다중 공유 링크들의 공정한 대역폭 분배를 위한 세션할당 알고리즘

심 재 홍[†] · 최 경 희[††] · 정 기 현[†††]

요 약

본 논문에서 다중 공유 링크들을 가진 스위치를 위한 세션할당 알고리즘을 제안한다. 제안 알고리즘은 서비스 클래스들에게 사전에 예약된 대역폭을 보장하고, 동일한 서비스 클래스에 속한 세션들에게는 서로 다른 공유 링크를 통해 전송되어도 가능한 비슷한 지연을 제공하고자 한다. 이러한 QoS를 제공하기 위해 다중 공유 링크를 위한 새로운 스케줄링 모델을 정의하고, 이를 기반으로 새로운 세션의 연결 설정 시 이를 어떤 공유 링크에 할당할 것인지를 결정하는 경험적 세션할당 알고리즘을 제안한다. 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.

A Session Allocation Algorithm for Fair Bandwidth Distribution of Multiple Shared Links

Jae-Hong Shim[†] · Kyung-Hee Choi[††] · Gi-Hyun Jung[†††]

ABSTRACT

In this paper, a session allocation algorithm for a switch with multiple shared links is proposed. The algorithm guarantees the reserved bandwidth to each service class and keeps the delay of sessions belonging to a service class as close as possible even if the sessionsare allocated to different shared links. To support these qualities of services, a new scheduling model for multiple shared links is defined and a session allocation algorithm to decide a shared link to be allocated to a new session on the connection establishmentis developed based on the model. The proposed heuristic algorithm allocates a session to a link including the subclass with the shortest (expected) delay that subclasses of the service class the session belongs to will experience. Simulation results verify that a switch with multiple shared links hiring the proposed algorithm provides service classes with fairer bandwidth allocation and higher throughput, and guarantees reserved bandwidth better than the switch hiring other session algorithms. It also guarantees very similarservice delay to the sessions in the same service class.

키워드 : 스위치 스케줄링(Switch Scheduling), 서비스의 품질(Quality of Service), 공정한 대역폭 할당(Fair Bandwidth Allocation), 예약된 대역폭 보장(Reserved Bandwidth Guarantee)

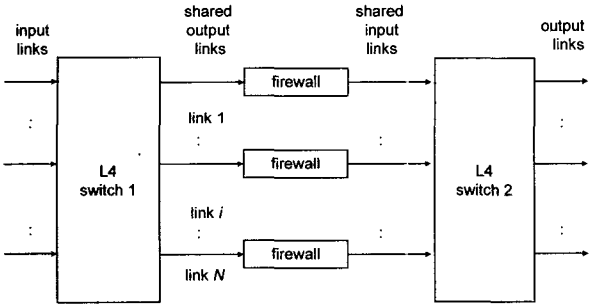
1. 서 론

오늘날 초고속 통신망과 스위칭 기술의 발전으로 고속으로 패킷 전송이 가능해졌지만, 역으로 방화벽(firewalls)[1], 침입탐지시스템(intrusion detection systems)[2] 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 링크들(*multiple shared links*)을 사용하여 이들 시스템들을 연결할 수 있으며((그림 1) 참조), 이를 통해 가격대 성능향상을 이룰 수 있다. (그림 1)(a)에서 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(routing) 결정을 하고, 이에 따라 선택된 출력 링크를 통해 패킷들을 전송한다.

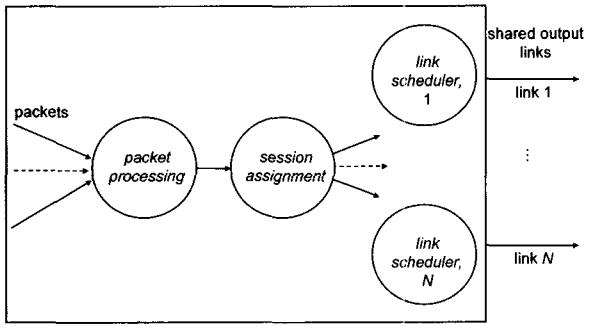
본 논문에서는 패킷 필터링, 주소변환, 레벨-n 라우팅 결정 등의 기능들을 지원할 수 있는 L4 또는 그 이상의 계층을 지원하는 스위치들을 대상으로 하며, 이들 기능은 (그림 1)(b)의 패킷 처리(packet processing) 모듈에 의해 수행된다. 세션(session)은 TCP 연결, IP 플로우 등과 같은 종단간

※ 본 논문은 2003년도 조선대학교 학술연구비의 지원을 받아 연구되었음.
† 정 회 원 : 조선대학교 인터넷소프트웨어공학부 교수
†† 정 회 원 : 아주대학교 정보통신전문대학원 교수
††† 정 회 원 : 아주대학교 전자공학부 교수
논문접수 : 2003년 10월 22일, 심사완료 : 2004년 1월 9일

(end-to-end) 연결된 패킷 스트림으로 정의된다. 세션의 연결 설정 시 이를 어떤 공유 링크를 통해 전송할 것인지를 결정해야 하며, 이때 임의의 한 공유 링크를 선택할 수 있다. 그러나 한번 연결이 설정된 세션은 방화벽 또는 침입탐지시스템에서 그 세션의 상태가 추적되므로 그 세션에 속한 모든 패킷들은 동일 링크를 통해 전송되어야 하는 제약조건을 가진다. 세션을 어떤 링크에 할당할 것인지는 (그림 1)(b)의 세션할당(session assignment) 모듈에 의해 결정된다.



