A New Seamless Handoff Mechanism for Wired and Wireless Coexistence Networks

Pyung Soo Kim[1], Hak Goo Lee[2], and Eung Hyuk Lee[1]

[1] Dept. of Electronics Engineering, Korea Polytechnic University, Shihung City, 429-793, Korea
pskim@kpu.ac.kr
[2] Mobile Platform Lab., Digital Media R&D Center, Samsung Electronics Co., Ltd, Suwon City, 442-742, Korea

Abstract. This paper deals with design and implementation of seamless handoff mechanism between *wired* and *wireless* network adapters for a system with both network adapters. A unique virtual adapter is developed between different adapters and then an IP address is assigned to the virtual adapter. As a general rule, when both network adapters are available, the wired adapter is preferred due to its faster transmission speed than the wireless adapter. When wired communication via the wired adapter gets disconnected while in service, the disconnection of wired adapter is automatically detected and then wireless handoff occurs by mapping information on the wireless adapter to the virtual adapter. According to the proposed handoff mechanism, the session can be continued seamlessly even when handoff between wired and wireless network adapters occurs at lower level in a network application where both IP address and port number are used to maintain session. To evaluate the proposed handoff mechanism, actual experiments are performed for various internet applications such as FTP, HTTP, Telnet, and then their results are discussed.

1 Introduction

Most of today's computers are operating on Microsoft Windows systems which allow installations of multiple network adapters such as wired adapter and wireless adapter. In Windows systems, not only with different types of network adapters, but also with same type of network adapters, when one communication medium disconnects, all sessions of internet applications that are communicating through the corresponding adapter get disconnected automatically [1], [2]. It's because, under TCP, information on an adapter is stored in TCP control block (TCB), and once adapter disconnection is notified by TCB, TCP automatically cuts off the corresponding sessions. In other words, disconnection of an adapter means the IP address assigned to the adapter can no longer be used. Thus, when a handoff occurs from one adapter to another where different IP addresses are assigned to each network adapter, Windows systems don't support seamless handoff. It's because the session must be newly made with the

L.T. Yang et al. (Eds.): HPCC 2005, LNCS 3726, pp. 14–23, 2005.

IP address of the new network adapter, since the IP address assigned to the old network adapter cannot be used any more. To solve this problem, Windows systems provide "Bridge" function that allows multiple network adapters to be combined into one virtual adapter. However, although Bridge allows multiple network adapters to share same IP address, it has some shortcomings. Even with the Bridge, once a network adapter gets disconnected, the virtual adapter notifies protocol driver about disconnection. Thus, all application sessions using TCP/IP protocol driver automatically get disconnected. In addition, this function has too long handoff latency. It is observed via the experiment that handoff took about 30 seconds, which results in the timeout of TCP applications under Windows systems.

In order to solve the problems addressed above, this paper proposes a new seamless handoff mechanism between wired (IEEE 802.3) and wireless (IEEE 802.11) network adapters for a system with both network adapters. A unique virtual adapter is developed between different adapters and then an IP address is assigned to the virtual adapter. As a general rule, when both network adapters are available, the wired adapter is preferred due to its faster transmission speed than the wireless adapter. When wired communication via the wired adapter gets disconnected while in service, the disconnection of the wired adapter is automatically detected and then wireless handoff occurs by mapping information on the wireless adapter to the virtual adapter. Through the proposed handoff mechanism, since IP address does not change, the session can be continued seamlessly even when handoff between wired and wireless network adapters occurs at lower level in a network application where both IP address and port number are used to maintain session. Finally, to evaluate the proposed handoff mechanism, actual experiments are performed for various internet applications such as FTP, HTTP, Telnet, and then their results are discussed.

In section 2, the existing network driver system is briefly shown and its limitations are discussed. In section 3, a new seamless handoff mechanism is proposed for a system with both wired and wireless network adapters. In section 4, actual experiments are performed for various internet applications. Finally, in section 5, conclusions are made.

