Channel Bonding in Linux Ethernet Environment using Regular Switching Hub

Chih-wen Hsueh, Hsin-hung Lin and Guo-Chiuan Huang
{chsueh, lsh, hgc89}@cs.ccu.edu.tw
Real-Time Systems Laboratory
Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi, Taiwan 621, R.O.C.

Abstract

Bandwidth plays an important role for quality of service in most network systems. There are many technologies developed to increase host bandwidth in a LAN environment. Most of them need special hardware support, such as switching hub that supports IEEE Link Aggregation standard. In this paper, we propose a Linux solution to increase the bandwidth between hosts with multiple network adapters connected to a regular switching hub. The approach is implemented as two Linux kernel modules in a LAN environment without modification to the hardware and operating systems on host machines. Packets are dispatched to bonding network adapters for transmission. The proposed approach is backward compatible, flexible and transparent to users and only one IP address is needed for multiple bonding network adapters. Evaluation experiments in TCP and UDP transmission are shown with bandwidth gain proportionally to the number of network adapters. It is suitable for large-scale LAN systems with high bandwidth requirement, such as clustering systems.

Keywords: IEEE 802.3, channel bonding, Link Aggregation, LAN, bandwidth.

1. Introduction

Bandwidth is critical in most network systems, especially for those with large data and communication transmission, for example, clustering and parallel systems. There are many products developed to increase host bandwidth in a LAN environment, such as CISCO EtherChannel [1], Intel Link Aggregation [2], Sun Trunking [3], Linux Bonding [4], etc. Although bandwidth can be increased using these approaches, most of them need special hardware support which means extra costs are needed. For example, a switching hub with Link Aggregation Standard [5] support is few times more expensive than a regular one.

Nowadays, low-end network adapters and switching hubs are very inexpensive. They support 100BASE-TX and Full-Duplex and still have very high filtering/forwarding rates. Network bandwidth of a host can be easily increased by using multiple network adapters simultaneously. However, one IP address is needed for each adapter so that it is not practical in large-scale systems. In this paper, we present an approach to make use of capability of regular switching hubs to increase the bandwidth of connected hosts.

The proposed approach is implemented as two Linux kernel modules [6], [7], [8], [9] between IP layer [10], [11], [12] and physical Ethernet interface drivers. One module creates a pseudo Ethernet device when loaded and bonds multiple physical Ethernet interfaces. It receives packets from IP layer and dispatches them to the underlying physical Ethernet interfaces. The other module maintains an IP/MAC addresses table, which contain mappings between each IP address and its corresponding MAC addresses of multiple bonding physical Ethernet interfaces in a LAN environment. When the pseudo device transmits a packet, it queries the IP/MAC addresses table and changes the source and destination MAC addresses of the packet accordingly. The host bandwidth of multiple network adapters is merged within a single IP address. The bandwidth is increased proportionally to the number of bonding network adapters without any modification to hardware and operating system.

Supported in part by a research grant from the ROC National Science Council under Grants NSC-89-2213-E-194-056

The rest of this paper is organized as follows. The next section describes background knowledge of packet handling. Section 3 details the design and implementation of the proposed approach in a Linux LAN environment. In Section 4, we measure and analyze the results of the proposed approach. This paper is concluded in Section 5.

2. Background

In order to have a clear view of how packets are handled before transmission, we introduce the packet handling in a switching hub and Linux network traffic control in this Section.

Packet Handling in Switching Hub

The Ethernet switch controller in a switching hub controls the flow of input packets. We take RealTek RTL8308B controller as an example to illustrate the packet handling in a switching hub. RealTek RTL8308B is an 8-port 10/100Mbps Ethernet switch controller [13]. It can operate in full-duplex mode and supports non-blocking 148800 packets/second wire speed forwarding rate and IEEE 802.3x flow control. The brief architecture of RealTek RTL8308B is shown in Figure 1. RealTek RTL8308B has a built-in 2MB bits DRAM as packet storage buffer. When a packet is coming, the received data flows into FIFO queue first and then is moved into Packet Buffer by the DMA Engine. The address look-up table consists of 8K entries of hash table and 128 entries of CAM. When a packet is received from a port, the switching logic hashes the destination MAC address to get a location index to the 8K-entry hash table and at the same time compares the destination MAC address with the contents of the 128-entry CAM. If a valid location index is found in the hash table or the CAM comparison is match, the received packet is forwarded to the corresponding destination port. Otherwise, the packet is broadcasted to all ports and the switching logic hashes the MAC address to get a location

