시스템 가상시간을 사용하지 않는 효율적인 Fair Queueing

준회원 이 준 엽*, 종신회원 이 승 형*

A Computationally-Efficient of Fair Queueing without Maintaining the System Virtual Time

Jun Yeop Lee*, Seung Hyong Rhee* *Regular Members*

요 약

패킷에 대한 스케줄링(Scheduling)은 현재 Best-effort 방식으로 운용되는 인터넷에서 서비스를 차별화하여 전송품질보장(QoS: Quality of Service)을 제공하기 위해 필수적인 기능이다. 지금까지 제안된 많은 스케줄링 알고리즘들은 간단한 구현에 의한 고속의 처리를 장점으로 하는 Round-Robin 방식과 정확한 서비스를 장점으로 하는 Fair Queueing 방식으로 나눌 수 있다. 특히, 기존의 Fair Queueing 알고리즘들은 모든 플로우(Flow)에 대한 상태를 스케줄러가 관리하기 때문에 계산량 및 구현상의 복잡성에 따른 문제를 안고 있다. 본 논문에서는 패킷의 서비스 순서 결정을 위한 계산을 각 플로우가 분산처리 하는 방식을 제안하여 스케줄러의 계산량을 대폭 줄이는 동시에 기존 Fair Queueing 알고리즘 수준의 정확성을 제공한다. 또한, 제안된 방식은 입력 큐에서의 스케줄링이 필요한 대형 라우터의 설계에 응용될 수 있다.

ABSTRACT

Packet scheduling is an essential function to gurantee a quality of service by differentiating services inthe Internet. Scheduling algorithms that have been suggested so far can be devided into Round-Robin methods and Fair Queueing methods Round-Robin methods have the advantage of high-speed processing through simple implementations, while Fair Queueing methods offer accurate services. Fair Queueing algorithms, however, have problems of computational overheads and implementation complexity as their schedulers manage the states of every flow. This paper suggests a new method in which each flow performs the calculation in a distributed way to decide the service order. Our algorithm significantly reduces the scheduler's computational overheads while providing the same level of accuracy with the previous Fair Queueing algorithms.

Ⅰ. 서 론

패킷에 대한 스케줄링(Scheduling)은 현재 Best-effort 방식으로 운용되는 인터넷에서 사용자들에게 서비스의 차별화 혹은 전송품질의 보장을 제공하기 위해 필수적인 기능이다. 지금까지 제안된 많은 스케줄링 알고리즘들은 간단한 구현에 의한 고속의 처리를 장점으로 하는 Round-Robin 방식[1,2]과 정확한 서비스를 장점으로 하는 Fair Queueing 방식[3,4,5]으로 나눌 수 있다. 계산량이이 적으면서도 각 플로우에 차별화 된 서비스를 제공할 수 있는 DRR[1]은 현재 많은 상용제품에 응용되어 사용되고 있다[2]. 그러나 이러한Round-Robin 방식의 스케줄링 알고리즘은 모든 플로우를 정해진 순서대로 서비스하는 특성 때문에 정확한 서비스를 제공하는 데에는 한계가 있다. 한편, Fair Queueing 방식의 알고리즘

* 광운대학교 전자공학부 네트워크 시스템 연구실(mjuze@explore.gwu.ac.kr)
논문번호 : 020259-0603, 접수일자 : 2002년 6월 3일

