고정크기 패킷 네트워크 환경에서 할당율에 비례한 저지연 한계를 제공하는 계층적 라운드-로빈 알고리즘

(A Hierarchical Round-Robin Algorithm for Rate-Dependent Low Latency Bounds in Fixed-Sized Packet Networks)

편 기 현 †

(Kihyun Pyun)

요 약 보장서비스에서 실시간 패킷 스케쥴링 알고리즘은 높은 네트워크 유용도와 확장성있는 구현의 양쪽 모두를 성취해야만 한다. 여기서 네트워크 유용도는 승인하는 실시간 세션의 수를 나타낸다. 불행히도, 현존하는 스케쥴링 알고리즘은 확장성있는 구현에 문제점을 갖거나 성취할 수 있는 네트워크 유용도가 낮다. 가령 타임스템프에 기반한 알고리즘은 N이 세션의 수를 나타낼 때 O(log N) 스케쥴링 복잡도를 가진다. 반면 라운드-로빈 알고리즘은 O(1) 복잡도를 가지지만 성취할 수 있는 네트워크 유용도가 낮다. 이 논문은 확장성을 잃지 않으면서도 높은 네트워크 유용도를 성취할 수 있는 스케쥴링 알고리즘을 제안한다. 제안하는 알고리즘은 서로 다른 시간 구간 크기에 대해서 다중 라운드를 활용하는 계층적 라운드-로빈(H-RR) 알고리즘이다. 이 알고리즘은 우선 순위 큐를 사용하는 PGPS 알고리즘이 제공하는 것과 비슷한 지연의 한계를 제공하지만, 구현 복잡도가 상수라는 큰 장점을 갖는다.

키워드 : 실시간 서비스, 보장 서비스, 패킷 스케쥴링, 확장성

Abstract In the guaranteed service, a real-time scheduling algorithm must achieve both high level of network utilization and scalable implementation. Here, network utilization indicates the number of admitted real-time sessions. Unfortunately, existing scheduling algorithms either are lack of scalable implementation or can achieve low network utilization. For example, scheduling algorithms based on time-stamps have the problem of O(log N) scheduling complexity where N is the number of sessions. On the contrary, round-robin algorithms require O(1) complexity, but can achieve just a low level of network utilization. In this paper, we propose a scheduling algorithm that can achieve high network utilization without losing scalability. The proposed algorithm is a Hierarchical Round-Robin (H-RR) algorithm that utilizes multiple rounds with different interval sizes. It provides latency bounds similar to those by Packet-by-Packet Generalized Processor Sharing (PGPS) algorithm using a sorted-priority queue. However, H-RR requires a constant time for implementation.

Key words : Real-time service, guaranteed service, packet scheduling, scalability

1. Introduction

For the last several decades, guaranteed service has been studied to guarantee delay bounds of real-time packets through the network [1,2]. Since network bandwidth is reserved in advance for this guaranteed service, it is important to achieve a high level of network utilization, which here indicates that a large number of real-time sessions are admitted [3,4]. In addition, the most important challenge is to realize a scalable implementation.

Unfortunately, existing scheduling algorithms either are lack of scalable implementation or can achieve low network utilization. For example, scheduling algorithms based on time-stamps transmit packets using a sorted priority queue [4-20]. A

· This work was supported by Korea Research Foundation Grant (KRF-2004-003-D00240).

† 종신회원 : 전북대학교 전자정보공학부 교수
khpyun@chonbuuk.ac.kr

논문접수 : 2004년 5월 14일
심사완료 : 2004년 11월 22일

sorted priority can be implemented in O(1) complexity using special hardware such as a sequencer [21] or a systolic array [22]. However, such hardware drives up the cost of a router. In addition, it is not clear whether such a hardware can support tens of thousand of sessions. On the contrary, round-robin algorithms, such as Deficit Round Robin (DRR) [23], fix an order of service to achieve a scalable implementation. However, all previous round-robin algorithms can achieve just a low level of network utilization [19]. The main reason is that if the number of sessions increases, the guaranteed latency bounds become significantly large. Moreover, all sessions have similar large bounds, irrespective of their allocated bandwidths. For high network utilization, it is highly desirable to have short latency bounds for high-rate sessions.

