A Hierarchical Weighted Round Robin EPON DBA Scheme and Its Comparison with Cyclic Water-Filling Algorithm

Chan Kim*, Tae-Whan Yoo, Bong-Tae Kim

Broadband Convergence Network Laboratories

Electronics and Telecommunications Research Institute

161 Gajeong-Dong, Yuseong-goo, Daejeon, 305-350 KOREA

(ckim*,twyoo, bkim)@etri.re.kr

***Abstract*—A H-WRR (Hierarchical Weighted Round-Robin) EPON (Ethernet Passive Optical Network) DBA (Dynamic Bandwidth Allocation) algorithm is devised and investigated. WRR table entries having reports are scanned to generate the gate which is limited to the token size of the entry. LLIDs (Logical Link IDs) are grouped into classes and maximum delay control is provided per class and minimum idle state polling rate is configurable for each ONT (Optical Network Terminal). The scheme's bandwidth allocation behavior and delay performance are shown through VHDL simulation with highly random self-similar traffic and comparison is made with previous CWF (Cyclic Water-filling) DBA [4, 5]. We conclude that in general, WRR DBA outperforms cyclic DBA and the H-WRR DBA scheme provides very low delay and high throughput.**

I. EPON AND DBA

EPON (Ethernet Passive Optical Network) [1] standardized by the IEEE802 is regarded as the most promising scheme due to its low overhead and low cost nature and is now being deployed in volume especially in Asia. Though GPON (Gigabit Passive Optical Network) standardized by ITU-T is considered as the solution in North America and Europe, technically it is still at its initial stage. EPON is mature in technology and has already much wider deployment. It is likely that the EPON vs. GPON competition will continue for the next several years.

EPON comprises an OLT (Optical Line Termination) in the central office and many ONTs in the residential area that are connected by single fiber and splitter as shown in Fig. 1. The unit of data transport is the Ethernet frame and a tag called LLID (Logical Link ID) is used in the preamble to mark the source or destination ONT. The LLID is assigned to each ONT during the discovery process. In downstream, the LLID indicates the destination ONT, and in the upstream, in indicates the source ONT. Using LLIDs, bridges in the OLT or ONT can perform the bridge operation of source MAC learning, destination MAC look-up and forwarding, etc. as in other topologies.

Downstream optical signal is broadcast to all ONTs and upstream optical signal sent by an ONT goes only to the OLT. To avoid contention in the upstream signals, the OLT should arbitrate the upstream transmission of the ONTs allocating upstream bandwidth resource to them while doing it. This operation is called DBA (Dynamic Bandwidth Allocation).

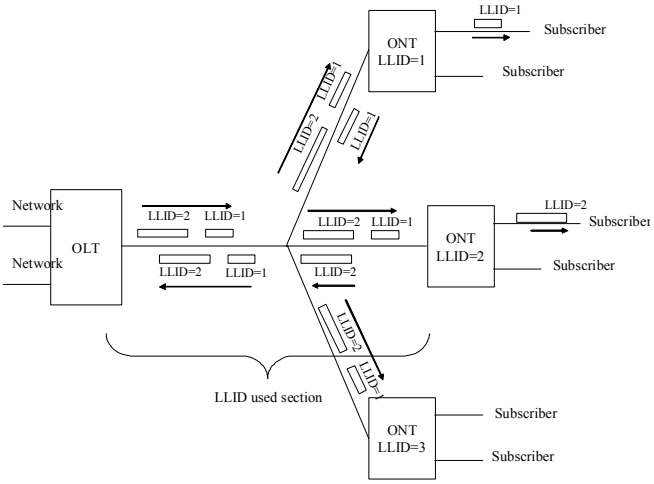

Fig. 1. Usage of LLID in EPON

The upstream bandwidth request is delivered using report frames containing the needed grant length and gate frames are delivered to the ONTs which contain the grant information in the form of (start time, length) pair. This control is done through MPCP (Multi-Point Control Protocol).

To arbitrate upstream transmission, timers are used in OLT and ONTs that is inserted to the MPCP frame when it is transmitted. ONTs' timer is synchronized to that of OLT and OLT measures the RTT (Round Trip Time) of each ONT by subtracting from the OLT timer the timestamp delivered from the ONT when it receives a frame. Before gate transmission, the OLT subtracts corresponding RTT value from the start time to compensate for the difference in the distance between ONTs.