(a) 다중 공유 링크들을 가진 스위치들



(b) 스위치 1에서의 내부 패킷 흐름

(그림 1) 스위치들간의 연결 및 내부 구성

서비스 클래스(*service class*)는 통신 프로토콜, 트래픽 종류, 서비스 종류, 관리 효율성, 또는 기타 기준에 따라 그룹화된 세션들의 집합으로 정의된다. 다중 링크 스위치는 공유 링크들의 전체 대역폭(bandwidth)을 클래스들에게 공정하게 할당하면서 요구된 QoS(quality of services)를 제공해야 한다. 가장 중요한 QoS로서 클래스들의 수와 그들의 트래픽 패턴에 영향을 받지 않고 각 클래스에게 사전에 예약된(reserved) 대역폭을 보장할 수 있어야 한다. 두 번째 QoS로서임의의 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[3-5]. 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[9-14]. 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 전송될 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 다중 공유 링크를 위한 링크 스케줄링 모델을 정의하고, 이를 기반으로 새로운 세션의 연결 설정 시 이를 어떤 링크에 할당할 것인지를 결정하는 경험적 세션할당 알고리즘을 제안한다. 이 외에도 본 논문에서는 다양한 세션할당 알고리즘들을 고안하고 검토하였으며, 시뮬레이션을 통해 이들의 장단점을 비교 분석한다. 제안된 알고리즘은 새로운 세션이 소속된 클래스의 각 링크에 할당된 세션들의 예측된 지연들 중 가장 작은 지연을 가진 링크에게 그 세션을 할당함으로써, 서로 다른 링크를 통해 서비스된 동일 클래스의 임의의 두 세션의 지연 차이를 최대한 감소시킬 수 있다. 또한 제안 알고리즘을 채택한 스위치는 다른 세션할당 알고리즘을 채택한 것에 비해 공유 링크 자원을 보다 효율적으로 사용하고 공정하게 대역폭을 할당함으로써, 서비스 클래스들에게 사전에 예약된 대역폭을 보다 확실히 제공할 수 있다.

2. 관련 연구 및 QoS

2.1 공정한 링크 스케줄링

본 절에서는 단일 링크 스케줄링 기술에 대해 간단히 살펴보고자 한다[6]. GPS(Generalized Processor Sharing)[7]는 패킷들이 무한정 작게 나누어질 수 있다고 가정하고 단일 링크를 통해 여러 세션들이 서로 다른 전송률로 동시에 전송될 수 있는 유체(fluid) 서비스 모델을 기반으로 하는 이상적인 스케줄링 알고리즘이다. GPS는 세션들에게 예약된 대역폭 보장, 서비스 분산의 공정성, 격리효과(isolation), 리키 버켓(leaky bucket) 트래픽 패턴을 따르는 세션들에게 고정된 종단간 지연(end-to-end delay bound) 등의 QoS를 제공한다.

GPS-관련 PFQ(Packet Fair Queueing) 알고리즘은 유체

시스템을 시뮬레이션하며 GPS 유체 서비스 모델하의 전송 시간을 예측하여 이를 기반으로 패킷들을 서비스하는 GPS의 패킷 단위(paket-by-packet) 스케줄링 알고리즘이다. PFQ 알고리즘들은 전송할 데이터가 존재하는 한 해당 링크를 놀리지 않고 계속해서 전송하며(work conserving), 상응하는 하나의 시스템 가상시간(system virtual time or potential)을 관리한다[8]. 시스템 가상시간은 감소하지 않는 시간 함수이며, 서버(PFQ)에 의해 전송된 총 데이터의 양을 반영하기 위해 계속해서 수정(증가)된다. 또한 시스템 내의 각 세션(또는 클래스)도 상응하는 하나의 가상시간을 가진다. 한 세션의 데이터가 도착하기 시작할 때, 그 세션의 가상시간은 처음에 시스템 가상시간과 같은 값을 가진다. 이후 연속적으로 계속해서 데이터가 도착하면 해당 세션의 가상시간은 서버로부터 받은 서비스 양을 반영하기 위해 계속 수정된다. 시스템 가상시간을 관리하는 방법과 서버에 의해 전송될 다음 패킷을 선택하는 패킷 선택 정책에 따라 다양한 PFQ 알고리즘들이 존재한다. 이러한 PFQ의 예로 WFQ(Weighted Fair Queueing)[9], WF^2Q(Worst-case Fair Weighted Fair Queueing)[10], WF^2Q+[11], SCFQ(Self-Clocked Fair Queueing)[12], VirtualClock[13], FFQ(Frame-based Fair Queueing)[14], SPFQ(Starting Potential based Fair Queueing)[14] 등을 들 수 있다. 이들은 공정성, 지연, 순간 패킷 유입량(burstiness), 계산 복잡도 등에서 차이점을 가진다[15].

기존의 PFQ 알고리즘들은 단일 링크를 스케줄링하기 위한 목적으로 설계 되었으므로, 이들만 활용해서는 다중 공유 링크를 가진 스위치에서 요구되는 QoS를 지원할 수 없다. 따라서 본 논문에서는 이들 알고리즘을 적절히 활용하는 새로운 스케줄링 모델을 제시하고, 효율적인 세션할당 알고리즘을 제안하고자 한다.

2.2 QoS 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 ϕ_i를 클래스 C_i의 공정분배율(fair share ratio)이라 하며, 임의의 링크 l의 링크 전송률(transmission rate)을 r_l로 표현한다. 공유 링크들의 총 링크 전송률 $r=\sum_{l=1}^{N} r_l$이라 할 때, C_i를 위해 $\phi_i \cdot \gamma$의 대역폭이 예약되며, $\sum_{i=1}^{M} \phi_i \cdot r \le r$이다. 본 논문에서는 편의상 정규화된(normalized) 링크 전송률을 사용한다. 따라서 $r=1$이고, $1 \le l \le N$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(back-logged) 되었다고 한다. 단일 링크를 위한 GPS[7]의 정의를 따르면서, 다중 공유 링크들을 위한 스케줄러의 공정성(fairness)을 다음과 같이 정의한다.

[정의 1] 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.

공정성은 두 가지 중요한 특성을 가진다[11]. 첫째, 백로그 되지 않은 클래스의 남는 대역폭을 백로그된 클래스들에게 할당해 주되 그들의 공정분배율에 비례하여 할당해 준다. 둘째, 시간구간 $[t_1, t_2]$에서 백로그된 클래스 C_i에 대해 수식 $W_i(t_1, t_2) \ge \phi_i W(t_1, t_2)$이 성립한다. 만약 동일 조건하의 단일 링크상에서 PFQ 알고리즘이 사용된다면, 두 번째 특성은 곧 C_i에게 예약된 대역폭을 보장할 수 있음을 의미한다. 이유는 단일 링크상에서 백로그된 클래스가 존재하는 한 PFQ 알고리즘은 항상 $r(t_2 - t_1) = W(t_1, t_2)$가 성립하기 때문이다[7]. 여기서 $r(t_2 - t_1)$는 구간 $[t_1, t_2]$에서 공유 링크들에 의해 전송 가능한 서비스 총량이다. 그러나 다중 공유 링크를 가진 스위치에서는 공정성이 제공된다고 해서 예약된 대역폭까지 보장하는 것은 아니다. 즉, $r(t_2 - t_1) \ne W(t_1, t_2)$가 성립할 수 있기 때문이다. 극단적인 예로 세션할당 알고리즘이 하나의 링크에만 모든 세션을 할당하고 다른 링크들에는 어떠한 세션도 할당하지 않았을 경우이다.