2 Limitations of Existing Network Driver System

There is a kernel level library released by Microsoft to allow Windows systems to have networking capability. This library is called Network Driver Interface Specification (NDIS) [3]. The normal NDIS is formed with miniport drivers and protocol drivers as depicted in Fig. 1. The miniport driver is used to run network adapters and communicate with the protocol driver. The protocol driver such as TCP/IP or IPX/SPX/NETBIOS services obtains binding handle through a binding process with the miniport driver. In the binding process, the protocol driver makes bindings with all working miniport drivers. Up to the protocol driver belongs to the operating system's kernel level, and applications and others belong to user level [4]. During booting sequence of Windows systems, NDIS

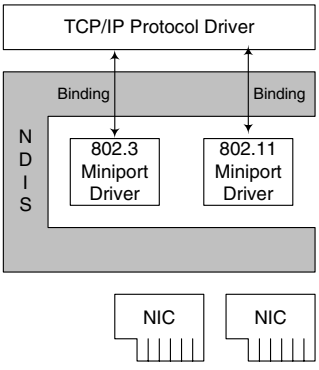

Fig. 1. Existing network driver system

initializes the miniport driver of registered adapter first. Then, as the protocol driver gets initialized, it binds with each miniport driver. Binding handle acts as a key used in transmission or reception of packets through corresponding adapter.

It is noted that the session must be newly made with the IP address of the new network adapter when a handoff occurs from the old network adapter to the new network adapter, since the IP address assigned to the old network adapter cannot be used in the new network adapter any more. Therefore, Windows systems using the existing network driver system of Fig. 1 cannot support the seamless handoff, because different IP addresses are assigned to each network adapter.

3 Proposed Seamless Handoff Mechanism

In order to solve the problems addressed in Section 2, this paper proposes a new seamless handoff mechanism between *wired* and *wireless* network adapters for a system with both network adapters.

3.1 New Network Driver System

Firstly, an intermediate driver is developed newly to modify packets sent from protocol driver then sends them to the miniport driver, and vice versa. The intermediate driver resides in between protocol driver and miniport driver, communicates with both miniport and protocol drivers as shown in Fig. 2. Note that the intermediate driver doesn't always exist, but exists only when it is needed by the developer. The intermediate drive generates the virtual protocol driver on the bottom and the virtual miniport drive on top. In other words, if we take a look at the virtual protocol drive and miniport driver on the bottom of the intermediate driver, then the intermediate driver works as the virtual protocol driver; and if we take a look at the virtual miniport driver and protocol driver on top of the intermediate driver, then the intermediate driver works as the virtual miniport driver. At this point, the virtual miniport driver is recognized by

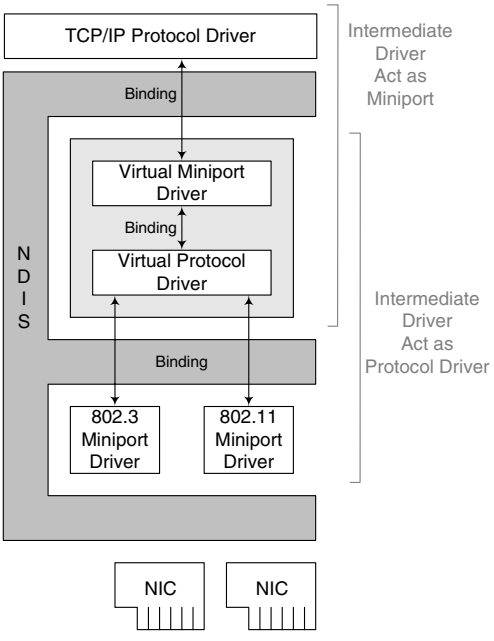

Fig. 2. New network driver system

actual upper level protocol driver as just another adapter [4]. The reason why the intermediate driver and the virtual drivers are in use is to solve some of the problems for the seamless handoff. Firstly, when there are two network adapters, each adapter must have different IP address, therefore, seamless communication is compromised during handoff. Due to this reason, a virtual adapter is used, whereas the actual protocol driver binds only with the virtual adapter to operate. This way, the layers above the protocol driver use the IP address assigned to this virtual adapter to communicate. Secondly, the intermediate drive can filter out data, which is sent to TCP/IP protocol driver in the event of a network adapter's network connection ends, to prevent session disconnection. Through this process, layers above the protocol driver do not know what is going in the lower level. Thirdly, use of the intermediate driver gives advantage of changing packet routing in real-time. By selecting the optimal adapter to be used for communication in accordance with the network connection status, packets can be transmitted and received through the optimal adapter. When this is realized, the handoff between wired and wireless network adapters can be done without disrupting the connection.

