비 작업보존형 라운드로빈 스케줄러

정진우*

Non-Work Conserving Round Robin Schedulers

Jinoo Joung*

요 약

패킷 스위칭 네트워크에서의 QoS 보장을 위한 많은 연구가 지난 10여 년간 진행되었다. 이들 중 많은 수가 IntServs 기반의 플로우별 대역폭 할당과 보장을 위한 여러 가지 시그널링, 스케줄링 방법에 관한 것이나, 구현의 복잡성으로 인해 실제 네트워크에서 실현된 예가 드물다. 이러한 복잡성을 극복하기 위해서 최근 플로우 통합 (Flow Aggregation)이 제안된 바 있다. 통합된 플로우 기반의 스케줄링을 통해 지연시간을 보장해 주기 위해서는 플로우 간의 공정한 통합이 요구되며, 이를 위해서 스케줄러가 비 작업보존방식으로 동작해야 한다. 본고에서는 가장 간단하면서 널리 쓰이는 스케줄링 기법인 Deficit Round Robin을 변형한, 비 작업보존 방식의 Round Robin with Virtual Flow (RRVF)를 제안하고 이를 바탕으로 공정한 플로우간 통합을 시도하였다. RRVF가 보장하는 지연시간 최대치를 연구하였으며 이를 적용한 플로우 통합시의 지연시간 최대치를 구하였다. 이를 통해 RRVF로 플로우를 통합하는 경우 대역폭 할당이 간단해짐과 동시에 네트워크 전체의 지연시간도 줄어드는 것을 알아내었다.

ABSTRACT

There have been numerous researches regarding the QoS guarantee in packet switching networks. IntServs, based on a signaling mechanism and scheduling algorithms, suggesting promising solutions, yet has the crucial complexity problem so that not enough real implementations has been witnessed. Flow aggregation is suggested recently to overcome this issue. In order to aggregate flows fairly so that the latency of the aggregated flows is bound, however, a non-work conserving scheduler is necessary, which is not very popular because of its another inherent complexity. We suggest a non-work conserving scheduler, the Round Robin with Virtual Flow (RRVF), which is a variation of the popular Deficit Round Robin (DRR). We study the latency of the RRVF, and observe that the non-work conserving nature of the RRVF yields a slight disadvantage in terms of the latency, but after the aggregation, the latency is greatly reduced, so that the combined latency is reduced. We conclude that the flow aggregation through RRVF can actually reduce the complexity of the bandwidth allocation as well as the overall latency within a network.

Keyword

QoS, Scheduling, DRR, Flow Aggregation, Latency

Ⅰ. 서 론

정확하게 할당된 만큼의 대역폭을 특정 플로우 혹은 클래스의 트래픽에 효율적으로 적용시키는 스케줄링 알고리즘은 패킷 네트워크의 서비스 품질 보장 메커니즘에 있어서 가장 중요한 부분이며 이에 대하여

* 상명대학교 소프트웨어학부

접수일자 : 2005. 8. 18

지난 10여 년간 많은 연구가 있었다. 초창기 연구 중 대표적인 것으로 Weighted Fair Queueing (WFQ) [1], Generalized Processor Sharing [2] 등을 들 수 있다. 이들 초기 스케줄링 알고리즘의 특징은 비트 레벨까지 정확하게 할당된 대역폭을 보장해 주지만 알고리즘의 복잡도가 크다는 것이다. 이러한 이상적인 스케줄링 알고리즘들을 Sorted priority based 방식이라고 하며 여러 가지 변형된 방식이 존재하지만 최선의 경우에도 그 복잡도는 O(log N) (N은 활성화 된 플로우의 개수)에 달한다 [3]. 기존 라운드로빈 스케줄링 방식을 활용한 알고리즘인 Deficit Round Robin (DRR) 알고리즘은 이러한 복잡도를 크게 줄여주면서도 이상적인 알고리즘인 WFQ등에 상당히 근접하게 대역폭을 보장해 줄 수 있는 방식이다 [4]. 라운드로빈 방식의 스케줄링 알고리즘의 복잡도는 경우에 따라 O(1)까지 줄일 수 있다. 하지만 DRR의 경우에도 복잡도와 스케줄러의 성능(여기서는 지연 시간 최대치, latency)은 반비례한다. 즉, O(1)의 복잡도를 얻기 위해서는 DRR스케줄러가 상당히 조악하게 동작하여야 하며 이 경우 각 플로우가 보장받는 지연 시간 최대치는 상당히 커져서 실제 멀티미디어 전송에 쓰기 힘들게 된다 [5].

