Windows 2000 에서의 데이터 링크 계층 접근을 위한 Packet Driver API 설계 및 구현

유 환석, 김 상하*
*충남대학교 컴퓨터과학과

e-mail : shkim@cclab.cnu.ac.kr

Design and Implementation of Packet Driver for Accessing DataLink Layer in Windows 2000

Hwan-Souk Yoo, Sang-Ha Kim*
*Dept. of Computer Science, ChungNam University

요 약

인터넷 소프트웨어 개발자는 인터넷 보안 소프트웨어, 네트워크 실습 교육용 소프트웨어 등의 다양한 서비스 개발을 위해, 데이터 링크 계층의 프레임에 직접 접근하는 방법을 요구한다. 이러한 개발자를 위해, Windows 2000 은 사용자 계층에서 인터넷 관련 네트워크 자원에 접근할 수 있는 Winsock, NPP, Internet Protocol Helper API 를 지원한다. 하지만, 개발자가 이러한 API, 특히 Winsock API 를 통해 직접 데이터 링크 계층의 프레임에 접근하는 데에는 한계가 있다. 따라서, 본 논문에서는 NDIS Library 를 이용하여 일반 개발자가 손쉽게 데이터 링크 계층의 프레임에 접근할 수 있는 packet driver API 를 설계하였다. API 를 구성하는 각 클래스는 계층적 구조로 설계하였으며 Visual C++를 사용하여 구현하였다.

1. 서론

인터넷이 일반화되면서 사용자들은 인터넷에 접근하기 쉬운 환경과 더욱 다양한 인터넷 소프트웨어를 요구하게 되었으며, 이에 따라 개발자도 인터넷 소프트웨어 개발에 많은 관심을 갖게 되었다. 하지만 개발자가 사용자의 다양한 요구사항을 만족시키기 위해서, 보다 다양한 인터넷 소프트웨어 개발 방법이 요구되고 있다. 인터넷 보안 소프트웨어, 인터넷 공유 프로그램, 네트워크 실습을 위한 소프트웨어 등은 데이터 링크 레벨의 프레임을 직접 접근하여야만 구현 가능하며, 이러한 소프웨어 개발 요구가 증가되고 있다. 그러므로 기존의 socket 을 이용한 인터넷 소프트웨어 개발방법으로는 사용자의 요구사항을 만족시키기 어렵게 되었다.

또한 사용자는 인터넷에 접근하기 위한 환경으로 Windows 를 더 선호하고 있다. UNIX 계열의 운영체제보다 사용하기 쉽고, 인터넷에 접근하기 위한 다양한 소프트웨어가 제공되기 때문이다. 개발자 역시 사용이 편리하고 다양한 개발도구가 지원되는 Windows 를 인터넷 소프트웨어 개발환경으로 더 선호하고 있다.

Windows 계열의 개발 환경을 살펴보면, 사용자 레벨의 인터넷 소프트웨어 개발을 위한 대표적인 API 는 Winsock API(Windows Sockets)이다. Winsock 은 사용이 쉽지만 데이터 링크 계층에 접근하는 데에는 한계가 있다.

그 밖의 API 로는 NPP(Network Packet Providers), IP Helper(Internet Protocol Helper), DNS API(Domain Name System API), DHCP API(Dynamic Host Configuration Protocol API), MADCAP(Multicast Address Dynamic Client Allocation Protocol) 등이 있으며, 이 API 중에서 네트워크 자원에 공용으로 접근 할 수 있는 것은 Winsock, NPP, IP Helper 이다. Winsock 에서는 링크계층까지 접근할 수 없고, NPP 는 읽기만 가능하고, IP Helper 는 ARP 만 가능하다.

