механізмів регулювання потоків при вибухових надходженнях трафіка;
- розробка механізмів швидкого й точного оцінювання доступної смуги пропускання після тимчасової втрати з'єднань.

Аналіз попередніх досліджень. Проміжне обладнання вирішує проблему вибухових надходжень трафіку за допомогою схем регулювання втрат пакетів, наприклад, схеми Additive Increase Multiplicative Decrease (AIMD) [2]. Керування трафіком на основі AIMD може застосовуватись не тількі в інтелектуальних комутаторах, а й в хостах мережі (за допомогою драйверів канального рівня). Задача точного оцінювання доступної смуги пропускання вирішується, наприклад, з використанням схеми Explicit Rate Estimation (ERE) (як у TCP Westwood [3]), або за допомогою додаткових протоколів (що реалізовано в ATCP [1], який також потребує створення схеми керування на нижніх рівнях стеку TCP/IP). У сучасних мережних технологіях також існує необхідність у розвантаженні стеку протоколів TCP/IP, ця задача дістала назву TCP Offloading. Ефективнішого використання мережних ресурсів можна досягти за рахунок оптимізації в стеці TCP/IP процедур доступу різних видів трафіку до каналу зв'язку та застосування алгоритмів керування TCP-потоком, призначених для конкретного типу середовища передачі даних. У рамках специфікації інтерфейсів мережних драйверів (NDIS), розробленої 3Com и Microsoft ще в 1989 році, в останній версії ОС Windows (Vista) з'явилась технологія TCP Chimney (див. рис.1), яка, співпрацюючи зі стандартним стеком TCP/IP, надає механізм реалізації протоколів дружніх до TCP-протоколу на рівні доступу до каналу зв'язку [4].

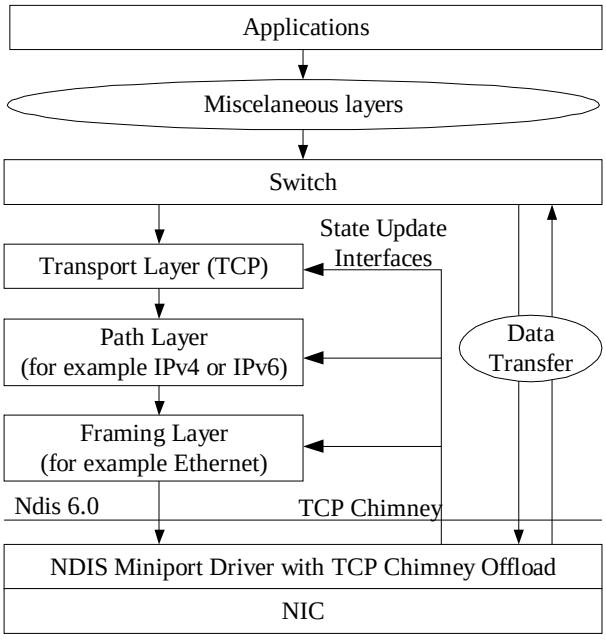


Рисунок 1 – Блок-схема інтерфейсів "TCP Chimney"

Складність розробки таких специфічних протоколів співставима із складністю розробки звичайних брандмауерів. Технологія TCP Chimney на сьогодніш-

ній день задіяна в дуже обмеженій мірі. У даній роботі зроблена спроба реалізувати протокол TCP на базі проміжного NDIS-драйверу, який здатен працювати в ОС Windows 98 та вище, і має можливості подібні до TCP Chimney. У роботі надано експериментальні результати тестування розроблених програмних засобів, що демонструють ефективність застосування TCP Offload у локальній мережі та можливість їх застосування для мультимедійного Інтернет-трафіку з використанням ефективних схем керування TCP-потоком.

Мета роботи. Розробка експериментальної реалізації протоколу TCP на рівні доступу до каналу зв'язку, який надасть можливість більш ефективного використання мережних ресурсів пріоритетним трафіком та співпрацюватиме зі звичайним стеком TCP/IP в ОС Windows.

Матеріал і результати дослідження. Для досягнення поставленої мети необхідно вирішити наступні задачі:

- розробити функціональну модель протоколу TCP;
- розробити проміжний драйвер канального рівня;
- реалізувати з його допомогою TCP-сесії;
- реалізувати запропоновану в [1] схему керування TCP-трафіком;
- перевірити якість роботи даної реалізації TCP-протоколу в локальній мережі та для Інтернет-трафіку.

Функціональна модель протоколу TCP

Для розробки моделі TCP-протоколу були використані матеріали сайту [4] та RFC 4614. Функціональну модель будь-якого протоколу можна представити діаграмою його станів.

