ИНФОРМАТИКА

УДК 004.353.2

ОБ ОДНОМ ПОДХОДЕ К ОПРЕДЕЛЕНИЮ ПРОПУСКНОЙ СПОСОБНОСТИ КАНАЛОВ ЛВС ПО ПРОТОКОЛАМ TCP/IP, ICMP И UDP

В.Н. Кулинченко, О.М. Демиденко, П.Л. Чечет

Гомельский государственный университет им. Ф. Скорины, Гомель, Беларусь

AN APPROACH TO IDENTIFY CHANNEL CAPACITY OF LAN ON TCP/IP, ICMP AND UDP PROTOCOLS

V.N. Kulinchenko, O.M. Demidenko, P.L. Chechat

F. Scorina Gomel State University, Gomel, Belarus

Рассмотрен один из подходов определения пропускной способности каналов связи локальной вычислительной сети по наиболее часто используемым протоколам. Этот подход реализован программным способом с использованием мультиплатформенной библиотеки PCAP.

Ключевые слова: *пакетное зондирование, пропускная способность, протоколы передачи, ЛВС, библиотека PCAP.*

One approach determining network bandwidth of local area network for the most commonly used protocols is describes. This approach is implemented in software using a multiplatform library PCAP.

Keywords: *packet sensing, bandwidth, protocols, LAN, library PCAP.*

Введение

Для измерения физических характеристик каналов передачи данных можно разработать и создать специальные технические устройства, но более привлекательной является возможность реализации их программного измерения на основе уже интегрированных в системы связи стандартных устройств. Существующие для этого методы получили название пакетных методов. Для анализа пропускной способности каналов локальной вычислительной сети достаточно эффективным является метод однопакетного зондирования.

1 Метод однопакетного зондирования

Для охвата максимального количества тестируемых сетей при использовании этого метода необходимо задействовать наиболее часто используемые протоколы ICMP, TCP/IP и UDP. Зачем использовать ICMP протокол, если есть остальные? Этому есть несколько причин:

– при использовании данного протокола программное обеспечение устанавливается не на клиенте, а только на хосте;

– широкая поддержка данного протокола, в отличие, например, от протокола SNMP, для которого дополнительно на клиенте должен быть запущен соответствующий сервис.

Как известно из проведенных натурных экспериментов, использовать данный протокол для получения более-менее точных цифр, значит использовать пакеты максимальной длины – 65535 байт, а поскольку их использование в ряде случаев затруднительно из-за блокировок фрагментированных ICMP пакетов, то эта методика становится менее привлекательной. Вот почему важно реализовать измерения по этой же методике, но с использованием других протоколов [1].

Функционально программа должна генерировать пакеты различной длины и обрабатывать полученные пакеты – отклики. Данный способ является универсальным решением при написании мультиплатформенной программы, но для её использования необходимы права суперпользователя в Unix и Linux и администратора в Windows. В случае использования протокола ICMP для операционной системы Windows XP SP3 и Windows Vista и 7 необходимо дополнительно отключить встроенный файервол или включить данную программу в список исключений, а также добавить в реестр параметр типа DWORD KEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \AFD \Parameters \DisableRawSecurity равный 1, поскольку возможность использования «сырых» сокетов в данных операционных системах была отключена компанией Microsoft по умолчанию. Вообще в Windows есть динамически подключаемая библиотека ICMP.dll, реализующая возможности работы с ICMP протоколом, но у нее наблюдаются проблемы при работе в составе мультипотоковой программы.

Для захвата пакетов при использовании протокола ICMP, также для возможности мультиплатформенности, использовалась бесплатная, свободно распространяемая библиотека PCAP (ее Windows аналог – библиотека WinPCap). Архитектура PCAP изначально разрабатывалась для операционных систем с открытым исходным кодом. Закрытые коммерческие операционные

системы остались без внимания разработчиков. Этот факт явился одной из причин, из-за которых практически все более-менее серьезные сетевые приложения написаны для UNIX, Linux, BSD. Windows в этом плане заметно отставала.

