The FPGA implementation of the RC-DBA algorithm in the EPON network

Jong-wook Jang, Hyun-jin Kang, and Hyoung-goo Jeon,

University of Dong-eui, Korea

Summary
In the upstream link of the EPON network, numerous ONUs receive the privileges to use the optical medium from the scheduler of the OLT, but not through the competition with others. Therefore, it is very important to select a proper DBA algorithm to allocate the band to each ONU in an effectively and fair manner. In our preceding study, we proposed the RC-DBA algorithm that complements many problems in existing DBA algorithms. In this paper, we designed the MAC scheduler for the OLT, which the proposed algorithm was applied to and implement it in the FPGA. In addition, in order to verify the operation of the scheduler, we developed the embedded Linux based testbed.

Key words:
EPON, RC-DBA algorithm, FPGA, Embedded Linux system

Introduction

The EPON network employs a point-to-multipoints architecture where many ONUs share the same optical medium while they are all connected to the same OLT. So, as shown in Figure 1, when the ONU transmits the data to the OLT, no collisions should occur between each ONU and every ONU should be given an equal privilege to access the medium.

In IEEE802.3ah EFM, the MPCP is standardized as a MAC protocol for the EPON but the packet scheduling algorithm, which is the essence of the dynamic frequency band allocation is excluded from the standardization, so that it can provide more flexibility for the EPON service provider. Many types of packet scheduling algorithms that have been previously proposed have problems as they do not guarantee the QoS that considers the traffic priority or support the fairness between each ONU. Hence, in the article [1], we proposed a new scheduling algorithm called RC-DBA that complements such shortcomings.

This study is an extensive research effort stretched out from [1]. Its purpose is to design the MAC scheduler in the OLT by applying the proposed RC-DBA algorithm to it and to develop the device driver and the test application program to verify the proposed algorithm. The design of the RC-DBA based scheduler is implemented in the LDS-2000 FPGA ver1.0 system by Corebell, which is composed of the embedded board equipped with the Intel PXA255 processor and the board with the built-in Cyclone EP1C12F324C8 FPGA chip.

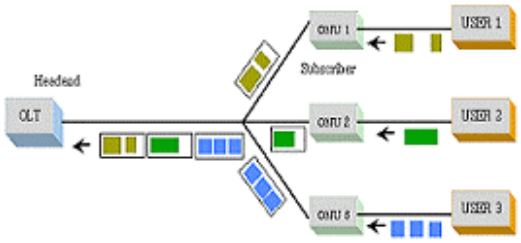


Fig. 1 The transmission of upstream in the EPON

The organization of this paper is as follows. In Chapter 2, we will describe the RC-DBA algorithm, which is the essence of the MAC scheduler as we wish to summarize the progress of our study up until now. In Chapter 3, we will begin to deal with the design of the scheduler more aggressively and in Chapter 4, we will investigate the implementation of the CPU interface logic used to interconnect with the test program and to implement the FPGA. In Chapter 5, we will examine the development and the execution of the device driver and the application program used to verify the operation of the scheduler. Finally, in Chapter 6, we will draw the conclusion in this paper.

2. Request Counter-DBA algorithm

In general, the DBA algorithm must satisfy the functions including the capability to control many priority queues for supporting the QoS, and to fairly allocate the bandwidth to each ONU as well as providing the compact algorithm to minimize the delay of the frame. The existing DBA algorithms[2, 3, 4, 5, 6] show excellent performance in supporting many priority queues and minimizing the frame delay but they do not support the function for the fair band allocation between each ONU. Therefore, in this study, we pointed out the problems in the existing ETRI's DBA algorithm[2] and proposed the new algorithm to supplement it.

Manuscript received June 5, 2006.
Manuscript revised June 25, 2006.

The ETRI's DBA algorithm provides each ONU with the bandwidth in proportional to net bandwidth for the upstream link according to the traffic priority. In this case, if a certain ONU is requesting the bandwidth larger than what is requested by other ONUs, then other ONUs may not receive the sufficient bandwidth. In order to resolve such a problem, the RC-DBA algorithm assigns a weight to each ONU having the same priority and allocates the upstream link bandwidth to each ONU in the increasing order of the its weighted priority[7].

