공유 링크들을 가진 스위치를 위한 세션 할당 알고리즘

안성용*, 이정아*, 심재홍**
*조선대학교 컴퓨터공학과
**조선대학교 인터넷소프트웨어 공학부
e-mail:dis@chosun.ac.kr

A Session Allocation Algorithm for a Switch with Multiple Shared Links

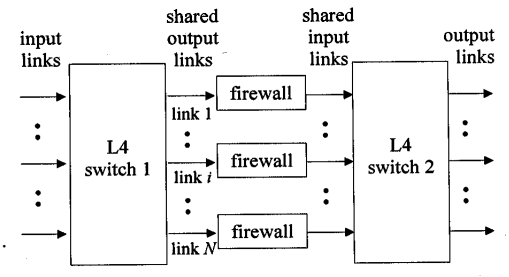
Seong-Yong Ahn*, Jeong-A Lee*, Jae-Hong Shim**
*Dept of Computer Engineering, Chosun University
**School of Internet Software Engineering, Chosun University

요 약

본 논문에서는 다중 공유 링크를 가진 스위치를 위한 세션 연결 설정 시 이를 어떤 링크에 할당할 것인지를 결정하는 경험적 세션할당 알고리즘을 제안한다. 제안 알고리즘은 새로운 세션이 소속된 서비스 클래스의 각 링크에 할당된 세션들의 예측된 지연들 중 가장 작은 예측 지연을 가진 링크에게 그 세션을 할당한다. 다른 세션할당 알고리즘에 비해 제안 알고리즘은 부클래스들의 예측지연 특성을 직접적으로 활용함으로써, 서비스 클래스들에게 사전에 예약된 대역폭을 제공하고, 동일한 서비스 클래스에 속한 세션들에게는 서로 다른 공유 링크를 통해 전송되어도 가능한 비슷한 지연을 제공한다는 것을 모의실험을 통해 확인했다.

1. 서론

오늘날 초고속 통신망과 스위칭 기술의 발전으로 고속으로 패킷 전송이 가능해졌지만, 역으로 방화벽, 침입탐지 시스템 등의 처리 속도 한계로 인해 이들 시스템들이 통신망을 흐르는 모든 패킷을 처리할 수 없는 상황이 초래되었다. 이를 해결하기 위한 방안으로 고속의 단일 링크를 사용하는 대신 보다 저속의 다중 공유 링크들을 사용하여 이들 시스템들을 연결할 수 있으며 (그림 1 참조), 이를 통해 가격대 성능향상을 이룰 수 있다. 그림에서 스위치 1은 입력 링크들을 통해 도착한 패킷들을 세션들로 분류한 후에 공유 출력 링크들 중의 하나로 전송한다. 방화벽은 패킷 필터링 과정을 거쳐 이 패킷들을 스위치 2에 전달한다. 스위치 2는 공유 입력 링크들을 통해 도착한 패킷들에 대해 라우팅 결정을 하고, 이에 따라 선택된 출력 링크를 통해 패킷들을 전송한다



(그림 1) 다중 공유 링크들을 가진 스위치들

다중 링크 스위치는 공유 링크들의 전체 대역폭을 클래스들에게 공정하게 할당하면서 요구된 QoS를 제공해야 한다. 가장 중요한 QoS로서 클래스들의 수와 그들의 트래픽 패턴에 영향을 받지 않고 각 클래스에게 사전에 예약된 대역폭을 보장할 수 있어야 한다. 두 번째 QoS로서 임의의 시간구간에서 동일 클래스에 속하는 두 세션들이 서로 다른 공유 링크를 통해 전송될 때, 이들 두 세션의 지연(delay)이 가능한 서로 비슷하도록 해야 한다.