따라서 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 요소이다.

[정의 2] 공정성과 고-처리율을 지원함으로써 모든 클래스들에게 그들의 예약된 대역폭을 보장할 수 있다면, 그 스케줄러는 **클래스간**(*inter-class*) 공정성을 지원한다.

예약된 대역폭 보장은 DOS(denial of services) 공격 또는 사용자들의 오류로 인한 특정 클래스(또는 일부 세션들)의 과도한 부하가 모든 링크 자원을 점령하는 것을 방지하고 다른 클래스들에게 영향을 주지 않는 격리(isolation) 효과를

제공한다[6]. 따라서 본 논문에서 지원하고자 하는 첫 번째 QoS는 클래스들에게 사전에 예약된 대역폭을 보장하여 클래스간 공정성을 제공하는데 있다.

두 번째 QoS는 임의의 시간구간에서 동일 클래스에 속하는 두 세션이 서로 다른 공유 링크를 통해 서비스될 때 이들 세션들의 서비스 지연 차이를 가능한 비슷하게 하는 것이다. 이는 비록 그 클래스에게 클래스간 공정성을 지원하였다 해도 그것에 속한 세션들은 어떤 공유 링크를 통해 전송되느냐에 따라 서로 다른 지연을 가질 수 있기 때문이다.

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.

[정의 3] 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) 공정성을 지원한다. 즉, $|D'_{i,j}(t_1, t_2) - D'_{i,l}(t_1, t_2)| \leq \Delta D$이고, $\Delta D \leq \infty$는 클래스-내부 공정성 측정 요소이다.

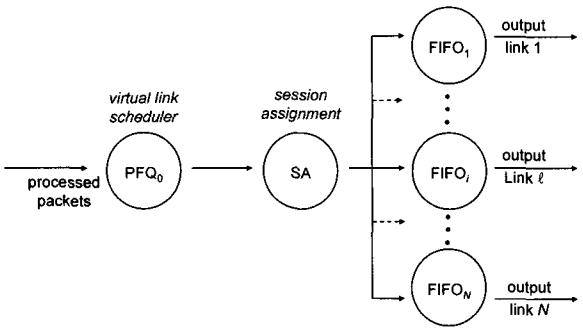
각 클래스에 소속된 세션들의 시작시간과 지속시간, 대역폭 요구량, 트래픽 패턴을 사전에 정확히 예측하기 힘들고, 또한 임의의 링크에 할당된 세션은 연결 설정 후에 다른 링크를 통해 전송될 수 없기 때문에, 위에서 기술한 세 측정 요소인 상한치 ΔF, ΔT, ΔD를 고정시킨다는 것은 매우 어렵다. 따라서 본 논문에서는 기존에 잘 알려진 단일 링크용 PFQ 알고리즘들[9-14]과 잘 설계된 세션할당 알고리즘을 적절히 연동시킴으로써 이들 상한치를 가능한 최소화 시키고자 한다.

3. 다중 공유 링크들을 위한 패킷 스케줄링

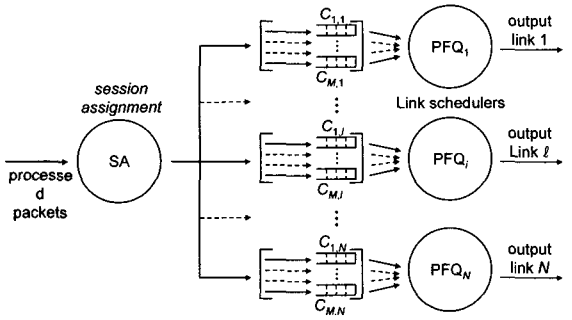
본 절에서는 클래스간 공정성과 클래스-내부 공정성 등의 QoS를 제공하기 위한 스케줄링 모델을 정의하고, 이를 기반으로 효율적인 경험적 세션할당 알고리즘을 제안한다.

3.1 다중 링크 스케줄링 모델

다중 공유 링크를 가진 스위치를 위한 스케줄링 모델로서 (그림 2)에 보인 두 가지 스케줄링 모델을 고려할 수 있다. 그림에서 SA(session assignment)는 세션의 연결 설정 시 어떤 링크를 통해 그 세션을 서비스할 것인지를 결정하고, 이후 그 세션의 모든 패킷들을 결정된 링크의 패킷 큐로 전달해 주는 역할을 담당한다. PFQ와 FIFO는 단일 링크를 위한 스케줄링 알고리즘으로 각 링크마다 하나씩 존재한다. PFQ는 기존의 잘 알려진 GPS 관련 공정한 링크 스케줄링 알고리즘들[9-14] 중의 하나이며, 해당 링크에서 각 클래스별로 하나의 큐를 관리하면서 그 클래스에 소속된 세션들의 패킷들을 도착 순서대로 큐에 보관한다. FIFO는 선입선출(first-in first-out) 알고리즘이며, 전송될 모든 패킷을 클래스에 상관없이 하나의 패킷 큐에 저장한다.



(a) PFQ-SA-FIFO 모델



(b) SA-PFQ 모델

(그림 2) 다중 공유 링크를 위해 고려되었던 두 스케줄링 모델

PFQ-SA-FIFO 모델은 세 단계를 가진 스케줄링 구조이다. 가상 링크 스케줄러인 PFQ_0에 의해 전송순서가 결정된 패킷들은 SA에 의해 서비스될 링크가 결정되며, 해당 링크의 스케줄러(FIFO)에 의해 실제 서비스된다. PFQ_0는 모든 공유 링크 전송률의 합과 동일한 전송률을 가진 하나의 가상 링크를 통해 모든 패킷이 서비스된다고 가정한다. 별도의 PFQ_0를 SA 앞에 배치한 이유는 단일 가상 링크를 통해 각 클래스에게 우선적으로 공정성과 예약된 대역폭을 제공할 수 있도록 모든 패킷의 전송순서를 미리 결정할 수 있는 이점을 제공하기 때문이다. PFQ-SA-FIFO 모델은 또한 단지 하나의 PFQ_0만을 사용하고 상대적으로 스케줄링 복잡도가 낮은 FIFO를 각 링크 스케줄러로 사용하기 때문에 SA-PFQ 모델보다 스케줄링 복잡도 측면에서 상대적인 이점을 가진다.