The scheduler in this paper assigns weights based on the following two criteria. First, it adds the total number of ONUs to the weight of each ONU that requested the bandwidth. Second, it assigns the weights to ONUs in the increasing order of the amount of bandwidth that they requested. For example, if a total of 16 ONUs exist and only 10 of them requested the bandwidth, then the ONU, which requested the largest amount of bandwidth, increases its weight by 10. Similarly, the remaining 9 ONUs add 9, 8, 7 … and 1 to their weights. As described above, in the first stage, the total number of ONUs is added to the weight. If a ONU requested the upstream link bandwidth but did not receive it, then it should be given a higher priority than other ONUs that did not request the bandwidth. So, if its weight is just increased by 1, then not every ONU can receive a fair amount of bandwidth that it deserves[8].

3. Design of the MAC Scheduler

In this paper, in order to make the hardware, we used a design method using the hardware technical language rather than the circuit diagram. In general, the techniques for a digital system design are achieved through the technology of the design specification, the ASM chart, the data path design and the ASM chart changed by control logic[9]. From now on, we would like to show you how to design the MAC scheduler based on these procedures.

3.1 Specification

In order to design a digital system, first you need to determine the basic design specification for the hardware to be manufactured. Especially, the input/output data format of communication systems is the most important factor determining the system performance. Basically, since the MAC scheduler in the OLT allocates the bandwidth using the status information of the priority queues of ONU and it has to send the allocation information for the bandwidth to the ONU again, it uses the report and gate message in the MPCP MAC control frame. The MPCP protocol in an actual EPON employs a 64 byte frame[10] structure. In this paper, for the convenient implementation, the field not related to the operation of the scheduler is eliminated. Figure 2 represents a reduced format of the report and gate message and its size is 48 bits.

DA (4)	SA (4)	Opcode(00-02) (16)		High grant (8)	Middle grant (8)	Low grant (8)
DA	SA	Q_NUM (8)	Bitmap (8)	High report	Middle report	Low report

Fig.2 Format of the gate and report frame

3.2. Design by ASM chart

When the report message sent by the ONU arrives at the OLT, then the internal scheduler allocates the upstream link bandwidth to each ONU according to the RC-DBA algorithm. Once the grant information is calculated, it will be added to the gate message and then transmitted to each ONU. If the overall operation of the scheduler is defined as above, the corresponding operation of the module to be implemented can be divided into the following four categories. They include the input of the report message, the weight calculation according to the request for the priority queue, the bandwidth allocation and finally, the gate message output. When the start signal is turned on in the initial status, the report message input to the system will be initiated and when the outputs from the ONU are completed, the control circuit will apply the value of 1 to the done signal to terminate the operation of the system. Now, let's take a close look at the operation of each part through the ASM chart.

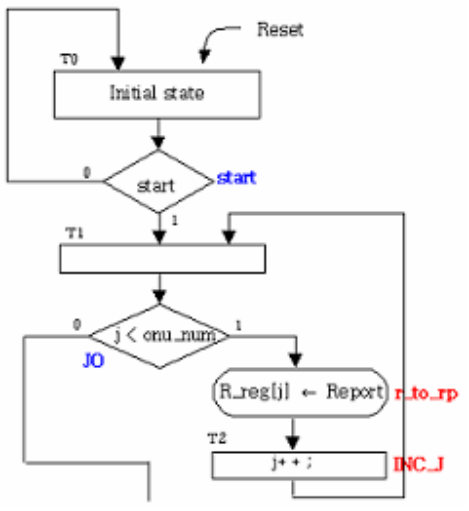


Fig.3 Input of the report message

3.2.1. Input of the Report frame

Figure 3 represents the ASM chart corresponding to the report message input. When the start signal is received in the initial status(T0), it receives as many report message as the number of ONU supported by the OLT. It sorts out the request information for each priority queue in the report message and stores it in a separated register called "length"

3.2.2. Computation of the weight

Figure 4 is the ASM chart corresponding to the weight calculation in the RC-DBA algorithm. The weight of the ONU increases depending on whether the request information from T9 to T10 exists. When the update_cnt signal is generated, the weight is increased by the total number of ONUs. T11 through T14 increases the weight of ONUs in the increasing order of the requested amount. The m_to_tl signal is used to search for the ONU with the largest requested amount and the ccts signal is used to increase the weight.