결국 서비스 품질 보장형 스케줄러는 전체 라우터나 스위치 설계에 있어서 중대한 요소이며, 시스템 내에서 처리해야할 플로우의 개수가 많아짐에 따라 복잡도가 증가하여 전체 시스템 성능의 bottleneck이 되는 부분이다. 이러한 문제를 해결하는 방법으로 플로우 통합 (Flow Aggregation)이 제시되었다 [6]. 여기서는, 라우터나 스위치의 특정 포트에서 출력되는 두 개 이상의 플로우들을 공정하게 통합하여 보내주면 다음번 라우터/스위치에서 하나의 플로우로 보고 처리를 하여도 각각의 플로우가 보장받는 지연 시간 최대치는 동일하게나 심지어는 작아질 수 있다는 것이 증명되었다. 공정하게 통합한다고 하는 것은, 간단히 설명하면, 각각의 플로우간의 기존에 할당받은 대역폭의 비율을 통합된 플로우내에서도 보장받는다는 것이다. 이를 위해, 만약 하나의 플로우가 활성화되어 있지 않은 경우에는 통합 플로우 내의 각 플로우가 할당받은 대역폭의 총합 이상의 속도로 전송되어지지 않을 필요가 있다. 이는 실제 출력포트의 대역폭이 통합되는 플로우의 총 요구 대역폭보다 보다 큰 경우 스케줄러가 비작업보존 (Non-Work Conserving) 방식으로 동작하여야 한다는 것을 의미한다. DRR을 비롯한 기존 라운드로빈 방식의 스케줄러들은 모두 작업보존형 (Work Conserving) 스케줄러들이다. 본고에서는 기존 DRR을 변형한 비 작업보존형 스케줄러를 제안하고 이를 바탕으로 플로우 통합을 이루며, 그 성능을 분석한다.

Ⅱ. DRR의 지연시간

DRR은 다른 지연시간 최대치를 보장해 주는 여러 가지의 다른 스케줄러와 함께 *latency-rate server* (LR server)임이 증명이 되었다 [8]. LR server는 *latency*라는 값으로 특징지어지는데 이 *latency*는 해당 플로우가 활성화되기 시작할 때 첫 번째 패킷이 겪을 수 있는 지연시간의 최대치로 해석할 수 있다. DRR이 보장하는 *latency* Θ_i는 여러 문헌에서 제시되었다. LR server 개념을 처음으로 확정한 [8]에서는 다음과 같이 간단한 값을 제시하였다.

$$\Theta_i = \frac{3F - 2Q_i}{C}. \tag{1}$$

여기서 $F = \sum_{k=1}^{N} Q_k$는 *frame length*로 표현되며, 라운드가 한번 돌아가면서 서비스되는 bit수의 평균치를 의미한다. (1)에서 알 수 있듯이 *latency, 지연시간 최대치*는 quantum 크기와 밀접하게 관계가 있다. 따라서 복잡도를 낮추기 위해서 Q_i가 커진다는 것은 그만큼 세밀하지 못한 제어를 의미하며, 이 경우 각 플로우가 보장받는 지연시간 최대치는 상당히 커진다. Quantum 값을 패킷 크기보다 작게 유지하여 지연 시간 최대치를 일정한 크기 이하로 낮추면서 구현상의 복잡도를 낮추고자 하는 시도들이 있었다[5][7]. [5]에서 구한 일반화된 DRR의 지연시간 최대치는 다음과 같다. 여기서 $\overline{L_i}$는 플로우 i의 최대 패킷 크기이다.

$$\Theta_i = \frac{1}{C}\left[(F - Q_i)\left(1 + \frac{\overline{L_i}}{Q_i}\right) + \sum_{j=1}^{N} \overline{L_j}\right]. \tag{2}$$

Ⅲ. 플로우 통합

Latency-rate server로 대변되는 일련의 스케줄러들은, leaky bucket등의 방법으로 입력되는 traffic의 대역폭이 적절하게 조절된다면 네트워크 노드 안에서 겪는 지연시간의 최대치를 보장해 주므로 결국 네트워크 전체에서의 지연시간의 최대치를 보장해 준다는 장점을 가지고 있어서 서비스 품질 (QoS) 보장에 있어서 필수적인 요소로 등장하였다.