그러나 PFQ_0를 활용하는 방법은 두 가지 문제점이 있다. 먼저, 각 패킷은 가상의 링크를 통해 전송되는 시간만큼의 서비스 지연($\delta = L/r$)을 더 가지게 된다. 여기서 L은 패킷 길이이며, r은 가상 링크 전송률이다. 따라서 모든 패킷에게 δ만큼 더 증가된 지연을 부과한다. 둘째, PFQ_0에 의한 서비스 지연은 경우에 따라 속도가 빠른 공유 링크의 유휴상태를 초래할 수 있다. 즉, PFQ_0가 없을 경우 SA을 통과해 바로 지정된 공유 링크에서 전송될 수 있는 패킷을 PFQ_0가

δ만큼 더 지연시켰을 경우 그 만큼 해당 링크는 유휴상태로 더 대기해야 하는 경우가 발생할 수 있기 때문이다. 이는 곧 시스템 전체의 패킷 처리율을 감소시키는 요인이 될 수 있다. 따라서 본 논문에서는 PFQ_0의 이러한 문제점을 고려하여 SA-PFQ 모델을 채택하기로 결정했다. SA-PFQ 모델에서 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 전송률이다.

3.2 경험적 세션할당 알고리즘

본 논문에서 정의된 SA-PFQ 스케줄링 모델에서 각 링크의 PFQ는 단일 링크용 공정한 스케줄링 알고리즘이므로, 다중 공유 링크를 가진 스위치에선 링크들에게 세션을 적절히 할당하는 세션할당(SA) 알고리즘의 역할이 매우 중요하다. 세션할당은 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 전송률로 나눈 정규화된 대역폭 요구량을 의미한다.

이러한 고려 사항들을 단적으로 반영하는 것이 각 부클래스의 예측지연(expected delay)이다. SA-PFQ모델에서 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 l상의 부클래스 $C_{i,l}$의 예측지연, $D_{i,l}(t)$는 다음과 같은 특성을 가진다.

① 클래스 C_i를 제외한 각 클래스의 부하가 공유 링크들에게 균일하게 분산되었다면, 각 링크에 할당된 C_i의 부클래스들 중 가장 작은 부하를 가진 부클래스 $C_{i,l}$가 다른 링크들에 할당된 C_i의 부클래스들 보다 더 짧은 $D_{i,l}(t)$를 가진다.

② 클래스 C_i의 부하가 각 링크에게 균일하게 분산되었다면, 가장 작은 링크 부하를 가진 링크 l의 C_i의 부클래스 $C_{i,l}$가 C_i의 다른 부클래스들 보다 더 짧은 $D_{i,l}(t)$를 가진다.

③ 시간 Δt에서 모든 클래스의 부하가 링크들에게 균일하게 분산되어 있더라도, 임의의 Δt에 대해 구간 $[t-\Delta t, t)$ 동안 C_i의 부클래스들 중 가장 작은 부하를 가졌던 부클래스 $C_{i,l}$가 C_i의 다른 부클래스들 보다 더 짧은 $D_{i,l}(t)$를 가진다.

④ 모든 클래스의 부하가 링크들에게 불균일하게 분산되었다면, C_i의 부클래스들 중 다른 클래스들의 부클래스들에 비해 상대적으로 가장 작은 부하를 가진 $C_{i,l}$가 C_i의 다른 부클래스들 보다 더 짧은 $D_{i,l}(t)$를 가진다.