Удобство отладки, наличие удобной системы помощи, быстрая разработка приложений и визуальных интерфейсов и масса других положительных качеств у средств разработки приложений (Delphi, C++ Builder, Visual Studio) и отсутствие возможности низкоуровневого управления сетевым интерфейсом привели к возникновению необходимости создания архитектуры, аналогичной PCAP, для семейства ОС Windows. Естественно, разработчики архитектуры WinPCAP не стали изобретать велосипед, а адаптировали существующую архитектуру PCAP для ОС Windows. Процесс переноса PCAP на платформу Windows заключался в адаптации PCAP-драйвера и библиотеки libpcap для работы под Win32. Оригинальная версия libpcap написана на языке С с учетом возможности переноса библиотеки на различные версии UNIX. ОС Windows не поддерживает всех вызовов POSIX-систем, однако предоставляет некоторые аналогичные им функции через API. Модели памяти UNIX и Windows одинаковы (Windows и большинство UNIX являются 32-х битными ОС) и имеют аналогичный размер целых чисел [2].

Версия PCAP для Win32 основана на драйвере захвата пакетов, структура и принцип действия которого аналогичен его предшественнику для UNIX. Это значительно облегчает процесс переноса приложений из одной операционной системы на другую. Библиотеки и функции ОС UNIX, отсутствующие в Windows (например, getnetbyname) и необходимые для компиляции libpcap на Windows-машине, разработчикам пришлось включить в исходный код драйвера. Часть исходного кода, отвечающая за взаимодействие с сетевым адаптером, была изменена для поддержки его NDIS-драйвера. В соответствии с обозначениями, принятыми в оригинальной версии libpcap, исходный код для взаимодействия с драйвером находится в наборе файлов pcap-XXX.c (и соответствующий ему pcap-XXX.h), где XXX – указывает на операционную систему (например, pcap-linux.c). К уже существующим файлам были добавлены pcap-win32.c и pcap-win32.h.

Основному изменению подвергся принцип взаимодействия приложения пользователя с драйвером захвата пакетов. Libpcap для Win32 взаимодействует с аппаратным обеспечением через интерфейс, предоставляемый динамической библиотекой packet.dll (в отличие от Windows, в ОС UNIX сетевой адаптер или модем «виден» как стандартный файл, поэтому нет необходимости в использовании промежуточных библиотек, достаточно просто создать пакет необходимой структуры и записать его в этот файл). Это не влияет на нормальную работу libpcap, однако может создать определенные проблемы программисту, желающему получить доступ непосредственно к драйверу захвата пакетов. Например, в ОС UNIX возможно использовать системный вызов SELECT для того, чтобы узнать, поступил ли пакет на вход адаптера. В ОС Windows такая возможность отсутствует.

Программист может использовать функции libpcap для обеспечения работоспособности исходного кода приложения на различных операционных системах, но при этом возможности приложения будут ограничены (например, libpcap не позволяет отправлять пакеты через сетевой интерфейс). Если программист решит воспользоваться функциями packet.dll, то его приложение будет работать только под управлением ОС семейства Win32, однако при этом возможности приложения будут практически неограниченными.

Для обеспечения максимальной совместимости исходного кода libpcap различных ОС, часть кода, предназначенная для ОС Windows, отделена от остального кода директивами #ifdef и #ifndef. Например:

```
#ifdef WIN32
/* исходный код для Windows */
#endif
```

Это позволяет компилировать исходный код libpcap как на ОС Windows, так и на UNIX.

2 Алгоритм работы программной реализации метода однопакетного зондирования

Упрощенная блок-схема алгоритма работы программы изображена на рисунке 2.1.

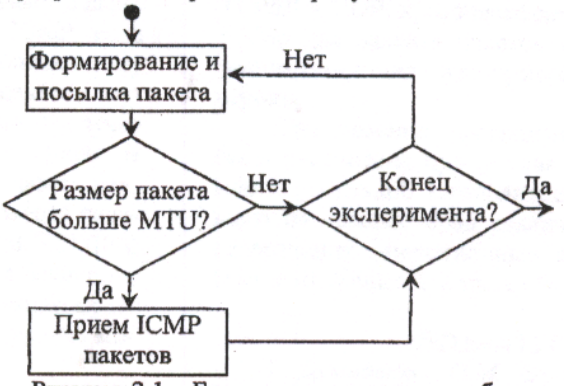

Рисунок 2.1 – Блок-схема алгоритма работы программы по протоколу ICMP

Все функции для работы с «сырыми» сокетами, а также сокетами TCP/IP и UDP содержаться в файле raw_socket.cpp проекта. Функции для работы с сокетами следующие:

– после создания сокета функцией WSASocket, вызывается функция rs_init, инициализирующая библиотеку по работе с сокетами Windows;

– rs_set_raw задает параметры «сырого» сокета;

– rs_exit завершает работу с библиотекой по работе с сокетами Windows;

– rs_send_ICMP формирует заголовок и сам пакет ICMP и вызывает функцию rs_send_IP;

– rs_send_IP непосредственно формирует заголовок IP и производит посылку пакета;

– Get_Message служит для получения данных эксперимента по протоколам TCP/IP и UDP;

– Put_Control_Message служит для отправки контрольных сообщений от клиента к серверу по протоколам TCP/IP и UDP;

– Send_Information_To_Client служит для отправки результатов эксперимента сервера к клиенту по протоколам TCP/IP и UDP;

– Receive_Information_From_Server служит для приема результатов эксперимента от сервера для их обработки и отображения клиентом по протоколам TCP/IP и UDP.

