개선된 Latency의 DRR 분석

정회원 정 진 우*

Analysis of the DRR with Improved Latency

Jinoo Joung* ***Regular Member***

요 약

패킷 스위칭 네트워크 노드에서 정해진 대역폭을 보장해주는 스케줄링 기법 중 많은 수가 Latency-Rate (LR) server의 범주에 포함된다. LR server로 이루어진 네트워크는 각 노드에서의 latency의 합이 그대로 전체 네트워크의 latency가 된다는 성질을 가지고 있어서 각 노드의 latency를 구하면 전체 네트워크에서 겪을 수 있는 최대 delay를 쉽게 얻을 수 있다. Deficit Round Robin (DRR)은 이러한 LR server의 하나로 가장 구현이 간단하다는 장점 때문에 실제 여러 종류의 시스템에서 채택하였다. 본 연구에서는 이러한 DRR을 개선한 "DRR with Instant Service" (DRR-IS) 기법을 제시하고 분석하였다. DRR-IS가 LR server임을 증명하였고, 기존 DRR과 비교하여 전혀 복잡도의 증가가 없으면서 latency 성능은 30%가량 향상되었음을 밝혀내었다.

Key Words : DRR, Scheduling, QoS, Delay bound, Latency

ABSTRACT

Many of scheduling algorithms that provide a pre-defined bandwidth to a traffic flow fall into a category of Latency-rate (LR) server, the delay of whose network can be simply calculated by adding up individual "latencies" of each LR servers. Deficit Round Robin (DRR) is one of such LR servers and the simplest one to implement, so that it is adopted in many real systems. In this research we suggest an improved version of DRR, the DRR with Instant Service (DRR-IS), and analyze it. We have proved that the DRR-IS is still an LR server and have obtained its latency. The DRR-IS, compared with DRR, turns out to have the same complexity while provide about 30% better latency.

Ⅰ. 서 론

패킷 스위칭 네트워크에서의 품질보장(Quality of Service guarantee)의 문제는 오랜 시간 연구되어 왔으며 그 대표적인 approach중의 하나는, 세션 혹은 플로우라고 불리는 각 트래픽의 단위에 대하여 필요한 서비스 대역폭을 예약하고 이에 따라 네트워크 내에서 예약된 대역폭을 보장해 주는 방법인 IntServs이다. 정확하게 할당된 만큼의 대역폭을 특정 플로우의 트래픽에 효율적으로 적용시키는 스케줄링 알고리즘은 이러한 서비스 품질 보장 메커니즘에 있어서 가장 중요한 부분이며 이에 대하여 지난 10여 년간 많은 연구가 있었다. 초창기 연구 중 대표적인 것으로 Weighted Fair Queueing (WFQ)[1], Packetized Generalized Processor Sharing (PGPS)[2]과 이들의 variation을 들 수 있다. 이들 초기 스케줄링 알고리즘들은 Fluid-flow model을 이용하여 비트 레벨까지 정확하게 할당된 대역폭을 다시 packet-level로 approximate하여 보장해 준다는 장점이 있으나 알고리즘의 복잡도가 너무 크다. 이러한 이상적인 스케줄링 알고리즘들을 Sorted priority 방식이라고 하며 여러 가지 변형된 방식이 존재하지만 최

* 상명대학교 소프트웨어학부 (jjoung@smu.ac.kr)
논문번호 : KICS2005-10-418, 접수일자 : 2005년 10월 18일

선의 경우에도 그 복잡도는 O(log N) (N은 활성화된 플로우의 개수)에 달한다[3]. 간단한 라운드로빈 기반의 스케줄러들 중 Weighted Round Robin (WRR)은 homogeneous한 packet 크기를 가지는 환경에서는 적용 가능하나 packet 크기가 가변적인 일반적인 Internet환경에서는 정확하게 대역폭을 보장해 주지 못 한다. 이러한 문제를 해결하는, round robin 기반의 알고리즘인 Deficit Round Robin (DRR)은 복잡도를 크게 줄여주면서도 이상적인 알고리즘인 PGPS등에 상당히 근접하게 대역폭을 보장해 줄 수 있는 방식이다[4]. 이 후 DRR 스케줄링을 개선한 여러 가지 알고리즘들이 선 보였으나 그 근본 설계 취지에서 크게 벗어났다고 보기는 힘들다[5][6][7]. DRR로 대표되는 이러한 라운드로빈 방식의 스케줄링 알고리즘의 복잡도는 경우에 따라 O(1)까지 줄일 수 있다. 하지만 DRR의 경우에도 복잡도와 스케줄러의 성능 (여기서는 최대 지연시간)은 반비례한다. 즉, O(1)의 복잡도를 얻기 위해서는 DRR스케줄러가 상당히 조악하게 동작하여야 하며 이 경우 각 플로우가 보장 받는 최대 지연시간은 상당히 커져서 실제 멀티미디어 전송에 쓰기 힘들게 된다[8].