In this paper, we propose a scheduling algorithm that guarantees low latency bounds for high-rate sessions without losing scalability. The proposed algorithm is a Hierarchical Round-Robin (H-RR) algorithm that utilizes multiple rounds with different interval sizes. H-RR maintains a rate-based hierarchy over session queues to *separate* sessions into a number of rate groups. Each rate group has its own interval size that is independent of lower rate groups. This separation results in short latency bounds for high rate sessions. In fact, it provides latency bounds similar to those by Packet-by-Packet Generalized Processor Sharing (PGPS) algorithm [5] using a sorted-priority queue. However, H-RR requires a constant time for implementation. We assume that all packets have the same size as in ATM networks throughout the paper.

Although the idea about using a hierarchy is not new, the usage and the purpose are different from one another. For instance, H-FSC proposed in [10] uses a hierarchy. However, the purpose is for distributing bandwidth *fairly* according to the hierarchy, and is independent of the guaranteed service. Specifically, H-FSC consists of a fair scheduler that has the purpose of fair bandwidth distribution and a real-time scheduler that has the purpose of the guaranteed service. The real-time scheduler does *not* utilize the hierarchy. H-FSC is better than H-RR in the viewpoint of network utilization. However, in terms of complexity, H-RR is superior to H-FSC since H-FSC has the complexity of O(log N) where N is the number of sessions. Similarly, H-PFQ proposed in [12] uses a hierarchy for fair bandwidth distribution and has O(log N) complexity.

This paper is organized as follows: In Section 2, we discuss limitations of previous round-robin algorithms. In Section 3, we propose H-RR and describe how it works. In Section 4, we analyze the scheduling complexity and guaranteed latencies for sessions. Lastly, Section 5 concludes this paper.

2. Long Latency Problem in Previous Round-Robin Algorithms

Most rate-based algorithms belong to the general class of schedulers called the Latency-Rate (*LR*) servers [19]. In *LR* servers, a scheduler is characterized by two parameters, a rate and a latency for each real-time session. The rate means the allocated service rate. The latency represents intuitively the worst-case delay needed to guarantee the rate. Once a scheduler is modeled by an *LR* server, we can derive the delay bound for a session. Note that low latencies result in low delay bounds. For a detailed discussion about *LR* servers, see [19].

In most cases, existing round-robin algorithms such as Deficit Round Robin (DRR) [23], DRR+ [23], and Nested DRR [24], provides very long latencies for sessions. First, let us consider DRR proposed in [23]. DRR is a variant of the weight round-robin that handles different packet sizes. Each session i is allocated a weight w_i. During a round, each backlogged session transmits the packet amount of the weight w_i one after another. In the worst-case, a session has to wait until one entire round is complete for its next service. In DRR, the latency for a session is proportional to the worst-case round interval, nearly irrespective of the allocated bandwidth. If there exists a large number of sessions. the interval becomes significantly large. Furthermore, just a few sessions can also incur large latencies. For example, if we allocate weight one for a 1 Kbps session, a 1 Mbps

session requires weight 1000. Thus, a few high-rate sessions result in a long round interval due to big weights.

Although DRR+ proposed in [23] slightly shortens latencies, its achievable latencies are still long. DRR+ separates real-time sessions from non-real-time sessions. The queues for real-time sessions are served before those for non-real-time sessions in each round. Thus, only real-time sessions (independent of non-real-time sessions) participate in constituting latencies for them. Note that, however, the interval of a round directly depends on the total number of real-time sessions. In addition, we cannot avoid allocating big weights for high-rate sessions.