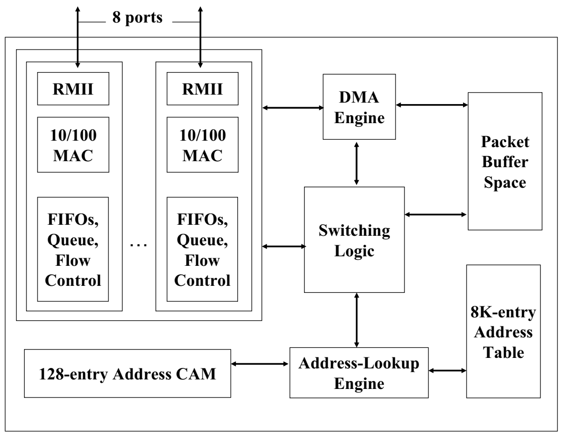

Fig. 1. **Brief Architecture of Switching Hub (RealTek RTL8308B)**

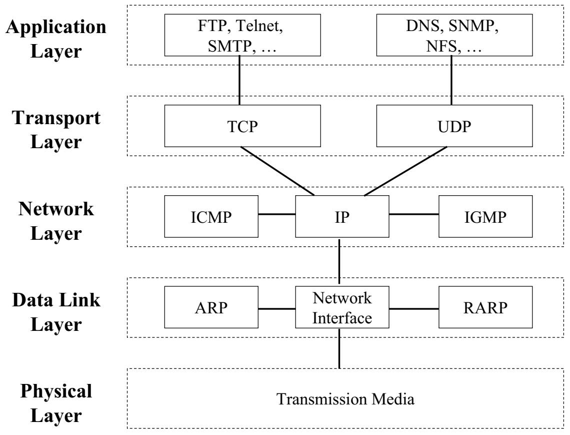

Fig. 2. **Linux Network Protocol Stack**

index to the hash table. If a hash collision occurs, the MAC address will be put into the 128-entry CAM.

Packet Flow in Linux Protocol Stack

As shown in Figure 2, Linux protocol stack is based on TCP/IP and is normally considered as a 4-layer system [10]. In Linux, a packet is represented by a common data structure called socket buffer structure throughout all protocol layers. By this way, parameters and payloads of a packet would not need to be copied between different protocol layers. Figure 3 illustrates the abstraction of the traffic path in Linux. A packets is first sent from an application to the transport layer (TCP or UDP layer) through a socket when the application sends some data. After the packet is handled in transport layer, it is then sent to the network layer (IP layer). The route of packets is determined in the network layer. If the destination of the packet is another computer, the network layer sends it to the data link layer. The data link layer sends packets via an available output device which is usually a network adapter.

When a packet is arrived, the input interface checks whether the packet is for this computer, for example, a network adapter checks the destination MAC address field when receiving a packet. If so, the network interface driver sends the packet to the network layer. The network layer checks the destination of the packet. If the packet is for this computer, the network layer sends it to the transport layer and finally to the application. Otherwise, the packet is sent back to an output device.

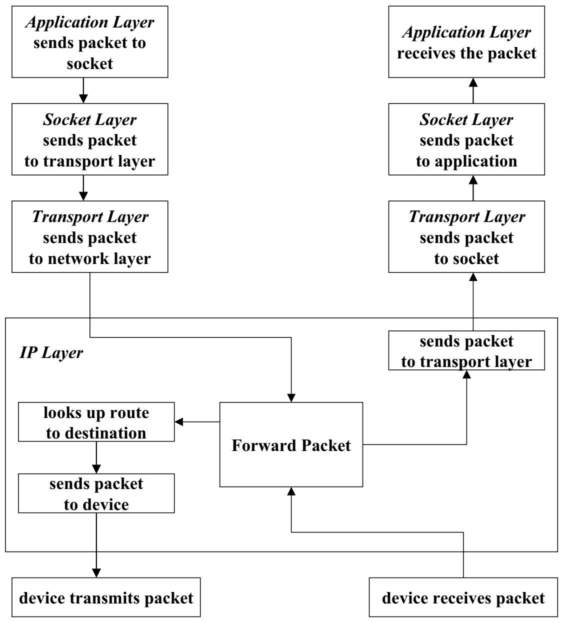
Application Layer sends packet to socket
Socket Layer sends packet to transport layer
Transport Layer sends packet to network layer
IP Layer
looks up route to destination
sends packet to device
Forward Packet
sends packet to transport layer
device transmits packet
device receives packet
Application Layer receives the packet
Socket Layer sends packet to application
Transport Layer sends packet to socket

Fig. 3. **Linux Traffic Path**

3. Design and Implementation

The proposed approach is implemented as two Linux kernel modules, BondingPlus and ARP+. BondingPlus is responsible for dispatching packets to bonding network adapters. ARP+ maintains a one-to-many mapping table between an IP address and MAC addresses of bonding network adapters in a LAN environment.

