유/무선 Seamless 핸드오프를 위한 설계 및 구현

Design and Implementaion for Wired/Wireless Seamless Handoff

이학구* 김평수*, 김선우*, 김영근*
HakGoo LEE, PyungSoo KIM, SunWoo KIM and YoungKeun KIM

Abstract - This paper proposes design and implementation for Seamless Handoff method between adapters in a system environment where both wired and wireless adapters are present. First of all, by settingLayer 2 address of wired adapter to Layer 2address of wireless adapter, then generate virtual adapter on the above layer to make these two adapters operate on an IP address. Under the condition, when wired communication via the wired adapter gets disconnected while in service, wireless handoff occurs by mapping information on the wireless adapter to the virtual adapter. According to the method proposed in this paper, continuous session can be obtained even when handoff between wired and wireless adapters occurs at lower level in an application where both IP address and Port address are used to maintain session since IP address does not change.

Key Words : 802.3, 802.11, wired, wireless, handoff, NDIS, Intermediate Driver

1. 서론

현재 대부분의 컴퓨터 시스템 OS로 사용되는 Windows OS는 다수의 물리적 어댑터(Physical Adapter)를 설치할 수 있다. 그리고, 각각의 물리적 어댑터에 각기 다른 IP 주소를 부여하여 통신을 할 수 있다. 상대방과 세션을 설정하여 통신을 하고자 할 때는, 기본적으로 네트워크 계층(Network Layer)의 IP 주소와 전송 계층(Transport Layer)의 포트(Port) 주소를 쌍(Pair)으로 해서 통신을 하게 된다. 만약 통신도중 이 쌍 중에 하나라도 바뀌게 되면 통신 세션은 바로 끊기게 된다 [1].

Fig.1 은 Windows OS의 네트워킹 기능을 부여하기 위해서 마이크로소프트(Microsoft)사에서 제공하는 커널 레벨의 라이브러리를 나타낸다. 이를 NDIS(Network Driver Interface Specification)라고 말을 한다.

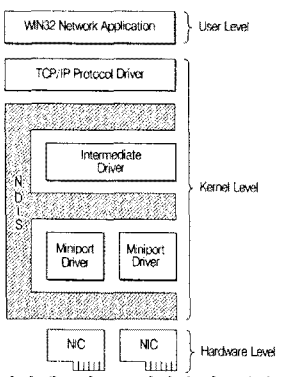

Fig.1. 전체 네트워크 드라이버 시스템의 구성도

미니포트 드라이버(Miniport Driver)는 물리적 어댑터를 구동하기 위한 드라이버로써 드라이버 개발자가 NDIS의 미니포트 스펙(Miniport Spec.)에 맞게 설계하여 구현한다. 인터미디엇 드라이버(Intermediate Driver)는 프로토콜 드라이버(Protocol Driver)와 미니포트드라이버 사이에 존재한다. 잘 알려져 있는 TCP/IP나 IPX/SPX/NETBIOS 등은 NDIS 상위 계층에서 동작하는 프로토콜 라이버에 속한다. 프로토콜 드라이버까지 OS의 커널 레벨에 속하고, 애플리케이션 등은 사용자 레벨이 된다 [2].

Windows OS에서는 이기종의 물리적 어댑터 뿐만 아니라, 동일 기종의 물리적 어댑터에서도 하나의 통신 매체가 끊어지면 해당 어댑터를 통해 통신하는 애플리케이션의 세션 자체를 자동으로 끊게 되어 있다. 이는 TCP의 경우 어댑터에 대한 정보를 TCB(TCP Control Block)에 저장하고 있는데, 어댑터가 끊어졌다는 정보가 TCB에 전달되면 TCP가 자동으로 해당 세션을 끊어 버린다. 이는 어댑터가 끊어졌다는 것은 곧 해당 어댑터에 할당된 IP 주소를 더 이상 사용할 수 없는 것이 자명하기 때문이다 [3].

본 논문에서는 현재 Windows OS에서 유/무선 어댑터간 Seamless 핸드오프가 지원되지 않는 문제점을 해결 하기 위해서 인터미디엇 드라이버를 새롭게 설계하고 구현하여 그 효과를 실험을 통해 입증하였다.

2장에서는 본론을 통해 새롭게 설계한 인터미디엇 드라이버의 구성 및 동작에 대해서 구체적으로 설명하였고, 3장에서는 제안한 인터미디엇 드라이버를 구현하고 실험을 통해 그 효과를 확인하였으며, 4장에서는 결론 및 향후 연구 과제에 대해 기술하였다.