Nested DRR proposed in [24] alleviates the long latency problem resulting from big weights for high-rate sessions. However, it does not handle the major problem that the worst-case round interval depends on the total number of sessions. While DRR serves the w_i amount of packets from session i as a whole, nested DRR interleaves the service of the w_i amount in such a way that just one basic unit, *e.g.*, one packet, is served in a series of sub-rounds. In the case of high-rate sessions, this interleaving shortens waiting times for their next service. Note that, however, nested DRR does not discriminate the service order between high- and low-rate sessions. Thus, a high-rate session still has to wait for *all* other sessions to finish their basic unit service.

3. Proposing Algorithm Description

In this section, we propose a hierarchical round-robin (H-RR) scheduling algorithm to provide low latencies for high-rate sessions. Note that the latencies for those sessions are not affected by either the total number of sessions or high rates.

The H-RR maintains a rate-based hierarchy over session queues. In the level 0, there is a node called the root, which represents the link. Each node n has a positive integer weight w_n to receive a weight-proportional bandwidth from the parent. The weight of the root is always one. A non-leaf node has the purpose of splitting its share of bandwidth into the children. A leaf node is used to serve a session. So, only a leaf node has a pointer to a session queue. The upper level a session is allocated a leaf node at, the larger bandwidth the session receives. To guarantee a throughput for a node, we enforce a limit on the worst-case weight sum $W^{\max}$ of siblings, *i.e.*, the children of the same parent. Be careful that $W^{\max}$ does not mean the weight sum of nodes at the same level. On the level l, there can exist $(W^{\max})^l$ nodes at the maximum. The value of $W^{\max}$ is chosen by the network administrator. A node with weight w at the level l is guaranteed $\frac{w}{(W^{\max})^l} r$ bandwidth where r is the link bandwidth. When a session requires a certain bandwidth, we determine the level at which a leaf node is constructed and then choose the weight value of the leaf node. Note the since we have fixed the worst-case weight sum at each level, we already know the minimum bandwidth of a node with weight 1 at each level. This knowledge enables us to choose the level of a leaf node to be constructed. After fixing the level, we can determine the weight in a straightforward manner.

Figure 1 illustrates an example hierarchy for a 100 Mbps link. In the hierarchy, the worst-case weight sum $W^{\max}$ is set to 10. The root has two children, A_1 and A_2. The node A_1 has two children, B_1 and B_2. The nodes B_1, B_2, and A_2 point to the queues for the session S_1, S_2, and

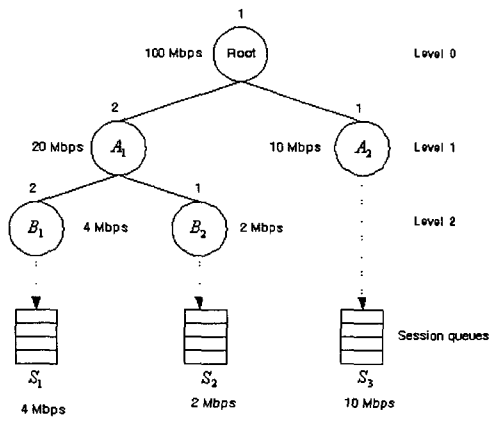

Figure 1 Example - a rate-based hierarchy

S_3, respectively. The weight of A_1 and B_1 is two. The nodes A_2 and B_2 have weight one. Then, the sessions S_1, S_2, and S_3 are guaranteed 4, 2, and 10 Mbps, respectively.

To avoid examining empty session queues, each non-leaf node maintains a service list, which is a list of indices to the backlogged children. We say that if a session queue has at least one packet, both the session and the leaf node that points to the queue are in a backlogged period. Similarly, we say that a non-leaf node is in a backlogged period if there exists a backlogged leaf node among all descendent nodes. A newly backlogged session may make all the nodes in the path from the leaf to the root newly backlogged. Entries of service lists are created by appending the index to the newly backlogged node. Similarly, entries of service lists are removed whenever a session queue becomes empty. In that case, the index to the leaf node that points to the session queue is removed from the service list of the parent. If the service list becomes null, the same removal operation is propagated upward recursively.