The weight is calculated for each ONU with respect to its traffic priority queue. The use of weight is not necessarily applied to the high priority corresponding to the fixed and assured bandwidth, but in this paper, it is applied to every level of priority.

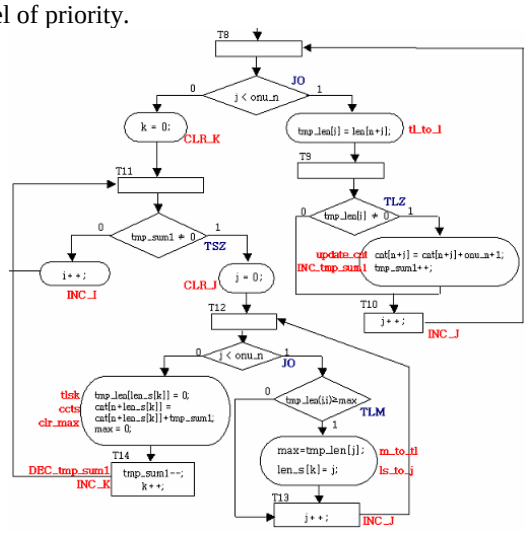


Fig.4 Computation of the weight

3.2.3. Allocation of the bandwidth and Gate frame

When the weight calculation is completed, the bandwidth allocation in proportional to the result from the weight calculation will be performed. Figure 5 is the ASM chart corresponding to such channel allocation. The bandwidth allocation will start with the queue with the highest priority and the entire amount requested by this queue will be permitted. The middle and low priority allocates the half of the remaining bandwidth to each ONU and this also guarantees the best effort bandwidth request. When the bandwidth allocation is completed, the gate message is created by the make_grant signal.

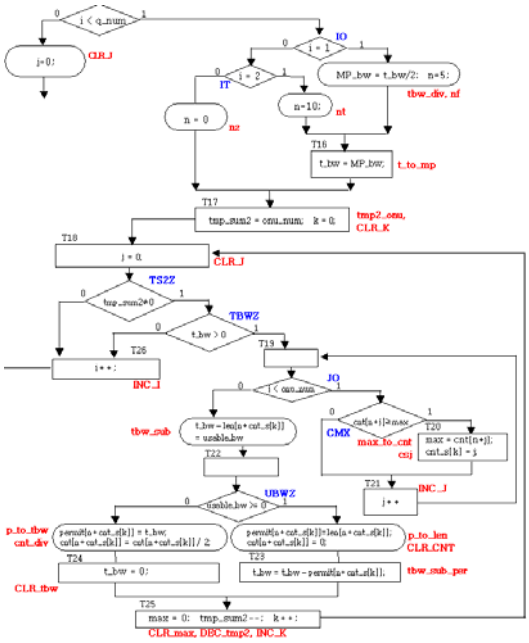


Fig.5 Allocation of the bandwidth

4. The implementation of the FPGA

When the ASM chart is prepared, the hardware should be designed based on the ASM chart using verilogHDL. The designed module creates the test vector and verifies the function in the simulation tool. After that, the module synthesizes it into the gate level using the synthesizer tool.

4.1. Simulation

In the simulation, the following conditions are used.

① The total number of ONUs : 5 units
② The number of priority queues that can be supported : 3 units
③ The number of time slots than can be allocated each time : 15 units.

As shown in Table 1, when the report message is entered into the test vector, the gate message as shown in Table 3 should be generated by the RC-DBA algorithm. If you calculate the weight, then you will have the result as shown in Table 2. The method for assigning the weight is applied to the queues with the same priority. For the high priority queues, the total number of ONUs (that is 5) is added to

