An Improved Cyclic Water-Filling EPON DBA Providing Priority Scheduling and Faster Allocation Time

Chan Kim*, Tae-Whan Yoo, Bong-Tae Kim
(ckim*,twyoo, bkim)@etri.re.kr
ETRI, 161 Gajeong-Dong,Yuseong-goo,Daejeon,305-350 KOREA
Tel : +82-42-860-5773 Fax : +82-42-860-5213

Abstract - **A cyclic water-filling EPON(Ethernet Passive Optical Network) DBA(Dynamic Bandwidth Allocation) algorithm and its implementation in an ASIC are described with future improvements. Every cycle, short static gates are generated to collect reports and dynamic gates are generated according to the reports collected in previous cycle. In each cycle, unit length is additively allocated to the ONUs in a cyclic fashion until all the requests are satisfied or no resource is left. Four parallel engines are used to process the requests. As improvements, priority will be considered, and the processing burden will be evenly distributed to the 4 parallel engines in any case.**

Keywords – EPON, DBA, Water-filling

I. EPON AND DBA

FTTH(Fiber To The Home) is becoming a practical solution for access network and has already been deployed in some countries. Among the FTTH technologies, EPON (Ethernet Passive Optical Network) standardized by the IEEE802 [1] is the most promising scheme due to its internet friendliness and low cost nature.

EPON comprises an OLT (Optical Line Termination) in the central office and many ONU (Optical Network Unit)s in the residential area. A single fiber connects the OLT to the residential area and passive splitter is used to connect many ONUs in that area to the fiber. The unit of data transport is the Ethernet frame and a tag called LLID(Logical Link ID) is used in the preamble to make the PON topology appear as many point-to-point links. The LLID is assigned to each ONU during the discovery procedure. In downstream, the LLID indicates the destination ONU, and in the upstream, in indicates the source ONU. Using LLIDs, bridges in the OLT or ONU can perform the bridge operation of source MAC learning, destination MAC look-up and forwarding, etc(Fig.1) as in other topologies.

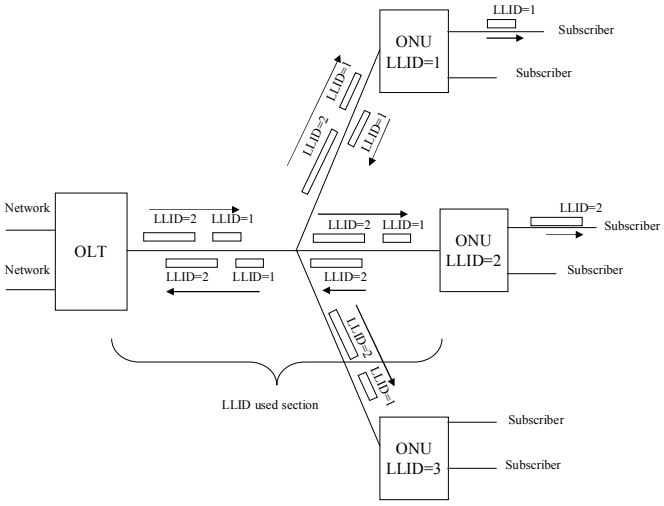


Fig. 1. Usage of LLID in EPON

Downstream optical signal is broadcast to all ONUs and upstream optical signal sent by an ONU goes only to the OLT. To avoid contention in the upstream direction, the OLT should arbitrate the upstream transmission of the ONUs allocating upstream bandwidth resource to them. This operation is called DBA (Dynamic Bandwidth Allocation). The upstream bandwidth request is delivered using report frames containing the needed grant length and gate frames are delivered to the ONUs which contains the grant

information in the form of (start time, length) pair.

To arbitrate upstream transmission, timer is used in OLT and ONUs. ONUs' timer is synchronized to that of OLT and OLT measures the RTT (Round Trip Time) of each ONU by subtracting from the OLT timer the timestamp delivered from the ONU. Before gate transmission, the OLT subtracts corresponding RTT value from the start time to compensate for the distance difference.

We have developed EPON master [2][3], slave chips. The master ASIC integrates two EPON ports and each port connects up to 64 ONUs. The DBA employs a simple cycle-based water-filling algorithm providing stable and high enough performance with necessary bandwidth provisioning capability. Fig. 2 shows the picture of the master ASIC which is being used in a commercial service in the city of Gwang-Joo for about 1500 subscribers as of Dec. 2006.

Fig. 2 Picture of the EPMC chip

We are revising the chip with some improvements in the algorithm and implementation architecture. This paper is on the improved version of the CWF (Cyclic Water-Filling DBA) algorithm. The organization of the paper is as follows. Our DBA algorithm is explained in section 2 and the implementation is described in section 3. The improvements are explained together with indications. Section 4 gives a brief result of the performance and section 5 concludes.