Let us see how the H-RR works using service lists. When the link is idle, service slots are given to the root one-by-one. A service slot is defined to be the time needed to transmit one packet in the link speed. If a non-leaf node including the root receives a service slot, it always passes the slot to the head node from the service list. In the case that the head node remains backlogged after using the current slot and have received the same number of service slots as its weight, its index is moved to the end of the list. At last, each service slot arrives at a leaf node. Then, the leaf node transmits one packet in the session queue during the slot.

```
Enqueueing:on arrival of packet p to session s
  enqueue(s, p);
  if(session s is newly backlogged)
    let the node n be leaf for session s;
    addToList(n);

addToList(n)
  let p be parent(n);
  append( list(p), n);
   CNUM_p = CNUM_p + 1;
  if(p != root and CNUM_p == 1)
    addToList( p );
```

Figure 2 Enqueueing module for H-RR

```
Dequeueing:
  while( TRUE ) do
    if(CNUM_root > 0 )
      giveServiceToken( root );
giveServiceToken(n)
  if(n is a non-leaf node)
    empty=giveServiceToken(head(list(n)));
    if(empty == TRUE )
      deletehead(list(n));
      CNUM_n = CNUM_n - 1;
      return (CNUM_n == 0);
    else
      c_n = c_n + 1;
      if(c_n == w_n )
        c_n = 0;
        movehead(list(n));
      return FALSE;
      else /* if n is a leaf node */
        send(queue(n));
        return isEmpty(queue(n));
```

Figure 3 Dequeueing module for H-RR

We present the pseudo-codes for enqueueing and dequeueing modules in Figure 2 and Figure 3, respectively. The notations used in the codes are summarized in Table 1. Note that all the functions can be implemented in a constant time.

Table 1 Notations used for pseudo-codes

Notation	Description
$CNUM_n$	The number of backlogged children of the node n.
c_n	The received number of service slots.
w_n	The weight of node n.
append(l, n)	Append an index to n to the service list l.
deletehead(l)	Delete the head from the service list l.
enqueue(s, p)	Enqueue p to the queue for sessions s.
isEmpty(q)	Returns TRUE if the queue q is empty.
list(n)	Returns the service list of node n.
movehead(l)	Moves the head to the end of the service list l.
parent(n)	Returns the parent node of node n.
queue(n)	Returns the session queue pointed by leaf node n.
send(q)	Send one packet from the queue q to the link.