본 연구에서는 모든 round robin 방식의 스케줄링 알고리즘에 적용 가능한, Instant Service라고 하는 새로운 스케줄링 paradigm을 제안하고 이것을 DRR에 적용하고, 이를 정확히 분석하였다. 몇 가지 실제 scenario를 상정해서 성능을 직접 비교하여 본 결과 이 스케줄러 (DRR-IS)는 복잡도를 그대로 유지하면서도 기존 DRR과 비교하여 약 30% 이상의 성능을 개선한다.

Ⅱ. DRR and its latency

DRR은 활성화 되어있는 플로우들을 라운드로빈 방식으로 서비스하는 데에 있어서 플로우별로 deficit counter를 유지하면서 이 값을 이용하여 플로우 사이에 다르게 적용된 공정성을 보상해 주는 기법이다. 요구하는 대역폭에 따라 미리 결정된 quantum값이 플로우별로 주어져서 이 quantum값에 따라 플로우 간의 상대적인 가중치가 결정이 된다. Deficit counter 와 quantum값은 bit로 표시되며, 플로우 i에 대한 quantum값과, k번째 라운드가 끝난 시점에서의 deficit counter값을 ϕ_i와 D_i^k로 표시하기로 하자. 플로우 i의 k+1번째 라운드에서는, $\phi_i + D_i^k$ 보다 작은 범위 안에서 큐의 앞에서부터 차례로 패킷들이 서비스된다. 이 후 deficit counter값 (D_i^{k+1})을 서비스된 패킷들 크기의 총합과 $\phi_i + D_i^k$와의 차이로 경신한다. 만약 가장 앞에 있는 패킷의 크기가 $\phi_i + D_i^k$ 보다 크다면 해당 플로우는 해당 순번에서는 서비스를 받지 못하고, 다만 deficit counter값을 ϕ_i만큼 증가된 값으로 경신하게 된다. 이러한 방식의 DRR은 복잡도를 O(1)까지 낮출 수 있다는 장점 때문에 크게 각광을 받아서 널리 쓰이고 있다. 하지만 O(1)의 복잡도를 얻기 위해서는 ϕ_i를 플로우 i의 최대 패킷크기 (L_i) 보다 크게 유지해서 라운드에서 서비스 받지 못하는 일이 없어야 하며, 그렇지 않은 경우는 최악의 경우 O(N) 까지 복잡도가 증가한다[9].

DRR은 지연시간 최대치를 보장해 주는 여러 가지의 다른 스케줄러와 함께 latency-rate server (LR server)임이 증명 되었다[10]. LR server는 latency라는 parameter로 특징지어지는데 이 latency는 해당 플로우가 활성화되기 시작할 때 첫 번째 패킷이 겪을 수 있는 지연시간의 최대치로 해석할 수 있다. LR server로 구성된 네트워크는, 지연시간 측면에서 전체 네트워크를 하나의 equivalent한 노드로 볼 수 있다. 이 equivalent 노드의 latency는 실제 노드들의 latency의 합과 같으며, 이 성질을 이용해서 전체 네트워크에서 특정 플로우가 겪을 수 있는 최대 delay를 계산해 낼 수 있다는 커다란 장점을 가지고 있다. DRR의 latency는 다음과 같이 제시 되었다[10].

$$(3F - 2\phi_i)/r \qquad (1)$$

여기서 F는 frame length로 표현되며, 라운드가 한 번 돌아가면서 서비스되는 bit수의 평균치를 의미한다. r은 output link의 bandwidth를 말한다.

Ⅲ. DRR-IS and its analysis

본 연구에서 고려되는 모든 scheduler는 항상 store-and-forward 방식으로 동작한다. $A_i(\tau,t)$를 시간구간 $(\tau,t]$ 동안 플로우 i로부터 들어오는 데이터의 양이라 하고, $W_i(\tau,t)$를 같은 기간 동안 플로우 i가 서비스 받는 데이터의 양이라 하자. 이 두 값은 packet-by-packet 모델로 분석되는 본 연구에서는 항상 패킷의 입력이 완료되는 순간 혹은 전송이 완료되는 순간에만 증가한다. 한 편, 플로우 i가 시간

t에 backlog 되었다라고 하는 것은 $Q_i(t) = A_i(0,t) - W_i(0,t)$로 정의된 Queue length $Q_i(t)$가 0보다 큰 값을 가지는 것으로 정의된다. 이에 따라 비활성화 되어 있던 플로우의 패킷이 입력되는 순간 backlog가 시작되며, 플로우의 활성화는 곧 backlog 기간의 시작과 동일하다.