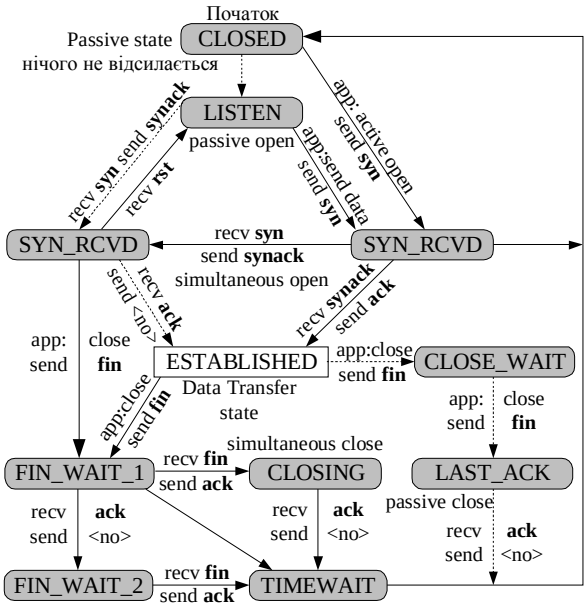


Рисунок 2 – Діаграма станів протоколу TCP [4]

→ - нормальний перехід для клієнта;

→ - нормальний перехід для сервера;

app: - зміна стану при видачі команди сервером;

recv - зміна стану при отриманні пакету;

send - відсилка пакету на даному переході.

Відповідно до RFC 793 усього в протоколі TCP налічується 11 станів, що встановлюються в порядку, представленому на рис. 2 за допомогою відповідних керуючих пакетів, головними з яких є:

SYN - для встановлення з'єднання і його початкових налаштувань (параметрів безпеки, пріоритету, розміру сегментів, нумерації пакетів, встановлення таймерів та вікон передачі).

FIN – сигнал завершення прийому/передачі.

RST – для негайного розриву з'єднання (наприклад при порушенні протоколу обміну даних).

ACK – для підтвердження прийому відісланих пакетів та розрахунку показників керування TCP-трафіком.

Розробка проміжного драйверу

У роботі для реалізації TCP-сесій в обхід стеку протоколів ОС Windows був розроблений проміжний NDIS драйвер, що працює на канальному рівні та є проміжною ланкою між протокольними драйверами та драйверами мережних адаптерів. Сучасна архітектура NDIS-драйверів майже цілком будується на базі Windows Driver Model (WDM), що надало змогу реалізувати драйвер, здатний працювати в усіх ОС, починаючи з Windows 98. Можливості даного драйверу подібні до можливостей комерційного пакету WinpkFilter, що розміщено на сайті [8]. На базі тих функцій, що підтримує й WinpkFilter (запити до мережних адаптерів та протоколів, сбір статистики, фільтрація трафіку, відправка пакетів, обробка подій), в проміжному NDIS-драйвері реалізована функціональність протоколу TCP. Робота з цим драйвером з режиму користувача та встановлення TCP-з'єднань може виконуватись за допомогою окремих програмних додатків. Можлива реалізація роботи і через сокети, що потребує розробки відповідного BSP (Base Sockets Provider), за рахунок чого є можливість використання можливостей TCP Chimney не тільки для Windows Vista.

Реалізація TCP-сесій

При відсилці або прийомі **syn-**пакету драйвер TCP створює блок керування передачею та окрему процедуру очікування подій, а також буфери прийому й відсилки пакетів. Слід сказати, що багато з можливостей NDIS є недостатньо документованими, відсилка пакету драйвером потребує виконання наступних процедур:

```
//виділяємо структуру ndis-пакету з окремого пулу
NdisAllocatePacket(…);
//розміщуємо kernel-mode буфер з пакетом даних,
//що надсилається, в структурі MDL
NdisAllocateBuffer(…);
//приєднуємо MDL до ndis-пакету
NdisChainBufferAtBack(…);
//надсилаємо ndis-пакет
NdisSendPackets(…);
// далі можлива обробка помилок
…
// для використаного пулу ndis-пакетів
// звільнюємо структури даних пакету
// в процедурі ProtocolSendPacketComplete
```

NdisUnchainBufferAtBack(…);
NdisFreeBuffer(…);
NdisReinitializePacket(…);

Драйвер очікує надходження пакетів-підтвердження в процедурі очікування TCP-подій, яка фільтрує вхідні пакети на основі даних блоку керування передачею.