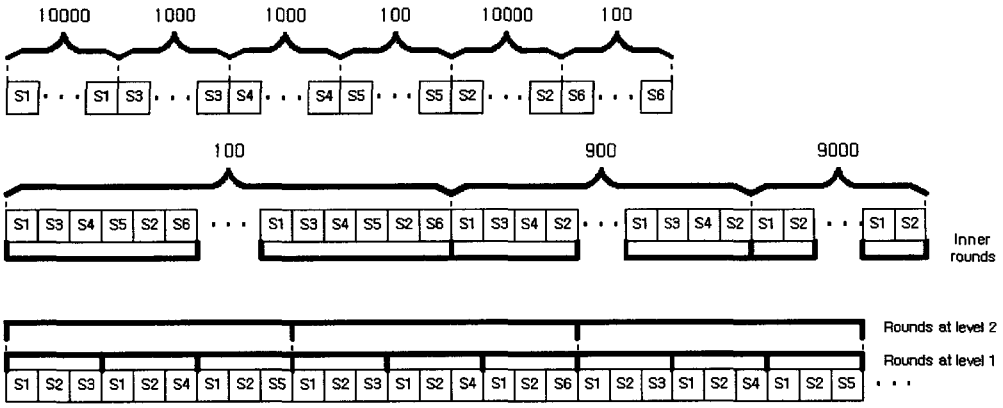

Figure 4 An execution of a round in DRR, nested DRR and H-RR

Figure 4 illustrates an example execution of one round in DRR, nested DRR and H-RR to give an intuition on how they operate in different ways. In the example, we assume that there exists two 10 Mbps sessions called $S1$ and $S2$, two 1 Mbps sessions called $S3$ and $S4$, and two 100 Kbps sessions called $S5$ and $S6$ in a 100 Mbps link. Also, we assume that the minimum bandwidth allocation unit is 1 bps and all packet sizes are equal to 1 Kb. In DRR and nested DRR, since the weight for a 1 Kbps session is set to one, the weights for each 100 Kbps, 1 Mbps and 10 Mbps session are set to 100, 1000 and 10000, respectively. In H-RR, the worst-case weight sum W^{max} is set to 10. Then, a leaf node at level l with weight one is guaranteed $(100/10^{l})$ Mbps. In this case, $S1$ and $S2$ are allocated a leaf node with weight one in the level 1, $S3$ and $S4$ are allocated a leaf node with weight one in the level 2, and so on.

4. Analytical Results

In this section, we present the scheduling complexity and latency bounds for sessions in H-RR.

From the pseudo-codes, H-RR algorithm requires O(D) scheduling complexity where D is the depth of the hierarchy. Since D is in practice a small constant, H-RR can be regarded is a constant time algorithm. For example, let 1 Kbps be the minimum allocation unit in a 1 Gbps link. If we set W^{max} to 10, the maximum depth becomes just six.

H-RR can provide a delay bound for a real-time session if the session is allocated a leaf node that receives more bandwidth than its traffic rate. Each session is accepted if the delay bound is short enough to satisfy its delay requirement. In terms of *LR* servers, since the latency for a session determines the delay bound [19], short latencies are better than long. It is impossible to provide short latencies for all sessions. We believe that it is highly desirable to give preference to high-rate sessions in such a way that those sessions receive short latencies, instead of long latencies for all sessions.

Theorem 1 characterizes H-RR by an *LR* server.

Theorem 1 Let session i be allocated a leaf node n at the level l in H-RR. Let t_0 be the beginning of a backlogged period of the session i. Let us denote by $W_i(t_1, t_2)$ the transmitted packet amount from the session queue during the interval $(t_1, t_2]$. Then, at any time t during the backlogged period,

$$W_i(t_0, t) \geq \max(0, \frac{w_n r}{(W^{max})^l}(t - t_0 - \frac{L(W^{max})^l}{r})) \quad (1)$$

where r is the link bandwidth, w_n is the weight of the node n, L is the packet size, and W^{max} is the worst-case weight sum.

Proof of Theorem 1. At a time t during the backlogged period, let the node n be in the K-th round from the time t_0. We regard the last round before the time t_0 as the 0-th round. Let us denote the ending time of the k-th round by r_k. Then, we can write t_0 and t as follows:

$$t_0 \in [r_0, r_1) \text{ and } t \in (r_{K-1}, r_K]. \quad (2)$$

During the k-th round, the packet amount of

(w_nQ) bytes is transmitted from the session queue pointed by the leaf node n. Thus, if we sum the transmitted amount from the queue starting from the first round to the K-th round,

$$W_n(r_0, r_K) = Kw_nL. \quad (3)$$

If $(W^{\max})^l$ service slots are given to the root, at least one round of service around the node n and its siblings is completed. Since the packet amount of L bytes can be transmitted at the maximum during a service slot, for each interval $(r_{k-1}, r_k]$,

$$r_k - r_{k-1} \le \frac{1}{r} L(W^{\max})^l. \quad (4)$$

By summing over k from Equation (4),

$$r_K - r_0 \le \frac{K}{r} L(W^{\max})^l. \quad (5)$$

From Equation (6),

$$K \ge \frac{(r_K - r_0)r}{L(W^{\max})^l}. \quad (6)$$

From Equation (3) and Equation (6),

$$W_n(r_0, r_K) \ge \frac{w_nLr}{L(W^{\max})^l}(r_K - r_0). \quad (7)$$

From Equation (2), $W_i(t_0, t)$ is no less than the accumulated served amount until the end of the K-th round minus the served amount at the K-th round. That is,

$$\begin{aligned} & W_n(t_0, t) \\ \ge\ & W_n(r_0, r_K) - w_nL \\ \ge\ & \frac{w_nr}{(W^{\max})^l}(r_K - r_0) - w_nL \\ =\ & \frac{w_nr}{(W^{\max})^l}\left(r_K - r_0 - \frac{(W^{\max})^l w_nL}{w_nr}\right) \\ \ge\ & \max\left(0, \frac{w_nr}{(W^{\max})^l}\left(r_K - r_0 - \frac{(W^{\max})^l L}{r}\right)\right) \\ \ge\ & \max\left(0, \frac{w_nr}{(W^{\max})^l}\left(t - t_0 - \frac{(W^{\max})^l L}{r}\right)\right). \end{aligned}$$