모든 라운드로빈 스케줄러들은 활성리스트(Active list)라고 하는 가상의 큐를 유지하고 있으면서 새로운 플로우가 활성화되는(backlog) 경우 해당 플로우를 활성리스트에 추가하고, 기존 플로우가 비활성화되는 경우에는 활성리스트에서 제거하는 방식으로 활성화되어있는 플로우들의 정보를 update한다. 여기서 일차원적으로 (linked list 방식으로) 구성되어있는 활성리스트의 어느 위치에 새로운 플로우를 추가하는가가 우리가 제안하는 Instant Service의 핵심이다. 기존방식의 라운드로빈 스케줄러는, 현재 서비스 받고 있는 플로우와는 상관없이 활성리스트의 시작(Head)과 끝(Tail)이 고정적으로 존재한다. 이 경우 새로운 플로우가 활성화되었을 때, 이 플로우가 기존 활성리스트의 새로운 끝이 되며, 이에 따라 라운드 내에서 서비스 받는 순서가 결정된다. Instant Service는 기존의 라운드로빈 방식의 스케줄러에 다음과 같은 추가적인 policy를 적용하여 불필요하게 라운드 내에서 기다리는 것을 방지한다.

1. 새롭게 backlog되는 플로우는 backlog 시점에 서비스 받고 있는 플로우에 대한 서비스가 종료되는 직 후에 서비스를 받는다.
2. 만약 하나 이상의 플로우가 동시에, 즉 기존 플로우의 한 번의 서비스기간 내에 backlog되면 이들 간의 순서는 임의로 정할 수 있다.

위에서 기술한 Instant Service 방식을 DRR에 적용하여 분석한다. 다음과 같은 시스템을 고려하자. 전체 플로우의 개수가 v이며 이 전체 플로우의 집합을 V라 하자($\mathrm{n}(V) = v$). 이 중 n개의 플로우가 backlog되어 있으며 따라서 이들 n개의 플로우가 DRR policy에 의해서 서비스 받고 있다. 이 플로우들의 집합을 N이라 하자. 새로운 플로우 i가 backlog되는 시점에 서비스를 받고 있는 플로우를 g라 하자. $g \in N$이다. 이 플로우 g가 서비스 받고 있는 중에 플로우 i와 함께 새롭게 backlog된 플로우의 수를 플로우 i를 포함해서 m이라 하고 이들 집합은 M이라 하자. 이 경우 플로우 g가 서비스 받고 있는 동안 같이 backlog된 M에 속한 플로우간의 순서는 임의로 정해질 수 있으며 최악의 경우 플로우 i는 이 중 가장 마지막이 될 수 있다. 여기서 더 나아가 DRR 시스템 (M,N)을 집합 N에 속한 플로우들이 backlog된 상태에서 M에 속한 플로우들이 플로우 $g(g \in N)$의 서비스 시간 내에 모두 backlog되는 시스템으로 정의한다. 또한 다음과 같이 특정 event들의 구체적인 발생 시점들에 대한 정의가 필요하다.

T_0: 플로우 i의 backlog이 시작된 시간, 즉 이 기간의 첫 번째 패킷이 들어온 시간.
t_k: 시스템의 k번째 라운드가 끝난 시간.

여기서 t_k는 새로운 플로우들이 backlog되기 이전에 이미 라운드에 포함된 n개의 플로우 중 마지막 플로우에 대한 서비스가 끝나는 시간을 말하며 이것은 플로우 g가 한 라운드의 마지막 플로우가 아니었다면 새로운 플로우들이 backlog된 이 후의 라운드에서도 마찬가지이다. 즉, 새롭게 backlog되는 m개의 플로우는 라운드의 중간에 끼어들어간다고 볼 수 있다.

τ_k: k번째 라운드 내에서 플로우 i가 받는 서비스의 끝나는 시간.

그림 1은 플로우 i가 동시에 backlog된 플로우들 중 가장 마지막에 서비스 받는다는 가정 하에 각각의 시점을 표시한 그림이다.