Відкриття клієнтського TCP-з'єднання включає наступні кроки: відсилка **syn(n,0)**, прийом **syn-ack(n+1,k)**, відсилка **ack(n+1,k+1)**. Відкриття серверного з'єднання, відповідно: отримання **syn(n,0)**, відсилка **syn-ack(n+1,k)** та прийом **ack(n+1,k+1)**, де **n, k –** початкові номери (ISN) клієнтського та серверного сегментів. У подальшому **n** і **k** зростають на величину надісланих/підтверджених даних. Закриття з'єднання виконувалось шляхом обміну **fin**-пакетами або відсилкою пакету **rst** (рис. 2).

Керування трафіком TCP

Після встановлення з'єднання при отриманні ACK-пакетів розраховуються наступні показники керування потоком даних:

WND – вікно характеризує розмір буфера приймача трафіка або кількість даних, що можна надіслати без підтвердження;

RTT – колообіжні затримки означають час, що пройшов після відсилки пакету підтвердженого даним **ack**;

RTO – таймаут повторних передач пакетів, що розраховується за алгоритмом Якобсона [2]:

$$RTT = aRTT + (1-a)M \quad (1)$$

$$D = bD + (1-b)|RTT - M| \quad (2)$$

$$RTO = RTT + 4*D \quad (3)$$

де α=0,875 та β=0,75.

CWND – вікно перевантажень, що контролює доступну смугу пропускання TCP-з'єднання. Існує велика кількість оригінальних алгоритмів керування вікном перевантажень, серед яких слід відмітити: Vegas, NewReno (що застосовується в Windows) та Westwood, що для бездротових швидкісних каналів може дати ліпший результат ніж NewReno [3].

Для локальної мережі обрано наступний алгоритм керування вікном перевантажень:

$$cwnd_{k+1} = \begin{cases} cwnd_k + 1 & if \quad cwnd_k < ssthresh \\ cwnd_k + 1/cwnd_k & otherwise \end{cases} \quad (4)$$

де *ssthresh*=3. Для Інтернет-трафіку обрано алгоритм розрахунку *cwnd* подібний до TCP Vegas [7], що встановлює вікно перевантажень на базі RTT-затримок і підходить для протяжних каналів зв'язку.

Тестування реалізації TCP-протоколу

Для тестування роботи даного протоколу реалізовано декілька прикладних програм. Для перевірки роботи TCP-сесій була використана мережа Fast Ethernet що включає клієнтську машину (джерело трафіка) та сервер, що має підключення до мережі Інтернет із максимальною швидкістю передачі – 512 кбіт/сек. На клієнтській машині використовувались наступні програми на базі NDIS-драйверу: програма відсилки даних по протоколу TCP у локальній мережі, програма для роботи з GET та POST запитами протоколу HTTP [4]. На серверній машині використовувався звичайний TCP-сервер та сніфер пакетів реалізований на базі Winpcap. Загальна структура експериментальної мережі представлена на рис. 3.

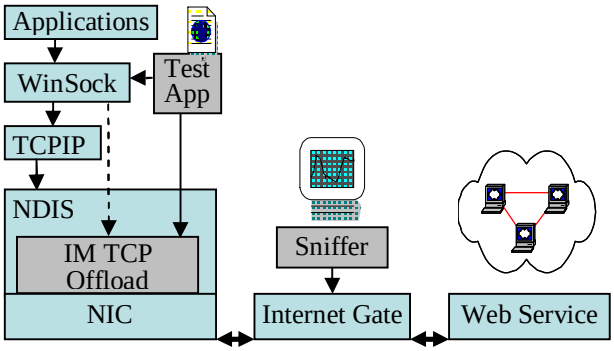
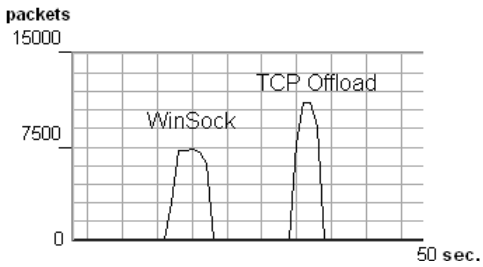


Рисунок 3 – Структура експериментальної мережі

Проведено тестування роботи протоколу в локальній мережі шляхом обміну даними з серверним TCP додатком. Виконувалась відсилка 40000 пакетів встановленого розміру. На рис.4а, б представлені графіки швидкостей передачі пакетів мінімального та максимального розміру (64 і 1514 байт) для звичайного сокетного TCP-з'єднання та з'єднання на базі TCP Offload.

а)



б)

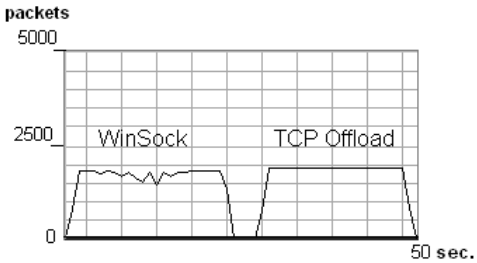


Рисунок 4 – Передача пакетів для звичайного (WinSock) та розробленого (TCP Offload) з'єднань: а) 64 байт; б) 1514 байт

Результати тестування передачі пакетів стандартним та розробленим протоколами TCP у локальній мережі наведені в табл. 1.