이 논문에서는 NDIS (Network Driver Interface Specification) Library 를 사용하여 사용자 계층에서 인터넷 통신 프로토콜 계층 중 데이터 링크 계층에 쉽게 접근할 수 있는 packet driver API 를 설계 및 구현하였다. 이 API 는 Windows System 의 대표적인 개발 툴인 Visual C++에서 사용 할 수 있는 클래스로 구현되어 있으며, 각 계층별 계층적 호출 구조를 가지고 있다.

이 논문은 다음과 같이 구성된다. 2 장에서는 Windows 2000 Network API 의 구조 및 특징에 대해 살펴본다. 3 장에서는 NDIS 의 구조와 NDIS 드라이버와 WIN32 API 를 이용한 packet driver API 구조와 이를 이용한 예제(Repeater) 구현에 대하여 살펴본다. 4 장에서는 향후 발전 방향에 대하여 알아본다.

2. Windows 2000 Network API 의 구조 및 특징

Windows 2000(WIN2K)을 위한 개발 툴에서는 인터넷 소프트웨어 개발을 위하여 몇 가지 API 를 제공하고 있다. 이 장에서는 이 API 에 대한 구조와 특징 및 각 API 와 Protocol 사이의 관계에 대하여 알아본다.

2.1 Winsock (Windows Sockets)

Windows 계열 대부분의 인터넷 소프트웨어가 사용하고 있는 API 이다. Winsock 은 프로토콜이 아니라 전송 프로토콜 계층에 접근하기 위한 인터페이스에 불과하며 DLL 형태로 Windows 시스템에 존재하고 있는 WS2_32.dll 파일이 이에 해당한다.

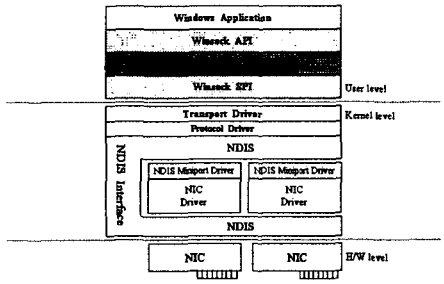

그림 1. Winsock 의 구조 [1]

그림 1 에서의 WS2_32.dll 은 상위 계층으로는 API 를 통하여 사용자 프로그램에 인터페이스를, 하위 계층으로는 SPI(Service Provider Interface)를 통하여 전송 프로토콜 계층에 인터페이스를 제공한다.[1]

Winsock 은 전송 프로토콜 계층과 인터페이스를 이루고 있으므로 기본적으로 그에 해당하는 TCP 나 UDP 에만 접근할 수 있다. 이를 확장하기 위해서는 방법으로는 소켓(socket)의 프로토콜 종류를 *SOCK_RAW* 하여 사용하는 방법이 있다. 이 방법은 ICMP 의 접근에 적당하며 TCP, UDP, IGMP 접근을 위해 사용할 수 있다.

Winsock 의 확장이 더 필요한 경우는 setsockopt() 함수와 *IP_HDRINCL* 옵션을 사용하여 IP 헤더(Header)부분까지 확장이 가능하다. 이 방법은 소켓에게 보내지는 데이터의 앞부분에 IP 헤더가 포함된 것을 알려주고 전송 프로토콜 레벨에서는 IP 헤더를 만들지 않는 대신 사용자가 작성한 IP 헤더를 사용하게 된다. 그러므로 사용자가 IP 패킷 전체를 만들고 전송할 수 있게 된 것이다. [4]

하지만 Winsock 에서는 전송에 대한 확장만이 허용할 뿐 수신에 대한 확장은 지원하지 않으므로 확장을 위해서는 Network Monitor SDK 중 NPP 를 사용한다. [1]

2.2 NPP (Network Packet Providers)

MS(Microsoft)사에서는 시스템 관리를 위하여 SMS (Systems Management Server) 를 개발하였다. 이 중에서 네트워크의 문제점 발견과 정보 수집을 위한 네트워크 모니터링 시스템에서는 데이터 링크 계층의 프레임에 대한 수집이 필요하게 되었다. 이를 지원하기 위해 개발한 것이 NPP (Network Packet Providers)이다. [1]

그림 2 는 NPP 의 구조를 나타내고 있다. NPP 는 커널 계층의 네트워크 모니터링 시스템 드라이버 (Nmnt.sys)로부터 얻은 네트워크 정보를 Network Monitor UI, Monitor Applications, NPP Applications 등에게 COM Interface 를 통하여 전달한다. NPP 는 프레임 수집 기능을 수행하는 부분과 트래픽에 대한 통계 정보를 관리하는 부분으로 구성되어있다.