저자 소개
* 正 會 員 : 삼성전자 DM연구소 Mobile Platform Lab.

2. 본론

2.1. 인터미디엇 드라이버의 구성

인터미디엇 드라이버는 Fig.1처럼 NDIS 전체 계층 중에서 미니포트 드라이버와 프로토콜 드라이버 사이에 위치한다. 인터미디엇 드라이버의 역할은 프로토콜 드라이버로부터 내려오는 패킷을 목적에 맞게 변경하여 미니포트 드라이버로 내려 보내고, 미니포트 드라이버로부터 들어오는 패킷을 받아 목적에 맞게 변경하여 프로토콜 드라이버에게 전달하는 역할을 수행한다. 이와 같은 인터미디엇 드라이버의 특성을 이용하여 본 논문에서는 Fig.2과 같은 구조의 인터미디엇 드라이버를 설계 하였다.

본 논문에 따른 인터미디엇 드라이버는 하위에 가상 프로토콜 드라이버(Virtual Protocol Driver)와, 상위에 가상 미니포트 드라이버(Virtual Miniport Driver)를 생성한다. 즉, 인터미디엇 드라이버 하위의 가상 프로토콜 드라이버와 미니포트 드라이버를 보면, 인터미디엇 드라이버는 가상 프로토콜 드라이버의 기능을 하고, 인터미디엇 드라이버 상위의 가상 미니포트 드라이버와 프로토콜 드라이버를 보면, 인터미디엇 드라이버는 가상 미니포트 드라이버의 기능을 한다. 이 때 가상미니포터 드라이버는 상위 실제 프로토콜 드라이버가 보기에는 오직 하나의 어댑터로 인식이 된다.

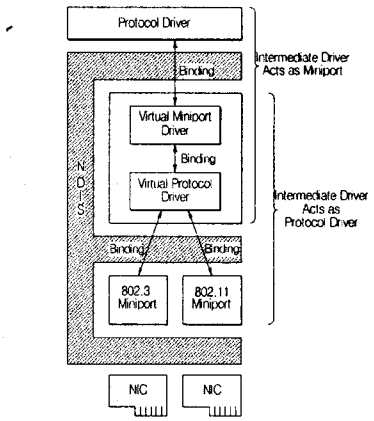

Fig.2. 본 논문에서 제안하는 드라이버 시스템의 전체 구성도

이와 같이 인터미디엇 드라이버와 그 내부에 가상 어댑터를 만드는 이유는 서론에서 언급한 Seamless 핸드오프를 위한 몇 가지 문제점을 해결하기 위해서이다. 첫째는 물리적 어댑터가 두 개이면 각각의 어댑터에 각기 다른 IP 주소를 부여해야 하기 때문에 핸드오프 시 통신의 연속성 부여가 불가능하다. 이런 이유로 가상 어댑터를 하나 두고, 실제 프로토콜 드라이버는 가상 어댑터와만 바인딩하여 동작한다. 둘째는 물리적 어댑터의 네트워크 커넥션이 끊기는 정보가 TCP/IP 프로토콜 드라이버에게 곧바로 전달되어 세션이 끊겨 버리는 문제를 인터미디엇 드라이버단에서 필터링(Filtering)을 통해 프로토콜 드라이버로의 전달을 막을 수가 있게 된다. 셋째는 인터미디엇 드라이버단에서는 패킷의 라우팅을 실시간으로 바꿀 수 있는 장점이 있다. 그래서 네트워크 커넥션의 상황에 맞추어서 통신에 사용할 최적 어댑터를 실시간으로 선택하여 패킷을 최적 어댑터를 통해 송수신을 하면 된다. 이렇게 되면 유/무선 어댑터간 핸드오프가 상위 실제 프로토콜 드라이버와 애플리케이션에게 영향을 미치지 않고, 실시간으로 이루어질 수 있다.

2.2. 동작 순서

Fig.3 처럼 유무선 어댑터 모두 DISCONNECT 상태에서 CONNECT 이벤트가 발생하면, 우선 인터미디엇 드라이버는 어떤 어댑터가 연결상태가 되었는지를 인지한다. 연결 정보를 바탕으로 바인딩 핸들 리스트에 있는 해당 어댑터에 연결 정보를 업데이트한다. 이후 인터미디엇 드라이버는 곧바로 상위 프로토콜 드라이버에게 커넥션 상태를 보고하고, 연결된 어댑터가 유/무선 인지를 판단하여 유선 어댑터인 경우, 유선 어댑터를 통해 통신을 실시한다. 유선 어댑터를 통해 통신을 하던 도중 무선 어댑터의 CONNECT 이벤트가 발생한 경우에는, 인터미디엇 드라이버는 무선 어댑터의 연결을 인지하고, 이 정보를 바인딩 핸들 리스트에 있는 무선 어댑터의 연결 정보만 업데이트한다.