Таблиця 1 – Показники передачі TCP-пакетів в локальній мережі

Index	packet length = 64		packet length = 1514	
	WinSock	TCP Offload	WinSock	TCP Offload
Packets/sec	6780	8890	1900	1950
Throughput (%)	3,57	4,95	21,14	22,11
Mean RTT (ms)	104,00	76,70	173,60	171,60

Із рис.4б видно, що для з'єднання з використанням NDIS-драйверу отримано більш стабільну швидкість передачі пакетів ніж для звичайного TCP-з'єднання. Використання TCP Offload у локальній мережі дозволило майже в півтора рази збільшити швидкість передачі малих пакетів (рис. 4а), та дещо покращити якість передачі для пакетів максимального розміру (рис. 4б), що досягнуто, в першу чергу, за рахунок зменшення колообіжних затримок трафіку.

Для тестування роботи розробленого TCP-протоколу в мережі Інтернет було створено програмне забезпечення, що дозволяє надсилати мультимедійні дані за протоколом HTTP за допомогою методу POST. Метод POST є широковживаним, оскільки за його допомогою реалізується, наприклад, обмін XML-даними Web-службами Інтернет. Була реалізована відправка графічних зображень на сервери: mail.ru та vkontakte.ru, приблизно за тим ж самим алгоритмом, за яким працює Інтернет-браузер, але оскільки метод POST є потенційно небезпечним для Web-серверів, виникли проблеми із встановленням чітких правил безпеки при відсилці даних. Планується проведення експерименту з передачею мультимедійного WEB-контенту з використанням алгоритму керування потоком адаптивної версії TCP-Vegas [7].

Висновки. Отримано експериментальне підтвердження ефективності використання розробленого TCP-протоколу в локальній мережі та його якісну роботу при передачі мультимедійного HTTP-трафіку в мережі Інтернет. Представлена в даній роботі реалізація TCP Offloading дозволяє поглибити кооперацію між транспортним та канальним рівнями моделі OSI, надаючи більш спрощений та ефективний доступ до каналу зв'язку. В ході даної роботи встановлено, що технологія TCP Offloading може застосовуватись в першу чергу для трафіку, який потребує мінімальних RTT-затримок, точного встановлення міжкадрових інтервалів й швидкості передачі, та узгодженної передачі поряд з іншим трафіком TCP/IP. Розроблено програмні засоби, що надають можливість швидкого створення в ОС Wndows TCP-подібних протоколів не тільки в режимі ядра, а й в звичайному користувальницькому режимі. Крім того, на базі розробленого програмного забезпечення можлива реалізація засобів підтримки стандартного стеку TCP/IP з використанням схеми AIMD [5] або шляхом створення допоміжних протоколів подібних до протоколу ATCP [1].

ЛІТЕРАТУРА

1. Ka-Cheong Leung, Victor O. K. Li, "Transmission Control Protocol (TCP) in wireless Networks: Issues, Approaches, and Challenges", IEEE Communications Surveys & Tutorials, Vol. 8, No. 4, 4th Quarter 2006. http://www.comsoc.org/pubs/surveys.

2. Ki Baek Kim, "Design of Feedback Controls Supporting TCP based on Modern Control Theory", INRIA Rocquencourt, Report No. 5014, Nov. 2003 – 37 p. http://hal.inria.fr/inria-00071570/en.

3. R. Wang *et al.*, "TCP With Sender-Side Intelligence to Handle Dynamic, Large, Leaky Pipes," *IEEE JSAC*, vol. 23, no. 2, Feb.2005. http://ieeexplore.ieee.org

4. TCP Chimney Offload and RSS Documentation. [Online]. http://support.microsoft.com/?kbid=912222.

3. M. Haeri, A.H. Mohsenian Rad, "TCP Retransmission Timer Adjustment Mechanism Using System Identification", Proceeding of the 2004 American Control Conference, Boston. http://ieeexplore.ieee.org.

4. Телекоммуникационные технологии. [Online]. http://book.itep.ru

7. K.N. Srijith, Lillykutty Jacob1, A.L. Ananda, "TCP Vegas-A: Improving the Performance of TCP Vegas" / Computer Communications, No.28 (2005), pp. 429–440.

8. NT Kernel Resources Web Pages. [Online]. http://www.ntkernel.com.

Стаття надійшла 25.01.2008.
Рекомендовано до друку доц.
Сисюком Г.Ю.