IDelaydC Interface 는 트래픽에 대한 정보를 파일형태로 저장하여 결과를 알려며, **IRTC** Interface 는 실시간 트래픽에 대한 정보를 수집하여 고정된 크기의 버퍼에 저장한다. **IESP** interface 는 수집된 프레임에 대한 자세한 통계정보를 파일로 제공해준다. **IStats** interface 는 각종 Filter 수준의 통계 정보를 제공해준다. (예. 수집한 프레임의 수와 크기, Multicasting 을 통해 전송 받은 프레임의 수와 크기) [1]

하지만 NPP 는 Monitoring 을 위해 개발되었기 때문에 정보수집에 관련된 오퍼레이션만을 수행할 수 있고, 데이터 링크 레벨의 전송 기능은 지원하지 않는다.

2.3 IP Helper (Internet Protocol Helper)

IP Helper 는 Internet Protocol 을 사용하는 컴퓨터의 관리에 중점을 둔 API 라고 볼 수 있다.

IP Helper 는 다음과 같은 기능을 가지고 있다.

- 네트워크 설정에 관련된 정보 관리

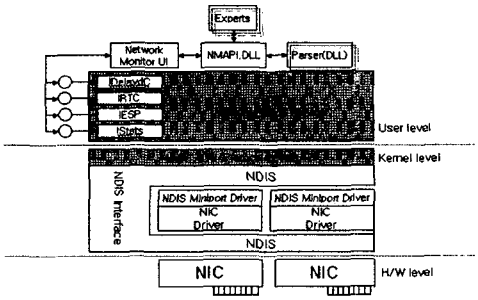

그림 2. NPP 의 구조 [1]

- 네트워크 인터페이스에 관련된 정보 관리
- 로컬 컴퓨터의 IP 계층 설정 관리
- ARP 기능 수행 및 ARP Table 관리
- 프로토콜의 통계정보
- Routing Table 관리
- 네트워크 설정의 변경에 대한 알림 기능

IP Helper 에서는 사용자 계층에서 ARP 를 직접 수행할 수 있으며, ARP Table 의 열람, 추가, 삭제 가능하다. 하지만 단순 ARP 만 지원하므로 RARP, Proxy ARP, Gratuitous ARP 는 구현 불가능하다.

IP Helper 기능은 NDIS Library 를 사용하여 모두 구현 가능하지만 사용자 레벨에서는 커널의 NDIS 에 직접 접근할 수 없으므로 IP Helper 를 사용하여 개발자에게 보다 편리한 개발 환경을 제공해주고 있다.

2.4 API 와 프로토콜의 관계

표 1 은 지금까지 알아본 API 와 프로토콜과 대하여 API 별 지원 가능한 프로토콜을 요약한 것이다.

API	지원가능 프로토콜	비고

Winsock (RAW Mode)	TCP UDP (IP) IP Multicasting IP Broadcasting (ICMP) (IGMP)	Asynchronous Read/Write 가능
NPP	Datalink Layer	Promiscuous Read
IP Helper	ARP	RARP, Proxy ARP, Gratuitous ARP 불가능

표 1. API 와 프로토콜과의 관계

일반적으로 네트워크 프로그래밍을 위한 API 는 다음과 같은 요구 사항이 있다.

