Повышение производительности беспроводной сети

Е. А. Сторожок

Дальневосточный федеральный университет, Владивосток

Аннотация: Предлагается использование промежуточного драйвера в стеке сетевых драйверов для коррекции метода доступа CSMA/CA. Каждый узел сети осуществляет передачу своих пакетов в отведённые интервалы времени. Это делает доступ к разделяемому каналу связи детерминированным, что в условиях интенсивного сетевого трафика способствует повышению производительности сети WI-FI.
Ключевые слова: промежуточный драйвер, метод доступа, беспроводная сеть, коллизия, среда передачи данных.

С каждым годом увеличивается количество пользователей глобальной сети Internet. В связи с быстрым темпом развития портативной техники наиболее актуальным способом подключения к интернету является беспроводное подключение.

В сети Wi-Fi предусматривается использование метода доступа к единой среде передачи данных CSMA/CA [1-5]. Метод носит вероятностный характер, который не гарантирует успешность передачи сообщения в случае высокой интенсивности сетевого трафика. По этой причине сети, использующие данный метод доступа, плохо приспособлены для решения задач управления в реальном масштабе времени. Некоторые видоизменения алгоритма доступа CSMA/CA позволяют устранить такие недостатки сетей Wi-Fi, как:

- невозможность использования таких сетей для решения задач управления в реальном масштабе времени;
-невозможность приоритетного обслуживания абонентов.

В статье приведены результаты исследований возможности коррекции метода доступа CSMA/CA путём применения промежуточного драйвера в стеке сетевых драйверов.

В случае повышения интенсивности сетевого трафика, когда процент потерянных пакетов превысит установленный порог, время использования канала связи начинает делиться между узлами сети. Механизм включения (выключения) временного разделения основан на использовании статистического метода последовательного анализа [6]. Это метод статистического исследования при проверке гипотез, при котором после каждого наблюдения производится анализ всех предыдущих наблюдений случайной величины x. В нашем случае применения этого статистического метода проверяются две конкурирующие гипотезы: H_0- «Временное разделение канала необходимо»; H_1- «Необходимости временного разделения канала нет». В роли случайной величины x выступает количество потерянных пакетов на m-м испытании. Метод последовательной проверки гипотезы H_0 относительно гипотезы H_1 основан на применении на каждой стадии эксперимента (m-м испытании) некоторого правила принятия одного из трёх возможных решений. Правило определяет три попарно-непересекающиеся области R_{m0}, R_{m1} *и* R_m множества всех возможных выборок $(x_1, \ldots, x_m)$ объёма m, таких, что если наблюдённая выборка, начиная с $m=1$, попала в область R_{m0}, то принимается 1-е решение (гипотеза H_0), если в область R_m, то эксперимент продолжается и проводится очередное $m+1$- е испытание.

Область R_{m1} называется *критической областью* W и однозначно определяется ошибками двух родов. *Ошибка 1-го рода*, если отклоняется гипотеза H_0, в то время как она истинна. *Ошибка 2-го рода*- если гипотеза H_0 принимается, в то время как истинна конкурирующая гипотеза H_1. Вероятность ошибки 1-го рода α равна вероятности попадания наблюдённой выборки в критическую область W, вычисленной при гипотезе H_0, а вероятность ошибки 2-го рода β- вероятности непопадания выборки в область W, вычисленной при гипотезе H_1.

Процесс последовательной проверки характеризуется допускаемым риском, связанным с принятием неверных решений. Допускаемый риск определяется выбором четырёх чисел: ошибками 1-го рода (α) и 2-го рода (β), а также верхней (Q_0) и нижней (Q_1) границами областей принятия и отклонения проверяемой гипотезы.

Ошибки α и β и значения Q_0 и Q_1 выбираются на основе оценки последствий, к которым приводит неправильное решение, так чтобы вероятность принятия гипотезы H_0 не превышала величины α, когда истинное значение неизвестного параметра $Q<=Q_0$ и вероятность не принятия гипотезы H_0 не превышала β, когда $Q>=Q_1$.