BondingPlus

As shown in Figure 4, BondingPlus is a pseudo Ethernet driver which resides between IP layer and physical Ethernet interface drivers. When the pseudo driver is loaded into kernel, it creates a pseudo Ethernet device and registers it to kernel. We assign the pseudo device an IP address and a netmask as a usual network device. We can bond multiple physical Ethernet interfaces as slave devices of the pseudo device to the Ethernet interface pool (slave list) created by BondingPlus module. The MAC address of the pseudo device is set as the MAC address of its first slave device. Therefore, network packets which are transmitted by physical Ethernet interfaces are considered as transmitted by the pseudo Ethernet device. The pseudo device is to dispatch packets from IP layer to multiple physical Ethernet interfaces in its slave list. It is responsible for modifying the attributes of a socket buffer, including source MAC address, destination MAC address and output device. Then, BondingPlus finds an active physical Ethernet interface from its slave list to send out the socket buffer and finds an active physical Ethernet interface of the receiving host to receive the packets. The selection of output network adapter of the sending host and the input adapter of the receiving host is implemented in a round-robin manner. Other scheduling algorithms can be adopted for additional purposes, such as sending real-time packets[14].

ARP+

When a socket buffer is to be sent on the pseudo network driver, the kernel will fill the destination MAC address obtained by the ARP protocol to the Ethernet header of this socket buffer. Without modification to the ARP protocol, the bandwidth of the

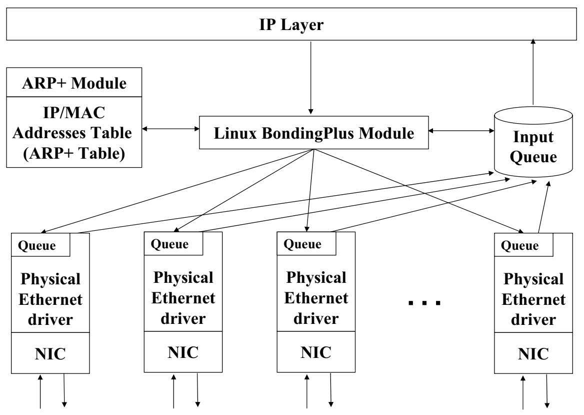

Fig. 4. **Pseudo Ethernet Driver**

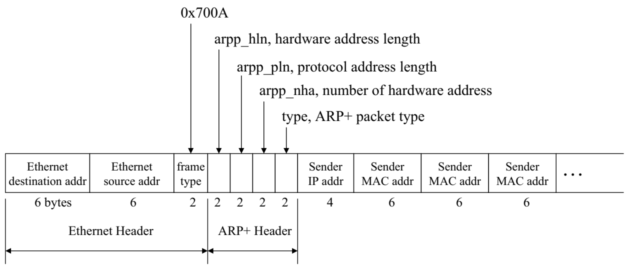
0x700A
arpp_hln, hardware address length
arpp_pln, protocol address length
arpp_nha, number of hardware address
type, ARP+ packet type
Ethernet destination addr
Ethernet source addr
frame type
Sender IP addr
Sender MAC addr
6 bytes
6
2
4
Ethernet Header
ARP+ Header

Fig. 5. **ARP+ Protocol Packet Type**

multiple slave Ethernet interface will not be merged since ARP protocol is an one-to-one mapping between an IP address and its corresponding MAC address so that we always obtain the same MAC address of a designated IP address from the ARP protocol. This means that although we can send out packets through multiple physical Ethernet interfaces, but we always send packets to the same Ethernet interface of a receiving host. In other words, although the outgoing bandwidth of a sending host can be increased, the incoming bandwidth of a receiving host is not increased.

To solve this problem, we design a APR+ protocol [14], through which a sending host knows all the MAC addresses of network adapters bonding to a designated IP address of a destination host. The pseudo Ethernet driver maintains a table (ARP+ table) of mappings between each IP address and its corresponding MAC addresses of multiple physical Ethernet interfaces in a LAN environment using the ARP+ protocol. As shown in Figure 5, we design a proprietary packet (ARP+ packet) which can only be understood and interpreted by the proposed pseudo Ethernet driver without interfering other existing protocols. When a host is loaded with the pseudo Ethernet driver, it broadcasts a $ARPP\_BROADCAST$ packet containing all the MAC addresses of its physical Ethernet interfaces. When a host receives the broadcasted packet, the pseudo Ethernet driver unicasts a $ARPP\_REPLY$ packet to notify the newly joined host with all its MAC addresses. Thus, a newly joined host is able to obtain the MAC address lists of all other hosts in the same LAN environment. An $ARPP\_CHANGE$ or an $ARPP\_CLEAR$ packet is broadcasted to notify other hosts to modify or clear the entry of their ARP+ table respectively. Since the mapping table is only created once and updated when network configuration is changed, the broadcasting overhead is negligible.