들은 가장 이상적인 서비스 모델인 GPS(Generalized Processor Sharing)[3]의 동작을 패킷 단위의 서비스로 모방하기 위한 것으로써, Round-Robin 방식의 알고리즘 보다 훨씬 정확한 서비스를 플로우들에게 제공할 수 있다. 그러나, GPS에서의 서비스 순서를 기준으로 하기 때문에 스케줄러는 모든 플로우의 모든 패킷에 대하여 도착시간 및 서비스완료 시간을 계산해야 하고, 이를 위해 시스템 차원의 가상적인 시간체계, 즉 System Virtual Time을 관리하여야 한다[3,4,5]. 이는 실제 구현 시에 스케줄러의 계산량을 증가시켜서 시스템의 전송성능을 저하시키며 시스템의 확장성(Scalability)에 문제를 야기하는 원인이 된다. 또한 이러한 방식은 스케줄러가 시스템의 모든 상태(State)를 관리하므로 출력 링크에서의 큐잉에만 적용할 수 있으며, 이를 대형 시스템의 입력링크에서의 큐잉에 적용하려는 경우에는 계산량 및 구현상의 복잡도는 기하급수적으로 증가하게 된다[6]. 본 논문에서는 Fair Queueing 방식의 스케줄러가 시스템 전체의 상태를 관리함으로 인한 계산량 및 구현상의 문제점을 없애는 동시에, 기존에 가장 정확한 성능을 낸다고 알려진 WF^2Q[5]와 유사한 수준의 서비스를 제공하는 새로운 패킷 스케줄링 알고리즘을 제안한다. 제안된 알고리즘에서는 시스템의 상태관리, 즉 각 패킷의 서비스 순서 결정을 위한 가상 시간 체계의 관리를 각 플로우에게 분산시키고, 스케줄러는 각 플로우에 의해 계산되는 정보에 의해 다음 서비스할 패킷을 선정하는 일만을 하게 된다. 각 플로우는 시스템의 상태에 무관하게 독립적인 별도의 클럭을 관리하므로, 본 논문에서 제안된 알고리즘을 DCFQ(Distributed-Clocked Fair Queueing)라는 이름을 사용한다. 본 논문의 구성은 다음과 같다. 2장에서 기존의 스케줄링 알고리즘에 대한 설명과 3장에서는 본 논문에서 제안된 DCFQ 알고리즘에 대한 설명과 간단한 예를 보이고 4장에 시뮬레이션 결과를 제시한다. 5장에서 기존의 패킷 스케줄링 방식들과 패킷 서비스 순서 및 계산량에 대한 비교를 한 후에 6장에서 DCFQ의 응용 및 향후 연구방향을 언급한다.

Ⅱ. 기존의 스케줄링 알고리즘들

GPS는 이상적인 스케줄링 방법으로 최대 최소 공유 할당(max-min share rate)을 제공해 준다. GPS에서 스케줄러는 서비스를 받기 위해 기다리고 있는 패킷들의 플로우에 대해, 각 플로우마다 무한대의 적은 양의 데이터를 round-robin방식으로 서비스를 해나가게 된다. 또한 각 플로우에 대한 가중치가 다를 경우 가중치에 따라 데이터가 서비스되는 시간은 비례하게 된다. GPS 서버는 work conserving 상태를 가정하며, GPS 서버는 다음과 같이 정의 된다. φ_i와 φ_j는 전체 대역폭(bandwidth)중에 각 플로우가 할당받은 비율을 나타내며, $S_i(\tau, t)$와 $S_j(\tau, t)$는 시간구간 τ 와 t사이에 각 플로우에서 서비스된 양을 나타낸다.

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} \geq \frac{\varphi_i}{\varphi_j}, \quad j=1,2,..,N \tag{1}$$

WFQ 스케줄링 알고리즘은 무한대의 적은 양을 서비스하는 GPS와는 달리 패킷 단위로 서비스를 하게 된다. WFQ에서 서비스되는 패킷의 순서는 가상 종료 시간(virtual finish time)에 따라 결정되며, 적은 가상 종료 시간을 가지고 있는 플로우의 패킷이 서비스를 받게 된다. 가상 종료 시간의 정의는 다음과 같다.

$$\begin{aligned} S_i^k &= \max\{F_i^{k-1}, V(a_i^k)\} \\ F_i^k &= S_i^k + \frac{L_i^k}{\varphi_i} \end{aligned} \tag{2}$$

플로우 i의 k번째 패킷의 도착 시간은 a_i^k로 표현되며, 플로우 i의 k번째 패킷의 길이를 L_i^k이라 하고, 이 패킷이 서비스가 시작될 시간과 서비스가 끝나는 시간은 각각 S_i^k와 F_i^k로 표현된다.

WF^2Q 스케줄링 알고리즘은 WFQ와 같은 가상 시간을 사용하며, 가장 큰 특징은 다음에 서비스될 패킷을 선정하는 과정에서, 선정 한 패킷의 가상 서비스 시작 시간은 GPS 스케줄링 알고리즘을 사용한 경우의 서비스 시작 시간보다 빨라서는 안되다는 조건을 사용한다.