II. Related Works and Water-Filling DBA Algorithm

In paper [4], a natural DBA algorithm is suggested where bandwidth is allocated in the order of fixed allocation, high priority and low priority requests in strict priority algorithm. When congested, the available length is allocated with weights proportional to the requests. [5] tried to solve the starvation problem of this previous work and shows better performance with respect to fairness. But these works, 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 between reports and gates (grants).

This paper is based on our previous work [2] and addresses its functional and performance upgrades to be made in the future. As before, the proposed and implemented water-filling DBA scheme does not need any floating point computation yet providing faster allocation time and control over priority in the requests.

Our DBA scheme is cycle based where short static gates are generated for each registered ONUs to make sure reports are collected every cycle and the remaining time is dynamically allocated to the ONUs according to the reports received from previous cycle. Fig. 3 shows the timing relation between downstream gates and upstream bursts. Reports are delivered at the end of every burst. Bandwidth is controlled by controlling the gate length per cycle.

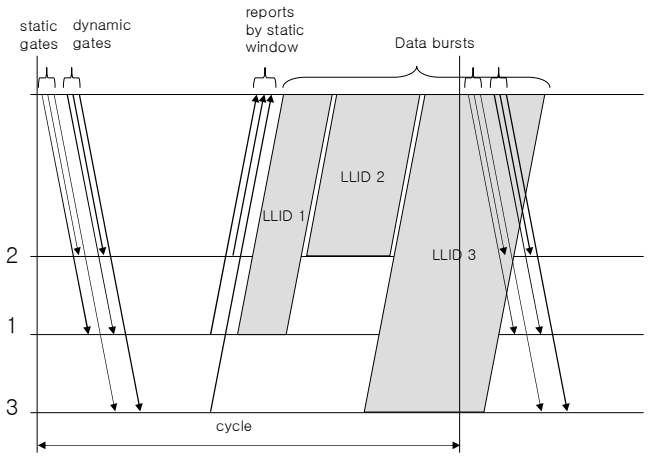


Fig. 3 report and gate timing in cycle

Allocating the remaining time to the ONUs is like water-filling in that unit length is subtracted from the available gate

resource time and added to each ONU in a cyclic fashion until the requests are all satisfied or the resource runs out.

This algorithm also makes it possible to put a limit in the dynamic gate length and to provide minimum guaranteed dynamic gate length for each ONU while considering priority information in the requests.

The original 8 priority queue reports are mapped to high and low priority, and then max-limited and converted to guaranteed (G), high (H) and low (L) values. Then, three DBA processing target variables of guaranteed (G), up-to-high (BH=G+H) and up-to-low (BL=G+H+L) are derived. Water-filling is done for these 3 phage variables. At each phase, unit lengths are cyclically allocated until the corresponding phase's request variables are satisfied for all the ONUs or the remaining gate runs out. Fig. 4 shows how the 3 target variables are calculated for actual water-filling processing phases.

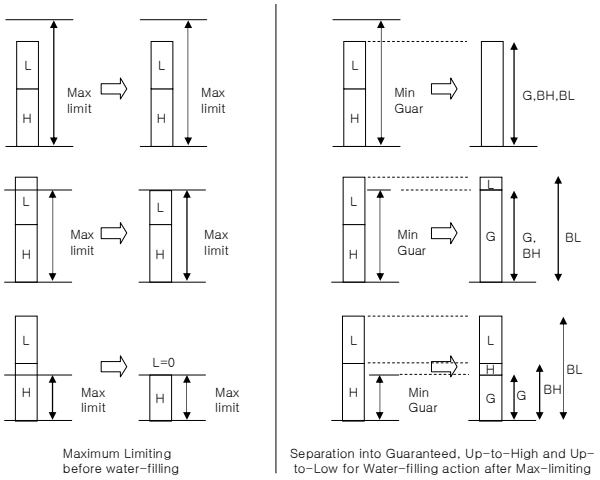


Fig. 4 pre-processing before water-filling

Fig. 5 shows the example of the allocation result using the proposed and implemented dynamic gate allocation algorithm for 2 different cycles. In the figure, each bar represents ONUs' requests. The lowest part of each bar is the minimum guaranteed gate length and the filled area is the allocated gate length. In cycles like case a), some ONUs' requests were not satisfied because no grant time is left at the final allocation stage. In cycles like case b), all the requests were satisfied and resource is left after DBA processing.

