사용자 기반 상대적 차별화를 위한 계층적 결손 보완 라운드-로빈 스케줄링 알고리즘

(A Hierarchical Deficit Round-Robin Packet Scheduling Algorithm for User-Oriented Relative Differentiated Services)

편 기 현 † 이 종 열 †† 조 성 익 ††
(Kihyun Pyun) (Jong-Yeol Lee) (Sung-Ik Cho)

요 약 네트워크 제공자들뿐만 아니라 인터넷 사용자들도 최선 서비스(best-effort service)를 뛰어 넘어 사용자들 간에 서로 다른 서비스 품질을 받기를 열망하고 있다. 이 논문은 사용자 세션 단위로 차별화 서비스를 제공할 수 있는 알고리즘을 제안한다. 제안하는 알고리즘은 결손 보완 라운드-로빈 알고리즘을 기반으로 확장한 계층적 결손 보완 라운드-로빈 알고리즘이다. 이 알고리즘의 주 장점은 응용의 타입을 별도로 구분하지 않으면서도 FTP와 같이 대역폭에 민감한 응용뿐만 아니라 VoIP와 같이 지연에 민감한 응용의 품질 차별화를 제공할 수 있다는 점이다. 품질 차별화 서비스를 제공함에 있어서 네트워크 제공자 측면에서 중요한 점은 예측성과 제어성이다. 우리는 수학적 분석과 모의실험 실험을 통해서 제안하는 알고리즘이 예측성과 제어성이 기존의 결손 보완 라운드-로빈 알고리즘보다 월등히 뛰어남을 보인다. 그럼에도 불구하고 계층적 결손 보완 라운드-로빈 알고리즘의 구현 복잡도는 O(1)이다.

키워드 : 패킷 스케줄링, 서비스 품질, 차별화 서비스

Abstract The Internet users as well as network providers are eager to have different qualities of service among users beyond the best-effort. In this paper, we propose a scheduling algorithm that provides a differentiated service in the granularity of user sessions. The proposed algorithm is a Hierarchical Deficit Round-Robin (H-DRR) algorithm that is an extension of an existing DRR algorithm. A main advantage is that H-DRR provides service differentiation for throughput-intensive applications such as FTP as well as delay-sensitive applications such as telnet or VoIP without distinguishing the types of applications. The most importance in providing a service differentiation in term of network providers is to have controllability and predictability. We show that H-DRR is superior to DRR in terms of controllability and predictability through both mathematical analysis and simulation experiments. Nevertheless, H-DRR requires O(1) complexity for implementation.

Key words : packet scheduling, quality-of-service, differentiated service

1. 서 론

네트워크 제공자들뿐만 아니라 인터넷 사용자들도 최선 서비스(best-effort service)를 뛰어 넘어 사용자들 간에 서로 다른 서비스 품질 제공을 열망하고 있다. 예를 들어, ADSL(Asynchronous Digital Subscriber Line) 네트워크 제공자는 사용자가 ADSL에 가입할 때 두 단계의 서비스 등급, 즉, 고급(premium) 등급과 일반(normal) 등급을 두고, 일반 등급 사용자보다 고급 등급 사용자에게 더 나은 서비스를 제공하고자 한다. 더 나은 서비스의 의미는 응용 프로그램 타입에 따라서 다르다. 예를 들어, FTP 응용의 경우 더 많은 대역폭이 주어질 때 더 좋은 서비스를 받는다고 느낀다. 반면 텔넷(telnet) 혹은 게임 응용의 경우 더 짧은 패킷 지연이 더 나은 서비스를 의미한다. 따라서 다양한 응용 프로그램 관점에서 서비스 차별화(service differentiation)를 제공할 수 있는 모델이 필요하다. 서비스 차별화를 제공함에 있어서 또 다른 중요한 핵심은 구현 복잡도가 충

· 이 논문은 2004년도 전북대학교 신진교수과제(과제번호: 103782001)의 지원을 받았음
† 종신회원 : 전북대학교 전자정보공학부 교수
khpyun@chonbuuk.ac.kr
†† 비 회 원 : 전북대학교 전자정보공학부 교수
jong@chonbuk.ac.kr
sicho@chonbuk.ac.kr
논문접수 : 2004년 7월 15일
심사완료 : 2005년 9월 15일

분히 작아서 많은 수의 사용자들에게 서비스를 제공할 수 있어야 한다는 점이다.