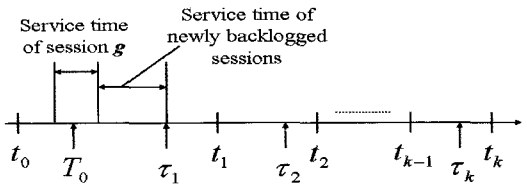

그림 1. DRR-IS 분석을 위한 시간 구간 설정

Lemma 1: 임의의 DRR시스템 (M,N), $\mathrm{n}(M) + \mathrm{n}(N) < v$, 에 대해서 다음의 부등식을 만족하는 (M',N'), $\mathrm{n}(M') + \mathrm{n}(N') \le v$, 이 항상 존재한다.

$$W_i^{(M,N)}(T_0,t) \ge W_i^{(M',N')}(T_0,t),$$
$$\text{for} \quad (M',N') \ne (M,N), \ \text{for} \ \forall t > T_0. \qquad (2)$$

Proof: 증명은 (2)를 만족하는 (M', N')의 예를 하나 보이는 것으로 충분하다. 원소의 개수 n(M)과 n(N)의 합이 전체 플로우 개수 v보다 작은 집합 M과 N에 대해서 항상 (M^*, N), $M^* = M \cup \{i^*\}$, 으로 표현되는 시스템이 존재한다. 관찰의 대상인 플로우 i는 항상 M의 원소이다. 따라서 i*와 i는 하나의 플로우 서비스 시간 내에 backlog되었으므로 i*가 i보다 먼저 서비스를 받을 수 있다. 이 경우 $\tau_k^{(M,N)} < \tau_k^{(M^*,N)}, \forall k > 0,$이다. 만약 i*가 i 이후에 서비스를 받는다고 하면 $\tau_k^{(M,N)} = \tau_k^{(M^*,N)}$이다. 어떤 경우이든지 $\tau_k^{(M,N)} \le \tau_k^{(M^*,N)}$가 성립한다.

한편 $W_i^{(M,N)}(T_0, \tau_k^{(M,N)}) = W_i^{(M^*,N)}(T_0, \tau_k^{(M^*,N)}), \forall k > 0,$ 이고 $W_i(T_0, t)$는 t에 대해서 monotonically increasing하는 함수이다. 따라서 $W_i^{(M,N)}(T_0, t) \ge W_i^{(M^*,N)}(T_0, t)$이고, Lemma가 성립한다.

Lemma 2: 주어진 임의의 N, $\mathrm{n}(N) \le \mathrm{n}(V)$, 에 대해서 다음의 부등식이 성립한다.

$$W_i^{(M,N)}(T_0, t) \ge W_i^{(V-N,N)}(T_0, t),$$
$$\text{for } \mathrm{n}(M) + \mathrm{n}(N) < \mathrm{n}(V), \text{ for } \forall t > T_0.$$

Proof: Lemma 2는 Lemma 1로부터 직접 유추 가능한 결과이다.

Lemma 2의 의미는, 플로우 n개가 backlog된 상황에서는 플로우 i를 포함한 나머지 전체 플로우 v-n개가 동시에 플로우 i와 함께 backlog되는 상황에서 W_i, 즉 플로우 i가 받는 서비스의 양이 최소화된다는 것이다.

Theorem 1:

$$W_i(T_0, t) \ge \max\left(0, \rho_i\left(t - T_0 - \frac{2F + \phi_{\max} - \phi_i}{r}\right)\right). \qquad (3)$$

여기서 $\phi_{\max} = \max_{j \in V, j \ne i} \phi_j$이다.

Proof: 위에서 정의한 바대로 T_0를 플로우 i가 backlog된 시점이라 하고 t_k를 k번째 round가 끝난 시간이라 하면, [4]에서 증명된 바와 같이 만약 플로우 i가 $(T_0, t_k]$ 동안 계속해서 backlog되어 있다면

$$W_i(T_0, t_k) \ge k\phi_i - D_i^k.$$

다음과 같은 집합들에 대하여 정의한다. A를 플로우 i가 backlog 된 라운드 이전부터 service받던 플로우들 가운데, 라운드 내에서 i 이후 service받게 되는 플로우들의 집합이라 하자. 비슷하게 B를 플로우 i가 backlog 되기 이전의 라운드부터 service 받던 플로우들 중, 라운드 내에서 i 이전에 service 되는 플로우들의 집합이라 하고, C를 플로우 i와 같은 시점에 backlog된 플로우들의 집합이라 하자. $A \cup B = N, C \cup \{i\} = M$이 성립한다. 여기서 같은 시점에 backlog되었다는 것은 플로우 i가 backlog되는 시점에 service를 받고 있던 플로우 g, $g \in B,$의 service round내에 backlog가 되었다는 뜻이다. 다음의 부등식이 성립한다.