- 가능한 낮은 데이터 링크 계층까지 접근 가능.
- Promiscuous 모드로 동작 가능.
- 사용자 정의 프로토콜을 사용할 수 있어야 함.
- 사용자 계층에서 쉽게 접근 가능.
- Asynchronous 모드로 동작 가능.
- 네트워크 인터페이스의 정보 열람 가능.
- 로컬 컴퓨터의 네트워크 설정 정보를 열람 가능.
- 장치 드라이버 계층의 정보를 질의/설정 가능.

Windows 2000 에서 사용할 수 있는 API 는 위의 요구사항을 부분적으로만 만족하며 가장 중요한 기능인 데이터 링크 계층에서 사용자의 직접 전송을 지원하지 않는다. 이 기능은 라우터의 포워딩(forwarding)과 같은 기능을 구현하기 위해서는 반드시 필요하다.

이를 문제를 해결하기 위하여 NDIS Library 를 이용한 packet driver 를 사용한다.

3. Packet Driver 의 설계 및 구현

Packet Driver 는 커널 계층의 NDIS Library 를 사용한 Protocol Driver 와 사용자 계층의 WIN32 API 를 사용한 WIN32 Packet Driver 로 구성 되어 있다.

3.1 NDIS (Network Driver Interface Specification)

그림 3 은 NDIS 를 사용하여 프로토콜 스택을 구성한 것이다. NDIS 는 다음과 같은 기능을 제공한다.

- **NDIS Miniport Driver**
  하드웨어와 상위 계층간의 통신을 담당한다. NIC 을 통한 데이터 전송,수신 기능이 있으며, 상위 계층의 드라이버를 위한 인터페이스 가지고 있다. 대부분 NIC 제조 업체에서 제공한다.

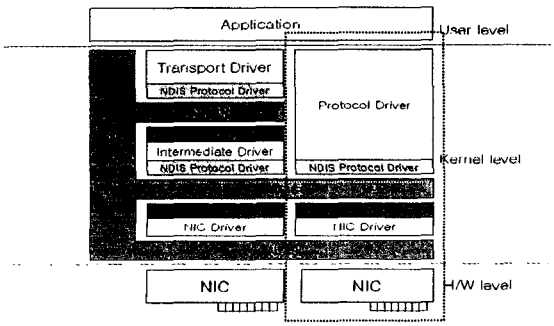

그림 3. NDIS 를 사용한 프로토콜 구성[2]

- **NDIS Intermediate Driver**
  새로운 계층을 추가하여 새로운 프로토콜 추가, 패킷 필터링, 다른 종류의 네트워크 미디어간의 패킷 교환 등의 기능을 수행할 수 있다. 상위 계층 인터페이스로 NDIS Miniport Driver 를 사용하며 하위 계층 인터페이스로 NDIS Protocol Driver 를 사용한다.
- **NDIS Protocol Driver**
  상위 계층의 Driver 와 하위 계층의 Driver 사이에서 인터페이스로 사용된다. 그림 3 에서 Intermediate Driver 가 상위의 Transport Driver 에게 접근하기 위해서는 Transport Driver 의 NDIS Protocol Driver 에 정의되어 있는 함수를 호출하며 Transport Driver 가 하위 계층 Driver 에 접근하기 위해서 NDIS 함수를 사용한다.

그림 4 는 사용자 계층에서 데이터 링크 계층 접근을 위한 Packet Driver 의 구조를 나타내고 있다. 사용자의 Protocol Driver 가 NDIS Protocol Driver Interface 를 사용하여 NIC 의 Miniport Driver 에 접근할 수 있게 되고 그에 따라 데이터 링크 계층에 사용자가 직접 접근할 수 있게 된다.

Packet Driver 구현에 중요한 Protocol Driver 에 대하여 자세히 알아보겠다.