Методика проведения последовательного анализа.

1) Рассчитываются значения:

$$b = (1-\beta)/\alpha \quad (5)$$

$$c = \beta(1-\alpha) \quad (6)$$

$$d = (1-Q_0)/(1-Q_1) \quad (7)$$

$$e = (1-Q_1)/(1-Q_0) \quad (8)$$

$$f = Q_1/Q_0 \quad (9)$$

2) Определяются значения чисел:

$$a_m = ln(c)/(ln(f)-ln(e)) + m*ln(d)/(ln(f)-ln(e)) \quad (10)$$

$$r_m = ln(b)/(ln(f)-ln(e)) + m*ln(d)/(ln(f)-ln(e)) \quad (11)$$

2) На m-м испытании вычисляется число потерянных кадров

$$d_m = \sum_{i=1}^{m} x_i \quad (12)$$

и проверяется условие $a_m < d_m < r_m$.

Если $a_m < d_m < r_m$ может быть принята любая гипотеза,

если $d_m \geq r_m$ принимается гипотеза H_0, а

если $d_m \leq a_m$ принимается гипотеза H_1.

Пример. Путём моделирования варианта сеанса связи проверить целесообразность принятия гипотезы H_0. Приём гипотезы H_0 («Временное разделение канала необходимо»), будем считать целесообразным, если ему соответствует вероятность успешной передачи пакета $W >= 0,8$. Пусть с вероятностью $\alpha \leq 0,02$ допустимо принятие решения о том, что приём гипотезы H_0 целесообразен, хотя $W < 0,7$, и пусть с вероятностью $\beta \leq 0,03$ также допустимо принятие альтернативной гипотезы H_1 («Необходимости временного разделения канала нет»), хотя $W > 0,9$. Таким образом, требуется проверить целесообразность приёма гипотезы H_0 при допускаемом риске:

$Q_0 = 1 - 0.9 = 0.1;\ Q_1 = 1 - 0.7 = 0.3;\ \alpha = 0.02;\ \beta = 0.03$

Решение. 1. Проверка целесообразности приёма гипотезы H_0.

а) По формулам (10, 11) рассчитываются a_m и r_m и заносятся в табл. 3.

б) Проводится моделирование. После каждого m = 1, 2, … вычисляется d_m, заносится в табл. 3 и сравнивается с a_m и r_m.

в) При m = 17 $d_m = r_m$. Проверка закончилась. Гипотеза H_0 принимается.

У каждого драйвера должна быть процедура DriverEntry [7-10], которая инициализирует структуру данных и ресурсы всего драйвера. Она является первой подпрограммой вызываемой диспетчером ввода-вывода после загрузки драйвера. Инициализация драйвера включает экспорт других точек входа, инициализацию используемых объектов и установку различных системных ресурсов.

Таблица 3

Результаты проверки целесообразности принятия гипотезы H_0

m	a_m	d_m	r_m		m	a_m	d_m	r_m
1	-2	0	3		12	0	2	5

2	-2	0	3		13	0	3	5
3	-2	1	3		14	0	3	5
4	-2	1	4		15	0	4	6
5	-2	1	4		16	0	5	6
6	-1	1	4		17	1	6	6
7	-1	1	4		18	1		6
8	-1	1	4		19	1		6
9	-1	2	5		20	1		7
10	-1	2	5		21	1		7
11	-1	2	5		22	2		7

Функция DriverEntry:

NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject,

IN PUNICODE_STRING RegistryPath)

Параметры функции DriverEntry:

1. Указатель на структуру DRIVER_OBJECT, который представляет адрес объекта драйвера;

2. Указатель на структуру UNICODE_STRING, который определяет путь к параметрам (подразделу) драйвера в реестре.