$$\tau_1 - T_0 \le \frac{1}{r}\left[\max_{j \in B}\left(\phi_j + D_j^0 - D_j^1\right) + \sum_{j \in C}\left(\phi_j - D_j^1\right) + \left(\phi_i - D_i^1\right)\right].$$

$$\tau_2 - \tau_1 \le \frac{1}{r}\left[\sum_{j \in A}\left(\phi_j + D_j^0 - D_j^1\right) + \sum_{j \in B}\left(\phi_j + D_j^1 - D_j^2\right) + \sum_{j \in C}\left(\phi_j + D_j^1 - D_j^2\right) + \phi_i + D_i^1 - D_i^2\right].$$

$$\tau_k - \tau_{k-1} \le \frac{1}{r}\left[\sum_{A}\left(\phi_j + D_j^{k-2} - D_j^{k-1}\right) + \sum_{B}\left(\phi_j + D_j^{k-1} - D_j^k\right) + \sum_{C}\left(\phi_j + D_j^{k-1} - D_j^k\right) + \phi_i + D_i^{k-1} - D_i^k\right].$$

$\tau_l - \tau_{l-1}$ 항을 l=2부터 k까지 모두 더하면,

$$\tau_k - \tau_1 \le \frac{1}{r}\left[\sum_A D_j^0 + \sum_B D_j^1 + \sum_C D_j^1 + D_i^1 - \sum_A D_j^{k-1} - \sum_B D_j^k - \sum_C D_j^k - D_i^k + \left(\sum_A \phi_j + \sum_B \phi_j + \sum_C \phi_j + \phi_i\right)(k-1)\right].$$

이것을 다시 $\tau_1 - T_0$과 더하면,

$$\tau_k - T_0 \le \frac{1}{r}\left[\max_{j \in B}\left(\phi_j + D_j^0 - D_j^1\right) + \sum_A D_j^0 + \sum_B D_j^1 + \sum_C \phi_j + \phi_i - \left(\sum_A D_j^{k-1} + \sum_B D_j^k + \sum_C D_j^k + D_i^k\right) + (k-1)F\right].$$

여기서 Lemma 2에 따라 $A \cup B \cup C \cup \{i\}$를 전체 플로우의 집합이라 할 때 위의 부등식이 성립한다. 또,

$$\max_{j\in B}(\phi_j + D_j^0 - D_j^1) + \sum_B D_j^1$$
$$= \max_{j\in B}\phi_j + \max_{j\in B}(D_j^0 - D_j^1) + \sum_B D_j^1$$

이며, $\max_{j\in B}(D_j^0 - D_j^1) = D_g^0 - D_g^1,\ g \in B,$ 라고 했을 때,

$\max_{j\in B}(D_j^0 - D_j^1) + \sum_B D_j^1 = D_g^0 - D_g^1 + \sum_B D_j^1 \le \sum_{j\in B}\phi_j$ 이다.

한 편 $\sum_{j\in A} D_j^0 \le \sum_{j\in A}\phi_j$ 이므로,

$$\tau_k - T_0 \le \frac{1}{r}\left[\max_{j\in B}\phi_j + \sum_A \phi_j + \sum_B \phi_j + \sum_C \phi_j + \phi_i + (k-1)F\right]$$
$$\le \frac{1}{r}\left[\phi_{\max, j\neq i} + kF\right]$$ 이고,

따라서 $k \ge \frac{1}{F}\left[r(\tau_k - T_0) - \phi_{\max, j\neq i}\right]$ 이다.

위에서 구한 k의 부등식을 바탕으로, 다음을 얻을 수 있다.

$$W_i(T_0, \tau_k) = W_i(T_0, t_k)$$
$$\ge \frac{1}{F}\left[r(\tau_k - T_0) - \phi_{\max}\right]\phi_i - D_i^k$$
$$\ge \rho_i(\tau_k - T_0) - \frac{\phi_{\max}\phi_i}{F} - D_i^k.$$

여기서 두 가지의 경우로 나누어서 고려한다.

Case 1: 플로우 i의 k 번째 라운드에서의 데이터 전송량이 $(\phi_i - D_i^k)$ 보다 큰 경우.