We have implemented an EPON ASIC chipset and found that our previous work (CWF, Cyclic Water-Filling [4, 5]) has unavoidable upstream delay and granularity problem in bandwidth control. So we have investigated other DBA schemes and found that simple WRR DBA scheme provides lower delay and higher throughput.

The organization of the paper is as follows. Some previous works are explained in section 2 and the new H-WRR algorithm is explained in section 3. Performance simulation

result is shown in section 4. Section 5 concludes it with future works.

II. Related Researches and Previous Works

There are numerous paper on EPON DBA. Here some relevant previous researches will be explained. In [2], a most natural DBA algorithm is described where bandwidth is allocated in the order of fixed allocation, high priority and low priority requests in strict priority algorithm. When bandwidth cannot be assigned as requested, the available length is allocated with weights proportional to the requests. Another research [3] tried to solve the starvation problem of this previous work [2] and it shows better performance with respect to fairness. But these previous works [2, 3], like most others in the literatures, need floating point operation which is difficult to implement without CPU or DSP and does not consider the timing latency between reports and gates (grants).

As our pervious work, we've developed an EPON ASIC chipset where cyclic water-filling DBA algorithm was implemented which doesn't need floating point operation [4, 5]. The chip is being used in commercial service in the city of Gwang-joo for about 1500 subscribers in 2006.

Fig. 2 shows the picture of the ASIC and the master card. It provides stable and high enough performance with necessary minimum guaranteeing and maximum limiting capability for each LLID.

The DBA is cycle based where short static gates are generated for each registered ONTs to make sure reports are collected every cycle and the remaining time of the cycle is dynamically allocated to the ONTs according to the reports collected during the previous cycle.

Allocating the remaining time to the ONTs is like water-filling in that unit length is subtracted from the available gate resource time and added to each ONT in a cyclic fashion until the requests are all satisfied or no available time is left. Upstream delay more than the cycle time is always present regardless of the traffic amount and the unit length in the water-filling causes scalability and granularity problem when expanding it to many, like more than 128 LLIDs.

Fig. 3 shows this CWF DBA's timing relationship. The actual water-filling action is performed between the static gate generation and dynamic gate generation.

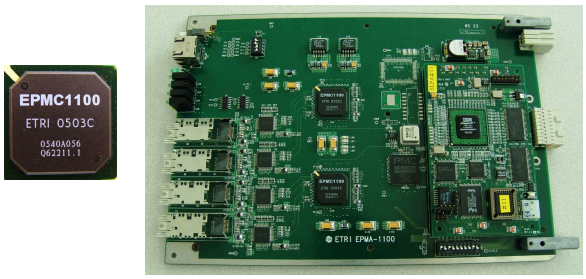

Fig. 2. Picture of the EPMC chip and board

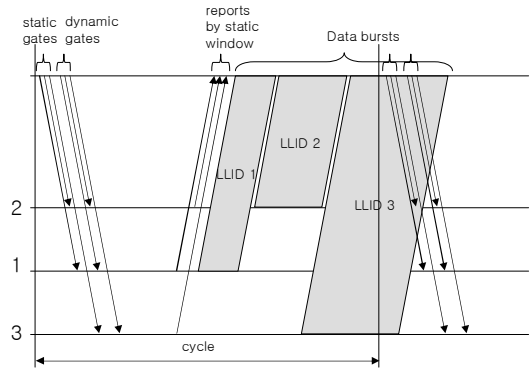

Fig. 3. CWF DBA's report and gate timing

As a continuing work, we've looked at various scheduling algorithms and found that simple WRR (Weighted Round Robin) scheme widely used in packet scheduling gives good performance [6] and investigated its performance through simulation. Ref. [7] provides hierarchical round-robin approach to provide delay control in ATM networks. The IPACT algorithm, with its adaptive cycle time, provides lower delay and maximum use of bandwidth [8]. The WRR scheme, because the report/gate feed back latency causes the report/gate actions to be interleaved, functions like IPACT algorithm. There is also another research employing WRR scheme in EPON master DBA [9].