TABLE I
TEST BED

System Parameters	Settings
CPU	Intel Celeron 1.2GHz
Memory	256MB
Operating System	Mandrake 8.1
Kernel Version	2.4.18
Network Adapter	Intel 21143, 82559
Switching Hub	DLink DES-1024R+

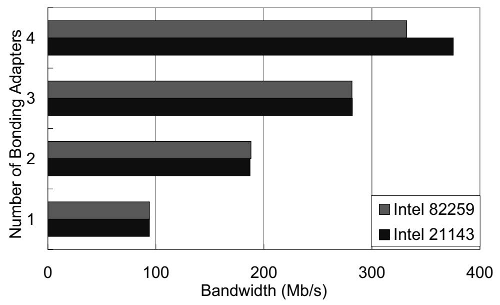

Fig. 6. **TCP Transmission**

Backward Compatible

The proposed approach is transparent to users and is backward compatible to hosts without channel bonding. If a host is not loaded with the proposed modules, which means it can not interpret ARP+ packet and will not reply so that there will be a null entry in the ARP+ tables for other hosts with channel bonding. However, the MAC address can still be obtained through the original ARP protocol. Hosts with channel bonding are able to send packets to the one without it. Moreover, since the MAC address of the pseudo Ethernet device is set as the MAC address of the first physical Ethernet interface in its slave list. The host without channel bonding is able to obtain the MAC address of hosts with channel bonding using ARP protocol and sends packets to them. Consequently, the bandwidth of the host without channel bonding is not increased. In order to avoid the bonding physical Ethernet interfaces to reply an ARP request, which may confuse other hosts, NOARP flag is set to all slave devices and the ARP request is only replied by the pseudo Ethernet device.

Performance Evaluation

We perform the following experiments to evaluate the bandwidth improvement when multiple network adapters are bonded into a system. The experiment environment contains two hosts whose system parameters are listed in Table 1. Two hosts were directly connected to the switching hub with four network adapters respectively.

Two types of network adapters, Intel 21143 and Intel 82559, are tested in both TCP and UDP transmission in a LAN environment. The drivers of Intel 21143 and Intel 82559 network adapters used in the experiments are ”tulip” and ”eepro100” respectively which can be found in Linux kernel source tree. The performance is measured by Netperf [15] package, a networking performance benchmark, using a client/server architecture. The client program of Netperf runs on a host generates packets to the server program on the other host. As shown in Figure 6 and Figure 7, the bandwidth of TCP and UDP transmission is increased proportionally to the number of bonding adapters. The bandwidth of

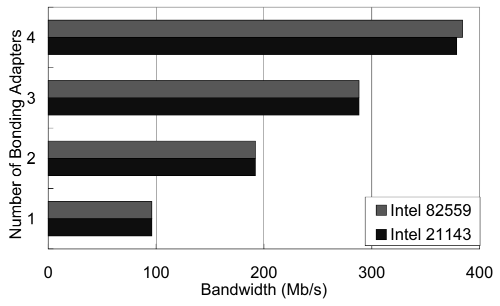

Fig. 7. **UDP Transmission**

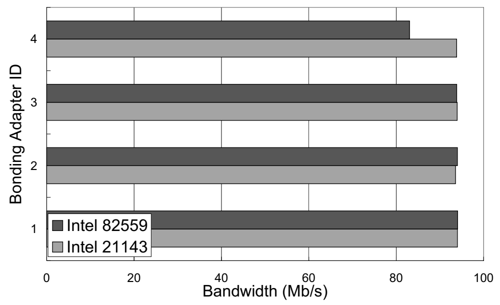

Fig. 8. **Load Balancing in TCP Transmission**

UDP transmission is slightly higher than TCP because of its less overhead.

We measure the bandwidth of each bonding adapters using Netperf. Figure 8 and Figure 9 show the results of the experiments. The bandwidth contribution of each adapters is almost equivalent both in TCP and UDP transmission. Outgoing and incoming network packets are equally distributed on all bonding adapters and thus load balancing on each bonding adapters is achieved in the proposed approach.