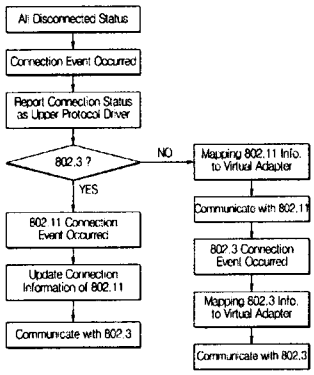

Fig.3. CONNECT 이벤트 시의 동작 흐름도

만약, 모두 DISCONNECT 상태에서 CONNECT 이벤트가 발생했을 때, 유선 어댑터가 아니고 무선 어댑터인 경우에는, 얻어진 연결 정보를 바인딩 핸들 리스트에 있는 해당 어댑터의 연결 상태 정보를 업데이트하고, 무선 어댑터를 최적 어댑터로 선택하게 된다. 그런 다음, 무선 어댑터의 정보를 가상 어댑터에 매핑한다. 그리고, 무선 어댑터를 통해서 통신을 수행한다.

이후 무선 어댑터를 통해 통신 수행중에 유선 어댑터의 CONNECT 이벤트가 발생한 경우에는, 마찬가지로 검출 된 정보를 바인딩 핸들 리스트에 업데이트하고, 업데이트된 정보를 바탕으로 최적 어댑터가 무선에서 유선 어댑터로 변경이 된다. 그리고 가상 어댑터에 유선 어댑터의 정보를 매핑한다. 결국 유선 어댑터를통해 통신을 수행하게 된다.

Fig.4를 참조하면, 유선 어댑터를 통해 통신을 수행하다가, DISCONNECT 이벤트가 발생하면,인터미디엇 드라이버는 DISCONNECT 된 어댑터가 유선 어댑터인지를 판단하고, 유선 어댑터인 경우에는 바인딩 핸들 리스트에 해당 어댑터의 연결 상태 정보를 업데이트한다. 이후 곧바로 무선 어댑터가 최적 어댑터로 선택되어 가상 어댑터에 무선 어댑터 정보를 매핑한다. 그리고, 무선 어댑터를 통해 통신을 수행한다.

DISCONNECT 된 어댑터가 유선 어댑터가 아닌 경우에는 유선 어댑터의 현재 커넥션 상태를 확인한다. 확인 결과 CONNECT 된 상태이면 유선 어댑터를 통해 통신을 계속 수행

한다. 그러나 확인 결과 DISCONNECT 된 상태이면, 상위 프로토콜 드라이버로 DISCONNECT 된 상태를 보고하고, 가상 어댑터에 유선 어댑터의 정보를 매핑하고 모두 DISCONNECT 된 상태가 된다.

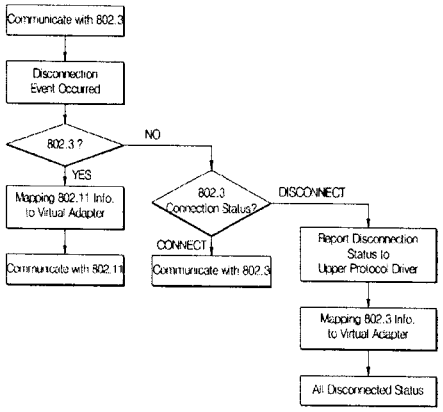

Fig.4. CONNECT 이벤트 시의 동작 흐름도

3. 실험 및 결과

새롭게 설계한 인터미디엇 드라이버가 설치된 컴퓨터에서 TCP를 사용하는 동작 특성이 각기 다른 애플리케이션을 실행하여 Correspondent Node와 통신을 하고 있는 상황에서 유/무선 어댑터간 핸드오프가 발생했을 때, 물리적인 핸드오프 시간을 측정하였다.