DRR 스케줄링 알고리즘은 각 플로우의 패킷 길이를 가변의 패킷으로 고려해서, 공평한 서비스를 받을 수 있도록 고안된 스케줄링 방법이다. 각 플로우들은 deficit 계수라는 변수를 가지고 있으며, 이 값은 처음에 0으로 초기화된다. 스케줄러는 각 플로우들을 번갈아 가며 서비스를 해주는데 한 퀀텀(quantum)만큼의 양을 서비스하게 된다. 각 플로우의 앞에 있는 패킷의 크기가 퀀텀의 크기보다 작으면 서비스를 하게 되고, 그렇지 않으면 그 퀀텀의 양은 각 플로우의 deficit 계수에 더해진다. 따라서 한 번 라운드할 때 한 개의 패킷이 서비스되는 것이 아니라 플로우 앞에 위치한 패킷의 크기가 매우

작다면, 여러 개의 패킷이 한 번 라운드에 서비스될 수도 있다.

Ⅲ. DCFQ: 새로운 알고리즘의 제안

대역폭 R의 출력 링크에 N개의 플로우 N={1,2,...N} 이 있다고 가정하고 플로우 i는 R_i의 서비스 율을 할당받아서 $\Sigma R_i \le R$이라고 가정한다. 플로우 i는 다른 플로우의 상태 혹은 전체 링크의 상태에 무관하게 독립적으로 자신의 상태를 관리하고, 패킷 스케줄러는 각 패킷 및 시스템에 대한 정보를 관리하지 않으며 각 플로우가 제공하는 정보에 따라 다음 서비스할 패킷을 선택한다. 플로우 i의 큐에 대한 정보는 증가 카운터 C_i와 레지스터 D_i에 의해 관리된다. 스케줄러는 D_i의 값을 비교하여 서비스의 순서를 정하게 되며, 따라서 스케줄러는 각 패킷에 대한 정보 및 가상 시간 등의 시스템 정보를 관리할 필요가 없으며, 각각의 큐에 대한 정보는 분산되어 있고 서로 독립적으로 관리된다. P_i^k를 플로우 i의 k번째 패킷이라고 하고 그 길이가 L_i라고 한다. P_i^k가 DCFQ에서 서비스가 시작되는 시간을 S_i^k라 하면, 카운터 C_i의 시간대 t에서의 증가율은 다음과 같이 정의된다.

$$\frac{d}{dt} C_i = \begin{cases} 0, & \text{if } S_i^k \le t \le S_i^k + \frac{L_i^k}{R} \\ R_i, & otherwise \end{cases} \quad (3)$$

즉, C_i는 자신의 패킷이 서비스 받는 동안에는 증가하지 않으며, 그 외에는 R_i의 비율로 단조 증가한다. L_i를 플로우 i의 맨 처음에 위치한 패킷의 크기라 할 때, 플로우 i는 C_i와 D_i의 값을 계속 갱신한다. 이 D_i는 스케줄러가 다음에 서비스할 패킷을 결정하는데 기준이 되며 다음과 같이 정의된다.

$$D_i = \begin{cases} \frac{C_i}{L_i}, & flow\ i\ is\ backlogged \\ 0, & otherwise \end{cases} \quad (4)$$

D_i의 값은 큐가 비어있지 않은 플로우의 경우에 서비스를 받지 못하는 동안 계속 증가하며, 그 증가율은 R_i 및 L_i의값에 따라 결정된다. P_i^k 가 선택되어 서비스를 받게 되는 경우에 C_i의 값은 다음과 같이 조정된다.

$$C_i = \max(C_i - L_i^k, 0) \quad (5)$$

DCFQ의 동작을 간단한 예를 들어 설명한다. 3개의 플로우가 용량이 100인 출력 링크를 통해 서비스를 받고 있으며, t=0의 시각에 각 플로우에는 크기가 100인 많은 패킷들이 대기하고 있다고 가정한다. 또한 세 플로우는 2:1:1의 비율로 서비스가 할당되어 있다. 즉, 모든 i=1,2,3 및 k=1,2,...에 대하여 L_i^k = 100이며, C_1 = 50, C_2 = C_3 = 25이다. 그림 1은 이 경우에 패킷의 서비스 순서를 나타내고 있는데, 스케줄러가 선정하는 플로우의 D_i 값이 음영으로 표시되어 있다.