Возвращаемые значения:

Если подпрограмма успешно выполняется, она должна возвратить STATUS_SUCCESS. Иначе, она должна возвратить одно из ошибочных значений состояния, которые определены в ntstatus.h.

Прежде чем начать функционировать, сетевой промежуточный драйвер должен выполнить как минимум 4 функции в подпрограмме DriverEntry:

1. NdisMInitializeWrapper — регистрация драйвера в среде NDIS;

2. NdisIMRegisterLayeredMiniport — регистрация точек входа MiniportXxx промежуточного драйвера (регистрация функций виртуального адаптера);

3. NdisRegisterProtocol — регистрация точек входа ProtocolXxx (регистрация функций виртуального драйвера протокола);

4. NdisIMAssociateMiniport — функция сообщает NDIS о том, что существуют верхний и нижний интерфейсы для минипорт драйверов и драйверов протокола, принадлежащие к одному промежуточному драйверу.

Для того чтобы драйвером можно было управлять из пользовательского уровня, необходимо создать и проинициализировать функции управления [10]. Для организации взаимодействия кода пользовательского режима с драйвером режима ядра, используется диспетчер ввода-вывода (I/O Manager). Подсистема ввода-вывода управляется пакетами. Большинство запросов ввода-вывода представляется пакетами запросов ввода-вывода (I/O request packets, IRP), передаваемых от одного компонента подсистемы ввода-вывода другому.

Объявляется массив указателей функций-обработчиков — IRP (I/O Request Packet) для регистрации точек входа в собственные рабочие функции драйвера:

```
PDRIVER_DISPATCH MajorFunctions [IRP_MJ_MAXIMUM_FUNCTION +
1];
```

Выделение блока памяти для массива указателей (MajorFunctions):

```
NdisZeroMemory(MajorFunctions, sizeof(MajorFunctions));
```

Параметры функции NdisZeroMemory:

1. Основной виртуальный адрес блока памяти. Тип PVOID;
2. Количество байт, которые заполнятся нулями. Тип ULONG.

Инициализация точек входа и связывание имен функций управления с массивом функций-обработчиков:

Major Functions [IRP_MJ_CREATE] = DeviceOpen;

Major Functions [IRP_MJ_CLOSE] = DeviceClose;

Major Functions [IRP_MJ_READ] = DeviceRead;

Major Functions [IRP_MJ_DEVICE_CONTROL] = DeviceControl.

Для создания нового объекта устройства и регистрации точек входа для подпрограмм IRP-обработчиков драйвера устройства, выполняется вызов следующей функции:

NdisMRegisterDevice (NdisWrapperHandle, &NtDeviceName, &Win32DeviceName, MajorFunctions, &DeviceObject, &NdisDeviceHandle);

Для взаимодействия пользователя с промежуточным драйвером, было разработано консольное приложение на языке программирования C. Приложение позволяет задать для текущего компьютера временной интервал, в течение которого разрешена отправка пакетов, включать/выключать детерминированный режим, просмотр текущего временного интервала, выгружать драйвер, включать/выключать вывод отладочных сообщений.

Основными используемыми WinAPI функциями для связи с драйвером являются: CreateFile, ReadFile, DeviceIoControl и CloseHandle.

Для того чтобы открыть устройство и получить доступ к объекту, используется функция CreateFile. CreateFile создает или открывает файл или устройство ввода/вывода, чаще всего это файл, файловый поток, служба,

каталог, физический диск, консольный буфер, коммуникационные ресурсы и другие.