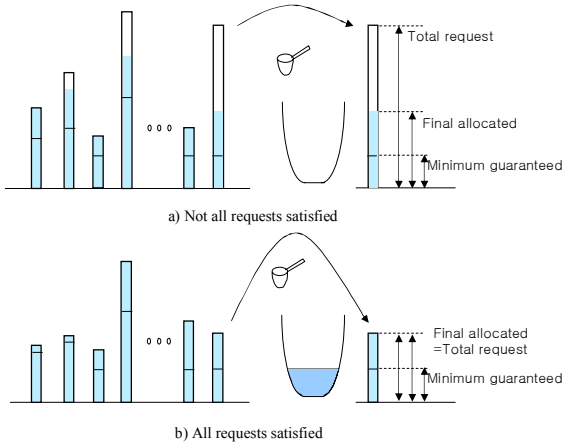


Fig. 5. Water-filling DBA algorithm

The water-filling algorithm can be represented as below where AV is total available gate length at given time.

(fixed allocation subtraction)

$$AV = AV - \sum_{i=1}^{N} FixedGateLength_i;$$

(CPU reserved length subtraction)

$$AV = AV - \sum_{i=1}^{M} CpuGateLength_i \text{ (if any)};$$

(adjustments for maximum limiting)
If ($HighPriorReq_i >= MaxLimit_i$)
HighPriorReqi = MaxLimiti;LowPriorReqi = 0;
Elsif ($HighPriorReq_i < MaxLimit_i < HighPriorReq_i + LowPriorReq_i$)
$HighPriorReq_i$ = $HighPriorReq_i$; $LowPriorReq_i$ = $MaxLimit-HighPriorReq_i$;
(separation into 3 target variables)
If ($HighPriorReq_i + LowPriorReq_i <= MinGuar_i$)
$G_i = BH_i = BL_i = HighPriorReq_i + LowPriorReq_i$;
Elsif ($HighPriorReq_i <= MinGuar_i < HighPriorReq_i + LowPriorReq_i$)
$G_i = BH_i = MinGuar_i$; $BL_i = HighPriorReq_i + LowPriorReq_i$;
Elsif ($MinGuar_i < HighPriorReq_i$)
G_i = $MinGuar_i$; BH_i = $HighPriorReq_i$; BL_i = $HighPriorReq_i$ + $LowPriorReq_i$;
(Water-Filling for Minimum Guaranteed Request)
While (not all G_i satisfied) and (AV>UnitLength){
If G_i not satisfied {
$Alloc_i = Alloc_i$ + UnitLength;
AV=AV-UnitLength; }
increment i;} / now Gi is satisfied */*
(Water-Filling for High Priority Request)
While (not all BH_i satisfied) and (AV>UnitLength){
If BHi not satisfied {
Alloci = Alloci + UnitLength;
AV=AV-UnitLength; }
increment i;} / now BHi is satisfied */*
(Water-Filling for Low Priority Request)
While (not all BL_i satisfied) and (AV>UnitLength){
If BLi not satisfied {
Alloci = Alloci + UnitLength;
AV=AV-UnitLength; }
increment i;} / now BLi is satisfied */*

III. IMPLEMENTATION OF THE DBA ALGORITHM

Fig. 6 shows the block diagram of the EPON master's main part controlling the EPON specific things including DBA. The operation is clearly understood so details will be not explained here. Note that the grant resources consumed by static gates and CPU gates are considered at the DBA block. The DBA gates are generated after static gate generation.

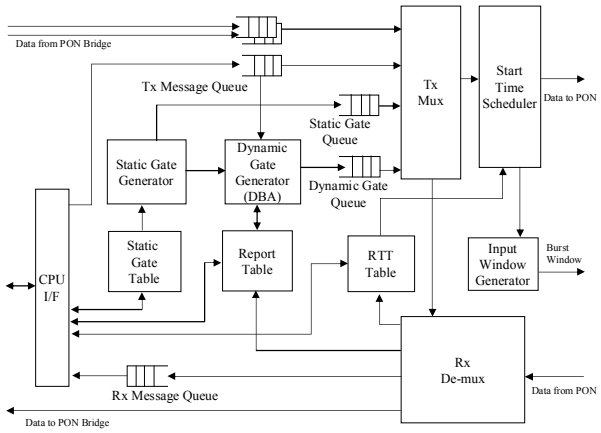


Fig. 6. Master's MAC Control Part

The start time scheduler determines the start time so that allocated grants do not overlap. It also subtracts corresponding ONU's RTT value from the start time before sending the frame and inserts timestamp. The start time scheduler also makes sure that the start time received by the ONU is always a future time with at least certain minimum distance apart so that the ONUs can setup for the grant usage.