3.1.1 NDIS Protocol Driver 등록

NT Service 가 시작되어 해당 Driver 를 Loading 과정 중, DriverEntry 함수가 호출되어 Protocol Driver 의 등록이 시작된다. NDIS_PROTOCOL_CHARACTERISTICS 구조체와 *NdisRegisterProtocol()* 함수를 사용하여 Protocol Driver 의 이름, 버전을 등록하고 각 핸들에 대한 함수를 지정하게 된다. 이는 하위 계층의 수행결과를 상위 계층으로 전달하기 위해 방법으로 하위 계층의 수행결과에 해당하는 핸들이 가르치고 있는 함수를 호출함으로써 이루어진다. 특별히 핸들의 이름에 xxxCompleteHandler 인 핸들은 비동기 모드 수행을 위한 함수로 비동기 모드를 사용하려면 반드시 구현하여야 한다.

예) SendCompleteHandler = PacketSendComplete

다음과 같이 설정되어 있다면 Protocol Driver 가 패킷을 보내기위해 *NdisSendPackets()* 함수를 호출하고 정상적으로 수행되면 운용체제 시스템에 의해 Protocol Driver 의 *PacketSendComplete()*함수가 수행 되어 그 결과를 알 수 있다.

DRIVER_OBJECT 구조체는 사용자가 커널 계층으로 접근하기위해 WIN32 API 사용할 때 WIN32 API I/O 함수와 Protocol Driver 에 정의되어 있는 함수와의 대응 관계를 나타내고 있다. 사용자가 *CreateFile()*함수를 사용하려면 하위 계층에서 IRP_MJ_CREATE 에 해당하는 함수가 반드시 존재해야만 가능하다.

Ex) MajorFunction[IRP_MJ_CREATE] = PacketOpen

3.1.2 I/O Control 및 Request/Query

사용자 계층에서 I/O Control 을 위하여 WIN32 API 중 *DeviceIoControl()*함수를 사용하면 커널 계층에서는 Protocol Driver 의 정의에 의해 *PacketIoControl()* 함수가 실행이 된다. PacketIoControl 은 두가지의 방법으로 Request 를 처리한다. Request 가 NDIS Driver 에 관련된 것이면 PACKET_OID_DATA 구조체에 의해 NDIS 드라이버에게 정보를 요청한다. 하지만 일반적인 I/O Control 이라면 커널의 상

위 드라이버가 하위 드라이버에게 정보를 요청하거나 질의하는데 가장 많이 쓰이는 IRP Function 을 사용하여 처리한다.[2]

Packet Driver	WIN32 API	Protocol Driver
PacketOpenAdapter	CreateFile	PacketOpen
PacketSendPacket	WriteFile	PacketWrite
PacketReceivePacket	ReadFile	PacketRead
PacketRequest	DeviceIoControl	PacketIoControl
PacketWaitPacket	GetOverlappedResult	Complete Function

표 2 계층별 함수의 대응관계

3.2 WIN32 API 를 이용한 WIN32 Packet Driver

사용자 계층의 Packet Driver 는 WIN32 API 를 사용하여 구성되어 있고, 사용자의 편의를 위하여 클래스로 구성되어 있다. 기존의 C 스타일로 쓰여진 Packet Driver 를 Visual C++에서 사용할 수 있게 클래스화하고 이를 상속 받아 CNetworkInterface 클래스로 확장 하였다. CNetworkInterface 클래스에는 Asynchronous 수행을 위한 정적크기의 tail-drop 버퍼를 가지고 있으며, 쓰레드 생성을 위한 부분과 여러 쓰레드에서 버퍼에 동시에 접근을 방지하는 Critical Section 을 설정하였다.