클래스들에게 위의 두 QoS를 제공하기 위해 스위치는 먼저 총 부하를 공유 링크들에게 적절히 분배하는 부하분배(load balance)와 클래스들에게 그들의 예약된 대역폭에 비례하여 링크 대역폭을 할당하는 공정한 링크 스케줄링(fair link scheduling)을 지원해야 한다. 이들 두 기술에 관해서는 그 동안 많은 연구가 진행되었다. 그러나 기존의 부하분배는 주로 시스템 전체의 성능 향상에 초점을 두었고, 클래스들에게 차별화된 QoS 제공은 고려하지 않았다[1]. 또한 기존의 공정한 링크 스케줄링 기술은 단일 링크 상에서만 공정성(fairness), 예약된 대역폭 보장, 지연 등을 제공하고, 다중 공유 링크들에 대해서는 이들 기능을 직접적으로 지원하지 못했다[2]. 따라서 다중 공유 링크들을 가진 스위치에서 단순히 이들 두 기술의 적용만으로는 위에서 언급된 두 QoS를 제공하기 어렵다. 이유는 세션들을 어떤 링크들에 할당하느냐에 따라 그들이 소속된 클래스들에게 제공되는 QoS가 달라지기 때문이다. 세션들이 할당된 링크들의 부하와 속도 등이 모두 다르고 한번 특정 링크에 할당된 세션은 도중에 다른 링크를 통해 전송

될 수 없기 때문에 다중 링크 스위치에서 세션할당 전략은 매우 중요한 스케줄링 문제가 된다.

본 논문에서는 다중 공유 링크를 가진 스위치에서 클래스들에게 사전에 예약된 대역폭을 보장하고, 동일한 클래스에 속한 세션들에게 서로 다른 공유 링크를 통해 전송되어도 가능한 비슷한 서비스 지연을 제공하고자 한다. 이를 위해 새로운 세션의 연결 설정 시 이를 어떤 링크에 할당할 것인지를 결정하는 경험적 세션할당 알고리즘을 제안한다.

2. QoS의 정의

본 절에서는 다중 공유 링크를 가진 스위치에서 클래스들에게 지원하고자 하는 두 QoS를 수식을 통해 보다 명확히 정의하고자 한다. N개의 다중 공유 링크들을 가지고 M개의 서비스 클래스를 지원하는 스위치를 고려해 보자. 상수 i를 클래스 C_i의 공정분배율(fair share ratio)이라 하며, 임의의 링크 l의 링크 전송율(transmission rate)을 r_l로 표현한다. 공유 링크들의 총 링크 전송율 $r = \sum_{l=1}^{N} r_l$ 이라 할 때, C_i를 위해 $\phi_i \cdot r$의 대역폭이 예약되며, $\sum_{i=1}^{M} \phi_i r \le r$이다. 본 논문에서는 편의상 정규화된(normalized) 링크 전송율을 사용한다. 따라서 r = 1이고, $1 \le l \le N$에 대해 $r_l \le 1$이다.

$W(t_1, t_2)$를 시간구간$[t_1, t_2]$에서 공유 링크들을 통해 스케줄러에 의해 전송된 서비스(데이터)의 총량이라 하고, $W_i(t_1, t_2)$를 동일 구간에서 클래스 C_i를 위해 서비스된 데이터 양이라 하자. 만약 임의의 시간구간에서 클래스 C_i에 소속된 세션의 수가 공유 링크들의 수보다 적어도 같거나 많고 각 세션의 패킷들이 끊임없이 시스템에 도착하여 큐에 쌓인다면, 클래스 C_i는 그 시간구간에서 백로그(backlogged) 되었다고 한다. 다중 공유 링크들을 위한 스케줄러의 공정성(fairness)을 다음과 같이 정의한다.

정의1: 임의의 두 개의 백로그된 클래스 C_i와 C_j에 대해, 시간구간 $[t_1, t_2]$에서 두 클래스에게 제공된 정규화된 서비스(normalized service)의 차이가 상한치에 의해 고정될 수 있다면, 그 스케줄러는 **공정**(*fair*)하다. 즉, $\left|\frac{W_i(t_1,t_2)}{\phi_i} - \frac{W_j(t_1,t_2)}{\phi_j}\right| \le \Delta F$ 이어야 한다. 여기서 $\Delta F \le \infty$는 스케줄러의 공정성 측정 요소이다.