III. Hierarchical WRR DBA Algorithm

Fig. 4 shows how the H-WRR (Hierarchical WRR) scheme works. All the LLIDs are grouped into several classes. Basically there is one entry for each LLID and they are visited continuously in round-robin fashion. So at any given moment, a single LLID is processed. Every time an entry is visited, if there is a non-zero report, gate is generated which is as big as the reported queue size, but the gate length is limited by the token size of the entry (Of course, sync time and laser on/off time are added to the gate length in all cases). Once gate is generated, the report value is cleared. This is an important aspect of the algorithm. If we manage WRR entries and report table separately, we could have multiple WRR entries for an LLID but this doesn't help much because of the report latency. If we do not clear the report value but only subtract the generated gate length from the report value, it damages the WRR operation because in EPON, due to the report latency, the queue sizes are not available to the OLT scheduler and this makes LLIDs with high report value be serviced consecutively thus breaking the report-gate interaction for the LLID.

Having token size too small, the maximum bandwidth a single ONT can use is limited by the RTT. At best, the max bandwidth is limited by token_size/RTT. So we cannot use too small token size. Further, small token size causes bigger upstream overhead.

Without any class mechanism, the maximum gate interval for an LLID is the sum of the token size of all the entries. In this case, if an ONT has very low traffic and all other ONTs have big token sizes and high traffic, all ONTs including the low traffic ONT will experience large delay. Therefore, we

need to provide guaranteed low delay for some premium service ONTs. To have priority control over upstream delay, hierarchy mechanism is introduced. Here LLIDs are each classified into a class, an LLID corresponding to a class. Since the MPCP gate frame contains only aggregate gate information for the LLID, not for each priority, there is no means to deliver gate information for each class in an ONT. The only way we can deliver prioritized gate for each priority is to divide traffic's priority into different LLIDs as a commercial company does [10]. So using this scheme, the upstream traffic will be classified and stored into separate queues that belong to different LLIDs according to the frame contents such as VLAN tag's user priority.

In the H-WRR scheme, each class has maximum tenure period, maximum total allocation length per a tenure and yield period. When processing each class, the scheduler cyclically visits the entries belonging to the class and allocates gate lengths according to the received queue reports and the token sizes. When gate allocation occurs, the gate length is added to the total allocated gate length for that class. This repeats until either the maximum tenure period expires or the total allocation length reaches the maximum value for the class, when the scheduler yields to the next lower class. Of course the lower class has its own total allocation length and maximum tenure period and has the same mechanism to yield to its lower class. When any lower class is being processed, all the higher classes' yield time counter counts and if the yield period expires, the processing is switched to the higher class with yield time-out.

Simply said, any class yields to its lower classes when maximum tenure period or maximum total allocation is reached and is deprived of its service by one of its higher classes when the yield counter of that higher class reaches the threshold.

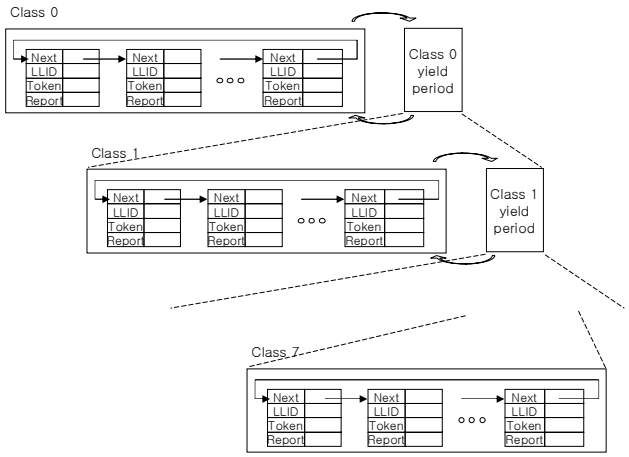

Fig. 4. H-WRR DBA scheme

Since there is no "cycle" concept, for each LLID, the report and gate response time is only bound by RTT and other ONTs' report and gate activities. If there are many idle ONTs, the report and gate response time decrease and upstream delay is reduced. This is the best nature of this WRR scheme. Since any ONT for which gate has been issued recently does not receive the chance during the visit until its report is updated, the following time is naturally allocated to other ONTs in the class. This makes the report and gate phases to be naturally interleaved. Also, for any class, since the scheduling order is maintained in WRR fashion even after it is switched to other classes and switch back in, there is no problem in the fairness unlike most cycle based DBA.

Also, to prevent an idle ONT from not receiving the gate for a long time, the gate generation is monitored and if an ONT didn't receive the gate for a predefined time, a static gate is generated for the ONT for polling. This minimum polling rate for an ONT is programmable for each ONT and this controls the maximum initial delay. The polling interval should be appropriately determined to control the maximum initial delay and basic upstream overhead.