이 경우 최대 데이터 전송량은 $2\phi_i - D_i^k$ 이다. 플로우 i가 서비스 받는 시점 이전의 모든 t에 대해,

$$W_i(T_o, t) \ge$$
$$\max\left(0, \rho_i(\tau_k - T_0) - \frac{\phi_i\cdot\phi_{\max}}{F} - D_i^k - 2\phi_i + D_i^k\right)$$
$$= \max\left(0, \rho_i(\tau_k - T_0) - \frac{r}{F}\cdot\phi_i\left(\frac{\phi_{\max}}{r} + \frac{2F}{r}\right)\right)$$

최소 $(\phi_i - D_i^k)$의 데이터가 전송되므로,

$t \le \tau_k - \left(\frac{\phi_i - D_i^k}{r}\right)$, $\tau_k \ge t + \frac{\phi_i}{r}$, 이고 따라서,

$$W_i(T_o, t) \ge \max\left(0, \rho_i\left(t - T_0 - \frac{1}{r}(2F + \phi_{\max} - \phi_i)\right)\right).$$

Case 2: 플로우 i의 k 번째 라운드에서의 데이터 전송량이 $(\phi_i - D_i^k)$ 이하인 경우.

플로우 i가 서비스 받는 시점 이전의 모든 t에 대해, $W_i(T_0, t) \ge$

$$\max\left(0, \rho_i(\tau_k - T_0) - \frac{\phi_i\cdot\phi_{\max}}{F} - D_i^k - \phi_i + D_i^k\right)$$
$$= \max\left(0, \rho_i(\tau_k - T_0) - \frac{\phi_i\cdot\phi_{\max}}{F} - \phi_i\right)$$
$$= \max\left(0, \rho_i\left(\tau_k - T_0 - \frac{\phi_{\max}}{F} - \frac{F}{r}\right)\right).$$

이 경우는 $\tau_k \ge t$ 이며,

$$W_i(T_0, t) \ge \max\left(0, \rho_i\left(t - T_0 - \left(\frac{F + \phi_{\max}}{r}\right)\right)\right).$$

$F > \phi_i$ 이므로 Case 1이 최종적으로 Theorem을 도출한다.

Theorem 1과 LR server의 정의에 따라 다음의 결론을 얻을 수 있다.

Corollary 1: Deficit Round Robin with Instant Service (DRR-IS)는 Latency-Rate Server이며 그 latency는 $(2F + \phi_{max} - \phi_i)/r$ 이다.

Theorem 1에서 구한 latency가 tight한 bound인지 확인하기 위해서는, 해당 부등식을 만족시키는 경우를 한 가지 찾아내는 것으로 충분하다. 다음의 경우를 고려해 보자. 플로우 i가 backlog되는 시점 T_0에 v-1개의 플로우, 즉 나머지 모든 플로우가 이미 backlog되어 있다. T_0는 또한, 정확하게 플로우 g의 서비스의 시작시간이다. 플로우 g의 서비스가 끝나면 DRR-IS policy에 따라 플로우 i의 서비스가 시작되는데 플로우 i의 최초 패킷의 길이는 $\Delta\phi_i$이고 두 번째 패킷의 길이는 ϕ_i이다. 이런 경우 첫 번째 라운드에서 플로우 i는 첫 번째 패킷만, 두 번째 라운드에서는 두 번째 패킷만 서비스 받는다. T_0에서부터 첫 번째 라운드가 끝나는 데까지 걸리는 시간은

$$\left(\phi_g + D_g^0 - D_g^1 + \Delta\phi_i + \sum_{j\neq i, j\neq g}(\phi_j + D_j^0 - D_j^1)\right)/r,$$

두 번째 라운드 시작부터 플로우 i의 두 번째 패킷이 서비스 받는데 까지 걸리는 시간은 $(\phi_g + D_g^1 - D_g^2 + \phi_i)/r$ 이다. 여기서

$D_j^1 = 0, \ \forall j \neq i, \ \forall j \neq g,$

$D_g^2 = 0,$

$D_j^0 = \phi_j - \Delta\phi_j, \forall j \neq i,$

라 하고, 더 나아가 $\max_{j \neq i} \phi_j = \phi_g$, $\Delta\phi_i = \Delta\phi_g << F$ 이라 하면, T_0 에서 시작해서 두 번째 패킷이 서비스를 받는데 까지 걸리는 시간은

$$\left(2\sum_{j \neq i}\phi_j + \phi_g + \phi_i\right)/r = \left(2\sum_{j=1}^{v}\phi_j + \phi_{\max} - \phi_i\right)/r.$$

따라서, 이러한 시나리오에서의 latency는 $(2F + \phi_{\max} - \phi_i)/r$ 로, Theorem 1에서 제시한 latency와 일치한다. 다른 경우로, 단 하나의 플로우 g 만이 backlog되어 있는 상태에서 플로우 i를 포함한 v-1개의 모든 나머지 플로우가 동시에 backlog되는 경우에도 마찬가지로 Theorem 1의 latency값이 정확히 구하여 진다는 것이 파악되었다.