다중 공유 링크 스위치에서 대역폭 보장을 위해서는 공정성 뿐 아니라, 모든 링크 자원들을 충분히 활용하는 **고-처리율(high throughput)**도 함께 제공되어야 한다. 이를 위해 시간구간 $[t_1, t_2]$에서 백로그된 클래스가 존재할 경우, $r(t_2 - t_1)$와 $W(t_1, t_2)$의 차이를 상한치에 의해 고정시킬 수 있어야 한다. 즉, $|r(t_2 - t_1) - W(t_1, t_2)| \le \Delta T$ 이어야 한다. 여기서 $\Delta T \le \infty$는 고-처리율 측정 요소이고, $r(t_2 - t_1)$는 구간 $[t_1, t_2]$에서 공유 링크들에 의해 전송 가능한 서비스 총량이다.

정의2: 공정성과 고-처리율을 지원함으로써 모든 클래스들에게 그들의 예약된 대역폭을 보장할 수 있다면, 그 스케줄러는 **클래스간(inter-class) 공정성**을 지원한다.

또 다른 QoS는 임의의 시간구간에서 동일 클래스에 속하는 두 세션이 서로 다른 공유 링크를 통해 서비스될 때 이들 세션들의 서비스 지연 차이를 가능한 비슷하게 하는 것이다.

클래스 C_i의 세션들 중 링크 l에 할당되어 서비스되는 세션들의 집합을 클래스 C_i의 부클래스(subclass) $C_{i,l}$라 하고, $D'_{i,l}(t_1, t_2)$를 시간구간 $[t_1, t_2]$에서 부클래스 $C_{i,l}$에 속한 세션들의 서비스된 모든 패킷들의 평균 지연(delay)이라 하자.

정의3: 서로 다른 링크에 할당된 클래스 C_i의 임의의 두 부클래스 $C_{i,j}$와 $C_{i,l}$ 에 대해 $(j \neq l)$, 시간구간 $[t_1, t_2]$에서 두 부클래스의 서비스된 모든 패킷들의 평균 지연 차이가 상한치에 의해 고정될 수 있다면, 스케줄러는 **클래스-내부(intra-class) 공정성**을 지원한다. 즉, $|D'_{i,j}(t_1,t_2) - D'_{i,l}(t_1,t_2)| \le \Delta D$ 이고, $\Delta D \le \infty$는 클래스-내부 공정성 측정 요소이다.

3. 경험적 세션 할당 알고리즘

다중 공유 링크를 가진 스위치는 세션의 연결 설정 시 어떤 링크를 통해 그 세션을 서비스할 것인지를 결정하고, 이후 그 세션의 모든 패킷들을 결정된 링크의 패킷 큐로 전달해 주는 역할을 담당하는 세션 할당 알고리즘을 가진다. PFQ(packet fair queueing)는 단일 링크를 위한 스케줄링 알고리즘으로 각 링크마다 하나씩 존재한다. PFQ는 기존의 잘 알려진 GPS[2] 관련 공정한 링크 스케줄링 알고리즘들[3-6] 중의 하나이며, 해당 링크에서 각 클래스별로 하나의 큐를 관리하면서 그 클래스에 소속된 세션들의 패킷들을 도착 순서대로 큐에 보관한다. 부클래스 $C_{i,l}$은 링크 l에 할당된 클래스 C_i의 세션들이며, 각각의 부클래스 $C_{i,l}$를 위해 $\phi_i \cdot r_l$ 의 대역폭이 예약된다. 여기서 ϕ_i는 클래스 C_i의 공정분배율이며, r_l는 링크 l의 링크 전송율이다.

각 링크의 PFQ는 단일 링크용 공정한 스케줄링 알고리즘이므로, 다중 공유 링크를 가진 스위치에선 링크들에게 세션을 적절히 할당하는 세션 할당 알고리즘의 역할이 매우 중요하다. 세션 할당은 각 링크의 PFQ와 연동하면서 공정한 대역폭 할당, 고-처리율, 예약된 대역폭 보장 등의 클래스간 공정성과 클래스-내부 공정성 등의 QoS를 제공해야 한다. 이를 위해 링크들에게 전체 부하를 적절히 분산함과 동시에 각 링크에선 새로운 세션이 소속된 클래스 뿐 아니라 다른 클래스들의 부하도 함께 고려하여야 한다. 여기서 부하(load)는 서비스 중인 세션들의 현재 대역폭 요구량을 상응하는 링크 전송율로 나눈 정규화된 대역폭 요구량을 의미한다.