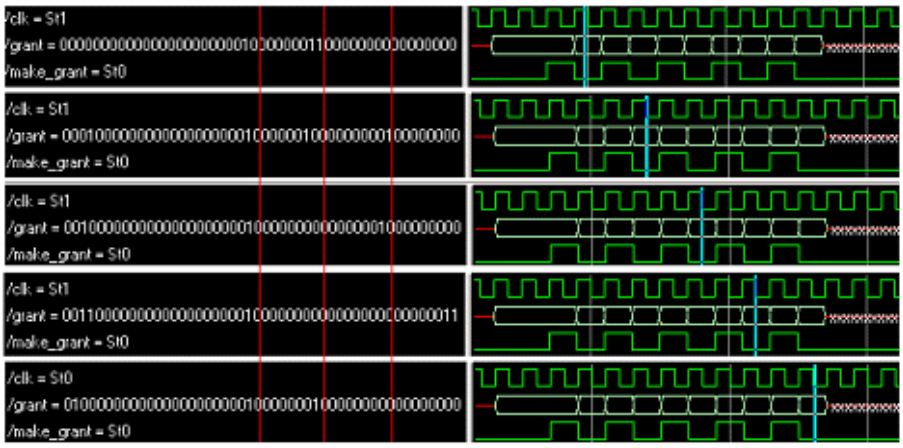


Fig.6 Results of gate messages

ONU1, ONU2 and ONU5 depending on whether they made a request or not. And, the queues with lower priorities, namely, ONU2, ONU1 and ONU5 are increased by 3, 2 and 1, respectively. For the middle and low priorities, the same method is used to calculate the weight. Figure 6 is the resulting waveform corresponding to the case when the input for the test vector in Table 2 is used in Modelsim EE/Plus 5.2c and the result is same as given in Table 3.

Table.1 Number of the request for timeslot

	onu1	onu2	onu3	onu4	onu5
high	3	4	0	0	2
mid	2	2	2	0	0
low	0	0	5	6	0

Table.2 Weight values

	onu1	onu2	onu3	onu4	onu5
high	7	8	0	0	6
mid	6	7	8	0	0
low	0	0	6	7	0

Table.3 Results of timeslot allocation

	onu1	onu2	onu3	onu4	onu5
high	3	4	0	0	2
mid	0	1	2	0	0
low	0	0	0	3	0

4.2. Design of the CPU interface logic and hardware synthesis

The accurate operation of the scheduler designed using verilogHDL has been verified. In this paper, we will examine whether the same result is obtained in the actual hardware by implementing it on the FPGA chip.

As mentioned in introduction, the scheduler module in the FPGA is controlled by the embedded Linux based device driver and the application program. For this purpose, the CPU interface logic has been added. The interface logic is a group of registers storing the data received from the CPU. The scheduler uses the report and gate messages with the size of 48 bits. Since the input/output bus between the LDS2000 embedded system and the FPGA board is 32 bits, in order to receive one report message with the size of 48 bits, the device driver partitions into three 16bit data and sends each of them to the interface logic. The interface logic combines them into the 48bits report data to be used by the scheduler module. The similar procedure is applied to the gate message and the CPU, which is the receiving end, combines the received data. Figure 7 illustrates the interface control signals necessary for the input and output data between the PXA255 CPU and the FPGA.

The MAC scheduler of the OLT that includes the CPU interface logic synthesized into the gate level using Quartus II, a synthesizer tool exclusively for the FPGA. The synthesized scheduler is configured and wired as a logic cell using 24% (2,938/12,060) of the Cyclone EP1C12F324C8 FPGA and the register uses 10%(1,349/12,795) while the actual value of fmax is 62.03MHz.

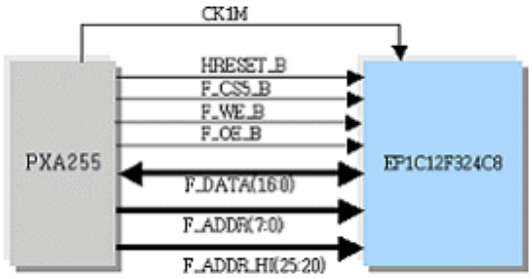


Fig.7 Interface control signals between the PXA255 and the FPGA

5. The development of the embedded linux based testbed

If a user wishes to handle the desired task using a specific device, he should first manufacture the device driver related to the equipment and then prepare the allocation program to perform the desired task. That is, the device driver should execute the data input/output process and the allocation program should specify the processing task that the user desires eventually[11].