지연시간과 더불어 scheduler의 중요한 성능지표인 공정성 (Fairness)에 대해서 알아보자. 이 공정성을 수량화하여 측정하는 지표로 *Fairness measure (FM)*가, 다음과 같이 두 플로우가 모두 backlog되어 있는 임의의 기간 동안 받는 normalized 서비스 차이의 최대값으로 제안되었다[3].

$$FM_{i,j} = \frac{1}{r}\max_{t_2 > t_1}\left|\frac{W_i(t_1,t_2)}{\rho_i / \sum \rho_k} - \frac{W_j(t_1,t_2)}{\rho_j / \sum \rho_k}\right|.$$

여기서 ρ_i 는 플로우 i가 할당받은 bandwidth이다. DRR의 *FM*은 [8]에서 구한 바와 같이 F, r, 그리고 플로우 i와 j의 최대 패킷길이에만 dependent하다. DRR-IS는 DRR과는 최초 backlog 시점 근처에서만 동작이 다르므로, DRR의 *FM*과 같은 값을 가진다. 따라서 공정성 측면에서 DRR-IS는 DRR과 동일한 성능을 보인다고 할 수 있다.

Ⅳ. Numerical results

DRR-IS의 latency를 다른 Latency-Rate server들과 비교하여 본다. 다음의 두 개의 시나리오를 고려한다. 등록된 모든 플로우가 backlog 되어있다고 가정하자.

첫 번째는 홈 네트워크 환경에서 Fast Ethernet 기술을 이용한, 동일한 다수의 실시간 멀티미디어 플로우 전송을 위해 네트워크 자원을 예약하여 이를 할당해 주는 경우이다. 링크의 대역폭 (r)은 100Mbps, 플로우는 8개로 모두 10Mbps의 대역폭을 요구한다. 그림 2는 첫 번째 시나리오를 각각 Packetized GPS, DRR, DRR-IS, Aliquem DRR[8]로 서비스했을 때를 비교한 결과이다. [8]에서는 최소 quantum 크기를 최대 패킷 크기로 설정하지 않고 일반적인 값 (흔히 패킷크기보다 작은 값)으로 하였을 때의 latency와 함께 구현방법 (Aliquem DRR)을 제시하였다. 그림에서 x축은 최대 패킷 크기이다. 각 플로우의 최대 패킷 크기는 Inter-frame gap에 해당하는 12byte와 Preamble 8byte를 포함해서 84byte부터 1538byte까지 변한다고 하였다. DRR과 DRR-IS에서 각 플로우의 quantum값은 모두 최대 패킷 크기와 동일하게 구현되었다고 가정한다. Aliquem DRR에서의 Quantum 크기는 100byte, 혹은 최대 패킷 크기가 100byte보다 작은 경우는 최대 패킷 크기로 설정하였다.

두 번째 시나리오는 비슷한 홈 네트워크 환경에서 30개의 플로우가 각각 1Mbps의 대역폭을, 5개의 다른 플로우가 각각 10Mbps를 요구하는 경우이다. 시나리오 1과 비교해서 전체 네트워크 utilization은 동일하지만 플로우의 수가 크게 증가하였다. 여기서 우리의 관찰 대상인 플로우는 1Mbps를 요구하는 플로우라고 하자. 그림 3은 이러한 시나리오 하에서 역시 각각 Packetized GPS, DRR, Aliquem DRR, 그리고 DRR-IS를 비교한 결과이다. 그림 2와 마찬가지로 최대 패킷 크기를 83byte에서 1538byte까지 변화시켰고, Aliquem DRR의 경우에 quantum 크기는 min(100byte, 최대 패킷크기)로 설정하고 계산하였다.