이러한 고려 사항들을 단적으로 반영하는 것이 각 부클래스의 예측지연(expected delay)이다. 스케줄링 모델에서 각 링크의 PFQ 알고리즘은 각 부클래스 $C_{i,l}$에 대해 하나의 패킷 큐를 관리한다. 부클래스의 예측지연은 해당 부클래스의 큐에서 대기 중인 패킷들의 예측되는 평균 지연을 의미한다. 임의의 시간 t에서 서비스 클래스 C_i에 소속된 새로운 세션이 연결될 때, 특정 링크 l 상의 부클래스 $C_{i,l}$의 예측지연을 $D_{i,l}(t)$로 표현한다. 예측지연의 다양한 특성들을 기반으로 본 논문에서는 SCDF(Shortest Class Delay First)라는 경험적 세션할당 알고리즘을 제안한다. 임의의 클래스 C_i에 소속된 새로운 세션이 도착할 때, 세션할당 알고리즘 SCDF는 각 링크에 할당된 C_i의 부클래스들 중 가장 작은 예측지연을 가진 부클래스(링크)에게 그 세션을 할당한다. 즉, $\min_{1 \le l \le N}(D_{i,l}(t))$인 그러한 링크 l을 선택한다.

임의의 시간 t에서 부클래스 $C_{i,l}$의 예측지연 $D_{i,l}(t)$는 PFQ에 의해 관리되는 $C_{i,l}$의 평균 큐 길이, $L_{i,l}(t)$를 이용

하여 계산할 수 있다. 즉, $D_{i,l}(t)=\frac{L_{i,l}(t)}{\phi_i r_l}$ 이며, $\phi_i \cdot r_l$은 $C_{i,l}$을 위해 예약된 링크 l의 대역폭이다. 평균 큐 길이는 과거의 값들을 기반으로 미래의 큐 길이를 예측하는데 널리 활용되는 지수(exponential) 평균 기법을 사용한다[7]. 즉, $L_{i,l}(t+1) = a\ T_{i,l}(t) + (1 - a)L_{i,l}(t)$이다. 여기서 $T_{i,l}(t)$ 는 $C_{i,l}$의 현재 큐 길이이며, a는 보다 최근 또는 과거의 큐 길이에게 부여될 상대적 가중치를 결정하는 상수 가중 인자(constant weighting factor, $0 < a < 1$)이다. (그림 2)는 제안 알고리즘인 SCDF를 보인 것이다.

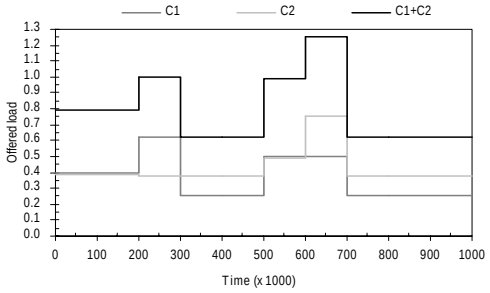
본 논문에서는 제안된 SCDF 외에 성능비교를 위해 가장 직관적인 또 다른 알고리즘을 고안하였다. MCUF(*Minimum Class Utilization First*)는 새로운 세션이 소속된 클래스 C_i의 부하가 가장 작게 할당된 링크에게 그 세션을 할당하는 알고리즘이다. C_i의 부클래스들 중 가장 작은 부하를 가진 부클래스가 할당된 링크 즉, $\min_{1\le l\le N}(U_{i,l}(t))$인 링크 l을 선택한다. 여기서 $U_{i,l}(t)$는 시간 t에서 부클래스 $C_{i,l}$에 속한 세션들의 부하(대역폭 요구량)의 합이다. MCUF는 각 클래스 부하를 링크들에게 균등하게 분산하고, 각 링크 스케줄러는 해당 링크 대역폭을 그 링크의 부클래스들에게 공정하게 할당함으로써, 정상적인 부하에서는 다양한 QoS를 제공할 수 있다.

```
SESSION_ASSIGNMENT(packet P) :

Let P be a packet to be passed to any link
Let P.sid be the session ID of P
Assume session P.sid belongs to a service class C_i

if (P is the first packet of P.sid) then
    minD ← infinite time
    for j ← 1 to N do
        D_{i,j}(t) ← L_{i,j}(t) / (φ_i r_j)
        if (D_{i,j}(t) < minD) then
            minD ← D_{i,j}(t)
            l ← j
        end if
    end for
    SessionToLink[P.sid] ← l
else
    l ← SessionToLink [P.sid]
end if
Q_{i,l} ← P   // Insert P to queue Q_{i,l} of C_{i,l}
```