□

The theorem tells that for the session i, H-RR can be characterized by the LR server with the rate $w_nr/(W^{\max})^l$ bps and the latency $L(W^{\max})^l/r$. Table 2 compares latency and complexity of H-RR with those of existing schedulers [19]. Low latency is desirable to prevent a session from over-provisioning bandwidth to achieve a low delay bound of the session. For example, suppose that a 50 Kbps session requires 100 ms delay bound. If a scheduler can guarantee only 200 ms delay bound for 50 Kbps sessions, the scheduler may reserve 100 Kbps bandwidth for a shorter delay bound of 100 ms. This bandwidth waste results in low network utilization. Table 2 shows that while DRR gives the *same* latency for all sessions, H-RR can give *different* latencies for different sessions.

Let us compare the latencies guaranteed by PGPS, DRR, and H-RR by an example. Suppose that the link bandwidth is 1 Gbps. Let the minimum bandwidth allocation unit be 1 Kbps. The packet size is assumed to be 1 Kb. In the case of H-RR, we set the worst-case weight sum $W^{\max}$ to 10. Then, Table 3 shows the latencies guaranteed by PGPS, H-RR, and DRR for each session with varying bandwidth requirements at the worst-case. In the table, DRR guarantees the same long latencies for any session, irrespective of bandwidth requirements. On the contrary, in both PGPS and H-RR, a short latency is guaranteed for a high-rate session although a relatively long latency is given to a low-rate session. Further, the latencies are the same in both algorithms in the example.

5. Conclusions

In this paper, we have proposed H-RR to provide rate-dependent low latencies for sessions with a

Table 2 Comparison of latency and complexity. ρ_i is the allocated rate for session i. L is the packet size. F is the total amount of data to be transmitted in a round. r is the link bandwidth. l is the level at which the leaf node is located.

scheduler	latency	complexity
PGPS	$\frac{L}{\rho_i} + \frac{L}{r}$	O(N)
VirtualClock	$\frac{L}{\rho_i} + \frac{L}{r}$	O(log N)
DRR	$\frac{F}{r}$	O(1)
H-RR	$\frac{L(W^{\max})^l}{r}$	O(1)

Table 3 The worst-case latencies for each session with varying bandwidth requirements

BW (bps)	10K	100K	1M	10M	100M
PGPS	100ms	10ms	1ms	100us	10us
H RR	100ms	10ms	1ms	100us	10us
DRR	1s	1s	1s	1s	1s

constant time implementation. We have shown that H-RR can provide a similar latency for a session compared to PGPS algorithm. Note that, however, PGPS has O(N) complexity where N is the number of sessions.

References

[1] S. Shenker, C. Partridge, and R. Gu´erin. *Specification of Guaranteed Quality of Service*, September 1997. RFC 2212.

[2] William Stallings. *High-Speed Networks: TCP/IP and ATM Design Principles*. Prentice Hall, 1998.

[3] H. Zhang. Service Disciplines for Guaranteed performance Service in Packet-Switching Networks. *Proc. IEEE*, 3(4):391-430, 1995.

[4] Hanrijanto Sariowan, Rene L. Cruz, and George C. Polyzos. SCED: A Generalized Scheduling Policy for Guaranteeing Quality-of-Service. *IEEE/ACM Tran. Networking*, 7(5):669-684, October 1999.

[5] A. K. J. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Service Networks: The Single Node Case. *IEEE/ACM Tran. Networking*, 1(3):344-357, June 1993.