Главный модуль программы main.cpp представляет пользовательский интерфейс и реализует функциональный алгоритм программы: по действию пользователя запускает процесс генерации пакетов, который может быть как с фиксированной, так и с переменной длиной пакета, изменяющейся в ходе эксперимента с заданным пользователем шагом, а также создает второй рабочий поток приложения, в котором осуществляется прием пакетов [3].

Основные функции главного модуля:

– функция btn1Click. Эта функция – обработчик пользовательского нажатия на кнопку «Начать», запускает передачу-прием пакетов. В зависимости от выбора пользователя используется генерация и передача пакета фиксированной длины либо переменной с определенным шагом. В случае использования протокола ICMP, если размер пакета меньше или равен максимальному блоку передачи (MTU), то посылка пакетов осуществляется асинхронно, в противном случае используется синхронная передача – прием. В случае фрагментации время прихода пакета равно времени прихода последнего фрагмента пакета. В случае использования протоколов TCP/IP и UDP посылка буферов осуществляется размером, заданным пользователем, а их возможная фрагментация осуществляется на сетевом уровне модели OSI;

– функция send_packet реализует непосредственную отправку пакетов, в которой происходит создание сокета, инициализация сокета, заполнение необходимых заголовков динамически получаемыми данными, установку признака фрагментации и посылку пакета.

Модуль recv.cpp отвечает за прием пакетов по протоколу ICMP. В данном модуле происходит обнаружение доступных на машине сетевых адаптеров, а также подключение по выбору к одному из них для фильтрации пакетов ICMP, как входящих, так и исходящих. Основная функция данного модуля – это функция packet_handler. Она вызывается каждый раз при приеме пакета, заданного в фильтре. В данной функции производится разбор пакета, идентификация фрагментированных пакетов и заполнение структуры sock_time. При использовании протокола TCP/IP функции сервера исполняются в модуле tcp_server.cpp, а UDP – UDP_server.cpp. Данные серверы реализованы отдельным потоком приложения для избежания эффекта «подтормаживания», а также для реализации возможности работы с главной формой приложения.

Основные структуры, используемые главным модулем приложения и модулем recv.cpp:

– структура sock_time. В данную структуру записывается идентификатор пакета, время посылки и приема пакета и высчитанная пропускная способность;

– структура сбора статистики globak_stat. В данную структуру для определенного размера пакета записываются и номинальная и эффективная пропускные способности, среднее время прохождения пакета, максимальная и минимальная номинальная пропускная способность, размер пакета, для которого эта статистика была набрана.

Заключение

Программная подсистема создавалась на языке C++ в интегрированной среде разработки Borland C++ Builder. Приложение использует «сырые» сокеты для генерации пакетов по протоколу ICMP и API сокеты Java по протоколам TCP/IP и UDP, кроссплатформенную библиотеку *PCap для захвата пакетов из сети, интерфейс реализован в виде web-приложения и командной строки.

При помощи программы опытным путем была рассчитана пропускная способность фрагмента локальной вычислительной сети, состоящего из компьютеров различных архитектур и содержащих операционные системы семейства Microsoft Windows, а также *nix.

ЛИТЕРАТУРА

1. *Демиденко, О.М.* Функциональные возможности программного комплекса адаптивной идентификации пользователей корпоративной сети / О.М. Демиденко, В.Д. Левчук, А.И. Кучеров // Проблемы физики, математики и техники. – 2010. – № 3 (4). – С. 69–73.

2. *Семенов, Ю.А.* Телекоммуникационные технологии : учебное пособие / Ю.А. Семенов. – ГНЦ ИТЭФ, 2010. – 600 с.

3. *Lai, K.* Measuring Link Bandwidths Using a Deterministic Model of Packet Delay / K. Lai. – Stanford University, 2001.

Поступила в редакцию 24.09.14.