The intermediate driver in accordance with this paper can be divided into virtual miniport driver and virtual protocol driver as shown in Fig. 3. The virtual miniport driver includes virtual adapter control module. The virtual adapter control module generates a virtual adapter to control binding of an actual protocol driver to the virtual adapter. Furthermore, to set up each network adapter's property during the binding of the virtual protocol driver and a network adapter

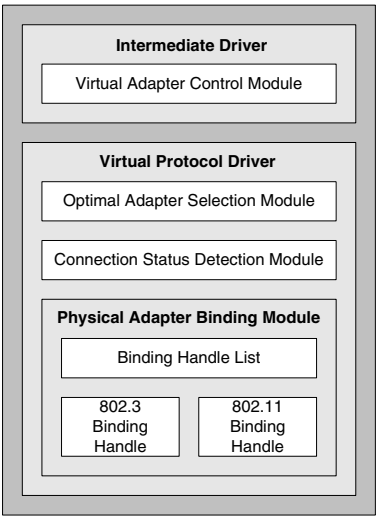

Fig. 3. Detailed diagram of the proposed driver system

at lower level, the virtual adapter control module sets all network adapters that are bound, to wireless adapter's link layer address. Here, under wired adapter, packet filtering property is set to promiscuous mode, which accepts all packets, and under wireless adapter, the property is set to direct/multicast/broadcast modes, which are typical. Hereinafter, the virtual adapter's link layer address is used as the wireless adapter's link layer address. In other words, under the wired adapter, since promiscuous mode is set, even though the virtual adapter's link layer address is set as the link layer address of the wireless adapter, corresponding packets can be transmitted and received. However, under the wireless adapter, since typical mode is used, the wireless adapter's link layer address is just used. Therefore, to upper protocol drivers, only one virtual adapter set with a link layer address is shown. Advantages from this are as follows. Firstly, by using the same link layer address, there is no need for retransmission of the address resolution protocol (ARP) packets to update ARP table in a hub [5]. Secondly, when assigning address dynamically using the dynamic host configuration protocol (DHCP), link layer address is used as one of options of DHCP protocol to identify the corresponding host. Here, by using the same link layer address again, DHCP server recognizes host as the same host and thus the same IP address gets assigned over and over [6]-[8]. Thirdly, when using link layer address in IPv6 (Internet Protocol version 6) environment assign IPv6 address through stateless auto configuration [9], [10], since the link layer address is the same, it is possible to assign an identical address.

When upper level protocol drivers request data of an adapter, the virtual adapter control module reports optimal adapter information selected by an optimal adapter selection module to an upper level protocol driver. Furthermore, the virtual adapter control module sends packet through binding handle of an optimal adapter chosen by the optimal adapter selection module. A virtual

protocol driver includes an optimal adapter selection module, a connection status detection module and a network adapter binding module. The optimal adapter selection module decides to which adapter to transmit or to receive in accordance with connection information of wired and wireless network adapter which is delivered from the connection status detection module. The connection status detection module detects connection status information of the wired and wireless network adapters, then provides the information to the optimal adapter selection module in real-time. The network adapter binding module generates binding handle list through binding with all active adapters. In the binding handle list, binding handle and status information for controlling bound network adapters get stored.

3.2 Seamless Handoff Mechanism Using New Network Driver System

As illustrated in Fig. 4, when a connection event occurs while all wired and wireless network adapters are in disconnection status, the connection status detection module is used to recognize which adapter is in connection status. Based on the connection status, connection information of the corresponding adapter that is in the binding handle list of the network adapter binding module gets updated. At first, this updated information gets sent to the virtual adapter control module in real-time; then, right away the virtual adapter control module reports the connection status to the upper level protocol driver. After that, the optimal adapter selection module determines whether the connected network adapter is wired or wireless, and if it is the wired connection, then a wired adapter gets selected and sent the information to the virtual adapter control module. If a

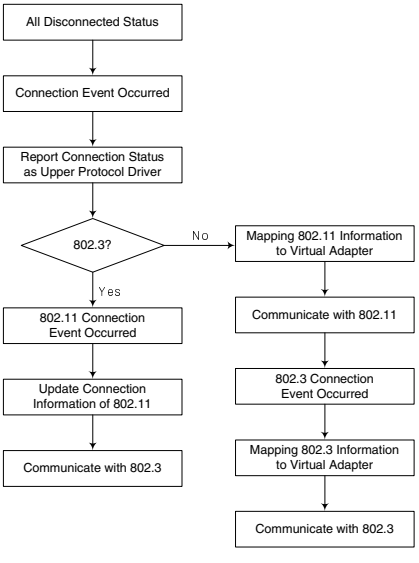

Fig. 4. Operation flow during connection event

wireless connection event occurs during wired communication, the connection status detection module recognizes connection of a wireless adapter, and this information updates only the connection information of the wireless adapter listed in the binding handle list of the network adapter binding module.