예측지연의 이러한 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, $\min_{1 \le l \le N}(D_{i,j}(t))$인 그러한 링크 l을 선택한다.

```
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 3) 제안된 SCDF 세션할당 알고리즘

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 이용하여 계산할 수 있다. 즉, $D_{i,j}(t) \leftarrow \frac{L_{i,l}(t)}{\phi_i r_l}$ 이며, $\phi_i \cdot r_l$은 $C_{i,l}$을 위해 예약된 링크 l의 대역폭이다. 예측지연 $D_{i,l}(t)$를 계산하기 위해 평균 큐 길이를 사용하는 이유는 클래스 C_i 뿐 아니라 다른 클래스에 소속된 세션들의 패킷 과도 유입(burstiness) 시 현재의 큐 길이는 이를 너무 민감하게 반영하기 때문이다. 그러나 시간 0에서부터 현재 시간 t까지의 일반적인 산술 평균 큐 길이는 매 순간 순간에 동일한 가중치를 부여하기 때문에 이 또한 적절하지 않다. 일반적으로 우리는 보다 최근의 큐 길이에 더 큰 가중치를 부여하기를 바란다. 이유는 최근의 큐 길이가 현재 뿐 아니라 미래의 큐 길이를 보다 더 밀접하게 반영할 수 있기 때문이다. 과거의 값들을 기반으로 미래의 큐 길이를 예측할 수 있는 널리 활용되는 기술은 지수(exponential) 평균 기법이다[16]. 즉, $L_{i,l}(t+1) = \alpha T_{i,l}(t) + (1-\alpha) L_{i,l}(t)$이다. 여기서

$T_{i,l}(t)$는 $C_{i,l}$의 현재 큐 길이이며, 는 보다 최근 또는 과거의 큐 길이에게 부여될 상대적 가중치를 결정하는 상수 가중 인자(constant weighting factor, $0<\alpha<1$)이다. 본 논문에서는 평균 큐 길이 계산을 위해 지수 평균 기법을 사용한다.

(그림 3)에 보인 제안 알고리즘 SCDF는 새로운 패킷이 도착할 때마다, 그 패킷이 이미 연결이 설정된 세션의 패킷일 경우 그 세션이 할당된 링크의 해당 부클래스의 패킷 큐($Q_{i,l}$)에 삽입한다. 그렇지 않고 새로운 세션의 첫번째 패킷일 경우, 그 세션이 소속된 클래스 C_i의 부클래스들 중 예측지연이 가장 작은 부클래스(링크 l)를 선택하고, 이후 그 세션의 모든 패킷들을 선택된 링크 l를 통해 서비스하게 한다.

4. 성능 평가

본 절에서는 제안된 SCDF알고리즘과 다른 세션할당 알고리즘과의 성능을 모의실험을 통해 비교 평가한다. 먼저 다양한 세션할당 알고리즘에 대해 간단히 논의하고, 이들의 비교평가를 위한 모의실험 환경을 기술한 후, 실험결과를 분석하도록 한다.

4.1 다양한 세션할당 알고리즘

본 논문에서는 제안된 SCDF 외에 다음과 같은 다양한 세션할당 알고리즘들을 고안하고 검토하였다. RR(*Round Robin*)은 클래스의 세션들을 연결 순서대로 링크들에게 순차적으로할당하는 가장 간단하고 기본적인 알고리즘이다. 각 클래스 C_i에 대해 최근에 할당한 링크의 인덱스 l_i를 저장하고 있다가 새로운 세션의 연결 요청 시 이를 l_i의 다음 링크에 할당한다. 즉, $l_i=((l_i+1) \bmod N)+1$인 링크 l_i를 선택한다. 여기서 N은 공유 링크들의 개수이다. RR은 세션 할당 시 각 링크의 부하를 전혀 고려하지 않기 때문에 특정 링크로 부하가 편중되는 문제가 발생할 수 있다.

보다 직관적인 방법은 각 클래스에 소속된 세션들을 그들의 부하를 고려하여 링크들에게 균등하게 분산하는 것이다. 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.

각 클래스의 부하가 링크별로 균등하게 분산되지 않아도 각 링크에는 공정한 링크 스케줄러인 PFQ가 있다. 따라서 클래스에 상관없이 전체 부하를 링크들에게 균등하게 분산해 준다면, 각각의 링크에서는 클래스별 공정분배율이 만족되지 않을지라도 전체적으로는 클래스별 공정분배율이 만족될 수 있다. 이런 고찰을 기반으로 MLUF(*Minimum Link Utilization First*)는 클래스에 상관없이 부하가 가장 적은 링크에게 새로운 세션을 할당하는 알고리즘이다. 즉, $\min_{1\le l\le N}(U_{*,l}(t))$인 그러한 링크 l을 선택한다. 여기서 $U_{*,j}(t)=\sum_{i=1}^{M}U_{i,l}(t)$이다. MLUF와 유사한 전략으로 각 링크에서 서비스를 위해 대기 중인 패킷들의 예측지연을 계산하여 이를 활용하는 방안을 고려할 수 있다. SLDF(*Shortest Link Delay First*)는 클래스에 상관없이 각 링크에 할당된 패킷들의 예측지연이 가장 작은 링크에게 새로운 세션을 할당하는 알고리즘이다. 즉, $\min_{1\le l\le N}(D_{*,l}(t))$인 링크 l을 선택한다. 여기서 $D_{*,l}(t)$는 링크 l의 각 부클래스의 패킷 큐에서 대기 중인 모든 패킷들의 예측지연이다. MLUF와 SLDF는 각 링크에게 총 부하를 균등하게 분산함으로써 링크 자원들을 효율적으로 사용하고, 다른 알고리즘에 비해 서비스된 모든 패킷들의 평균 지연이 가장 짧은 이점을 제공할 수 있다.

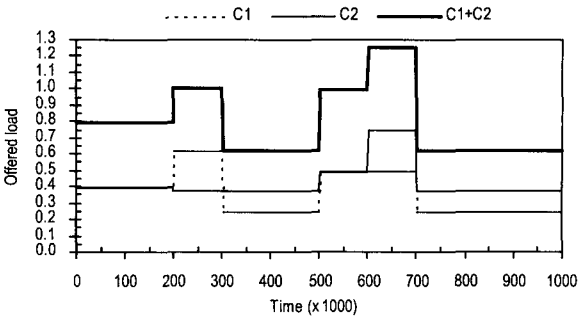
4.2 모의실험 환경