The DBA gate generation logic is composed of several blocks as in Fig. 7. The 4 engines each process 16 ONUs' requests and status information is combined and distributed from/to the engines so that the common gate resource value is updated correctly and the engines can process the water-filling operation seeing the common resource value and the processing status of other engines.

In the improved parallel organization, registered ONUs are evenly distributed to the 4 engines. This reduces the DBA processing time and prevents some time period from not being used due to failing to meet the minimum offset requirement.

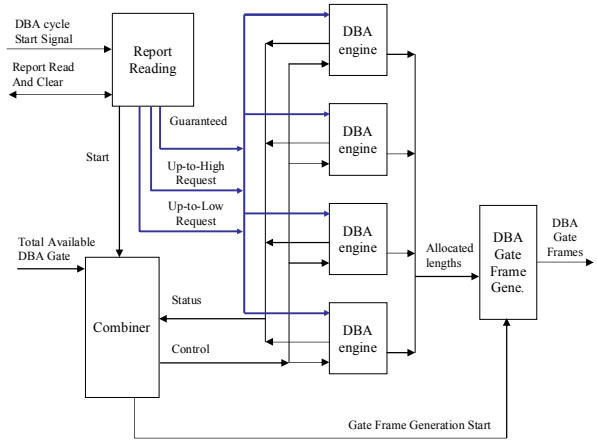


Fig. 7. DBA gate generator

Here, each block's function is described.

At the completion of static gate generation, DBA gate generation calculates the total available gate length provided. The report reading block reads the report table entries and puts limits in the report values and separates Gi, BHi and BLi values according to the maximum grant length and minimum guaranteed length set for each ONU. The reports are each cleared after reading. After these values are set-up, actual DBA processing starts.

Each engine processes 16 ONUs' requests. Each engine will have 3 three processing phases – G, BH and BL. For each phase, each engine repeats subtracting the basic unit from the common resource and adding it to its ONUs until they are satisfied or no more resource is left. As 4 engines process water-filling, the total available gate length is reduced by adequate amount every time unit lengths are allocated and the 4 engines monitor the total available length. If current phase completes for all ONUs, 4 engines proceed to the next phase at the same time. So all engines are in the same processing phase with same cyclic ONU indexes.

IV. PERFORMANCE SIMULATION

In this scheme, with cycle T and guard time G, the ideal total throughput can be calculated as (0.608 is time for report

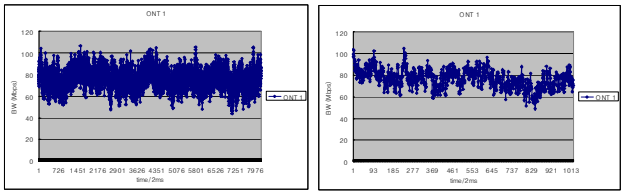
frame)

$$Throughput_{\max} = \frac{T - N*(G+0.608us)*2}{T}$$

With 1.024 ms cycle, 2.048 us guard time, this ideal maximum throughput for 16 ONUs is 91.7% and for 32 ONUs, it becomes 83.4%.

The performance of the CWF DBA algorithm was analyzed using VHDL simulation. This VHDL simulation runs exactly like the actual circuit including the simulation environments. Also, every 2 ms, the upstream frames' bandwidth and delay were measured by averaging for the period. This way, we can also see the transient behavior of the DBA algorithms.

To verify the performance and monitor the 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 Paxon[6] 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. 8 shows a typical traffic pattern (for 100% line rate, 64 byte frame) with self-similarity.

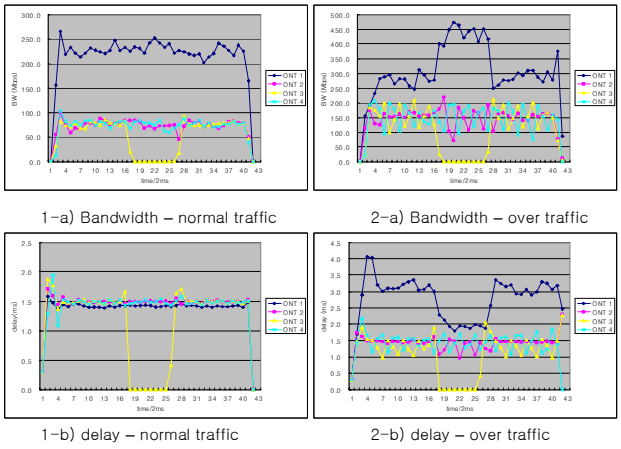


Bandwidth (0 ms ~ 81.92) Bandwidth (40.96 ms ~ 51.2 ms)

Fig. 8. Self-similar traffic sample

In the first scenario, we have 4 ONTs connected with ONT1 : 30%, ONT2 ~ ONT4 : 10% 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 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 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. 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.