For a given LLID, the minimum guaranteed bandwidth can be determined by the token sizes and the yield time for the classes. To add maximum limiting capability, separate leaky-bucket based rate limiter can be combined in gate generation. For each LLID, with predefined cycle time, a value is added to the rate token corresponding to the rate set for the LLID. During the WRR process, the gate length, which is calculated by the report value and limited by the WRR token size, is finally compared with the rate token. If there is enough rate token, the gate is generated and the gate length is subtracted from the rate token. If there isn't enough rate token, the gate is not generated and the report value is kept intact. Fig. 5 shows the rate limiting concept. A simple leaky bucket style rate limiting is available [11].

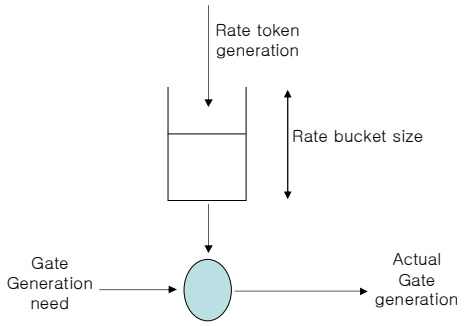

Fig.5. Rate Limiting

Comparison to other DBAs will be due here. IPACT [8] is not based on the current standard in that the OLT should generate the gate message exactly as earlier as RTT before the actual ONT transmission because the gate message doesn't contain the start time but the gate length. The authors suggested using escape code to insert the gate information to the downstream traffic at the right place. But current standard accommodates start time in the gate message so that there is flexibility between actual gate delivery time and the ONT's transmission window for the grant. In our algorithm, we have a constant called 'minimum offset' which defines the minimum offset between current time and the scheduled gate's start time at the time of gate generation. Therefore, the start time is sometimes pushed off to a future time so that the minimum

offset condition is met. After the start time is determined, the RTT value of the ONT is subtracted from the start time before gate frame transmission to compensate for the difference in the distance between ONTs. Also, in IPACT, the gate is calculated right after the report message is received in an interleaved manner. This is responding to the report right away without any cycle concept. The WRR mechanism is one way of implementing this 'generating gate immediately after a report is received' scheme. In all, this algorithm works like IPACT but conforms to the IEEE802.3 standard.

Some DBA algorithms like [2],[3] use floating point operation and it sometimes puts limitation when implementing it with FPGA or ASIC. Floating point calculation makes it possible to precisely implement fairness or more elaborate algorithms but in most cases, simple cyclic allocation or WRR based allocation provides enough performance.

IV. Performance Simulation

The performance of the H-WRR DBA algorithm was analyzed using VHDL simulation. The code for simulation can be used in the ASIC implementation with slight modification. Unlike other simulation techniques, this VHDL simulation runs exactly like the actual circuit including the simulation environments. In conventional even-driven simulation, many details which can affect the performance are sometimes ignored and not considered in the modeling thus making the simulation result different from the actual implementation. Also, every 2 ms, the upstream frames' bandwidth and delay were measured by averaging for the period. This way, unlike other paper's simulation result, we can also see the transient behavior of the DBA algorithms. Fig. 6 shows the VHDL test-bench for performance simulation. Every period, number of bits are read in from the files and added to the traffic source and every different period, statistics are gathered at the traffic destination and written to the result file.

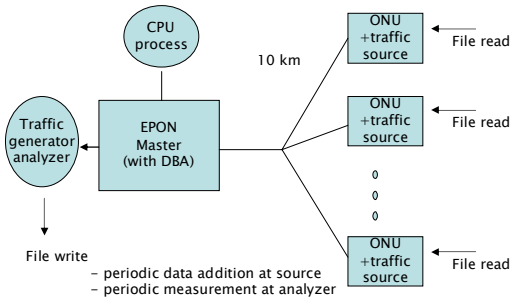

Fig. 6. VHDL simulation test-bench

To verify the performance and transient behavior, typical scenarios were selected and simulated. As traffic source, self-similar traffic was chosen which is regarded to be close to the actual network traffic. Traffic generator based on [12] and which is on the web was used to generate the traffic. When generating the sequence using the program Hurst parameter was set to 0.99 and the variance was set to 10 times big as the mean value to get a more dynamic traffic pattern with different seeds. Fig. 7 shows a typical traffic pattern (for 100% line rate, 64 byte frame) in which we can see self-similarity.