본 논문에서는 (그림 2)의 SA-PFQ 스케줄링 모델을 지원하는 패킷-기반 시뮬레이터를 개발하였다. 각 링크 스케줄러인 PFQ로는 WF^2Q+[11]를 사용하였다. WF^2Q+는 상한치(bounded)를 가진 지연을 제공하고 GPS와 견줄만한 공정성을 제공한다. 실험에서 부클래스들을 위한 큐 길이는 무한하다고 가정했으며, 두 개의 공유 링크를 가진 스위치를 대상으로 실험했다. 두 링크의 전송률 r_i는 각각 0.5이다. 결과분석의 편의를 위해 두 개의 클래스(C_1과 C_2)만 지원했으며, 이들의 공정분배율은 각각 0.5이다. SA-PFQ 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[11, 15, 17], 평균은 각각 250과 10,000이다. 각 클래스에는 항상 평균 20개의 세션들이 연결되어 서비스된다.

(그림 4)는 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[11].

DOS 공격과 같은 과도한 패킷이 도착하는 상황을 반영

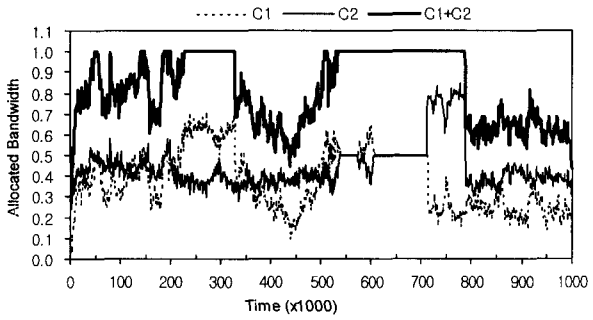
하기 위해 클래스 C_2의 첫 번째 세션으로 0.25의 과도한 부하를 가진 세션 $s_{2,2,1}$을 생성했다. 세션 $s_{2,2,1}$은 예외적으로 실험 시작부터 끝까지 존속하며, 링크 2(부클래스 $C_{2,2}$)에 할당되어 서비스된다. 시간구간 [200, 300]에서 클래스 C_1은 그것의 예약된 대역폭 0.5보다 더 많은 0.625의 과부하를 가진다. 반대로 클래스 C_2의 부하는 예약된 대역폭보다 낮은 0.375이며, $s_{2,2,1}$를 제외하면 0.125이다. 따라서 이 구간에서 $s_{2,2,1}$를 제외한 C_2의 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.



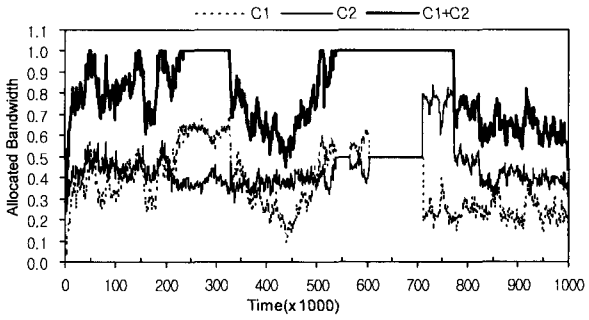
(그림 4) 클래스들에게 제공된 평균 부하

4.3 실험 결과 및 분석

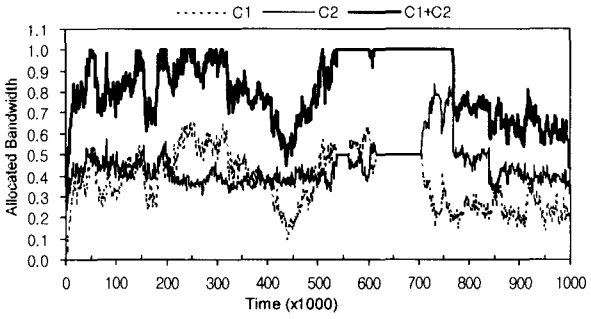
모의실험의 목적은 (그림 4)와 같이 시간대별 편중된 부



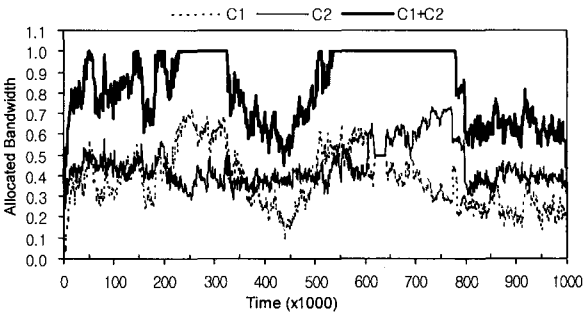
(a) SGLK



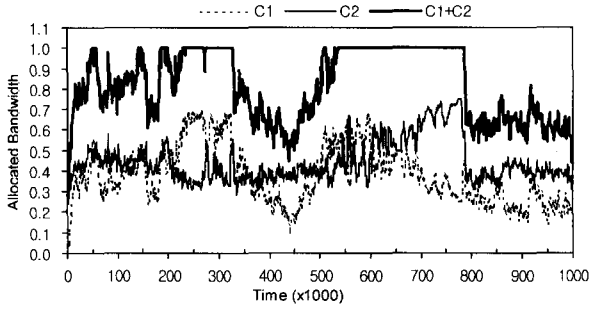
(b) SCDF



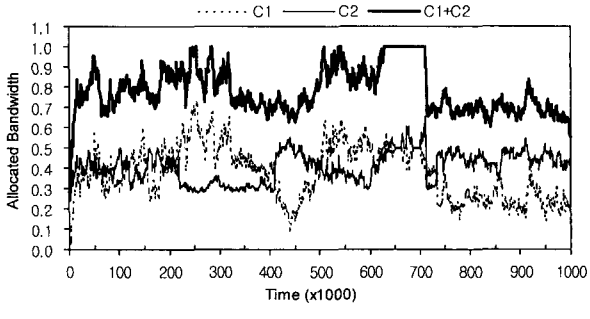
(c) MCUF



(d) MLUF

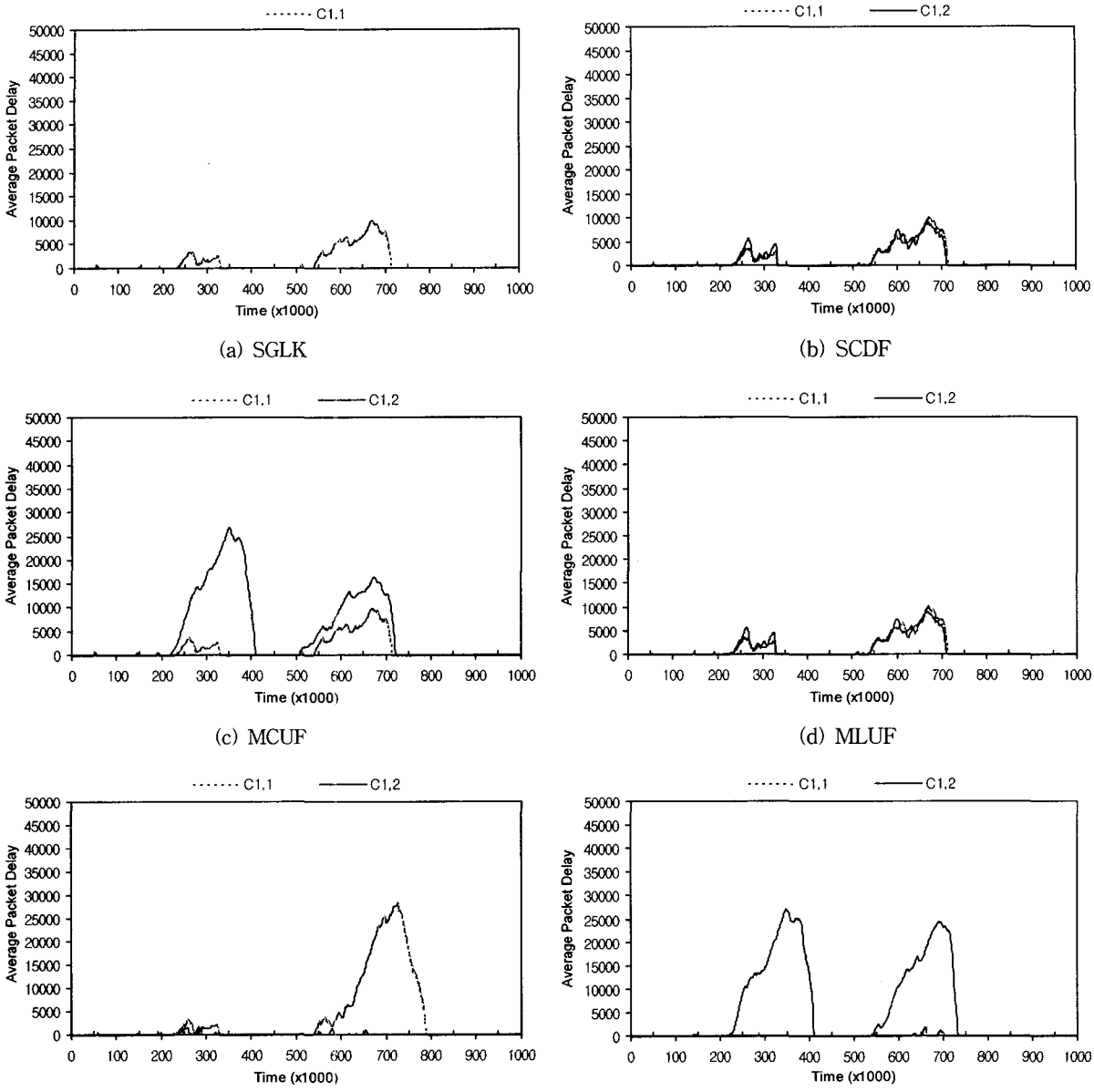


(e) SLDF



(f) RR

(그림 5) 할당된 대역폭



(a) SGLK (b) SCDF

(c) MCUF (d) MLUF

(e) SLDF (f) RR

(그림 6) 클래스 C_1의 각 링크별 부클래스의 평균 지연

하 또는 과부하가 주어졌을 경우, 각 세션할당 알고리즘이 클래스간 공정성 및 클래스-내부 공정성을 지원하는지의 여부를 확인하는 것이다. 먼저 각 알고리즘별 클래스간 공정성의 지원여부를 확인 해보자. (그림 5)는 (그림 4)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. (그림 5)(a) SGLK(single link)는 본 논문의 모의실험 대상인 두 개의 공유 링크를 가진 스위치의 총 링크 전송률과 동일한 전송률을 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 보이고, RR이 가장 좋지 않은 결과를 보인다. (그림 4)의 시간구간 [200, 300]에서 제공된 부하는 (그림 5)의 구간 [240, 340]에서 그 파급효과가 나타난다. 이 구간에서 SCDF, MLUF, SLDF는 총 링크 전송률과 동일한 대역폭(C_1+C_2)을 할당했으며, 또한 SGLK처럼 과부하를 가진 C_1에게 예약된 대역폭(0.5)을 제공한다. 그러나 MCUF와 RR은 총 링크 전송률보다 더 적은 대역폭을 할당하며, 고-처리율을 지원하지 못한다. MCUF는 C_2에게 SGLK와 비슷한 대역폭을 할당하지만 C_1에게는 SGLK에 비해 상

대적으로 적은 대역폭을 할당한다. 이는 MCUF가 C_2의 부하를 전혀 고려하지 않은 채 C_1의 부하를 각 링크에 균등하게 분산했기 때문이다. 앞 절의 (그림 4)에서 기술했듯이 $C_{2,1}$의 부하(0.125)가 $C_{2,2}$의 부하(0.25)보다 상대적으로 적기 때문에 C_1의 세션들을 상대적으로 부하가 적은 링크 1로 더 많이 할당해야 C_1에게 보다 많은 대역폭을 할당할 수 있다.

(그림 4)의 시간구간 [500, 700]에서 제공된 부하는 (그림 5)의 구간 [540, 710]에서 그 파급효과가 나타난다. 이 구간에서 SCDF는 SGLK와 가장 비슷한 결과를 보이며, 두 클래스에게 고-처리율과 공정성, 예약된 대역폭 할당 등을 지원한다. MCUF와 RR은 총 링크 전송률과 동일한 대역폭을 할당하지 못한다. MLUF와 SLDF는 링크 자원들을 충분히 활용하여 고-처리율은 지원하지만, 구간 [620, 710]에서 공정성을 지원하지 못한다. 즉, 두 클래스에게 모두 0.5의 대역폭을 할당해야 하나, C_2에게 상대적으로 더 많은 대역폭을 할당하고 C_1에게는 예약된 대역폭을 할당하지 못한다. 이는 MLUF와 SLDF가 각 클래스의 부하가 링크들에게 어떻게 분산되었는지에 상관없이 단지 링크 부하만 고려하여 세션들을 할당했기 때문이다.