Функция возвращает дескриптор (описатель), который может быть использован для получения доступа к файлу или устройству для различных типов ввода-вывода в зависимости от флагов, файла или устройства и указанных атрибутов.

```
HANDLE Device = CreateFile("\\\\.\\Passthru",
GENERIC_READ |GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, NULL);
```

Параметры функции CreateFile:

1. Имя файла объекта или устройства, которое будет создано или открыто. Тип LPCTSTR;

2. Тип доступа к объекту. Приложение может получить доступ только для чтения, доступ только для записи, для чтения-записи или доступ только для исполнения. Тип DWORD;

3. Режим совместного использования файла или устройства. Если параметр обнулен, объект не может быть совместно использован и не может быть открыт снова, пока дескриптор к файлу или устройству не закрыт. Тип DWORD;

4. Указатель на структуру SECURITY_ATTRIBUTES, которая содержит два отдельных, но связанных элементов данных: дополнительный дескриптор безопасности и булево значение, которое определяет, может ли возвращенный дескриптор быть наследован дочерним процессом. Тип LPSECURITY_ATTRIBUTES;

5. Вид действия над файлом или устройством, которое существует или не существует. Тип DWORD;

6. Атрибуты и флаги для файла. Тип DWORD;

7. Допустимый дескриптор к шаблонному файлу с правами доступа GENERIC_READ. Шаблонный файл предоставляет атрибуты файла и расширенные атрибуты для создаваемого файла. При открытии существующего файла, функция игнорирует этот параметр. Тип HANDLE.

Возвращаемые значения:

- Если файла или устройства не существует, то возвращаемое значение — INVALID_HANDLE_VALUE;

- Если функция выполняется успешно, то возвращаемое значение — открытый дескриптор (в нашем случае — Device) к указанному файлу (устройству, каналу, службе).

После успешного выполнения функции CreateFile, пользователю предлагается меню с выбором действий для управления драйвером.

Чтение данных из драйвера осуществляется при помощи WinAPI функции ReadFile. Функция принимает из драйвера текущее состояние детерминированного режима.

ReadFile(Device, &Inp, sizeof(Inp), &bytes, NULL);

Параметры функции ReadFile:

1. Дескриптор файла или устройства ввода/вывода, который был возвращен функцией CreateFile. Тип HANDLE;

2. Указатель на буфер, который принимает данные при считывании из файла или устройства. Тип LPVOID;

3. Максимальное количество байт, которые будут считаны. Тип DWORD;

4. Указатель на переменную, которая получает число считанных байтов. Тип LPDWORD;

5. Указатель на структуру OVERLAPPED. Эта структура требуется, если первый параметр был открыт с флагом FILE_FLAG_OVERLAPPED, иначе параметр может быть NULL. Тип LPOVERLAPPED.

Возвращаемые значения:

- Если функция успешно выполняется, возвращаемое значение не ноль (TRUE);

- Если функция завершается с ошибкой, возвращаемое значение — нуль (FALSE).

Для управления устройством используется универсальная функция DeviceIoControl.

Функция DeviceIoControl выполняет такие задачи, как доступ к устройству, получение информации, отправка, запись и чтение данных.

Таким образом, недостатки, присущие сетям, использующим метод доступа CSMA/CA, могут быть устранены при организации детерминированного доступа к каналам связи рабочих станций с точкой доступа. Детерминированный доступ используется в случае интенсивного сетевого трафика.

Предлагаемый способ повышения производительности применим и для сетей, использующих метод доступа *CSMA/CD*, даже если они реализованы на основе коммутатора. По причине ограниченности ёмкости буферного ЗУ коммутатора при высокой интенсивности потока заявок процент потерянных пакетов остаётся недопустимо высоким. Когда передача данных сталкивается с проблемой «бутылочного горлышка» для приёма и отправки пакетов на коммутаторах обычно используется метод *FIFO*: первый пришел — первый ушёл (*First In — First Out*). При интенсивном трафике это создаёт заторы, которые разрешаются крайне простым образом: все пакеты, не вошедшие в буфер очереди *FIFO* (на вход или на выход), игнорируются коммутатором и, соответственно, теряются безвозвратно.

Литература

1. Беделл П. Сети. Беспроводные технологии. М.: НТ Пресс, 2008. 448 с.