IntServs의 복잡도를 낮추기 위한 연구 중, 라우터가 유지해야 하는 각각의 플로우 상태정보를 패킷의 헤더에 기재하여 라우터 내에서 상태를 유지해야하는 필요를 해소하여 라우터의 복잡도를 감소시키는 방법[9]과, 출력포트가 동일한 플로우들을 통합하여 다음 라우터에서는 하나의 플로우로 보고 처리하여도 각각의 플로우 내의 패킷들이 겪는 지연시간 최대치가 보장되는 방법[6]이 주목할 만한 접근 방법이다. [6]에서는 다음의 내용이 증명되었다. 플로우 i와 j가 라우터 R1의 출력포트 s를 통해 다음 라우터 R2로 전송되어진다고 가정하자. 이 때 출력포트 s를 거쳐 나오는, i와 j가 합쳐진 플로우를 g라 하기로 하자. 만약 i와 j간의 통합이 공정하게 이루어진다면, R2에서 i와 j를 다른 플로우로 인식하고 처리하는 경우와 하나의 플로우 g로 인식하고 처리하는 경우 각각의 플로우가 보장받는 지연 시간 최대치는 유사하다. 좀 더 정확하게 말한다면, i와 j가 다른 플로우로 인식되는 경우의 R2에서의 지연시간 최대치 Θ_i, Θ_j를 각각 Q_i, $\overline{L_i}$의 함수 $f(Q_i, \overline{L_i})$과 Q_j, $\overline{L_j}$의 함수 $f(Q_j, \overline{L_j})$로 나타내면, 하나의 통합된 플로우 g로 인식되는 경우의 지연시간 최대치 Θg는 $f\left(Q_j, Q_j, \max(\overline{L_i}, \overline{L_j})\right)$로 표현 할 수 있다는 것이다. 극단적인 경우 Q_i, $\overline{L_i}$와 Q_j, $\overline{L_j}$가 같은 경우, 두 플로우는 통합되어도 전혀 지연시간 관련해서는 손실이 없고 오히려 늘어난 대역폭만큼 이득을 볼 수 있게 된다. 여기서 플로우 간의 공정한 통합이 이루어지기 위해서는 아래의 필요조건이 성립하여야한다.

1) 만약 둘 중 하나의 플로우가 활성화 되어있지 않은 경우, 예를 들면 j의 패킷이 큐에 없는 경우는 플로우 i는 $\rho_i + \rho_j = \rho_g$이상의 대역폭으로 서비스 되어서는 안 된다.
2) 만약 두 플로우 모두 활성화되어 있다면, 즉 큐에 대기하고 있는 패킷이 모두 있는 경우에는 두 플로우에 할당된 대역폭에 비례해서 서비스된다는 전제하에 ρ_g이상의 대역폭으로 서비스 받을 수 있다.

일반적인 Latency-rate server들은 2번 조건을 모두 쉽게 만족시킨다. 문제는 1번 조건을 만족시키는 스케줄러를 만들기 위해서는 작업보존 특성은 포기해야 한다는 것이다. 만약 스케줄러에 두 개의 플로우만 존재하는데 둘 중 하나의 플로우가 비활성화 된다면 나머지 하나의 플로우를 서버의 서비스 대역폭 C가 아닌 ρ_g로 서비스해야 되고 이 경우 필연적으로 비 작업보존형 스케줄러가 필요하다는 것이다. 특히 그 낮은 복잡도로 인하여 각광을 받고 있는 라운드로빈 기반의 스케줄러들은 특성상 작업보존형일 수밖에 없다는 문제점이 있다. 본고에서는 라운드로빈 기반의 비 작업보존형 스케줄러를 제안함으로써 간단하고 공정한 플로우 통합을 가능하게 하고자 한다.

Round Robin with Virtual Flow (RRVF)

이 방법은 가장 간단하게 공정한 통합 스케줄러를 구현하는 방법이다. 스케줄러에 등록되어있는 모든 플로우의 집합 A 중 활성화 되어있는 플로우의 집합을 A_T라 하자. 또한 통합될 플로우의 집합을 A_G라 하자. 이 때 $C-\sum_i \rho_i, \forall\, i \in A_G \cup A_T$의 대역폭을 가지는 가상의 플로우 v를 상정해서 다른 플로우들과 마찬가지로 v를 라운드로빈 방식으로 처리하는 방법이다. 이렇게 가상의 플로우를 생각함으로써 A_G에 속한 플로우에 대해서 1번 조건을 명확히 만족시킬 수 있다. 대신 2번 조건에서 허용된 패킷 스위칭의 statistical multiplex 이득을 취할 수 없다는 단점이 있다.