t=0에서 D_i는 모든 플로우 i에 대하여 같으므로 플로우 1의 패킷이 서비스를 받게 된다. 그 패킷의 서비스가 끝나는 t=1에 C_2=C_3=25의 값으로 증가하고 이에 따라 D_2=D_3=0.25의 값을 갖게 된다. 한편 플로우 1은 자신의 패킷이 서비스를 받는 동안 카운터가 정지하므로 t=1에서 D_1=0이 된다. 따라서 두 번째로는 플로우 2의 패킷이 서비스를 받고, 그 패킷의 서비스가 끝나는 t=2에서는 D_1=0.5, D_2=0, D_3=0.5가 되어 다시 플로우 1의 패킷이 서비스를 받는다. t=3에서는 플로우 3의 카운터가 75의 값을 갖게 되어 D3이 가장 큰 값을 가지므로 플로우 3의 패킷이 서비스를 받는다.

Time	C_1	D_1	C_2	D_2	C_3	D_3
0	0	0	0	0	0	0
1	0	0	25	0.25	25	0.25
2	50	0.5	0	0	50	0.5
3	0	0	25	0.25	75	0.75

그림 1. DCFQ의 예 (R1:R2:R3=2:1:1, R=100, Packet size =100)

Ⅳ. 시뮬레이션

본 장에서는 앞장에서 제안한 DCFQ 알고리즘을 그림 2와 같은 IP 네트워크 구조를 대상으로 시뮬레이션 한 결과를 제시한다. 본 장에서의 시뮬레이션 및 다음 장에서의 다른 알고리즘에 대한 시뮬레이션은 ns[7]를 사용하여 수행되었다. 그림 2에서 세 개의 소스 S1, S2, S3는 라우터 R1과 R2를 통해 싱크 S4로 packet을 전송하는 UDP 호스트들이며, R1에는 앞장에서 제안된 DCFQ 알고리즘을 적용하여 세 플로우에 대한 서비스를 하도록 한다.

S1, S2 및 S3는 on-off 소스들로서 각각 700Kbps, 400 Kbps 및 400Kbps의 Burst peak rate를 가지며, Burst time과 Idle time의 비율은 모두

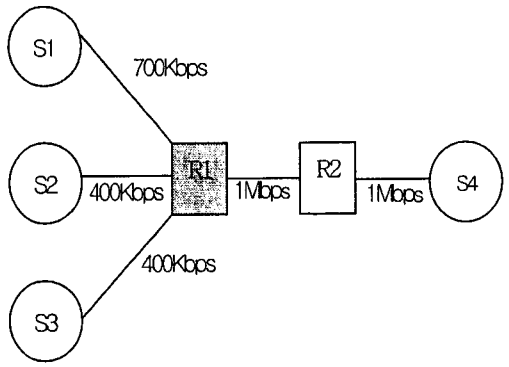

그림 2. 시뮬레이션 네트워크의 구성

2:0.1로 설정되었다. R1의 출력 링크 용량이 1 Mbps이므로 R1-R2 링크에서 병목이 발생하며 DCFQ에 의한 대역폭 관리가 이루어진다. S1과 S2에서 보내는 패킷의 크기는 1Kbytes, S3의 패킷은 200Bytes이며, 각 플로우에 대한 서비스 웨이트(Weight)는 φ_1=6, φ_2=3, φ_3=1로 설정하였다. 이러한 네트워크 설정에 대하여 시뮬레이션 한 결과는 그림 3과 같다.

그림에서 보듯이, 각 플로우는 소스의 전송률에 상관없이 미리 할당된 대역폭만을 사용하고 있음을 알 수 있다. 패킷의 서비스 순서 결정을 위하여 시스템 차원의 가상 시간 정보를 사용하지 않고 분산된 클럭(카운터)을 사용하여 Fair Queueing을 수행할 수 있음을 보여주고 있다. 그림에서 플로우의 수율(Throughput)이 불규칙하게 급변하는 이유는 다른 소스가 Idle time일 때 출력 대역폭에 생기는 여유분을 다른 플로우가 사용하기 때문이다.