이 논문은 사용자 세션 단위로 차별화 서비스를 제공할 수 있는 스케줄링 알고리즘을 제안한다. 제안하는 알고리즘은 기본 메커니즘이 [1]에서 제안된 DRR 알고리즘을 기반으로 확장한 계층적 결손 보완 라운드-로빈(H-DRR: Hierarchical Deficit Round-Robin) 알고리즘이다. H-DRR 알고리즘은 세션들의 큐 위에 서비스 등급에 기반을 둔 계층 구조를 유지하며 각 서비스 등급 내에서 사용자당 하나의 큐가 유지된다. 이 알고리즘의 주 장점은 응용의 타입을 별도로 구분하지 않으면서도 FTP와 같이 대역폭에 민감한 응용뿐만 아니라 VoIP와 같이 지연에 민감한 응용의 품질 차별화를 제공할 수 있다는 점이다. 품질 차별화 서비스를 제공함에 있어서 네트워크 제공자 측면에서 중요한 점은 예측성(predictability)과 제어성(controllability)이다. 예측성은 상위 등급 사용자들이 하위 등급 사용자들보다 더 나은 서비스를 받아야 함을 의미한다. 제어성은 네트워크 제공자가 서비스 등급들 간에 설정하는 인자 값에 비례해서 서비스 품질이 조절될 수 있어야 함을 의미한다. 우리는 수학적 분석과 모의실험 실험을 통해서 제안하는 알고리즘이 예측성과 제어성을 가짐을 보이며, 기존의 DRR과 비교했을 때 월등히 뛰어난 성능을 나타냄을 보인다. H-DRR 알고리즘의 구현 복잡도는 S가 서비스 등급의 수 일 때 O(S)이다. 그러나 대개 서비스 등급의 개수는 2-4개로 고정되므로 구현 복잡도는 상수로 간주할 수 있다.

문헌 [2]에서 제안한 상대적 차별화 서비스는 차별화하는 단위가 너무 커서 각 응용의 품질 요구를 만족시키기에 불충분한 점이 있다. 수많은 세션이 단지 몇 개의 등급으로 나누어지고, 각 서비스 등급 내의 세션들은 전혀 서로 구별되지 않기 때문이다. 최악의 경우 가장 높은 등급에 속한 세션이 트래픽의 군집성 (burstiness) 때문에 만일 순간적으로 무거운 트래픽 부하가 발생하면 같은 등급의 다른 세션들이 대역폭을 자신보다 더 많이 차지하고 심지어 하위 등급의 세션들보다 더 적은 대역폭을 받을 수도 있다. 이러한 방식은 세션 당 상태 유지가 필요 없기 때문에 코어 (core) 라우터에 적합하다. 그러나 에지 라우터는 세션들의 상태를 유지할 여력이 있고, 이를 통해서 코어 라우터의 품질 차별화 성능 문제를 보완할 수 있다면 의미 있는 일로 여겨진다.

이 논문의 구성은 다음과 같다. 2절은 관련 연구를 설명한다. 3절은 사용자 기반 차별화 서비스를 제공하는 계층적 결손 보완 라운드-로빈 알고리즘을 제안한다. 4절과 5절은 각각 수학적 분석과 모의실험 결과를 제시한다. 마지막으로 6절은 이 논문을 매듭짓는다.

2. 관련 연구

IETF에 의해 제안된 차별화 서비스는 확장성(scalability)을 위해서 코어 라우터(core router)에서 세션 당 상태(per-session state)를 유지하지 않고, 소수의 등급 간에 서로 다른 품질을 제공한다[3]. 각 라우터의 각 등급에 대한 서비스 품질은 사용된 Per-Hop Behavior(PHB), 예들 들면 EF(Expedited Forwarding) PHB[4], AF (Assured forwarding) PHB[5]에 의해서 결정된다. 특히 [2]에 제안된 상대적 차별화(relative differentiation) PHB는 상위 등급에 속한 패킷들이 하위 등급에 속한 패킷들보다 더 짧은 패킷 지연과 더 낮은 패킷 소실 확률을 준다. 그러나 수많은 세션들이 몇 개의 등급으로 병합되기 때문에 상위 등급에 속한 세션은 하위 등급에 속한 세션들보다 더 나은 서비스 (예: 대역폭)를 받음을 확신할 수 없다. 코어 라우터는 확장성 때문에 세션 당 상태를 유지하기 어렵지만 우리는 세션 당 상태를 유지하는 것이 에지 라우터(edge router)에게는 필수적이지는 않다고 믿는다.