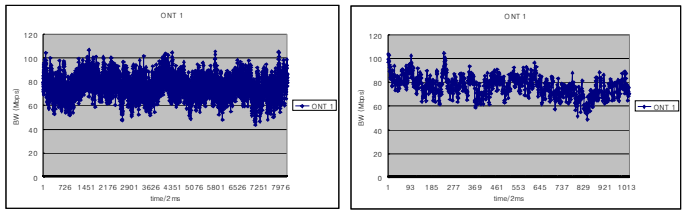

Fig. 7. Self-similar traffic sample

In the first scenario, we have 4 ONTs connected with ONT1 : 30%, ONT2 ~ ONT4 : 10% line-load all with 64 byte frame. Thus, total traffic is 60% - normal traffic load. Load is represented by the portion of traffic consisting of frame, preamble and minimum IFG (Inter-Frame Gap) relative to 1 Gbps. ONT3's traffic is turned off during a period. The fiber length is 10 km and ONT's upstream buffer size is 128 KB for each priority (In the two scenarios, only priority 0 traffic is applied. Priority scheduling in ONT was simulated too, but this paper is focused on OLT scheduler). In the second scenario, we have 4 ONTs but with each ONT having twice as much traffic, resulting in 120% load - an over-traffic case. ONT3's traffic is also turned off during the same period.

To see how much H-WRR algorithm outperforms the CWF (Cyclic Water-Filling) algorithm, simulation was performed first for CWF method for 85 ms. As can be seen in Fig. 8, when cycle was chosen to be 1 ms, mean delay is about 1.5 ms. Initial delay is also bound approximately to 1.5 ms because polling is performed every cycle time. There is no loss in the bandwidth utilization in both normal and over-traffic cases(10% load corresponds to 76.19 Mbps when 64 byte frame is used). In over-traffic case, during the period when ONT3 traffic is off, the excess bandwidth is utilized to deliver ONT1's traffic.

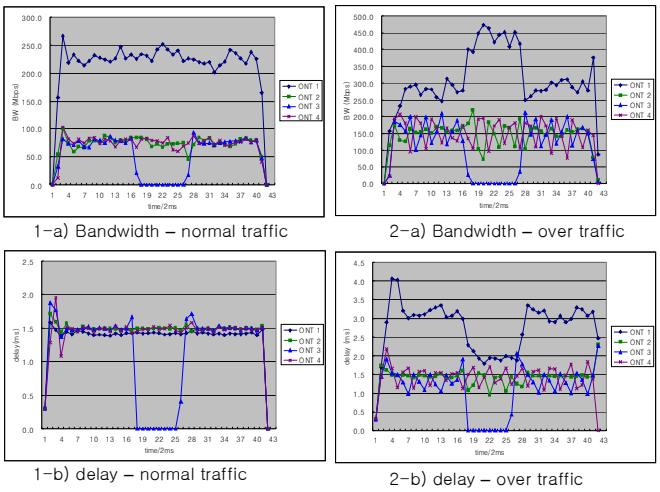

Fig. 8. Bandwidth and delay for CWF algorithm

Fig. 9 shows the simulation result for the same traffic using H-WRR algorithm. In the beginning, since the polling period during the idle period was set to 2 ms, it exhibits an initial build up of traffic and it takes some time until the effect of

back-logged traffic disappears. During this time, the delay and bandwidth settles down to equilibrium value. During the period when ONT3 traffic is off, the bandwidth that was allocated to ONT3 is utilized to deliver ONT1's traffic in over-traffic case. It shows about 0.5 ms upstream delay in normal traffic load which is very small compared to CWF algorithm. The upstream delay is bigger in over-load traffic (1 ms ~ 2 ms in this case, of course frame loss occurs) and the delay is determined by the ONT's buffer size. In both cases, there is no loss of upstream bandwidth.