If a connection event of a wireless adapter occurs while all others are in disconnection status, the connection information obtained by the connection status detection module updates the connection status information of the corresponding adapter listed in the binding handle list of the network adapter binding module. Upon the updated information, the optimal adapter control module selects the wireless adapter as the optimal adapter. And then, the virtual adapter control module maps the information on the wireless adapter into a virtual adapter. Then, communication gets performed via the wireless adapter. If a wired connection event occurs during wireless communication, the information detected by the connection status detection module gets updated into the binding handle list and based on the updated information, the optimal adapter changes from the wireless to the wired by the optimal adapter selection module. Furthermore, the wired adapter's information gets mapped on to the virtual adapter. And at last, communication gets performed via the wired adapter. According to Fig. 5, when a disconnection event occurs during the wired communication, the connection status detection module determines whether the disconnected adapter is wired or wireless. If the adapter is the wired adapter, the corresponding adapter's connection status information gets updated to the binding handle list of the network adapter binding module. Then, the wireless adapter becomes the optimal adapter via the optimal adapter selection module, and the wireless adapter information gets mapped to the virtual adapter. Then, communication gets performed via the wireless adapter. If the disconnected adapter is not wired, the current connection status of the wired adapter must be verified. Upon the verification, if the wired adapter is still in connection, then communication is performed via the wired adapter. However, if the wired adapter is in disconnected status, the status gets reported to the upper level protocol driver, and the wired adapter's information gets mapped on to the virtual adapter.

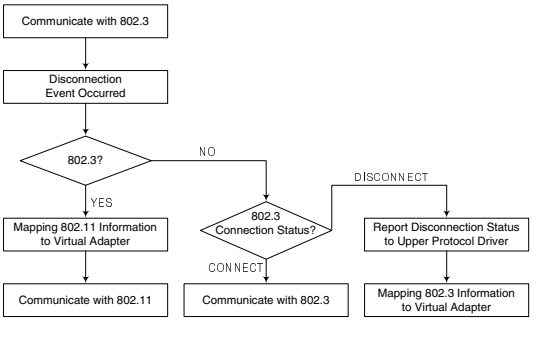

Fig. 5. Operation flow during disconnection event

4 Experiments and Results

To evaluate the proposed handoff mechanism, actual experiments are performed for various internet applications. The intermediate driver is realized on a portable PC using the driver development kit(DDK) [4] provided by Microsoft. The applications used in the experiment are FTP, HTTP, and Telnet. FTP supports large payload size and generates many packets for specific time in order to transmit large data, fast. HTTP's payload size and packet amount vary in accordance with amount of data each requested homepage provides. And for Telent, payload size and amount are relatively small since it supports interactive communication in text form.

In the experiment, the handoff latency is measured as follows. Firstly, a wired-to-wireless handoff occurs when wireless adapter takes over network connection as wired adapter loses the connection. And then, after mapping various information on the wireless adapter onto the virtual adapter, time it takes to transmit the first data packet has been measured. A wireless-to-wired handoff occurs when wired adapter connection is available during the wireless networking. And then, after mapping various information on the wired adapter onto the virtual adapter, time it takes to transmit the first data packet has been measured. Tick counter provided by Windows systems at kernel level is used as timer. Timing resolution per a tick of this tick counter is equivalent to approximately 16.3 *msec*. When testing each application, during data downloading FTP generated a handoff. For the case with HTTP, when downloading data linked to a homepage, handoff latency is measured during transmission of data via HTTP protocol. And for the case with Telnet, shell script to execute commands is made to run since packets can only be generated while sending or receiving commands through a terminal in Telnet.

Experimental results are shown in Fig. 6 and Table 1. Upon analyzing results, following two conclusions can be drawn. Firstly, wired-to-wireless handoff