[6,7]에서 연구된 계층적 공평 서비스 스케줄링 알고리즘은 계층 데이타 구조를 사용한다는 측면에서 우리가 제안하는 H-DRR 알고리즘과 유사하다. 그러나 계층적 공평 서비스에서 사용된 계층 데이타 구조는 서비스 할당율에 따라서 *미리 만들어지며 정적*이다. 반면, 제안하는 데이타 구조는 각 서비스 등급의 현재 사용자 수에 따라서 *가변적으로 변하며 서비스 등급에 할당되는 서비스 비율 또한 동적으로 변화*된다.

제안하는 H-DRR 알고리즘은 DRR 알고리즘을 근간으로 한다. 그러나 DRR 알고리즘은 사용자 기반 차별화 서비스 요구를 만족시키는데 부족하다. 그러나 대역폭 측면만 고려한다면 DRR 알고리즘도 어느 정도 성능 차별화 기능을 제공할 수 있다. 예들 들어, 고급 등급과 일반 등급이 존재하고 고급 등급에 속한 세션들이 일반 등급에 속하는 세션들보다 10배 더 많은 대역폭을 주고자 한다면, 고급 등급에 속하는 세션들에게 가중치 값으로 10을 주고, 일반 등급 세션들에게는 가중치 값을 1로 주면 된다. 그러나 패킷 지연의 경우 이러한 방식은 4절

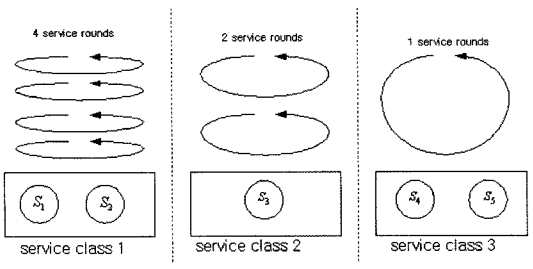

그림 1 예제

에서 보였듯이 두 등급 간에 별반 차별화 기능을 제공하지 못한다. 근본적인 이유는 DRR 알고리즘의 경우 각 세션이 서비스 받을 때까지 걸리는 지연 시간과 가중치 값은 서로 무관하기 때문이다. H-DRR과 DRR의 또 다른 큰 차이점은 DRR은 가중치 값이 고정되지만, H-DRR의 경우 가중치 값이 동적으로 계속 변화한다는 점이다.

3. 계층적 결손 보완 라운드-로빈 알고리즘

계층적 결손 보완 라운드-로빈(H-DRR) 알고리즘에 대한 쉬운 이해를 위해 예를 들어 보자. 우선 세 개의 서비스 등급이 있다고 가정하자. 서비스 등급 1은 두 세션 S_1과 S_2를 갖고 있다. 또한 서비스 등급 2는 세션 S_3을 갖고 있고, 서비스 등급 3은 세션 S_4와 S_5를 갖고 있다. 네트워크 제공자는 서비스 등급 1, 2, 3에게 각각 4 대 2 대 1의 비율로 서비스 품질을 상대적으로 차별화하고자 한다. 이 때 H-DRR의 주 아이디어는 그림 1에 보인바와 같이 서비스 등급 3에 있는 세션들에게 한 번의 서비스 라운드가 주어질 때 마다 서비스 등급 1과 2에게는 각각 네 번과 두 번의 서비스 라운드를 제공하는 것이다.