(그림 2) 제안된 SCDF 세션할당 알고리즘

4. 성능평가

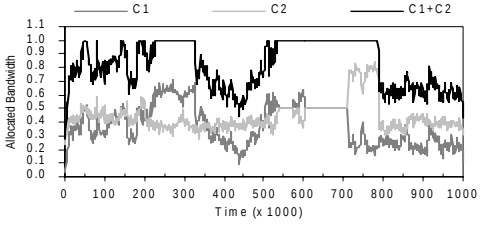
본 논문에서는 제안 알고리즘의 성능을 비교 평가하기 위해 패킷-기반 시뮬레이터를 개발하였다. 각 링크 스케줄러인 PFQ로는 WF^2Q+[5]를 사용하였다. 실험에서 부클래스들을 위한 큐 길이는 무한하다고 가정했으며, 두 개의 공유 링크를 가진 스위치를 대상으로 실험했다. 두 링크의 전송율 r_i는 각각 0.5이다. 결과분석의 편의를 위해 두 개의 클래스(C_1과 C_2)만 지원했으며, 이들의 공정분배율은 각각 0.5이다. 스케줄링 모델 정의에 따라 4개의 부클래스가 생성되며, 각 부클래스 $C_{i,l}$를 위해 0.25의 대역폭이 예약된다. 총 4,000개의 세션들이 두 클래스를 위해 교대로 생성되며, 그들의 연결 요청 간격(interval)과 존속 기간(life time)은 지수(exponential) 분포를 따르고, 평균은 각각 250과 10,000이다. 각 클래스에는 항상 평균 20개의 세션들이 연결되어 서비스된다.



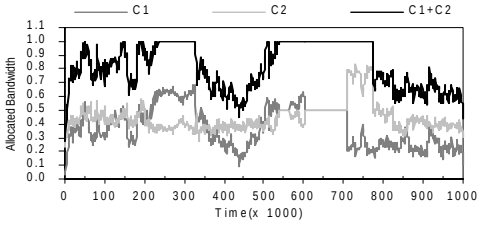
(그림 3) 클래스들에게 제공된 평균 부하

(그림 3)은 실험을 위해 각 시간별로 두 클래스를 위해 생성된 평균 부하(C_1과 C_2)와 이들의 합인 총 평균 부하(C_1+C_2)를 보여준다. X축은 시뮬레이션 시간을 보여 주며, 단위는 1000 시뮬레이션 시간이다. 클래스의 부하는 그 클래스에 속하는 세션들의 부하(대역폭 요구량)의 합이다. 각 세션의 부하는 평균 ρ를 가진 균등(uniform) 분포를 따르고, ρ는 그 세션이 소속된 클래스 부하를 그 클래스에 소속된 세션들의 평균 수(약 20개)로 나눈 값이다. 각 세션의 패킷 도착율은 평균 $\lambda(=\rho)$를 가진 포아송(Poisson) 분포를 따른다. 단순화를 위해 모든 패킷의 크기를 1로 하였다.