We evaluate the overhead of the proposed approach by performing the following experiments. We measure network and CPU utilization with one adapter on each host. We compare the results of the proposed approach with the one without channel bonding. We first measure the bandwidth of the adapter using Netperf. As shown in Table 2, the proposed approach only decreased the bandwidth of a adapter by 0.05%. Besides, we send and receive packets for 60 seconds using Netperf and measure the overhead on CPU utilization. The results are shown in Table 3. Our approach only increases 6.6% of CPU time in average. Therefore, the overhead of the proposed approach is little and can be neglected.

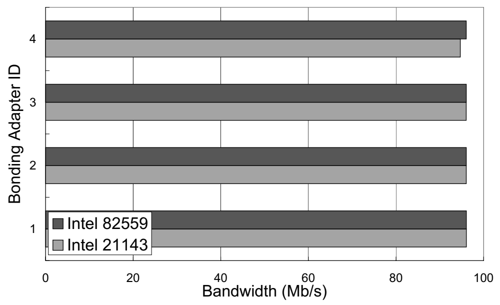

Fig. 9. **Load Balancing in UDP Transmission**

TABLE II
BANDWIDTH OVERHEAD

BondingPlus	Without	With
Bandwidth	94.05Mb/s	94.01Mb/s

TABLE III
SYSTEM UTILIZATION OVERHEAD

BondingPlus	Without	With
Minimum User Time	0s	0s
Maximum User Time	0s	0s
Minimum System Time	2.11s	2.33s
Maximum System Time	2.32s	2.49s
Average System Time	2.25s	2.40s

4. Conclusion

We proposed a new inexpensive approach to increase bandwidth of hosts connected to a regular switching hub in a LAN environment. Packets can be sent and received via multiple bonding network adapters with only one IP address. There is no modification to hardware and operating systems. The proposed approach is implemented as two Linux kernel modules and is backward compatible, flexible and transparent to users. An one-to-many mapping table of an IP address and its corresponding MAC addresses of multiple bonding network adapters in the LAN environment are kept by the ARP+ module in order to obtain the MAC addresses of a designated IP address. Packets from IP layer are dispatched by the proposed BondingPlus module to its underlying bonding network adapters and received by destination host in a round-robin manner. Bandwidth of a host can be increased proportionally to the number of bonding network adapters. It is suitable for large-scale systems in a LAN with high bandwidth requirement, such as clustering systems.

References

[1] “Etherchannel,” CISCO SYSTEMS. [Online]. Available: http://www.cisco.com/en/US/tech
[2] “Intel link aggregation,” Intel Corporation. [Online]. Available: http://www.intel.com/support/express/switches/53x/31460.htm
[3] “Sun trunking,” Sun Microsystems. [Online]. Available: http://wwws.sun.com/products-n-solutions/hw/networking/connectivity/suntrunking
[4] T. Davis, *Linux Bonding*. [Online]. Available: http://sourceforge.net/projects/bonding
[5] *IEEE 802.3 CSMA/CD Access Method*, IEEE Std., 2000. [Online]. Available: http://www.ieee802.org/3
[6] T. Aivazian, *Linux Kernel 2.4 Internals*. [Online]. Available: http://www.tldp.org/LDP/lki/index.html
[7] D. P. Bovet and M. Cesati, *Understanding the LINUX KERNEL*. O’REILLY, 2001.
[8] J. Crowcroft and I. Phillips, *TCP/IP and Linux Protocol Implementation*. WILEY, 2002.
[9] A. Rubini and J. Corbet, *LINUX DEVICE DRIVERS: Second Edition*. O’REILLY, 2001.
[10] R. Stevens, *TCP/IP Illustrated Volume 1*. Addison-Wesley Pub Co, 1994.
[11] G. R. Wright and W. R. Stevens, *TCP/IP Illustrated Volume 2*. Addison-Wesley Pub Co, 1995.
[12] D. Comer, *Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture*. Prentice Hall, 2000.
[13] “The rtl8308b specifications,” REALTEK CORPORATION. [Online]. Available: http://www.realtek.com.tw/downloads/downloads1-3.aspx?Keyword=rtl8308
[14] H. hung Lin, C. wen Hsueh, and G.-C. Huang, “Bondingplus: Real-time message channel in linux ethernet environment using regular switching hub,” in *The 9th International Conference on Real-Time and Embedded Computing Systems and Applications*, Feb. 2003.
[15] R. Jones, *Netperf*. [Online]. Available: http://www.netperf.org/