[6] A. K. J. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Service Networks: The Multiple Node Case. *IEEE/ACM Tran. Networking*, (2):137-150, April 1994.

[7] S. Jamaloddin Golestani. Network Delay Analysis of a Class of Fair Queueing Algorithms. *IEEE JSAC*, 13(6):1057-1070, August 1995.

[8] Jon C.R. Bennett and Hui Zhang. WF2Q: Worst-Case Fair Weighted Fair Queueing. In *INFOCOM*, pages 120-128, 1996.

[9] Leonidas Georgiadis, Roch Gu´erin, Vinod Peris, and Kumar N. Sivarajan. Efficient Network QoS Provisioning Based on per Node Traffic Shaping. *IEEE/ACM Tran. Networking*, 4(4):482-501, August 1996.

[10] Ion Stoica, Hui Zhang, and T. S. Eugene Ng. A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Services. *IEEE/ACM Tran. Networking*, 8(2):185-199, 2000.

[11] N. Figueira and J. Pasquale. An Upper Bound on Delay for the Virtual-Clock Service Discipline. *IEEE/ACM Tran. Networking*, 3(4):399-408, August 1995.

[12] J.C.R. Bennett and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. *IEEE/ACM Tran. Networking*, 5(5):675-689, October 1997.

[13] S. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. In *INFOCOM*, pages 636-646, 1994.

[14] Pawan Goyal, Harrick M. Vin, and Haichen Cheng. Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. *IEEE/ACM Tran. Networking*, 5(5):690-704, October 1997.

[15] Norival R. Figueira and Joseph Pasquale. A Schedulability Condition for Deadline-Ordered Service Disciplines. *IEEE/ACM Tran. Networking*, 5(2):232-244, April 1997.

[16] Pawan Goyal. *Packet Scheduling Algorithms for Integrated Services Networks*. PhD thesis, The University of Texas at Austin, August 1997.

[17] Debanjan Saha, Sarit Mukherjee, and Satish K. Tripathi. Multirate Scheduling of VBR Video Traffic in ATM Networks. *IEEE JSAC*, 15(6):1132-1147, August 1997.

[18] D. Stiliadis and A. Varma. Rate-Proportional Servers: A Design Methodology for Fair Queueing Algorithms. *IEEE/ACM Tran. Networking*, 6(2):164-174, April 1998.

[19] D. Stiliadis and A. Varma. Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms. *IEEE/ACM Tran. Networking*, 6(5):611-624, October 1998.

[20] Pawan Goyal and Harrick M. Vin. Generalized Guaranteed Rate Scheduling Algorithms: A Framework. *IEEE/ACM Tran. Networking*, 5(4):561-571, August 1997.

[21] Massoud R. Hashemi and Alberto Leon-Garcia. The Single-Queue Switch: A Building Block for Switches with Programmable Scheduling. *IEEE JSAC*, 15(5):785-794, June 1997.

[22] P. Lavoie and Y. Savaria. A systolic architecture for fast stack sequential decoders. *IEEE Tran. Communications*, 42(2/3/4):324-335, Feb./Mar./Apr. 1994.

[23] M. Shreedhar and George Varghese. Efficient Fair Queuing Using Deficit Round-Robin. *IEEE/ACM Tran. Networking*, 4(3):375-385, June 1996.

[24] Salil S.Kanhere and Harish Sethu. Fair, Efficient and Low-Latency Packet Scheduling Using Nested Deficit Round Robin. In *IEEE Workshop on High Performance Switching and Routing*, pages 6-10, 2001.

편 기 현

1995년 인하대학교 전자계산공학과(학사) 1997년 KAIST 전산학과(석사). 2002년 KAIST 전산학과(박사). 2002년~2003년 KAIST 전기및전자공학, 박사후 연구원 (Post Doctor). 2004년~현재 전북대학교 전자정보공학부 전임강사. 관심분야는 유무선 고품질 서비스, 유무선 패킷 스케줄링, 자원 관리, 저전력 시스템 소프트웨어