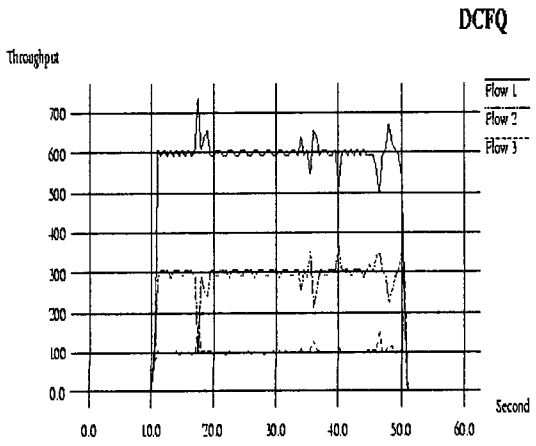

그림 3. 시뮬레이션 결과

V. 다른 스케줄링 방식과의 비교

5.1 패킷 서비스 방식

본 장에서는 DCFQ 알고리즘과 기존에 제안된 다른 패킷 스케줄링 방식과의 서비스 특성상의 차이에 대하여 비교한다. 그림 4는 DCFQ와 비교 대상인 GPS, WF^{2}Q, DRR, WFQ의 패킷 서비스 특성을 나타낸 것이다. t=0에서 스케줄러가 서비스가 시작될 때 6개의 플로우가 많은 수의 패킷을 각자의 큐에 저장하고 있다고 가정한다. 이때 출력 링크의 속도는 10이고 모든 패킷의 길이도 10이며, 각 플로우의 서비스 웨이트(Weight)는 φ_1=5, φ_2=...=φ_6=1으로 설정한다. 각 도표는 10의 출력 율로 10초간 패킷들이 서비스 받는 과정을 나타내며, 모든 스케줄링 방식은 Work Conservation의 원칙을 따르기 때문에 모든 시간대에서 출력 링크의 서비스 율은 10이 된다. 또한 각 도표 내의 직사각형들은 크기가 10인 패킷이 서비스 받는 속도와 서비스 시간을 나타내고 있다. 음영으로 표시된 부분은 50%의 서비스를 할당받은 플로우 1의 패킷들을 나타낸다.

(a)의 이상적인 GPS의 경우에는 모든 플로우가 할당된 비율로 동시에 서비스를 받으므로, 다른 플로우의 패킷이 하나 처리되는 동안에 플로우 1의 패킷은 다섯 개가 전송된다. (b)는 현재 GPS를 패킷 단위에서 가장 근사적으로 모방한다고 알려진 WF^{2}Q의 서비스 형태를 나타낸 것이다. 플로우 1의 패킷들과 다른 플로우의 패킷들이 하나씩 교대로 전송되어 할당된 서비스 웨이트를 만족함을 알 수 있다. (c)는 DRR 및 WFQ(혹은 PGPS)의 패킷 서비스 순서를 나타낸 것이데, 높은 웨이트를 갖는 플로우의 경우에는 GPS에서의 서비스 보다 훨씬 우선적인 전송이 이루어짐을 알 수 있다. 즉, 10의 시간 동안에 서비스 율은 GPS와 동일하지만, 웨이트가 높은 플로우의 패킷에 대해 우선적으로 전송이 된다. 이에 따라 플로우 1은 R1을 지나면서 Burst:Idle이 1:1인 On-Off 트래픽으로 변한다. 이는 인터넷의 경우처럼 Feedback 혼잡 제어를 사용하는 경우에 문제가 있으며[5], 음성이나 비디오 등의 실시간 트래픽의 전송에도 바람직하지 않다[8]. 한편, DCFQ의 경우에는 WF^{2}Q와 마찬가지로 웨이트가 높은 플로우에 편중되지 않는 서비스를 제공하여 플로우 1에게 일정한 전송률을 지원함을 알 수 있다.