Round Robin with Switched Virtual Flow (RRSVF)

여기서는 위의 RRVF에서 사용된 가상 플로우가 필요에 따라서 나타나거나 사라질 수 있다. 즉, A_G에 속한 플로우 중 어느 하나라도 비활성화 되는 경우에 가상 플로우는 $C-\sum_i \rho_i, \forall\, i \in A_G \cup A_T$의 대역폭으로 서비스되며, 그렇지 않은 경우에는 0의 대역폭으로 서비스 된다. 이렇게 함으로써 RRVF가 항상 C 이하의 대역폭으로 서비스하는 단점을 극복할 수 있다. 하지

만 플로우 활성 상태를 감시하고 있다가 필요에 따라 가상 플로우를 동작시키는 데에 따르는 구현상의 복잡성을 최소화 할 필요가 있다.

Ⅳ. RRVF과 RRSVF의 지연시간 최대치 및 플로우 통합 결과

Latency-Rate server에서 토큰 버킷 parameter (σ_i, ρ_i)로 제한된 플로우 i에 속한 패킷이 겪는 delay D_i의 최대치는 다음과 같이 표현된다[8].

$$D_i \le \frac{\sigma_i}{\rho_i} + \Theta_i.$$

여기서 Θ_i가 Latency, 혹은 지연시간 최대치이며, 특정 플로우 i가 활성화되는 시점에서 첫 번째 패킷이 실제로 서비스 받기 시작하는 시점까지의 시간의 최대치를 의미한다고 볼 수 있다. DRR의 지연시간 최대치로 (2)를 이용하기로 한다.

우리가 제안한 RRVF와 RRSVF는 모두 DRR을 기반하고 있으므로 (2)를 이용해서 지연시간 최대치를 계산할 수 있다. 먼저 RRVF 시스템의 지연시간 최대치를, 기존 DRR의 지연시간 최대치 Θ_i와 대비하여 Θ'_i라 하면 다음과 같은 관계가 성립된다.

$$\begin{aligned}
&\Theta_i' = \frac{1}{C}\left[(F'-Q_i')(1+\frac{\overline{L_i'}}{Q_i'}) + (\sum_{j=1}^{N}\overline{L_j})'\right],\\
&F' = F + Q_v,\\
&Q_i' = Q_i,\\
&\overline{L_i'} = \overline{L_i},\\
&(\sum_{j=1}^{N}\overline{L_j})' = \sum_{j=1}^{N}\overline{L_j} + \overline{L_v}.
\end{aligned} \tag{3}$$

여기서 Q_v와 $\overline{L_v}$는 가상 플로우 v의 quantum값과 최대 패킷 크기이다. Q_v는 다음과 같이 주어진다.

$$Q_v = \frac{Q_i}{\rho_i}(C - \sum_{j}^{N}\rho_j),\ \forall i.$$

(3)은 다음과 같이 정리된다.

$$\Theta_i' = \frac{1}{C}\left[(F+Q_v-Q_i)(1+\frac{\overline{L_i}}{Q_i}) + \sum_{j=1}^{N}\overline{L_j} + \overline{L_v}\right]. \tag{4}$$

(2)와 (4)를 이용해서 DRR과 RRVF 시스템의 지연시간 최대치의 차를 구하면 다음과 같다.

$$\Theta_i' - \Theta_i = \frac{1}{C}\left[Q_v(1+\frac{\overline{L_i}}{Q_i}) + \overline{L_v}\right].$$

여기서 $\overline{L_v}$는 구현 방법에 따라 임의의 작은 값으로 설정할 수 있으므로 큰 문제가 아니지만, $\frac{Q_v}{C}(1+\frac{\overline{L_i}}{Q_i})$ 항은 quantum 값을 충분히 작게 설정해서 구현하여도 일정한 값 이하로 작아지지 않는다. 이 부분에 대해서 6장에서 상세히 알아본다.