다음으로 각 알고리즘별 클래스-내부 공정성의 지원 여부를 확인 해보자. (그림 6)은 (그림 5)와 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. 두 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.

(그림 6)(c) MCUF의 시간구간 [250, 400]에서 $C_{1,1}$의 지연은 거의 없는 반면 $C_{1,2}$의 지연은 매우 크다. 이 구간에서 C_2의 부하가 $C_{2,1}$(0.125)보다 $C_{2,2}$(0.25)로 편중되어 있어도, MCUF는 C_1의 부하를 각 링크에 균등하게 분산하여 $C_{1,1}$과 $C_{1,2}$의 부하를 거의 동일(각각 0.3125)하게 만든다. 이 경우 링크 1의 부하($C_{1,1}+C_{2,1}$)는 링크 전송률(0.5)보다 적고 $C_{2,1}$의 남는 대역폭이 $C_{1,1}$에게 주어지기 때문에 $C_{1,1}$의 지연은 거의 없다. 반면, $C_{2,2}$는 예약된 대역폭과 동일한 부하를 가지므로 같은 링크에 할당된 $C_{1,2}$에게 주어질 남는 대역폭이 없다. 따라서 $C_{1,2}$의 지연은 상대적으로 커지게 된다. 동일 시간구간에서 MLUF와 SLDF는 SCDF와 SGLK보다 더 작은 평균 지연을 보인다. 그러나 이들은 그림으로는 보이지 않았지만 동일 구간에서 C_2의 두 부클래스인 $C_{2,1}$과 $C_{2,2}$가 상대적으로 매우 큰 지연을 가졌다.

(그림 6)(d) MLUF와 (그림 6)(e) SLDF의 시간구간 [550, 780]에서 $C_{1,2}$의 지연은 거의 없는 반면 $C_{1,1}$의 지연은 매우 크다. MLUF와 SLDF는 단지 링크들의 부하만 고려하여 세션들을 할당한다. 따라서 (그림 4)에서 기술한 것 같이 편중된 부하를 가진 세션 $s_{2,2,1}$이 링크 2에 존재하고 구간 [500, 600]에서 C_1과 C_2가 예약된 대역폭과 동일한 부하를 가질 경우, 링크 1의 부하($C_{1,1}+C_{2,1}$)는 링크 전송률과 동일한 부하를 가지지만 $C_{1,1}$은 예약된 대역폭(0.25)보다 더 많은 부하(0.375)를 가지고 $C_{2,1}$은 그 보다 더 적은 부하(0.125)를 가지게 된다. 반대로 링크 2에서는 $C_{1,2}$와 $C_{2,2}$가 각각 0.125, 0.375의 부하를 가진다. 그런데 구간 [600, 700]에서 C_2의 부하가 더 증가하여 과부하(0.75) 상태가 되면, $C_{2,1}$, $C_{2,2}$의 부하는 각각 0.25, 0.5로 증가한다. 이 경우 $C_{1,2}$는 예약된 대역폭보다 더 적은 부하를 가지므로 지연이 거의 없지만, $C_{1,1}$은 그 자체가 과부하이고 또한 이와 동일한 링크에 할당된 $C_{2,1}$이 충분한 부하를 가졌기 때문에 $C_{1,1}$의 지연은 매우 커지게 된다.

이상의 실험결과를 통해 RR은 각 클래스 및 링크의 부하를 전혀 고려하지 않기 때문에 가장 좋지 않은 결과를 보인다. MCUF는 과도한 대역폭을 요구하는 세션이 연결되거나 또는 특정 링크에 할당된 특정 클래스의 세션들이 조기에 종료되어 각 링크에 분산된 그 클래스의 부하가 불균형을 이룰 때, 모든 링크 자원을 충분히 활용하지 못하는 문제점이 있다. MLUF와 SLDF는 과부하가 걸릴 경우 특정 클래스의 세션들을 특정 링크로 편중되게 할당함으로써 클래스간 공정성과 클래스-내부 공정성을 지원하지 못한다는 것을 확인할 수 있다.

5. 결 론

본 논문에서는 다중 공유 링크를 가진 스위치를 위한 다중 링크 스케줄링 모델을 정의하고 세션 연결 설정 시 이를 어떤 링크에 할당할 것인지를 결정하는 경험적 세션할당 알고리즘 SCDF를 제안했다. SCDF는 새로운 세션이 소속된 클래스의 각 링크에 할당된 부클래스의 예측된 지연들 중 가장 작은 지연을 가진 부클래스(링크)에게 새로운 세션을 할당한다. 다른 세션할당 알고리즘에 비해 SCDF는 부클래스들의 예측지연 특성을 직접적으로 활용함으로써, 동일한 클래스에 속한 세션들에게 서로 다른 공유 링크를 통해 전송되어도 가능한 비슷한 서비스 지연을 제공한다는 것을 모의실험을 통해 확인했다. 또한 동일한 클래스의 세션들을 어느 한 링크에 상대적으로 과도하게 또는 적게 할당하는 것을 방지하여 총 대역폭을 보다 공정하게 할당하고, 총 부하를 링크들에게 균형되게 분산하여 과부하 시 링크 자원들을 충분히 활용하는 보다 높은 처리율을 제공했다.

본 연구팀은 향후 각 부클래스의 예측지연 특성을 수학

적으로 모델링하고 이를 증명하고자 한다. 이를 통해 각 부클래스의 현 부하가 다른 부클래스에게 어느 정도 영향을 미치는지 파악하고, 또한 각 부클래스의 예측지연을 측정하지 않고도 임의의 클래스의 부클래스들 중 상대적으로 가장 작은 부하를 가진 부클래스를 보다 쉽게 찾을 수 있을 것으로 기대한다.

참 고 문 헌

[1] C. L. Scuba and E. H. Spafford, "A Reference Model for Firewall Technology," Proc. of the 13th Annual Computer Security Applications Conference (ACSAC), San Diego, CA, pp.133-145, Dec., 1997.

