Quality-of-Service Scheduler for Next Generation Network Services

Hyung-Woo Choi[1], Young-Min Kim[1], Hong-Shik Park[1], Hyun-Yong Hwang[2], and Bum-Chul Lee[2]
[1]School of Engineering, Information and Communications University,
119 Munji-ro Yuseong Daejeon Korea 305-732
{joey, injesus01, hspark}@icu.ac.kr
[2]Electronics and Telecommunications Research Institute
138 Gajeong-ro Yuseong Daejeon Korea 305-350
{hyhwang, bclee}@etri.re.kr

***Abstract* — Flows have different characteristics depending on the type of application, and this leads to consideration of the QoS needed to guarantee that the application will work satisfactorily. However, the scheduling algorithms such as WFQ, DRR, or Hierarchical Scheduling algorithms are complex to guarantee QoS in per-flow level because of buffer management. Calendar queue is appropriate to service per-flow based scheduling because it is simple enough to implement and easy to manage buffers. However, the nature of the calendar queue – fixed slot size can degrade the network throughput. So we propose a QoS scheduler based on calendar queue with an algorithm to improve the network throughput. Also, we verify the performance of the proposed scheduler with ns2 simulator especially in terms of throughput.**

***Keywords* — QoS, Scheduler, Calendar queue.**

1. Introduction

The demands of high-quality of service are increased according to the rapid development of Internet technology. Also one of the key issues of the Next-Generation-Network (NGN) technology is Quality-of-Service (QoS) which guarantees throughput, delay, delay jitter, and loss rate. Recently, the representative services which need strict QoS are VoIP, IPTV and any other service may need QoS in near future. However, it is not easy to guarantee QoS because QoS-guaranteed service traffic and QoS-non-guaranteed service traffic are mixed in networks.

The traditional scheduling algorithms are work-conserving and require per-flow queues [2]-[6]. They are appropriate for QoS guaranteed services but not for QoS-non-guaranteed service. If they are applied to best-effort services, the rates for the flows should be dynamically chosen according to the current number of backlogged flows, which creates too high a burden on the system. Therefore, packet scheduling algorithms have generally been applied on a per-class basis.

N.S. Ko et al. proposes the QoS architecture for the flow-based router system [7]. The buffer is shared among flows, and the calendar queue is used for the scheduling packets [1]. The packets are stored in the shared memory in the system when they arrive in the system, and the scheduling times for the input packets are decided before sending them to calendar queue. If a packet is not appropriate to be serviced, it is dropped rather than being sent to the calendar queue.

However, the nature of the calendar queue – fixed slot size can degrade the network throughput. Basically calendar queue works based on the time slot. The time slot cannot be always fully filled with packets because it has to service variant-sized packets within a fixed slot time. So, most time slot is prone to having a blank period which we call "hole". This issue will be discussed in more detail at section 2.

So we propose a QoS scheduler which can service both guaranteed service and non-guaranteed service harmoniously in the mixed traffic situation. We tried to focus on the network resource utilization where it improves the throughput of non-guaranteed service traffic while it still guarantees the QoS of guaranteed service traffic.

We will discuss about the "hole" problem in section 2. The proposed QoS scheduler is presented in section 3. The performance evaluation results are given in section 4 and lastly conclusions are drawn in section 5.

2. Throughput Issues

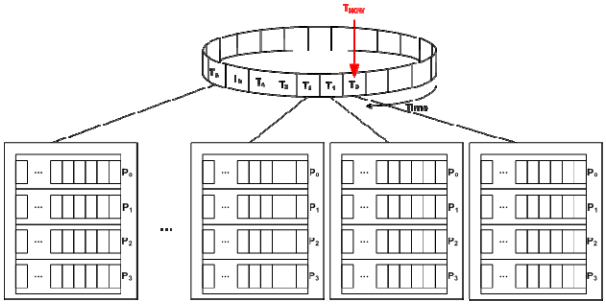

Figure 1. Calendar queue.

Calendar queue works based on the time slot and is non-work-conserving. It uses real-time clock and there are no specific buffers for each flows. This characteristic is very appropriate for the flow-based system because of easy buffer managements and controls. In the calendar queue every packet of which length is various is inserted into a time slot to be serviced. However, as the space of a time slot is limited the packet of which length is larger than the remained space of a time slot cannot be inserted into that slot and the time slot will

be serviced with the blank space (hole). Consequently, the scheduler will be in idle state during the blank space.