RRSVF의 경우는 두 가지 상태, 즉 스케줄러에 등록된 모든 플로우가 활성화 되어 있는 상태와 그렇지 않은 상태로 나누어서 생각해 볼 수 있다. 모든 플로우가 활성화되어 있다면 RRSVF의 지연시간 최대치는 (2)로, 그렇지 않는 경우의 지연시간 최대치는 (4)로 표현할 수 있다.

이번에는 RRVF와 RRSVF를 이용해서 공정하게 통합된 플로우가 다음 스케줄러에서 겪는 지연시간 최대치를 구해보자. 플로우 i가 통합되지 않은 상태로 DRR 스케줄러에서 서비스 되었을 때의 지연시연 최대치를 (2)의 Θ_i라 하고, 다른 플로우 j와 통합된 플로우 g로 DRR 스케줄러에서 서비스 되었을 때의 지연시간 최대치를 Θ_g라 한다면,

$$\Theta_g = \frac{1}{C}\left[(F-Q_g)(1+\frac{\overline{L_g}}{Q_g}) + \sum_{h=1}^{N^{Agg}}\overline{L_h}\right] \tag{5}$$

와 같다. 여기서 $Q_g = Q_i + Q_j$, $\overline{L_g} = \max\{\overline{L_i}, \overline{L_j}\}$, $N^{Agg} = N-1$, $\sum_{h=1}^{N^{Agg}}\overline{L_h} = \overline{L_1} + \overline{L_2} + \ldots + \overline{L_g} + \ldots + \overline{L_N}$로 표현된다. (두 개 이상의 플로우가 합쳐지는 경우도 그 지연시간 최대치를 구하는 식은 유사하다.)

V. 계산 결과

그림 1은 상기한 4장과 5장에서 설명된 시스템을 표현한 것이다. R1과 R2, 두 개의 라우터 혹은 스위치에 있는 여러 개의 출력포트와 이에 딸린 스케줄러 중 우리가 관심 있는 스케줄러를 각각 S1과 S2이라고 하자. S1은 RRVF를 채용한 플로우 통합 스케줄러이고 S2는 일반적인 DRR 스케줄러라고 가정하자. S1에 등록된 플로우 A 중 A_G에 속하는 플로우는 모두 R2의 S2로 향한다. 이를 통합된 플로우 g라 하면 S2는 자신을 통해서 서비스되는 다른 플로우들과 마찬가지로 g를 라운드로빈 방식으로 처리하는 것이다.

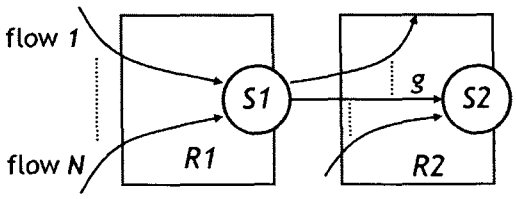


그림 1. 플로우 통합 시스템
Fig. 1. Flow Aggregation System

여기서 이러한 시스템에서의 지연시간 최대치를 구하여 일반 비 통합 시스템의 지연시간 최대치와 비교하면 네트워크상에서 한 번의 플로우 통합으로 만들어지는 효과를 파악할 수 있을 것이다.

다음의 두 개의 시나리오를 고려한다. 편의상 $A=A_T$, 즉 등록된 모든 플로우가 활성화 되어있다고 가정하자. 첫 번째는 홈 네트워크 환경에서 실시간 멀티미디어 전송을 위해 네트워크 자원을 예약하여 이를 할당해 주는 경우이다. 링크의 대역폭(C)은 100Mbps, S1을 지나가는 플로우는 총 8개로 모두 10Mbps의 대역폭을 요구한다. 이 중 4개의 플로우가 통합되어 S2에서 다른 4개의 10Mbps 대역폭의 플로우와 만난다. S1과 비 통합시스템에서의 S2의 경우, 각 플로우의 quantum값은 모두 100Byte로 동일하게 구현되었다고 가정한다. 통합 시스템에서의 S2에서는 Q_g는 400Byte, 나머지는 모두 100Byte로 구현한다고 가정한다. 가상 플로우의 최대 패킷길이 ($\overline{L_v}$)는 100Byte로 구현하였다고 하자. 표 1은 첫 번째 시나리오로 비 통합 시스템과 통합 시스템, 두 개의 시스템을 비교한 결과이다.