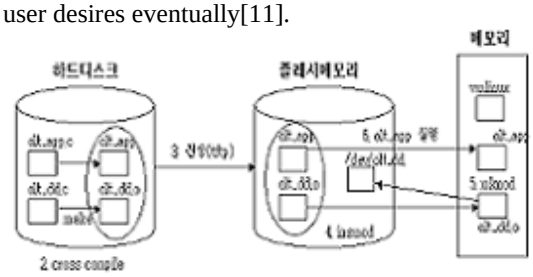


Fig.8 Operational procedure

Figure 8 illustrates the operational procedure from compiling of the application program to its execution. Since the embedded system, due to its limited characteristic, may not support the development environment for the program by itself, the program should be written and complied on a general Linux PC. The interior of the hard disk in the figure represents the procedure where the device driver, olt_dd.c and the application program olt_app.c are written on a general Linux PC and the object and execution files for the embedded Linux are created through the cross compiler. The module and the execution file that are created are sent to the embedded system which will be used by the user. In order to use the scheduler device in the FPGA chip, first, the module should be loaded into the kernel memory of the embedded Linux system using the insmod command and then registered in the OLT scheduler (device) using the mknod command. Then, the user may test the OLT scheduler device through the application program.

5.1. Device Driver

The device driver provides the function to control devices and it is provided from the kernel perspective. Depending on the type of device to control, it is divided into the block device driver, the network device driver, and the character device driver[11]. In this paper, the FPGA board is controlled by the character device driver.

```
struct file_operations OLT_fops = {
        open    : OLT_open,
        release : OLT_release,
        ioctl   : OLT_ioctl,
        write   : OLT_write,
        read    : OLT_read,
};
```

The essential parts of the device driver are as follows. The functions, init_module() and cleanup_module() are for loading and deleting of the modules. The functions, OLT_virtual_memory_allocate() and OLT_virtual_memory_free() are for obtaining and deleting the virtual addresses of the registers in the CPU interface logic.

```
ssize_t OLT_write(struct file *filp, const char *buf, size_t
length, loff_t *f_pos)
{
        char data[30];
        unsigned short  temp;

        copy_from_user(&data, buf, length);

        unsigned int *R_reg0_L =
        (unsigned int *)(Virtual_BasePtr + 0x3c);
                        ……
        temp = data[0]; temp = (temp<<8);
        temp = temp + data[1]; temp = temp & 0xffff;
        *R_reg0_H = temp; user_wait(500000);
                        ……
}
ssize_t OLT_read(struct file *filp, char *buf, size_t length,
loff_t *f_pos)
{
        char data[30];
        unsigned short temp = 0;
        unsigned int *G_reg0_L =
        (unsigned int *)(Virtual_BasePtr + 0x78);
                        ……
        temp = inw(G_reg0_H);
        data[0] = (temp & 0xff00) >> 8;
          data[1] = (temp & 0x00ff);
                        ……
        copy_to_user(buf, &data, sizeof(data));
}
```

The struct, OLT_fops is constructed based on the type of struct, file_operations provided by the kernel. The left side separated by a colon is the control operation that the module will process and the actual processing function corresponding to each control operation is given in the right side.

The function, OLT_write() moves the repost message sent from the application program to the device driver, using copy_from_user(&data, buf, length). Then it partitions it into 16bit data and writes it to the virtual address allocated to the report register in the CPU interface logic. The function, OLT_read() reads in the data in the address corresponding to the gate register of the FPGA and stores in the variable in the device driver.Then, it sends the data to the application program, using copy_to_user(buf, &data, sizeof(data)).

5.2. Application

The application program writes the report message, using the file descriptor returned by opening the OLT device and reads in the gate message sent from the device to generate the output.

```
fd = open("/dev/olt_dd", O_RDWR | O_SYNC);

while((key = main_menu()) != 0)
{
   switch(key)  {
   case '1':
      printf("start OLT's Scheduler\n");
      ioctl(fd, 1, flag); break;
    case 'w':
      printf("Put 5 Report packets\n\n");
      write(fd, buf_write, 30); break;
   case 'r':
       printf("Get 5 Grant packets\n");
       read(fd, buf_read, 30);
       Display(buf_read); break;
   case 'q':
       printf("exit\n"); exit(0); break;
  }
}
```

5.3. Results

Figure 9 illustrates a LDS2000 system. Since it uses the LDS2000 system combining ① the embedded board and ② the FPGA board as one, its experimental environment is very simple. The test application program is a console based program and its operation was verified using minicom on a host Linux PC. Figure 10 represents the entire procedure covering from loading the olt_dd.o module to executing the application program and deleting the olt_dd module. As shown below, it provides the same result as obtained from the simulation in Chapter 4.