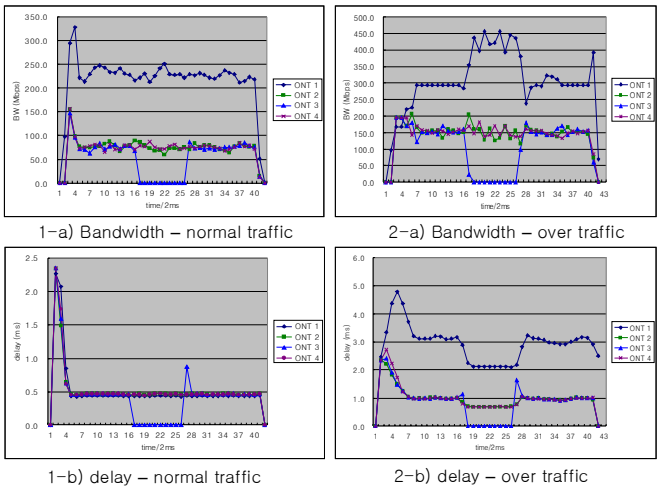

Fig. 9. Bandwidth and delay for H-WRR algorithm

A separate simulation scenarios was used for 32 ONTs to compare the performance of WRR and CWF. Here the traffic load increases evenly for all the ONTs with 4 steps. Fig. 10 shows the throughput and delay performance of WRR and CWF DBA algorithm with varying cycle time. The result can be regarded as the general performance comparison between any cyclic DBA and any WRR DBA (because only the general aspects of each scheme were used in this simulation). Though not shown in the picture, we could also see that the delay variance of CWF is bigger than that of WRR DBA.

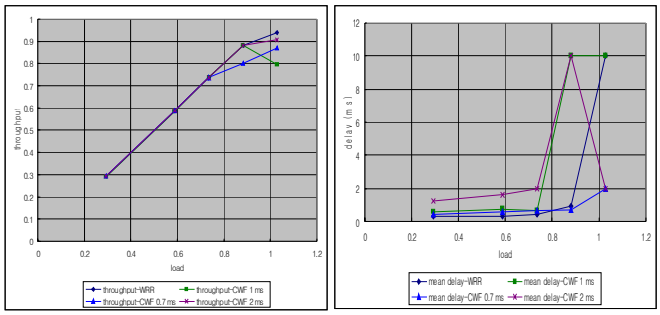

Fig. 10 comparison of CWF and WRR DBA

Having seen that WRR DBA outperforms CWF DBA, to further see the effect of H-WRR DBA's delay control using classification on LLIDs, another scenario was chosen where 8 ONTs are connected with each 12.5 % load (This sums up to 100 % load and this marginal situation is selected to see the class difference clearly). Only two classes were defined. ONTs 1~4 were set as class 0(high priority), and 5~8 were set as class 1(low priority). Also, ONT3's traffic is turned off in the meantime as before. Each class was set as table 1. All parameters are in unit of 16 bits, MPCP time-quanta.

Table 1. H-WRR class setting

Parameter	Class 0	Class 1
Max tenure period	0xA000	N.A.
Max total allocation	0xF000	0x7000
Yield period	0x5000	N.A.

Fig. 11 shows the effect of class setting using H-WRR DBA. Left side shows when there were no class setting and right side shows when class differentiation is in force. Class 0 has lower delay while class 1 experiences bigger delay. Though not shown, when the total traffic is not over-traffic, there still is some delay difference between classes.

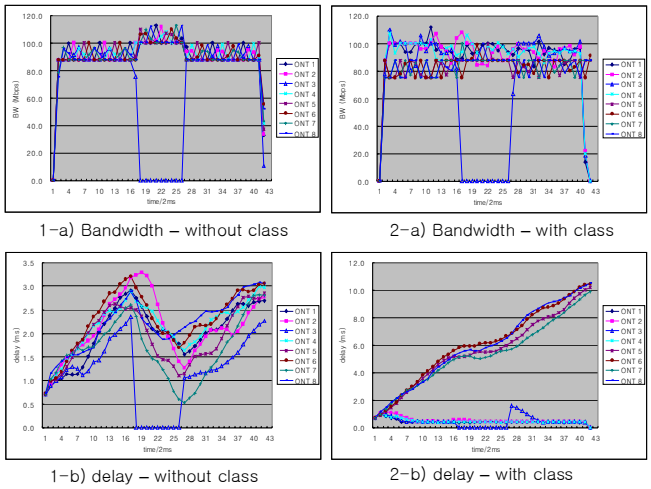

Fig. 11. Effect of class differentiation

To verify the rate limiting function, another scenario was selected for H-WRR DBA. We have 4 ONTs, each having 20% wire load (=152.38 Mbps). Rate limiting was applied. For ONTs 1, 2 and 4, 6 rate-token is added and for ONT 3, 12 rate-token is added every 100 MPCP time quanta. This corresponds to rate-limiting to 45.714 Mbps for ONT 1,2 and 4, and 91.43 Mbps for ONT 3. ONT 3's traffic stops during an interval as before. Fig. 12 shows the simulation result which indicates the good performance.