표 1. 시나리오1에 따른 두 시스템의 Latency
Table 1. Latencies of systems according to Scenario 1

	F	Q_i	Q_v	Q_g	Θ_i^{S1}	Θ_i^{S2}	Θ_i
비통합 시스템	800	100	-	-	1.856	1.856	3.712
통합 시스템			200	400	2.120	0.752	2.962

여기서 Θ_i^{S1}, Θ_i^{S2}는 각각 플로우 i가 S1, S2에서 겪는 지연시간의 최대치를 말하며, Θ_i는 이 둘을 합한 결과이다.

두 번째 시나리오는 다음과 같다. 첫 번째 시나리오와 마찬가지로 Fast Ethernet 환경이며, 링크 대역폭(C)은 100Mbps이라고 하자. S1에 12개의 플로우가 있는데 이 중 2개는 30Mbps의 대역폭을 요구하고, 나머지 10개는 1Mbps를 요구한다. 이 12개의 플로우 중 30Mbps 플로우 하나와 5개의 1Mbps 플로우가 통합되어 S2에서 다른 30Mbps 플로우 하나, 1Mbps 플로우 5개와 만난다. 따라서 비 통합 시스템인 경우 첫 번째 시나리오와 마찬가지로 S1과 S2가 동일하게 동작한다. 여기서 관심있는 플로우 i가 30Mbps 대역폭을 요구하는 플로우라 하고 Q_i를 300Byte로 구현했다고 하자. 표 2는 이에 따른 두 개의 시스템의 지연시간 최대치를 비교한 것이다.

표 2. 시나리오2에 따른 두 시스템의 Latency
Table 2. Latencies of systems according to Scenario 2

	F	Q_i	Q_v	Q_g	Θ_i^{S1}	Θ_i^{S2}	Θ_i
비통합 시스템	700	300	-	-	1.632	1.632	3.264
통합 시스템			300	350	1.784	0.748	2.532

VI. 결 론

본고에서는 플로우별로 대역폭을 할당해서 서비스해주는 기법에 대해서 고찰하였다. 라운드로빈 기반의 스케줄러, 특히 DRR과 그 지연시간 최대치에 대해서

알아보았고, 이어서 이들 플로우들을 통합하여서 다음 스케줄러로 넘겨주는 방법을 사용하기 위해서 필요한 스케줄링 기법인 RRVF와 RRSVF를 제시하고 이들의 성능에 대해서 알아보았다. 마지막으로 이러한 스케줄러를 통한 플로우 통합이 전체 네트워크 성능에 어떻게 영향을 미칠 것인가에 대해서 시나리오를 상정하여 분석하였다. 두 개의 시나리오에서 모두 비 작업보존 방식의 스케줄링으로 인해 약간의 지연시간 증가가 있지만 플로우 통합 이 후에는 오히려 지연시간이 크게 감소하였다. 이를 통해, 플로우를 통합함으로써 스케줄러의 복잡도를 낮추는 동시에 전체 지연시간의 최대치 또한 줄일 수 있다는 사실을 발견하였다. 이러한 사실은 현재 제시한 RRVF가 상대적으로 간단한, 개선의 여지가 있는 스케줄러라는 것을 감안하면 고무적인 결과이다.

참고문헌

[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 singlenode case," IEEE/ACM Trans. Networking, vol. 1, no. 3, pp. 344 - 357, June 1993.

[3] S. Golestani, "A Self-clocked Fair Queueing Scheme for Broadband Applications", in IEEE INFOCOM'94, 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] 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.

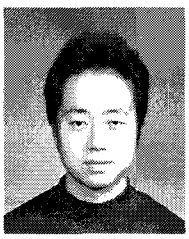
[6] J. A. Cobb, "Preserving Quality of Service guarantees in spite of flow aggregation", IEEE/ACM Trans. Networking, vol. 10, no. 1, Feb. 2002.

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

[8] 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.

[9] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in Proc. ACM SIGCOMM, pp. 81 - 94, Aug. 1999.

저자소개



정진우(Jinoo Joung)

1992 KAIST 전기전자공학과 공학사
1997 Polytechnic Univ., NY, USA, 공학박사 (Ph.D.)
1997~2001 삼성전자 중앙연구소
2001~2005 삼성종합기술원
2005~ 상명대학교
※관심분야: 컴퓨터 네트워크, System On Chip, Embedded systems, 무선 네트워크 및 통신