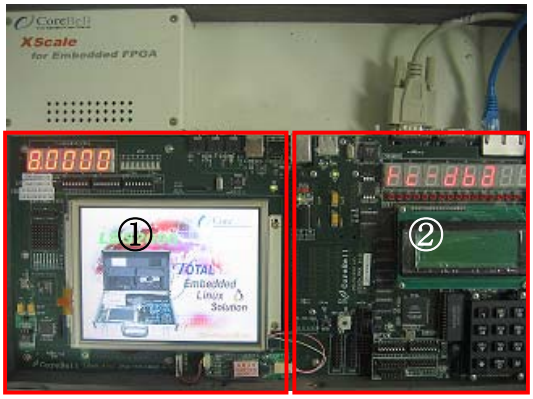
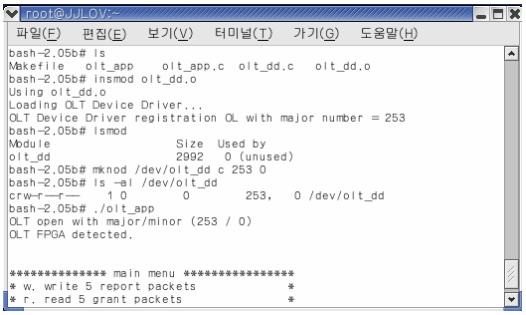
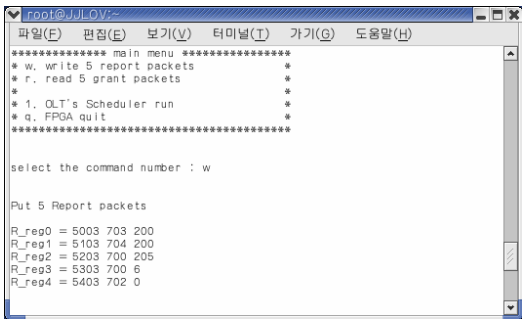
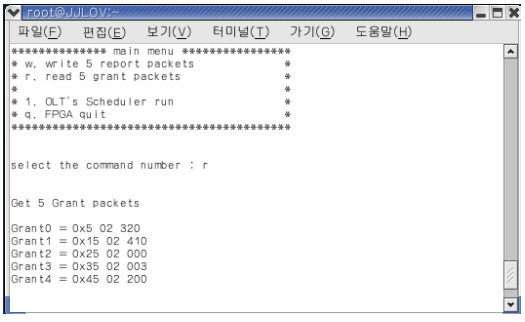


Fig.9 LDS2000 system







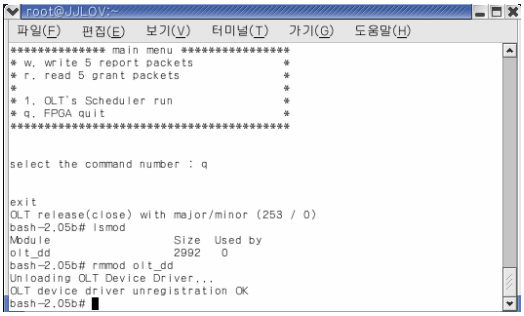


Fig.10 Execution of the verify program

6. Conclusion

Since the bandwidth allocation policy is excluded from the standardization in IEEE802.3ah EFM, it may create some problems in compatibility between the equipments provided by different manufacturers. On the other hand, the manufacturers are given opportunities to distinguish the performance of their products over other manufacturer's products. The RC-DBA algorithm was designed using the MPCP to support the QoS and to resolve the shortcoming of the existing DBA algorithms that can not support the fair bandwidth allocation between every ONU.