CHost 클래스에서는 시스템에 설치되어있는 NIC 을 발견하고 사용자에게 알려준다. 사용자는 NIC 의 이름을 이용하여 CNetworkInterface 클래스를 초기화 통합 관리를 위하여 CNetwork 클래스에 추가한다. CNetwork 클래스에는 모든 NIC 에 대한 포인터를 가지고 있기 때문에 CNetwork 클래스

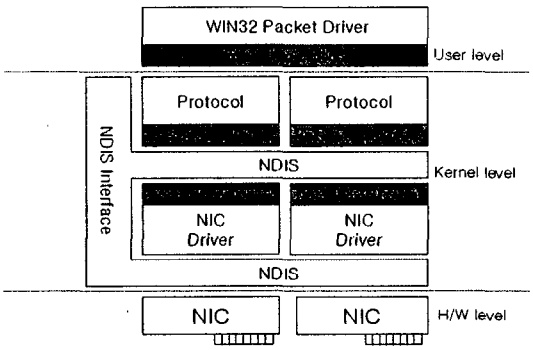

그림 4. Packet Driver 의 구조

만 가지면 모든 NIC 을 제어할 수 있다.

표 2 는 사용자 계층에서 Protocol Driver 까지의 계층적 구조를 나타난 것이다.

3.3 리피터(Repeater) 구현

그림 5 은 리피터를 Packet Driver API 를 사용하여 모델링한 것이다. 리피터의 원리는 NIC#1 으로 들어온 데이터 링크 계층의 프레임을 변환 없이 NIC#2 를 통하여 내보내는 것이다. 반대로도 동작하야 한다. NIC#1 의 입력버퍼를 NIC#2 의 출력버퍼로, NIC#2 의 입력버퍼는 NIC#1 의 출력버퍼로 연결하여 Repeater 를 구현하였다.

그림 6 은 리피터를 Class Diagram 을 나타낸 것이다.

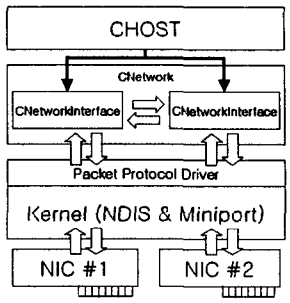

그림 5 Packet Driver 를 사용한리피터의 구조

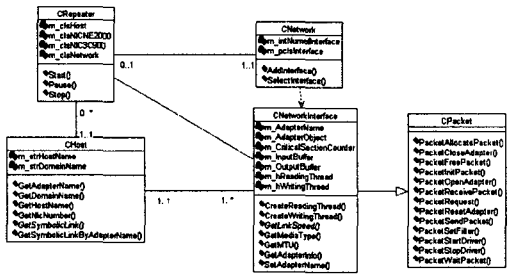

그림 6. 리피터의 클래스 다이어그램

4. 결론

본 논문에서는 Windows 2000 에서 인터넷 소프트웨어 개발을 위한 API 의 문제점을 알아보고 이를 해결하기 위해 NDIS 를 이용한 packet driver API 를 설계하고 구현하였다. Packet driver API 는 사용자 레벨에서 WIN32 API 를 사용하여 커널의 Protocol Driver 를 접근하여 데이터 링크 레벨의 프레임을 조작할 수 있다.

사용자 계층과 커널 계층의 입출력에 있어서, 사용자 계층보다 우선 순위가 높은 커널 계층에게 더 많은 자원이 할당되므로 packet driver 성능이 커널의 Driver 보다 상대적으로 낮게 감소한다. 이를 해결기 위해 Protocol driver 에 사용자 버퍼를 구성하는 방법과 사용자 계층에서의 비동기화 버퍼에 대하여 연구가 필요하다.

4. 참고 문헌

[1] MSDN, *Microsoft® Platform Software Development Kit(April 2000) Documentation*, Microsoft Corporation, 2000.
[2] MSDN, *Windows 2000 Driver Development Kit (March 2000) Documentation*, Microsoft Corporation, 2000.
[3] MSDN, *MSDN Library Visual Studio 6.0*, Microsoft Corporation, 1998.
[4] A. Jones, et al., *Network Programming for Microsoft Windows*, Microsoft®Press, 1999.
[6] W.R Stevens, TCP/IP Illustrated, Volume II The Implementation, Addison-Wesley, 1995.