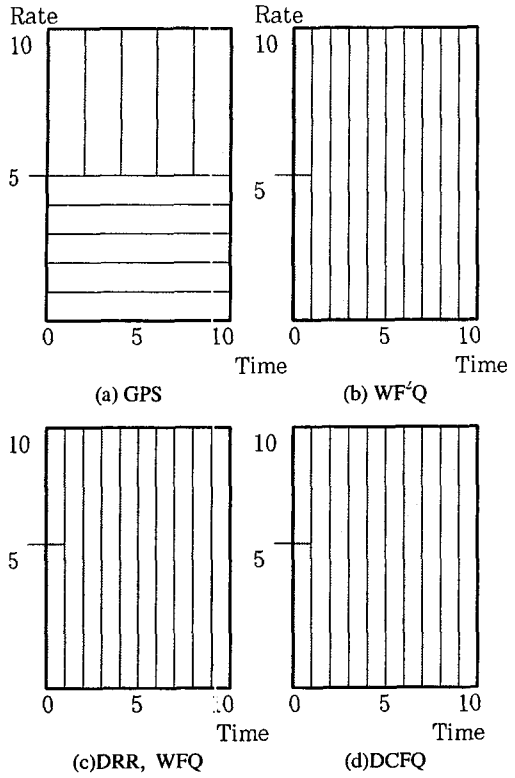

그림 4. 여러 패킷 스케줄링 방법들의 서비스 순서

그림 5는 그림 2의 네트워크에 6개의 플로우가 서비스 웨이트 φ_1=5, φ_2=...=φ_6=1을 갖는 경우에 WF²Q와 DCFQ의 수율(Throughput) 특성을 비교한 결과이다. 여기서 모든 패킷의 크기는 1 Kbytes로 하였으며, 플로우 1은 CBR 트래픽으로, 나머지 플로우들은 Burst:Idle = 2:0.4인 On-Off 트래픽을 가정한다.

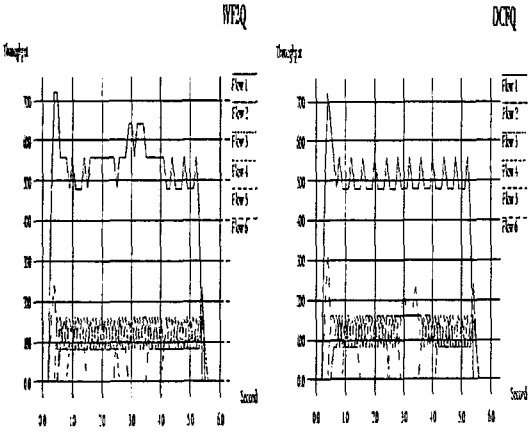

그림 5. WF²Q(왼쪽)와 DCFQ(오른쪽)의 수율 특성 비교

그림에서 보듯이 3sec와 3.5sec 사이에서 두 스케줄링 알고리즘의 서비스 특성을 비교할 수 있다. 그 시간에 6개 중 두 플로우가 Idle 상태에 있는데, 이로 인해 발생한 여유 대역폭을 WF²Q의 경우에는 가장 큰 서비스 웨이트를 갖는 플로우 1이 대부분 사용하게 되나, DCFQ에서는 웨이트가 작은 플로우들이 주로 이를 사용하게 됨을 알 수 있다. 따라서 DCFQ는 링크에 여유 용량이 발생하는 경우에 웨이트가 작은 플로우들을 우선하는 특성이 있다.

5.2 스케줄러의 계산량

표 1은 주요 스케줄링 알고리즘에서 스케줄러의 계산량, 즉 Enqueue와 Dequeue 연산에서의 패킷당 Work Complexity[11]를 나타낸 것이다. 플로우의 개수를 n이라 할 때, WFQ 및 WF²Q는 모두 O(log n)의 복잡도를 최적의 구현 시에 가지는데, 이는 Enqueue의 경우에는 가상 시간의 계산을 위해 모든 플로우에 대한 정보를 가져야 하며, 또한 Dequeue의 경우에는 모든 플로우를 비교하여 다음 서비스 패킷을 선정하여야 하기 때문이다. 이에 비해, SFQ는 Enqueue시에 O(1), DRR은 언제나 O(1)의 복잡도를 갖는다.