[2] S. Axelsson, "Research in Intrusion Detection Systems: A Survey," Technical Report 98-17, Dep. of Computer Engineering, Chalmers University of Technology, Dec., 1998.

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

[4] R. Russo, L. Kencl, B. Metzler, and P. Droz, "Scalable and Adaptive Load Balancing on IBM Power NP," Technical Report RZ-3431(#93699), IBM Zurich Research Laboratory, Jul., 2002.

[5] T. Wolf, P. Pappu, and M. A. Franklin, "Predictive Scheduling of Network Processors," Computer Networks, Vol.41, No.5, pp.601-621, Apr., 2003.

[6] H. Zang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," Proc. of the IEEE, Vol.83, No.10, pp.1374-1396, Oct., 1995.

[7] 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.

[8] D. Stiliadis and A. Varma, "Rate-Proportional Servers : A Design Methodology for Fair Queueing Algorithms," IEEE/ACM Transactions on Networking, Vol.6, No.2, pp.164-174, Apr., 1998.

[9] 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.

[10] J. C. R. Bennett and H. Zang, "WF2Q: Worst-Case Fair Weighted Fair Queueing," Proc. of IEEE INFOCOM'96, San Francisco, California, pp.120-128, Mar., 1996.

[11] 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.

[12] S. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications," Proc. of IEEE INFOCOM'94, Toronto, CA, pp.636-646, June, 1994.

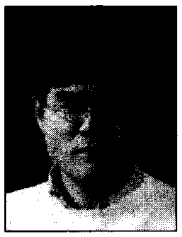
[13] L. Zang, "VirtualClock : A New Traffic Control Algorithm for Packet Switching Networks," ACM Transactions on Computer Systems, Vol.9, No.2, pp.101-124, May, 1991.

[14] 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.

[15] F. M. Chiussi and A. Francini, "A Distributed Scheduling Architecture for Scalable Packet Switches," IEEE Journal on Selected Areas in Communication, Vol.18, No.12, pp. 2665-2683, Dec., 2000.

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

[17] N. Ni and L. N. Bhuyan, "Fair Scheduling and Buffer Management in Internet Routers," Proc. of IEEE INFOCOM 2002, New York, June, 2002.



심 재 홍

e-mail : jhshim@chosun.ac.kr

1987년 서울대학교 전산과학과(학사)

1989년 아주대학교 컴퓨터공학과(석사)

2001년 아주대학교 컴퓨터공학과(박사)

1989년~1994년 서울시스템(주) 공학연구소

1999년~2000년 University of Arizona 객원연구원

2001년~2001년 아주대학교 정보통신전문대학원 BK21 전임연구원

2001년~현재 조선대학교 인터넷소프트웨어공학부 전임강사

관심분야 : 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템

최 경 희

e-mail : khchoi@madang.ajou.ac.kr

1976년 서울대학교 수학교육과(학사)

1979년 프랑스 그랑데꼴 Enseeiht대학(석사)

1982년 프랑스 Paul Sabatier대학 정보공학부(박사)

1982년~현재 아주대학교 정보통신전문대학원 교수

관심분야 : 운영 체제, 분산시스템, 실시간 및 멀티미디어 시스템 등

정 기 현

e-mail : khchung@madang.ajou.ac.kr

1984년 서강대학교 전자공학과(학사)

1988년 미국 Ilinois주립대 EECS(석사)

1990년 미국 Purdue대학 전기전자공학부(박사)

1991년~1992년 현대반도체 연구소

1993년~현재 아주대학교 전자공학부 교수

관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등