이 아이디어는 말로는 간단해 보이지만, 이 아이디어를 실현하는 스케줄링 알고리즘을 설계하는 데는 기술적으로 두 가지 어려운 문제점을 해결해야 한다. 첫째는 서비스 등급 내에서 서비스해야 하는 세션의 수가 고정되어 있지 않고, 패킷 도착 시간을 예측할 수 없기 때문에 현 시점에서 서비스해야 할 세션 수가 시간에 따라서 변화한다는 점이다. 심지어 승인 제어(admission control)를 하지 않기 때문에 서비스 등급에 속할 수 있는 세션의 최대 개수조차도 미리 알 수 없다. 따라서 각 서비스 등급에 주어져야 하는 서비스 양은 서비스 등급 간 서비스 라운드 제약 조건을 만족시키기 위해 현재 상태에 따라 작은 시간 구간에 대해서 동적으로 적응되어야 한다는 점이다. 둘째로 그럼에도 불구하고 고속의 구현을 위해서 스케줄링 알고리즘의 복잡도는 상수이여야 한다는 점이다.

본 논문은 이 두 가지 문제점을 해결하는 H-DRR 알고리즘을 제안한다. H-DRR은 세션 큐 위에 서비스 등급에 기반을 둔 계층을 유지한다. 고정된 수의 등급이 주어지면, 동일한 개수의 비잎 노드(non-leaf node)를 각 단계에 미리 하나씩 생성해 놓으며 각 단계에는 오직 한 개의 비잎 노드만이 존재한다. 서비스 등급 i에 속하는 세션이 들어오면, i 단계에 잎 노드(leaf node)를 하나 새롭게 생성하고 그 세션에게 할당한다. 그림 2는 그림 1에 보인 예에 대해서 형성된 계층 구조를 보여준다. 이 그림에서 세 개의 비잎 노드 C_1, C_2, C_3는

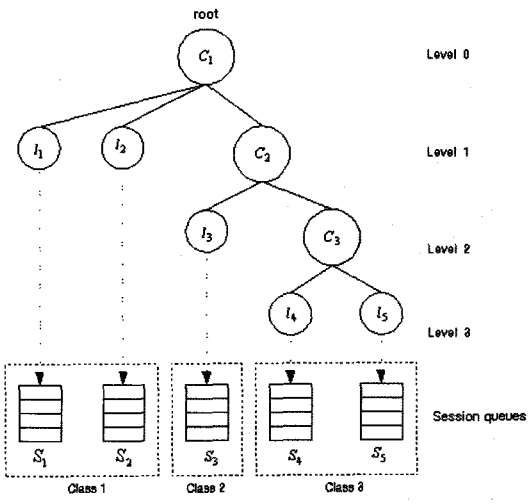

그림 2 H-DRR에서 형성된 계층 구조 예

각각 서비스 등급 1, 2, 3을 위해서 미리 형성된다. 서비스 등급 1에 속하는 세션들, 즉, S_1과 S_2는 1 단계에 있는 잎 노드 l_1과 l_2의 의해 가리켜진다. 마찬가지로 서비스 등급 2에 속하는 세션 S_3은 2 단계에 있는 잎 노드 l_3에 의해서 가리켜진다. 이 계층 구조에서 비잎 노드는 자신의 대역폭을 자식들에게 분배시키는 목적을 가지고 있다. 0 단계에 있는 특별한 비잎 노드는 루트라 부르며 전송선을 나타낸다. 루트는 우리가 서비스 슬롯(service slot)이라 부르는 단위 시간 동안 전송선 대역폭을 자식들에게 라운드-로빈 방식으로 하나씩 준다. 서비스 슬롯을 분배하는 방법에 대한 자세한 설명은 이 절의 끝으로 미룬다. 주목할 점은 비잎 노드는 서비스 슬롯을 자신이 직접 소모하지 않고 잎 노드에게 전달한다는 점이다. 반면 잎 노드는 서비스 슬롯을 소모하면서 자신이 가리키는 세션 큐에서 패킷을 전송한다. 따라서 오직 잎 노드만이 세션 큐에 대한 포인터를 갖는다.