As can be seen in Fig. 9, 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.



1-a) Bandwidth – normal traffic 2-a) Bandwidth – over traffic

1-b) delay – normal traffic 2-b) delay – over traffic

Fig. 9. Bandwidth and delay for scenarios 1 and 2

To verify that the minimum guaranteeing and priority control works correctly, a third scenario was deliberately devised. With 5 ONTs, the traffic high and low priority traffic composition, and the minimum guaranteed bandwidth were set as shown in Fig. 10. As before, ONT 3's traffic is turned off during a period as before.

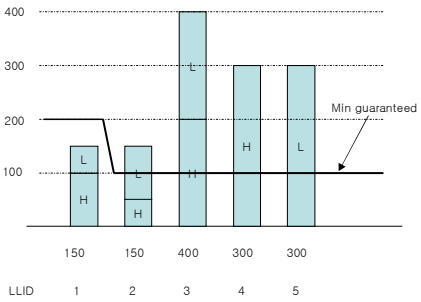


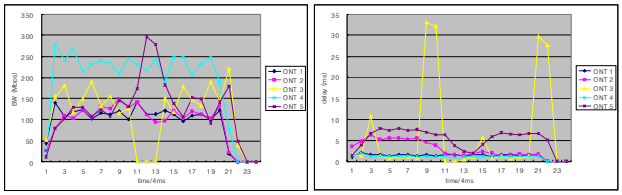
Fig. 10 traffic composition and min BW setting

If the DBA works as the algorithm, in steady state, the allocation should occur as below. Acc_BW value represents cumulative allocated bandwidth after processing.

Unit : Mbps

phase	1	2	3	4	5	Acc. BW
1st	150	100	100	100	100	550
2nd	0	0	100	200	0	850
3rd	0	50(50)	200(50)	0	200(50)	1000
Final	150	150	250	300	150	

The water-filling style allocation occurs only during 3rd phase. The number in parenthesis shows the allocated bandwidth using water-filling allocation. Fig. 11 shows the expected result. (The load is relative to 100Mbps wire speed, so 100Mbps wire load means 76.2 Mbps)



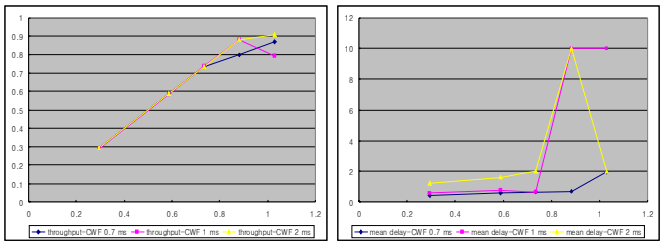
3-a) Bandwidth – scenario 3 3-b) delay – scenario 3

Fig. 11. Simulation result for scenario 3

With all the dynamic bandwidth allocation and priority control shown as before, the delay and throughput performance was simulated for 32 ONTs. Fig. 12 shows the performance for 32 ONTs for different loads and cycle time.

V. CONCLUSION

The previously implemented and modified DBA provides stability in over-traffic with enough upstream through-put. It



* Delay is shown clamped to 10 ms

Fig. 12. Performance for 32 ONTs

provides low worst case delay due to its periodic polling scheme. It also provides maximum limit and minimum guaranteed rate control for each ONU. But the performance degrades when more than 64 ONUs are attached due to wasted time during water-filling.

REFERENCE

[1] IEEE802.3ah standard, Sep. 2004.

[2] Su-Il Choi and Jae-Doo Heo, "Dynamic Bandwidth Allocation Algorithm for Multimedia Services over Ethernet PONs", *ETRI Journal*, Vol. 24, no. 6 Dec. 2002.

[3] Seong-Ho Jang, Jin-Man Kim and Jong-Wook 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] Chan Kim, Tae-Whan Yoo, Yool Kwon and Bong-Tae Kim, "Design and implementation of an EPON DBA algorithm", *BMW2005*, Whistler, B.C. Canada, Oct. 2005.

[5] Chan Kim, Tae-Whan Yoo, Yool Kwon and Bong-Tae Kim, "Design and Implementation of an EPON Master Bridge Function in an ASIC", *ISCC2006*, Pula-Cagliari, Italy, Jun. 2006.

[6] Vern Paxon, "Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic," ACM SIGCOM, *Computer Communications Review*, 27(5):5-18, 1997.