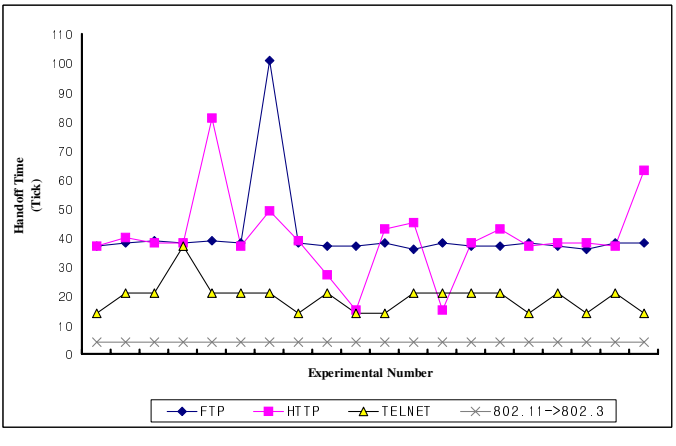

Fig. 6. Handoff Latency for Each Applications

Table 1. Mean value of handover latency

Handoff Direction	FTP	HTTP	Telnet
802.3 → 802.11	664msec	653msec	315msec
802.3 ← 802.11	25msec	25msec	25msec

latency takes as many as 10 times more than wireless-to-wired handoff latency, Secondly, a packet's payload size increases, or when packet amount per a unit time increases, handoff latency also increases. The first case is believed to happen because since 802.11 MAC protocol lacks reliability compare to 802.3 MAC protocol, some overhead had been added to 802.11 MAC protocol to overcome what it lacks [11], [12]. And the second case is believed to happen because packet size is large in different layers in accordance with Windows systems' internal scheduling rule; and in order to process many packets simultaneously, resource gets assigned to operations other than handoff. However, it is peculiar that a wireless-to-wired handoff remains constant regardless of the upper level application types. The reason for such behavior is believed to be happen because since reliability is always guaranteed for the wired adapter, and thus the impact the actual miniport driver, which drives the wired adapter, has on the actual throughput is insignificant since the driver is simple. Thus an analogy can be drawn here that the impact that the aforementioned overhead that 802.11 MAC protocol possesses is greater than that of resource sharing during a wired-to-wireless handoff. Therefore, the result, that nearly no impact is put on handoff even when packets are large, and resources being taken away to other layers due to the large number of packets, has been measured.

5 Conclusions

In this paper, the new seamless handoff mechanism between *wired* and *wireless* network adapters has been proposed for a system with both network adapters. The unique virtual adapter is developed between different adapters and then an IP address is assigned to the virtual adapter. As a general rule, when both network adapters are available, the wired adapter is preferred due to its faster transmission speed than wireless adapter. When wired communication via the wired adapter gets disconnected while in service, the disconnection of wired adapter is automatically detected and then wireless handoff occurs by mapping information on the wireless adapter to the virtual adapter. Through the proposed handoff mechanism, the session can be continued seamlessly even when handoff between wired and wireless network adapters occurs at lower level in a network application where both IP address and port number are used to maintain session. In order to evaluate the proposed handoff mechanism, actual experiments are performed for various internet applications such as FTP, HTTP, Telnet, and then their results are discussed.

In this paper, only the impacts of applications that are using TCP on handoff have been experiments. In the future, impacts that applications using transport

layer other than TCP have on handoff will be analyzed, and intermediate driver that efficiently corresponds to the impacts will be developed. Furthermore, the reasons for difference in amount of time it takes to handoff from a wired adapter to a wireless adapter in accordance with the characteristics of an application will be more carefully dealt with, and plan to design and develop an intermediate driver that gets less impacts from an application.

References

1. Forouzan, B. A.: TCP/IP Protocol Suite. McGraw-Hill (1999)
2. Wright, G. R. and Stevens, W. R.: TCP/IP Illustrated Volume 2. Addison-Wesley (1995)
3. Windows Network Data and Packet Filtering [Online], Available : `http://www.ndis.com` (2002)
4. Microsoft: Driver Development Kit Help Documentation. Microsoft Corporation, Redmond, WA (2002)
5. Stevens, W. R.: TCP/IP Illustrated Volume 1. Addison-Wesley (1994)
6. Droms, R.: Automated configuration of TCP/IP with DHCP, IEEE Internet Computing, **3** (1999) 45–53
7. Sun Microsystems, Dynamic Host Configuration Protocol (Whitepaper), (2000) 6–10
8. Park, S.H., Lee, M.H., Kim, P.S., Kim, Y.K.: Enhanced mechanism for address configuration in wireless Internet. IEICE Trans. Commun. **E87-B** (2004) 3777–3780
9. Thomson, S., Narten, T: IPv6 stateless address autoconfiguration, IETF RFC 2462 (1998)
10. Droms, R.: Deploying IPv6, IEEE Internet Computing, **5** (2001) 79–81
11. Gast, M. S.: 802.11 Wireless Networks. O'Reilly (2002)
12. ISO/ICE.: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. ANSI/IEEE Std 802.11 (1999)