Whenever the length of the packet to be inserted into last part of a time slot is larger than the remained time slot space, it makes a hole. Even if the holes are small, the accumulation of them effects on the throughput pretty much. If the MTU is 1500byte (12000 bits), the maximum hole size can be 11999 bits in the worst case. Because the maximum hole size follows MTU, it is determinate. However, the portion which the hole occupies varies according to the size of a time slot. For example, if the size of a time slot is 32us in the 1Gpbs link, then the scheduler can serve 4000byte of packets within a single time slot. In case of 16us, the scheduler can serve 2000byte of packets within a single time slot. So, the maximum portions of the hole in a time slot are 37.5% and 75% at 32us and 16us respectively. As the size of a time slot is larger, the influence of the hole on the throughput is alleviated. Figure 2 shows the trend of the throughput-bound according to the size of a time slot in the 1Gbps link.

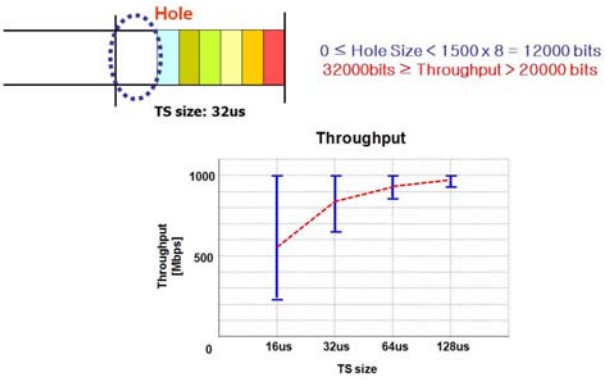

Figure 2. The trend of the throughput according to the size of time slot.

3. Proposed QoS Scheduler

Proposed scheduler is based on the calendar queue which operates based on the time-slot and is non-work conserving. The architecture of our proposed scheduler is shown at Figure 3. We suppose that it works micro-flow based. The scheduler identifies flow id and calculates TTS (Time-To-Start) for every incoming packets and these information is managed by the flow state table. A guaranteed service packet is directly inserted into time slot of calendar queue according to calculated TTS value and waits to be serviced because it has high-priority and should be serviced in real-time. A non-guaranteed service packet is not directly inserted into timeslot but waits in buffer because it has low-priority and is delay-tolerant.

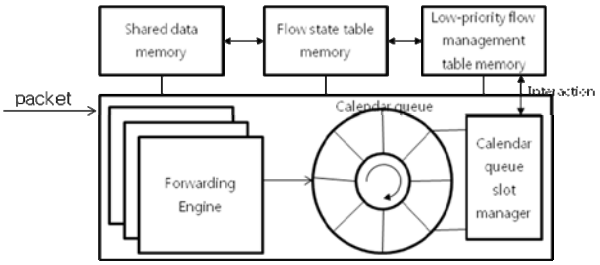

Figure 3. Modified switch architecture.

These non-guaranteed service packets are managed by the Low-priority flow management table and they are sorted by input ports and TTS values in the buffer. Guaranteed service packets are inserted into timeslot firstly and non-guaranteed service packets are inserted if there is an available space in a timeslot. Sorting by TTS it can guarantee the packet-ordering. In addition it can guarantee the fairness between input ports by finding an appropriate packet to an available space of a timeslot in DRR (Deficit Round Robin) scheme and sorting by input ports.

The calendar queue slot manager is for the management of the remain-space in the time slot. It has to keep tracking the size of the space of the time slot for every packet insertion in order to know the remain-space of the timeslot. Slot size $Slot_{size}(n)$ can be calculated recursively like in the equation (1).

$$Slot_{size}(n) = \{ Slot_{size}(n-1) - PL(i\text{-}th\ insert) \}, \qquad (1)$$

where $PL(i\text{-}th\ insert)$ is the length of *i-th* inserted packet.

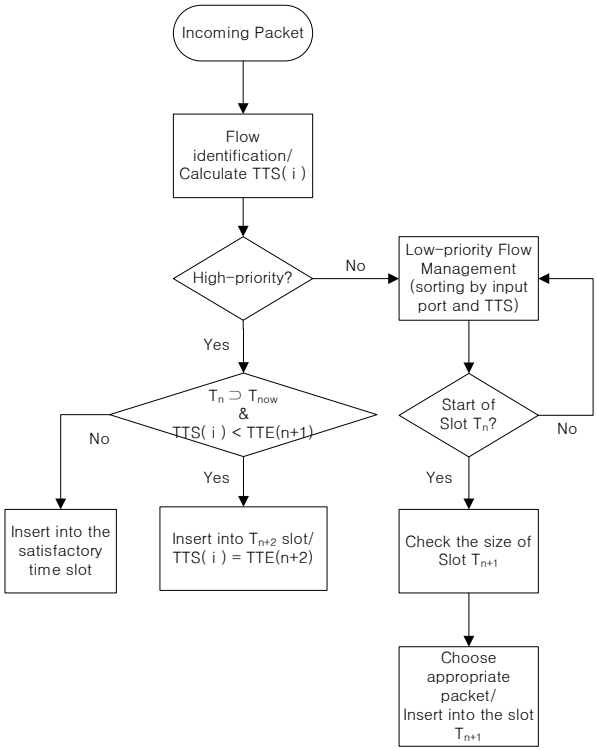

Figure 4. Scheduling flow-chart according to the priority.