In this paper, we implemented the scheduler for the channel allocation on the FPGA chip in the OLT which the RC-DBA algorithm is applied to and we also verified its performance by using the embedded Linux based device driver and the application program. This can be extended to the development of the MAC chip in the OLT in the future.

The ultimate goal of this study is to verify the operation of the RC-DBA algorithm by implementing the ONU module, which can transmit the report message to the scheduler and receive the gate message, on the FPGA chip and communicating with the actual OLT. For this purpose, we are currently engaged in designing the ONU module that can perform the operation related to the scheduler.

Acknowledgments

This work was supported by the Korea Industrial Technology Foundation and the Brain Busan 21 Project in 2006.

References

[1] Jang seong-ho, "Design and Performance evaluation of a RC-DBA algorithm Supporting fairness among ONUs for EPON", Department of computer engineering, graduate school, a thesis for a doctorate, 2004

[2] S.Choi and J. Huh, "Dynamic Bandwidth Allocation Algorithm for Multimedia Services over Ethernet PONs," ETRI Journal, Vol. 24, No. 6, pp. 465-468, Dec. 2002.

[3] H. Shimonishi, I. Maki, T. Murase, and M. Murata, "Dynamic Fair Bandwidth Allocation for Diffserv Classes," Proceeding of IEEE ICC, Vol. 4, pp. 2348-2352 Apr.-May 2002.

[4] Chadi M. Assi, Yinghua Ye, Sudhir Dixit and Mohamed A. Ali, "Dynamic Bandwidth Allocation for Quality-of-Service Over Ethernet PONs," IEEE Journal on Selected Areas in Communications, Vol. 21, No. 9, Nov. 2003.

[5] J. Moon, J. Park, and M. Lee, "Hybrid Bandwidth Allocation Algorithm To Support Multiple Services in Ethernet PON," Proceeding of ICACT 2003, pp. 692-696, Jan. 2003.

[6] Fu-Tai An, Yu-Li Hsueh, Kyeong Soo Kim, Ian M. White, and Leonid G. Kazovsky, "A New Dynamic Bandwidth Allocation Protocol with Quality of Service in Ethernet-based Passive Optical Networks," Proceeding of IASTED WOC 2003, pp. 383-135, Jul. 2003.

[7] Seng-Ho Jang and Jong-Wook Jang, "New DBA Algorithm Supporting QoS for EPON", The CS&CE International Multiconference on CIC 2004, Las Vegas, USA, Jul. 2004.

[8] Seong-Ho Jang and Jong-Wook Jang, "Performance Evaluation of a New DBA Algorithm Supporting Fairness and Priority for Ethernet-PON", the IASTED International Conference on OCSN 2004, Banff, Canada, Jul. 2004.

[9] Choi Byeong-yoon, "The design of the control logic for digital system by ASM Chart , http://hyomin.deu.ac.kr/~bychoi/system_lsi_2004.html

[10] 802.3 draft document - MPCP Message Format, http://www.ieee802.org/3/efm/baseline/hirth_1_0302.pdf

[11] Corebell Co., "The practical use of the Linux Device Driver", pp. 105-140, 2003

Jong Wook Jang received the M.S. degree in Computer science from Chungnam national university in 1991 and the PhD. Degree in Computer Engineering from Busan national university in 1995. He has been a Professor in the Department of Computer engineering, Dongeui University, Busan, Korea. His research interests include Ethernet-PON, Mobile MAC Protocol and low-power consumption Cross-layer Protocol.

Hyun Jin Kang received the B.S. degree in Computer Engineering from Dongeui University in 2005. She is now with the Department of Computer and Software Engineering, Dongeui University. Her research interests are in the areas of data communications and Embedded system.

Hyoung Goo Jeon received the B.S degree from Inha University, Incheon, Korea, in 1987 and the M.S. and PhD. Degrees from Yonsei University, Seoul, Korea, in 1992 and 2000, all in electronic engineering. From 1987 to 2000, he was with Electronic and Telecommunication Research Institute, Daejeon, Korea. Since 2001, he has been a Professor in the Information communication Engineering Department of Dongeui University, Busan, Korea. His research interests include WLAN, CDMA modems and digital communications