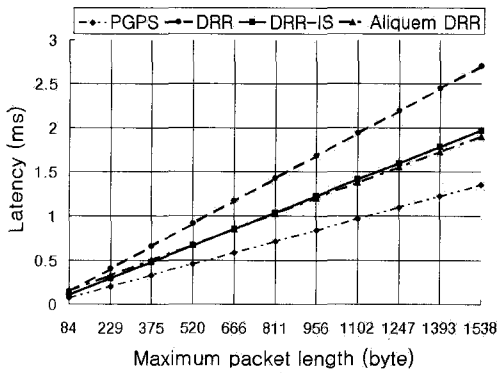

그림 2. Latencies of Schedulers at Scenario 1

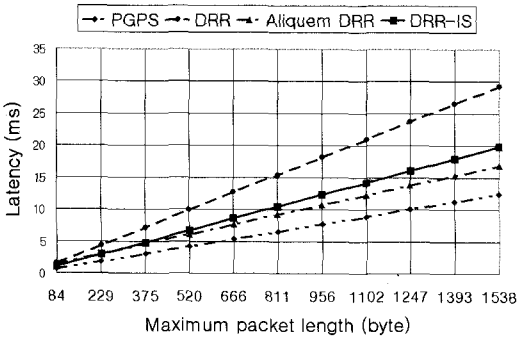

그림 3. Latency of Schedulers at Scenario 2

V. 결 론

두 개의 시나리오에서 공히 나타난 바와 같이, DRR-IS는 간단히 scheduling 순서를 바꾸는 것만으로 기존 DRR 알고리즘과 대비하여 30% 이상의 성능이 개선되는 효과를 보였다. 이것은 기존 DRR의 latency가 $3F$의 dominant 항을 가지는 것에 비해 DRR-IS의 latency는 $2F$의 dominant항을 가지는 것에서 기인한다. 하지만 Quantum 크기를 가변적으로 조절할 수 있는 Aliquem DRR의 성능과 비교해서는 비슷하거나 약간의 열세를 보인다. 이것은 그러나 Aliquem DRR에서 채용해야 하는 복잡한 2-level Active Queue List 하드웨어를 고려하면 큰 penalty는 아니라고 여겨진다. 더 나아가 Instant Service의 paradigm이 Aliquem DRR 에도 적용 가능하다는 것을 고려한다면 이 두 가지 알고리즘을 결합해서 최선의 DRR 스케줄링 알고리즘을 얻을 수 있을 것으로 사료된다.

간단히 활성리스트에의 삽입 위치를 바꾸는 것만으로 latency를 줄일 수 있는 것에 대해서는 다음과 같이 설명할 수 있다. WFQ이나 PGPS 등 가장 이상적인 스케줄러들은 bit단위의 미세한 대응을 하므로 상대적으로 좋은 latency 특성을 가지는 것이다. 이에 비해 실제적인 대부분의 LR server들은 플로우가 활성화되는 시점에서의 대응이 빠르지 못하다. 즉 플로우의 서비스 프로세스는 결국 할당된 서비스 rate이라는 equilibrium에 도달하지만 문제가 되는 것은 처음의 transient response이다. Instant Service 방식을 적용함으로써 다른 플로우들에 대한 penalty없이 새롭게 활성화되는 플로우가 필연적으로 받게 되는 불이익을 최소한으로 줄여주었고, latency 특성을 향상시켰다고 해석할 수 있다.

참 고 문 헌

[1] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," in *ACM SIGCOMM*, pp. 1-12, Sep. 1989.

[2] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344 - 357, June 1993.

[3] S. J. Golestani, "A Self-clocked Fair Queueing Scheme for Broadband Applications", in Proc. IEEE INFOCOM, pp. 636-646, 1994.

[4] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 375 - 385, June 1996.

[5] C. Guo, "SRR, an O(1) Time Complexity Packet Scheduler for Flows in Multi-Service Packet Networks," in *ACM SIGCOMM'01*, 2001.

[6] S. Ramabhadran and J. Pasquale, "Stratified Round Robin: A Low Complexity Packet Scheduler with Bandwidth Fairness and Bounded Delay," in *ACM SIGCOMM'03*, pp. 239-249, 2003.

[7] X. Yuan and Z. Duan, "FRR: a Proportional and Worst-Case Fair Round Robin scheduler", *IEEE INFOCOM'05*, 2005.

[8] L. Lenzini, E. Mingozzi, and G. Shea, "Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers," *IEEE/ACM Trans. Networking*, vol. 12, no. 4, pp. 681-693, Aug. 2004.

[9] S. S. Kanhere and H. Sethu, "On the latency bound of deficit round robin", in *Proceedings of the ICCCN*, Miami, Oct. 2002.

[10] D. Stiliadis and A. Varma, "Latency-Rate servers: A general model for analysis of traffic scheduling algorithms", *IEEE/ACM Trans. Networking*, vol. 6, no. 5 Oct. 1998.

정 진 우(Jinoo Joung) 정회원

1992년 KAIST 전기전자공학과 공학사

1997년 Polytechnic Univ., NY, USA, 공학박사 (Ph.D.)

1997년~2001년 삼성전자 중앙연구소

2001년~2005년 삼성종합기술원

2005년~현재 상명대학교