For every incoming packet it identifies flow id and calculates TTS value. TTS value is calculated based on the traditional rate-proportional server operation in the equation (2).

$$F_i^k = \max(F_i^{k-1}, t_i^k) + \frac{l_i^k}{\rho_i^k}, \qquad (2)$$

where t_i^k is an insertion-time to calendar queue, l_i^k is an packet length, and ρ_i^k is a guaranteed bandwidth for flow which the incoming packet is included in. In next step it verifies whether the incoming packet is guaranteed service or non-guaranteed service. If it is a packet of guaranteed service

then it should be inserted into a timeslot of calendar queue according to its TTS value. If it is a packet of non-guaranteed service then it waits in the buffer and is managed by the low-priority management table.

In order to insert non-guaranteed service packet into the remain-space of a timeslot where the size of space is variant we define the size of the remain-space of the next timeslot at a point of time where the current serviced timeslot is changed. For example as shown in Figure 5, if the scheduler is just start service the timeslot T_n, then it defines the size of the remain-space of the timeslot T_{n+1}. Also when the scheduler inserts a packet of guaranteed service into a timeslot, it should be considered that the remain-space of the timeslot is defined or not. If the remain-space is already defined then the incoming packet should be inserted the next timeslot.

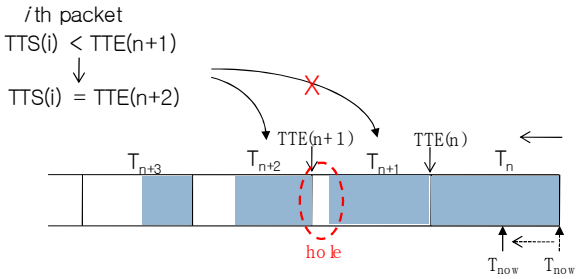

Figure 5. Timing to define the hole.

For example, when the scheduler services the timeslot T_n, the remain-space of the timeslot T_{n+1} is already defined. So, if the TTS value of an incoming packet indicates that it should be inserted into the timeslot T_{n+1} then it cannot be inserted into the timeslot T_{n+1} but inserted into the timeslot T_{n+2}. In addition the TTS value should be updated to the end time of the timeslot T_{n+2} (TTE(n+2)) in order to maintain the recursive equation of the TTS value.

The point of time where the current service timeslot is changed is also a time to insert a packet of non-guaranteed service into a timeslot. So at every this point of time the calendar queue manager defines the size of the remain-space of a timeslot and informs it to the low-priority flow management table. And then the scheduler picks up the very appropriate-sized packet in the low-priority flow management table to the size of the remain-space.

4. Performance Evaluation

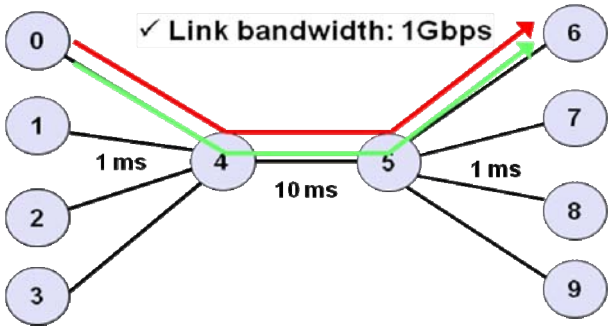

Figure 6. NS2 simulation topoloty

We perform a simulation using ns2 simulator to evaluate the performance of our proposed scheduler. Figure 6 shows our simulation topology. We suppose that the link bandwidth is 1Gpbs. The red line represent for class 1 traffic which requires strict QoS and this traffic is 200byte CBR traffic like VoIP traffic. The green line represent for class 2 traffic which is non-guaranteed traffic such as best effort traffic and this traffic is VBR traffic with variable packet length. The calendar queue as 2048 time slots and the size of each time slot is 32us. We compare the performance of our proposed scheduler with traditional calendar queue algorithm.