한 서비스 슬롯 동안, 세션 큐로부터 평균 Q 바이트가 전송되는데 Q값을 퀀텀 크기(quantum size)라고 부른다. Q 값은 최대 패킷 크기보다 작지 않는 정수이어야 한다. 일반적으로 패킷 크기는 가변이기 때문에 잎 노드는 각 슬롯 동안 정확히 Q 바이트를 전송할 수 있는 것은 아니다. 각 잎 노드 k는 그 이전 슬롯에서 결손된 서비스를 보상받기 위해서 보완 카운터 d_k를 유지한다. 초기에 보완 카운터 d_k는 0으로 리셋된다. 잎 노드 k가 서비스 슬롯을 받으면, 그 세션 큐가 비지 않고 또 전송된 양이 Q 바이트에다가 d_k를 더한 양보다 크지 않을 때 까지 패킷을 전송한다. 전송이 완료된 후 만일 그 세션 큐가 비면 d_k 값은 영으로 초기화된다. 그렇지 않으면, Q 더하기 d_k의 값에다가 전송된 양을 뺀 값

```
1  consumeServiceSlot( n )
2    d_n = Q + d_n;
3    while( d_n > 0 and not isEmpty(queue(n)) )
4      size = packetsize(head(queue(n)));
5      if( size ≤ d_n )
6        send(dequeue(queue(n)));
7        d_n = d_n − size;
8      else break;
9    return isEmpty(queue(n));
```

그림 3 H-DRR의 서비스 슬롯 소모 모듈

이 보완 카운터 d_k에 저장된다. 그림 3은 잎 노드에 주어진 서비스 슬롯을 소모하는 모듈에 대한 C 언어와 유사한 방식의 의사 코드(pseudo code)를 보여준다. 표 1은 이 논문에서 사용된 모든 의사 코드에 묵시적으로 사용된 함수들의 설명을 요약한 것이다.

패킷을 전송할 때 빈 세션 큐를 살피지 않기 위해 각 비잎 노드는 저장된 패킷이 있는 자식 노드들에 대한 색인을 항목으로 갖는 서비스 목록을 유지한다. 만일 세션 큐가 적어도 하나 이상의 전송할 패킷을 가지고 있으면, 우리는 그 세션과 그 세션 큐를 가리키는 잎 노드가 저장 기간(backlogged period) 내에 있다고 말한다. 비잎 노드의 경우 만일 그 자손 노드들 중에 저장 기간 내에 있는 잎 노드가 하나 이상 존재하면 저장 기간 내에 있다고 정의한다. 새롭게 저장 기간 내에 있게 된 세션은 그 경로 상에 있는 잎 노드부터 루트까지 모두 저장 기간 내에 있도록 만들 수도 있다. 서비스 목록의 항목들은 새롭게 저장 기간 내에 있게 되는 노드들의 색인을 첨가함으로써 생성된다. 또, 서비스 목록의 항목은 세션 큐가 비게 될 때 마다 제거된다. 이 경우 비게 되는 세션 큐를 가리키는 잎 노드에 대한 색인이 그 부모 노드의 서비스 목록으로부터 제거된다. 만일
그 서비스 목록의 항목이 모두 없어지면, 동일한 제거 행위가 위쪽으로 재귀적으로 전파된다.

서비스 슬롯이 분배되는 방법을 설명하기 이전에 노드의 가중치에 대해서 설명할 필요가 있다. 각 노드 n은 양의 정수인 가중치 w_n을 갖는다. 직관적으로 이 값은 자신의 형제 노드들과 비교했을 때 얼마나 많은 대역폭 (즉, 서비스 슬롯)을 자신의 부모로부터 받아야 하는 지를 의미한다. H-DRR 알고리즘은 노드가 잎 노드인가 그렇지 않은가에 따라서 가중치를 할당하는 방법이 전혀 다르다. 잎 노드는 오직 하나의 저장된 세션을 가질 수 있으므로 자신이 속한 서비스 등급에 관계없이 항상 가중치 값으로 1을 할당한다[5]. 주목할 점은 잎 노드의 경우 가중치 값이 항상 고정된 값 1이 사용되지만, 비잎 노드의 가중치 값은 미리 고정되어 있지 않다는 점이다. 우리는 서로 다른 시간 구간에 대해서 서로 다른 양의 패킷 서비스를 각 서비스 등급에 제공해야 하기 때문에, 각 서비스 등급에 대해 저장 기간 내에 있는 자식 노드들의 수에 따라서 적응적으로 변화되어야 한다.