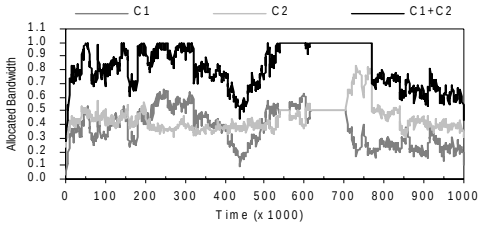
DOS 공격과 같은 과도한 패킷이 도착하는 상황을 반영하기 위해 클래스 C_2의 첫번째 세션으로 0.25의 과도한 부하를 가진 세션 $s_{2,2,1}$을 생성했다. 세션 $s_{2,2,1}$은 예외적으로 실험 시작부터 끝까지 존속하며, 링크 2(부클래스 $C_{2,2}$)에 할당되어 서비스된다. 시간구간 [200, 300]에서 클래스 C_1은 그것의 예약된 대역폭 0.5보다 더 많은 0.625의 과부하를 가진다. 반대로 클래스 C_2의 부하는 예약된 대역폭보다 낮은 0.375이며, $s_{2,2,1}$를 제외하면 0.125이다. 따라서 이 구간에서 $s_{2,2,1}$를 제외한 C_2의 나머지 세션들이 모두 링크 1에 할당된다 해도 $C_{2,1}$의 부하(0.125)는 링크 2에 할당된 $C_{2,2}$(단지 세션 $s_{2,2,1}$만 가짐)의 부하(0.25)보다도 작게 되는 편중된 부하를 갖게 된다. 또 다른 시간구간 [500, 700]에서 C_1은 자신의 예약된 대역폭과 동일한 부하를 가지는 반면 C_2는 구간 [500, 600]에서는 C_1과 동일한 부하를 가지다가 [600, 700]에서는 C_1보다 더 많은 과부하를 가진다.



(a) SGLK



(b) SCDF



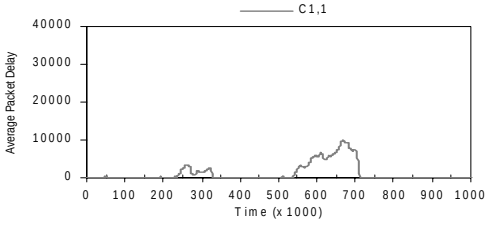
(c) MCUF

(그림 4) 할당된 대역폭

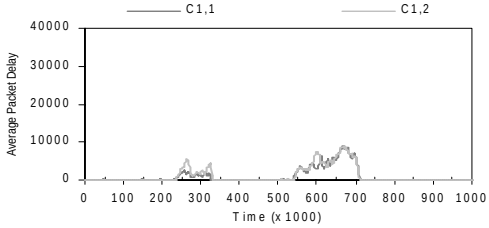
(그림 4)는 (그림 3)의 부하에 대해 서로 다른 세션할당 알고리즘을 채택한 다중 링크 스위치에 의해 각 클래스에게 할당된 대역폭(C_1, C_2)과 총 대역폭(C_1+C_2)을 보여 준다. 그림에서 SGLK(single link)는 본 논문의 모의실험 대상인 두 개의 공유 링크를 가진 스위치의 총 링크 전송율과 동일한 전송율을 가진 단일 링크를 가진 스위치에서 스케줄링된 결과이다. 이 결과는 단일 링크이므로 세션할당 알고리즘이 필요 없고 링크 스케줄러인 PFQ(WF^2Q+)에 의해서만 스케줄링된 결과이기 때문에, 다양한 세션할당 알고리즘들의 성능을 직접적으로 비교할 수 있는 최적의 결과이다.

전체적으로 볼 때 본 논문에서 제안하는 SCDF가 SGLK와 가장 비슷한 결과를 보인다. (그림 3)의 시간구간 [200, 300]에서 제공된 부하는 (그림 4)의 구간 [240, 340]에서 그 파급효과가 나타난다. 이 구간에서 SCDF는 총 링크 전송율과 동일한 대역폭(C_1+C_2)을 할당했으며, 또한 SGLK처럼 과부하를 가진 C_1에게 예약된 대역폭(0.5)을 제공한다. 그러나 MCUF는 총 링크 전송율보다 더 적은 대역폭을 할당하며, 고-처리율을 지원하지 못한다. 즉, C_2에게 SGLK와 비슷한 대역폭을 할당하지만 C_1에게는 SGLK에 비해 상대적으로 적은 대역폭을 할당한다. 이는 MCUF가 C_2의 부하를 전혀 고려하지 않은 채 C_1의 부하를 각 링크에 균등하게 분산했기 때문이다.