표 1. 주요 알고리즘들의 Work Complexity per Packet

Algorithm	Enqueue	Dequeue
WFQ[3], WF²Q[5]	O(log n)	O(log n)
SFQ[4]	O(1)	O(log n)
DRR[1]	O(1)	O(1)
DCFQ	O(1)	O(log n)

Fair Queueing 알고리즘의 경우에는 Round Robin과는 달리 Dequeue 연산은 O(1)에 수행될 수 없으며, 본 논문에서 제안된 DCFQ도 Dequeue의 경우는 복잡도가 O(log n)이고, 패킷 당 Work Complexity 상으로는 SFQ와 동일한 성능을 갖는다. 그러나, SFQ의 경우에는 스케줄러가 n 플로우의 모든 패킷들에 대해 가상 시간을 부여하는데 비해 DCFQ에서는 도착하는 패킷들에 대해서 별도의 연산이 없고 각 플로우의 레지스터 관리는 각 플로우가 분산하여 담당하므로, 실제의 계산량은 DCFQ가 훨씬 적다. 따라서, 스케줄러를 고려할 때, DCFQ는 기존의 Fair Queueing 알고리즘들 중에서 가장 적은 계산량을 필요로 한다.

Ⅵ. 결론 및 향후 연구 내용

본 논문에서 제안된 DCFQ는 기존의 Fair Queueing 알고리즘들의 확장성 및 구현성 문제를 해결하기 위해서 패킷의 서비스 순서를 정하기 위한 가상 시간의 관리를 각 플로우에 분산시키는 방법을 사용하였다. 스케줄러는 각 플로우가 제공하는 값에 따라 다음 패킷을 선정하며, 패킷의 도착 시에 필요한 가상 서비스 완료 시간의 연산도 불필요하게 된다. 이 알고리즘이 Round Robin수준의 계산량을 가능하게 하는 동시에 최적으로 알려진 WF^2Q 수준의 서비스를 제공함을 보였다.

분산하여 처리하는 특성으로 인하여, 여러 개의 Forwarding 엔진을 갖는 대규모의 라우터 설계에 응용될 수 있다. 즉, 라우팅을 마친 패킷들이 스위치를 통하여 큐잉 기능이 없는 인터페이스로 전달되는 경우에는 일반적인 스케줄링 알고리즘이 가정하는 출력 큐잉을 할 수 없으므로, 각 엔진에서 입력 큐잉을 구현할 필요가 있다. 본 논문에서 제안된 DCFQ를 이러한 입력 큐잉의 경우로 확장하는 연구 및 서비스의 공정성과 성능에 대한 해석적인 분석을 현재 진행 중에 있다.

참 고 문 헌

[1] M. Shreedhar, G. Varghese, "Efficient fair queueing using deficit round robin," *Proceedings of ACM SIGCOMM*, pp. 231-242, Sep. 1995.

[2] Planning for Quality of Service, White paper, *Cisco Systems*, Available at http://www.cisco.com

[3] A. Parekh, R. Gallager, "A generalized processor sharing approach to flow control: A single node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344-357, 1993.

[4] S. Golestani, "A self-clocked fair queueing scheme for broadcast applications," *Proceedings of IEEE INFOCOM*, pp. 636-646, May 1994.

[5] J.C.R. Bennett, H. Zhang, "Worst-case Fair Weighted Fair Queueing," *Proceedings of INFOCOM*, pp. 120-128, 1996.

[6] D. Stephens, H. Zhang, "Implementing distributed packet fair queueing in a scalable switch architecture," *Proceedings of IEEE INFOCOM*, pp. 282-290, 1998.

[7] The Network Simulator: ns-2, Available at http://www.isi.edu/nsnam

[8] S. Floyd, M. Handley, J. Padhye, "A comparison of equation-based and AIMD congestion control," In progress, Available at http://www.aciri.org

이 준 엽(Jun Yeop Lee) 준회원

1995년~2001년 : 광운대학교 전파공학과 (공학사)

2001년~현재 : 광운대학교 전자공학부 (공학석사)

<주관심 분야> 인터넷 보안, QoS, 혼잡제어

이 승 형(Seung Hyong Rhee) 종신회원

1984년~1988년 : 연세대학교 전자공학과 (공학사)

1988년~1990년 : 연세대학교 전자공학과 (공학석사)

1995년~1999년 : University of Texas at Austin, Dept. of ECE (Ph. D.)

1990년~1995년 : 국방과학연구소 연구원

1999년~2000년 : 삼성종합기술원 전문연구원

2000년~현재 : 광운대학교 전자공학부 전임강사

<주관심 분야> 인터넷 보안, QoS, 혼잡제어, 트래픽관리