표 1 의사코드에 사용된 표기

함수	설명
append(l,n)	노드 n에 대한 색인을 서비스 목록 l의 끝에 추가.
deletehead(l)	서비스 목록 l로부터 머리 항목을 지움.
dequeue(q)	큐 q로부터 머리 패킷을 추출.
enqueue(s,p)	세션 s의 큐에 패킷 p를 입력.
head(l or q)	서비스 목록 l 혹은 큐 q로부터 머리 항목 혹은 패킷을 돌려줌.
isEmpty(q)	만일 큐 q가 비었으면 TRUE를 돌려줌.
list(n)	노드 n의 서비스 목록을 돌려줌.
movehead(l)	머리 항목을 서비스 목록 l의 끝으로 옮김.
packetsize(p)	패킷 p의 크기를 돌려줌.
parent(n)	노드 n의 부모 노드를 돌려줌.
queue(n)	노드 n에 의해서 가리켜지는 큐를 돌려줌.
send(q)	큐 q로부터 패킷 하나 전송.

구체적으로 $i-1$ 단계에 위치한 비잎 노드 n을 고려하자. 이 노드 n은 서비스 등급 i에 속하는 세션들을 가리키는 잎 노드들의 부모 노드이다. 네트워크 사업자가 서비스 등급 $i-1$과 서비스 등급 i에 설정한 품질 차별 인자를 각각 Q_{i-1}과 Q_i로 표기 하자. 그러면, 이 두 서비스 등급 간 상대적 품질 차별화 비율은 Q_{i-1} 대 Q_i이고, 이 값이 변수 R_n에 저장된다고 하면, R_n 값은 다음과 같다.

$$R_n = \frac{Q_{i-1}}{Q_i}. \tag{1}$$

식 (1)에서 구현의 단순성을 위해서 각 품질 차별 인자 값 (예: Q_{i-1}, Q_i)은 R_n 값이 정수가 되도록 적절히 설정되었다고 가정한다. 이 경우 i 단계에서 한 번의 서비스 라운드는 R_n 번의 부라운드로 나누어져서, $i-1$ 단계에서 한 번의 라운드가 완결될 때마다 i 단계의 부라운드가 하나씩 완결되어야 한다. 매 첫 번째 부라운드의 시작에서 비잎 노드 n은 저장 기간 내에 있는 자식들의 가중치 값의 합을 변수 WS_n에 저장한다. 저장 기간 내에 있는 자식들의 수는 시간에 따라 변하므로 WS_n은 i단계에 있는 자식들 중 이번 라운드 동안 고려되어야 할 노드들의 수를 담고 있다. 만일 어떤 노드가 WS_n 값을 설정할 때 저장 기간 내에 있지 않다가 이 값을 설정한 후 저장 기간 내에 있게 되면 그 노드는 이번 라운드가 아니라 다음 라운드에 서비스 된다. WS_n 값을 계산해서 설정한 후 부라운드가 매번 새롭게 시작할 때 마다 비잎 노드 n의 가중치 w_n은 WS_n/R_n으로 설정되고, 노드 n이 서비스 슬롯을 하나 받을 때 마다 1씩 감소한다. 사실 노드의 가중치 값은 정수 값이어야 하므로 H-DRR은 정확히 WS_n/R_n이 아닌 WS_n/R_n 근처 값을 할당한다. 그러나 하나의 완전한 라운드를 구성하는 부라운드들의 합은 항상 정확히 WS_n 값과 일치해

5) 사실 4절에서 보였듯이 모든 잎 노드들이 동일한 가중치 값을 가지기만 한다면 어떤 양의 정수 값을 할당할 수 있지만 가중치를 1로 할당하는 것이 가장 좋은 성능을 보인다.

```
1  setWeight( n )
2    if( w_n == 0 )
3      if( RSR_n == 0 )
4        RSR_n = R_n;
5        if( n has the backlogged non-leaf node c )
6          setWeight( c )
7          WS_n = NUM_n + w_c - 1;
8        else
9          WS_n = NUM_n;
10       REM_n = WS_n;
11     w_n = REM_n / RSR_n;
12     REM_n = REM_n - w_n;
13     RSR_n = RSR_n - 1;
```

그림 4 비잎 노드 n에 대한 가중치 값을 결정하는 의사코드

야만 한다.