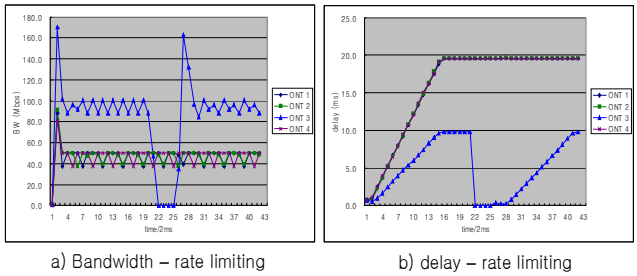

Fig. 12. Rate limiting effect

V. Conclusion and future works

Though the previously implemented CWF DBA provides stability in over-traffic with enough upstream throughput and low maximum delay, it has twice as much overhead because of the periodic static gates even for active ONTs providing less total throughput and larger average delay. When there are 128 LLIDs, this overhead by static gates takes too much portion.

The H-WRR DBA scheme provides very low delay with high throughput with appropriate parameter settings and the implementation has no scalability problem like CWF DBA. To have lower initial delay for the bursty traffic, it should have small polling cycle during the idle period and this can increase the basic overhead like CWF algorithm too. But this polling overhead is minimized for active ONTs.

The H-WRR DBA also has some limitation. First, it should rely on separate rate limiting mechanism to limit the maximum bandwidth per LLID. Second, it is not possible to provide per-LLID weighting other than the minimum guaranteed bandwidth weight defined by the token size and sum of other ONTs' token sizes.

We will investigate various ways to enhance the H-WRR DBA scheme. Some of the remaining issues are about separating the burst token size from the minimum, maximum bandwidth control and ways to harmonize it with static bandwidth allocation. Applying rate limiting based on actual input traffic monitoring and harmonizing the algorithm with the threshold reporting are some of the other future works.

References

1. IEEE802.3ah standard, Jul. 2004.
2. S. Choi and J. Heo, "Dynamic Bandwidth Allocation Algorithm for Multimedia Services over Ethernet PONs", *ETRI Journal*, Vol. 24, no. 6 Dec. 2002.
3. S. Jang, J. Kim and J. Jang, "Performance Evaluation of New DBA Algorithm Supporting Fairness for EPON", TENCON 2004. 2004 IEEE Region 10 Conference Vol. C, pp. 29 - 32 Vol. 3,21-24 Nov. 2004.
4. C. Kim, T. Yoo, Y. Kwon and B. Kim, "Design and implementation of an EPON DBA algorithm", *BMW2005*, Whistler, B.C. Canada, Oct. 2005.
5. C. Kim, T. Yoo, and B. Kim, "An Improved Cyclic Water-Filling EPON DBA with Priority Control and Faster Allocation Time", *ICACT2007*, Phoenix Park, Pyeong Chang. Korea, Feb. 2007.
6. H. J. Chao and Xiaolei Guo, "*Quality of Service Control in High-Speed Networks, Chapter. 4*," John Wiley & Sons. 2002,
7. C. R. Kalmanek, H. Kanakia and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks," *Proc. Of IEEE GLOBECOM*, Dec. 1990.
8. G. Kramer, B. Mukherjee and G. Pesavento, "Interleaved Polling with Adaptive Cycle Time(IPACT) : A Dynamic Bandwidth Distribution Scheme in an Optical Access Network," Photonic Network Comm., Vol. 4, no. 1, 2002, pp.89-107.
9. H. Naser and H. T. Mouftah "A Fast Class-of-Service Oriented Packet Scheduling Scheme for EPON Access Networks," IEEE Comm. Letters, Vol. 10, no.5, May 2006.
10. Teknovus TK3721 product brochure, http://www.teknovus.com/files/TK3721_PB_1.pdf
11. K. Yoshigoe and K. Christensen, "Rate Control for Bandwidth Allocated Services in IEEE802.3 Ethernet,", Proc. Of IEEE LCN 2001.
12. V. Paxon, "Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic," ACM SIGCOM, Computer Communications Review, 27(5):5-18, 1997.