2. Ковалевский В.Н. Аналитико-численное моделирование распределенных информационных систем с низким уровнем сетевого трафика // Инженерный вестник Дона, 2015, №3 URL: ivdon.ru/ uploads/ article/ pdf/ IVD_ 98_ kovalevsky. pdf_ 4b30a6c6f7. pdf
3. Скоба А. Н., Логанчук М.Л. Математическая модель функционирования распределённой информационной системы на базе архитектуры «файл – сервер» с учётом влияния блокировок // Инженерный вестник Дона, 2015, №3URL:ivdon.ru/uploads/article/pdf/IVD_188_scoba_loganchuk.pdf_76d4827f24.pdf
4. Buzen J.P. Computational Algorithms for Closed Queueing Networks with Exponential Servers // Commun. ACM. -1983. – Vol.16, №9.pp.527-531.
5. Antunes C.H. et al. A Multiple Objective Routing Algorithm for Integrated Communication Network // Proc. ITC-16.-1999.V.3b.-PP.1291-1300.
6. Абчук В.А., Матвейчук Ф.А., Томашевский Л.П. Справочник по исследованию операций. М.: Воениздат, 1979. 368 с.
7. Магда Ю. С. Основы разработки драйверов устройств в операционных системах Windows . М.: ДМК Пресс, 2008. 200 с.
8. Руссинович М. , Соломон Д. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер-класс. 4-е изд. М.: Издательско-торговый дом "Русская редакция", 2005. 992 с.
9. Солдатов В.П. Программирование драйверов Windows: Науч. - тех. изд.. 2-е изд. М.: ООО "Бином-Пресс", 2004. 480 с.
10. Гром. Как писать драйвера (часть 3) // URL: club.shelek.ru/viewart.php?id=32 (дата обращения: 20.06.2016).

References

1. Bedell P. Seti. Besprovodnye tekhnologii. [Network. Wireless technology]. M.: NT Press, 2008. 448 p.

2. Kovalevskiy V.N. Inženernyj vestnik Dona (Rus), 2015, №3 URL: ivdon.ru/ uploads/ article/ pdf/ IVD_ 98_ kovalevsky. pdf_ 4b30a6c6f7. Pdf
3. Skoba A. N., Loganchuk M.L. Inženernyj vestnik Dona (Rus), 2015, №3 URL: ivdon.ru/uploads/article/pdf/IVD_188_scoba_loganchuk.pdf_76d4827f24.pdf
4. Buzen J.P. Computational Algorithms for Closed Queueing Networks with Exponential Servers. Commun. ACM. .1983. . Vol.16, №9.pp.527-531.
5. Antunes C.H. et al. A Multiple Objective Routing Algorithm for Integrated Communication Network. Proc. ITC.16. 1999. V.3b. pp.1291-1300.
6. Abchuk V.A., Matveychuk F.A., Tomashevskiy L.P. Spravochnik po issledovaniyu operatsiy. [Handbook on operations research]. M.: Voenizdat, 1979. 368 p.
7. Magda Yu. S. Osnovy razrabotki drayverov ustroystv v operatsionnykh sistemakh Windows [The basics of developing device drivers in Windows operating systems]. M.: DMK Press, 2008. 200 p.
8. Russinovich M., Solomon D. Vnutrennee ustroystvo Microsoft Windows: Windows Server 2003, Windows XP i Windows 2000. Master-klass. [The internal device of Microsoft Windows: Windows Server 2003, Windows XP and Windows 2000. Master class]. 4-e izd. M.: Izdatel'sko-torgovyy dom "Russkaya redaktsiya", 2005. 992 p.
9. Soldatov V.P. Programmirovanie drayverov Windows [Programming drivers Windows]: Nauch.-tekh. izd.. 2-e izd. M.: OOO "Binom-Press", 2004. 480 p.
10. Grom. Kak pisat' drayvera (chast' 3). [How to write driver (part 3)]. URL: club.shelek.ru/viewart.php?id=32 (data obrashcheniya: 20.06.2016).