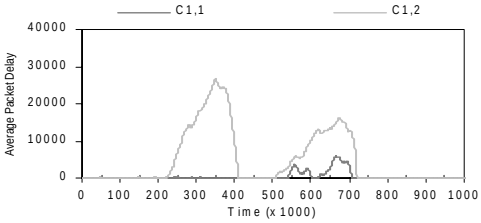
다음으로 각 알고리즘별 클래스-내부 공정성의 지원 여부를 확인 해보자. (그림 5)는 (그림 3)과 동일한 실험에서 링크 1과 링크 2를 통해 서비스된 클래스 C_1의 두 부클래스인 $C_{1,1}$과 $C_{1,2}$의 패킷들의 평균 지연을 보여준다. C_2의 두 부클래스의 평균 지연은 그림으로는 보이지 않았지만 C_1과 비슷한 현상을 보였다. 그림에서 각 시점의 평균 지연은 1000 단위시간 동안 서비스된 패킷들의 평균 지연이다. SGLK는 링크가 하나 뿐이므로 한 곡선만 보여준다. 그림에서 제안 알고리즘인 SCDF만 SGLK와 비슷한 평균 지연을 보이고, 링크 1과 링크 2를 통해 서비스된 패킷들의 지연 차이가 가장 작은 것을 확인할 수 있다. 즉, 가장 밀접한 클래스-내부 공정성을 지원한다.



(a) SGLK



(b) SCDF



(c) MCUF

(그림 5) 클래스 C_1의 링크별 부클래스의 평균 지연

5. 결론

본 논문에서 다중 공유 링크들을 가진 스위치를 위한 세션할당 알고리즘 SCDF를 제안했다. SCDF는 서비스 클래스들에게 사전에 예약된 대역폭을 제공하고, 동일한 서비스 클래스에 속한 세션들에게는 서로 다른 공유 링크를 통해 전송되어도 가능한 비슷한 지연을 제공하고자 한다. 제안된 알고리즘은 새로운 세션이 소속된 서비스 클래스의 각 링크에 할당된 세션들의 예측된 지연들 중 가장 작은 예측 지연을 가진 링크에게 새로운 세션을 할당한다. 모의실험을 통해 제안 알고리즘을 채택한 스위치가 다른 세션할당 알고리즘을 채택한 스위치에 비해 서비스 클래스들에게 보다 공정한 대역폭을 할당하고 높은 패킷 처리율을 제공하며 예약된 대역폭을 보다 확실히 제공한다는 것을 확인할 수 있었다. 또한 동일한 서비스 클래스의 세션들에게 보다 비슷한 서비스 지연을 제공한다는 것도 확인했다.

참고문헌

[1] L. Kencl and J. L. Boudec, "Adaptive Load Sharing for Network Processors," *Proc. of IEEE INFOCOM 2002*, pp. 545-554, New York, June 2002.

[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 Transactions on Networking*, Vol. 1, No. 3, pp. 344-357, June 1993.

[3] A. Demmers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Journal of Internetworking Research and Experience*, Vol. 1, No. 1, pp. 3-26, Oct. 1990.

[4] J. C. R. Bennett and H. Zang, "WF^2Q: Worst-Case Fair Weighted Fair Queueing," *Proc. of IEEE INFOCOM'96*, pp. 120-128, San Francisco, California, Mar. 1996.

[5] J. C. R. Bennett and H. Zang, "Hierarchical Packet Fair Queueing Algorithms," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, pp. 675-689, Oct. 1997.

[6] D. Stiliadis and A. Varma, "Efficient Fair Queueing Algorithms for Packet Switched Networks," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 2, pp. 175-185, Apr. 1998.

[7] W. Stallings, *Operating Systems: Internals and Design Principles*, 3rd ED., pp. 394-396, Prentice Hall, 1998.