실험을 실시한 애플리케이션은 FTP, HTTP, TELNET 이렇게 세가지이다. FTP는 대용량의 데이터를 신속하게 전송하기 위해서 Payload의 사이즈가 크고, 일정시간 동안 많은 패킷을 생성한다. HTTP는 Page에 대한 Request가 있을 때 Home Page의 정보에 따라서 Payload의 사이즈와 패킷의 양이 가변적이다. 그리고, TELNET의 경우에는 터미널에서 명령을 내리고, 그 결과를 주고 받는 Text형태의 Interactive한 통신이므로 Payload 사이즈와 패킷의 양은 상대적으로 적은 편이다.

이 실험에 사용한 Timer는 Windows OS의 커널 레벨에서 제공하는 Tick Counter를 사용하였다. 이 Tick Counter의 1 Tick당 Time Resolution은 약 16.3 ms이다.

실험 결과는 Fig.5, Table.1에 나타난 그림과 같다. 결과를 분석하면 다음과 같은 두 가지 결론을 얻을 수 있다. 첫째는 유선 어댑터에서 무선 어댑터로의 핸드오프 시간이 무선 어댑터에서 유선 어댑터로의 핸드오프 시간보다 많게는 10배 가량 시간이 더 걸린다는 것이다. 두 번째는 패킷의 Payload 사이즈가 커지거나, 단위 시간당 패킷의 수가 많아 질 경우 핸드오프에 소요되는 시간 또한 증가하게 된다는 것이다. 첫번째와 같은 결과는 802.3 MAC Layer Protocol 보다 802.11의 MAC Layer Protocol이 일반적으로 유선보다 떨어지는 무선에서의 신뢰성을 보완하기 위해 Overhead가 더 많기 때문에 발생하는 결과로 분석된다 [6]. 두번째 결과의 이유는 Windows OS 내부의 스케줄링 규칙에 따라, 다른 Layer에서 패킷 사이즈가 크고, 많은 수의 패킷을 동시에 처리하기 위해서 리소스가 핸드오프 이외의 작업에 더 많이 할당되어 발생하는 현상이다.

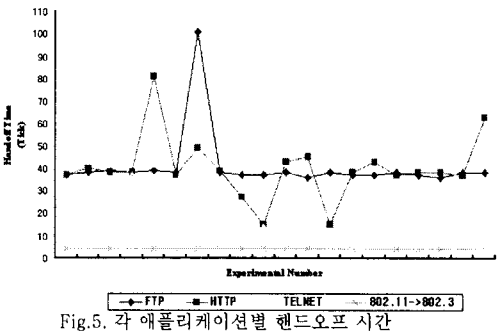

Fig.5. 각 애플리케이션별 핸드오프 시간

TABLE.1. 각 애플리케이션별 핸드오프 시간

	FTP	HTTP	TELNET
802.3 -> 802.11	40.75	39.9	19.35
802.11 -> 802.3	4	4	4

4. 결론

이상과 같이 본 논문에 의하면, 유선으로 연결된 상태에서 무선으로 핸드오프 하고자 할 때 하나의 가상 어댑터에 할당된 하나의 IP Address로 TCP/IP의 세션이 끊기지 않고 통신이 가능하다. 그리고 거의 Seamless한 핸드오프가 가능하다. 또한, 동일한 Layer 2 주소를 사용하므로 DHCP로 할당받은 IP Address도 통신의 연속성이 보장된다. 게다가 IPv6 환경에서도 Layer 2 주소를 가지고 IPv6 주소를 생성하는 Stateless Auto Configuration 환경에서도 아무 문제 없이 Seamless한 통신이 가능하다 [4][5].

이번 실험에서는 TCP를 사용하는 애플리케이션이 핸드오프에 영향에 대해서만 조사했다. 다음에는 TCP 이외의 Transport Layer를 사용하는 애플리케이션이 핸드오프에미치는 효과를 분석하고, 이에 효율적으로 대응할 수 있는 인터미디엇 드라이버를 보완 설계하도록 하겠다.

참 고 문 헌

[1] B. A. Forouzan, TCP/IP Protocol Suite, p.599-602, McGraw-Hill, First-Edition, 1999

[2] Driver Development Kit Help Documentation, Microsoft Developer Network, 2002

[3] G. R. Wright and W. R. Stevens, TCP/IP Illustrated Volume 2, p.803-805, Addison-Wesley, First-Edition, 1995

[4] R. Droms, Dynamic Host Configuration Protocol (DHCP), RFC, IETF, 1997

[5] S. Thomson and T. Narten, IPv6 Stateless Address Autoconfiguration, RFC, IETF, 1998

[6] M. S. Gast, 802.11 Wireless Networks, p.23-50, O'Reilly, First-Edition, 2002