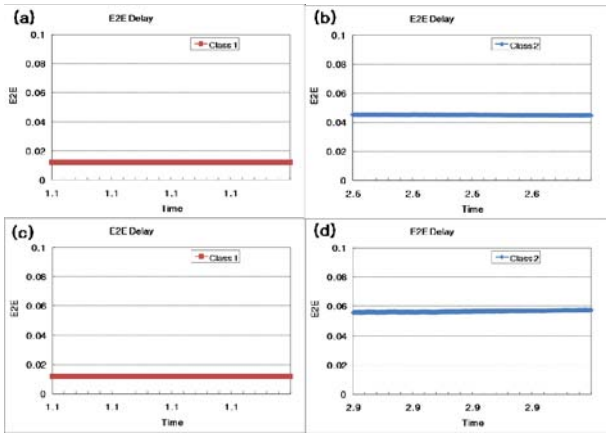

Figure 7. End-to-end delay when traffic load is 1.0

In Figure 7, the upper graphs (a), (b) are the end-to-end delay of our proposed algorithm and the below graphs (c), (d) are the end-to-end delay of the traditional algorithm. Both of the algorithms can support the guaranteed service (class 1) in the similar level but in case of non-guaranteed service (class 2) the proposed algorithm shows lower end-to-end delay. In the traditional algorithm non-guaranteed service packets should be waited during the hole at each time slot. However, in the proposed algorithm there is less wasted space of the time slot and the non-guaranteed service packets do not have to wait as much. This results in the lower end-to-end delay.

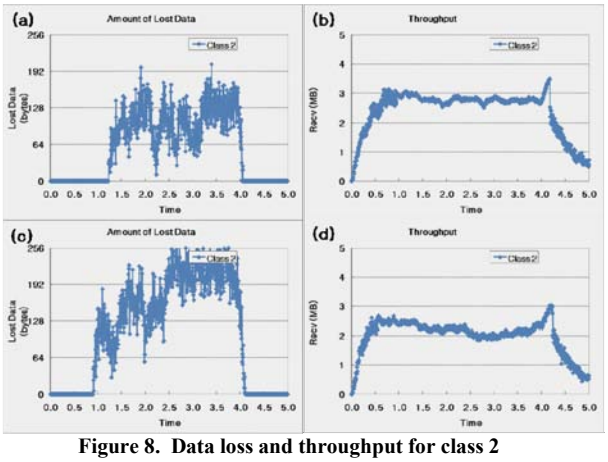

Figure 8. Data loss and throughput for class 2

We compare the data loss rate and the throughput of class 2 traffic. For the class 1 there is no data loss and throughput shows similar level in both algorithms, so we only compare

about class 2 traffic. The proposed algorithm shows lower data loss and higher throughput in Figure 7. This is because that we utilize the hole more effectively by searching appropriate packets of class 2. Traditional algorithm just abandons the hole as empty. However the proposed algorithm is trying to fulfill the time slots as full as possible. This difference makes the higher throughput.

5. Conclusions

In this paper we propose a QoS scheduler which is based on calendar queue. It can guarantee the QoS of the guaranteed service and also improve the end-to-end delay and the throughput of the non-guaranteed service by fulfilling the hole with non-guaranteed service traffic as full as possible. This can be more effective when the size of time slot is smaller. This scheduler is simple to implement and easy to manage the buffer, so it can be a candidate scheduler of the per-flow based switch. For the further works the searching algorithm to find appropriate sized packet to the hole is also important.

REFERENCES

[1] R. Brown, "Calendar Queues: A Fast 0(1) Priority Queue Implementation for the Simulation Event Set Problem," *Comm. of the ACM*, vol. 31, no. 10, Oct. 1998, pp. 1220-1227.

[2] K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *Proc. IEEE INFOCOM*, vol. 2, May 1992, pp. 915-924.

[3] B.H. Choi and H.S. Park, "Rate Proportional SCFQ (RP-SCFQ) Algorithm for High-Speed Packet-Switched Networks," *ETRI Journal*, vol. 22, no. 3, Sept. 2000, pp. 1-9.

[4] D.Y. Kwak, N.S. Ko, B. Kim, and H.S. Park, "A New Starting Potential Fair Queuing Algorithm with O(1) Virtual Time Computation Complexity," ETRI Journal, vol. 25, no. 6, Dec. 2003, pp. 475-488.

[5] DF.M. Chiussi, A. Francini, and J.G. Kneuer, "Implementing Fair Queuing in ATM Switches, Part 2: The Logarithmic Calendar Queue," Proc. IEEE INFOCOM, vol. 1, Nov. 1997, pp. 519-525

[6] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching," ACM Trans. Computer Systems, vol. 9, no. 2, May 1991, pp. 101-124.

[7] Nam-Seok Ko, Sung-Back Hong, Kyung-Ho Lee, Hong-Shik Park, and Nam Kim, "Quality-of-Service Mechanisms for Flow-based Routers," ETRI Journal, vol.30, no.2, Apr. 2008, pp.183-193.