그림 4는 비잎 노드 n의 가중치 값을 결정하는 의사코드를 보여준다. 함수 setWeight(n)은 만일 비잎 노드 n의 가중치 값이 호출 당시 0이 아닌 값을 갖는 경우, 즉 자식 노드들에 대한 부라운드가 현재 진행 중인 경우 아무 일도 수행 하지 않는다. 가중치 값 w_n은 오직 각 부라운드 시작 시점에만 0이 된다는 점을 인지해야 한다. 라인 3에서 라인 10은 만일 남은 부라운드의 수를 저장하는 RSR_n의 값이 0이면, 즉, 첫 번째 부라운드 시작 시점인 경우 가중치 값의 합 WS_n을 새롭게 결정한다. 라인 4에서 노드 n의 자식 노드들의 한 라운드는 R_n 개의 부라운드로 나뉘어 진다. 라인 9에서 만일 노드 n이 저장 기간 내에 있는 자식이 하나도 없으면 가중치 합 WS_n은 단순히 노드 n의 저장 기간 내에 있는 자식들의 수를 저장한 NM_n 값이 된다. 그렇지 않으면, 즉, 만일 노드 n이 저장 기간 내에 있는 비잎 노드, 가령 노드 c를 갖는 경우 함수 setWeight()은 재귀적으로 노드 c에게 적용된다. 이 경우 WS_n 값은 저장 기간 내에 있는 자식들의 수, 즉, NM_n 값에다가 노드 c의 가중치 값 w_c를 더한 것에 1을 뺀다 (라인 7). 여기서 1을 빼는 이유는 노드 c의 가중치 값이 이미 더해 졌으므로 NM_n 값에서 노드 c를 제외시키기 위해서다. 라인 10에서 우리는 WS_n 값을 이번 라운드에서 자식들이 받아야 할 서비스 슬롯의 총 수를 저장하는 REM_n 변수에 복사한다. 라인 11은 노드 n의 이번 부라운드에서 설정되어야 할 노드 n의 가중치 w_n의 값을 결정한다. 라인 12에서 라인 13은 다음 부라운드를 위해서 변수 RSR_n과 REM_n 값을 적절히 설정한다.

이제 H-DRR 알고리즘이 서비스 슬롯을 어떻게 분배하는 지를 살펴보자. 전송선이 패킷을 보낼 준비가 되면, 서비스 슬롯이 루트에 하나씩 주어진다. 만일 루트를 포함한 비잎 노드가 서비스 슬롯을 받게 되면, 항상

```
1  Dequeueing:
2    while( TRUE )
3      if(NUM_root > 0 ) giveServiceSlot( root );
4
5  giveServiceSlot( n )
6    next = head(list(n));
7    if( next is non-leaf )
8      setWeight( next );
9      empty = giveServiceSlot( next );
10     w_next = w_next - 1;
11     if( empty == TRUE)
12       deletehead(list(n));
13       NUM_n = NUM_n - 1;
14       return (NUM_n == 0);
15     else
16       if( w_next == 0 )
17         movehead(list(n));
18       returnFALSE;
19   else /* if n is a leaf node */
20     return consumeServiceSlot( n );
```

그림 5 패킷 출력 모듈

서비스 목록의 머리 노드에 그 슬롯을 전달한다. 머리 노드가 서비스 슬롯을 사용한 후에도 여전히 전송할 패킷이 남아있고 가중치와 동일한 개수의 서비스 슬롯을 받은 경우, 그 색인이 서비스 목록의 끝으로 옮겨진다. 결국 각 서비스 슬롯은 잎 노드에 도착한다. 그러면, 잎 노드는 그 슬롯 시간 동안 자신의 세션 큐에서 패킷을 전송한다. 구체적인 패킷 출력 모듈은 그림 5에 의사 코드로 나타내었다. 라인 2-3에서 H-DRR 알고리즘은 저장 기간 내에 있는 자식이 존재하면 루트 노드에게 서비스 슬롯을 하나씩 준다. 각 서비스 슬롯은 라인 3에서 giveServiceSlot() 함수를 호출함으로써 적절한 세션에게 전달된다. 이 함수에서 노드 n이 서비스 슬롯을 받을 때(라인 5), 서비스 목록의 머리 노드를 선택해서 *next*로 명명한다(라인 6). 만일 *next*가 잎 노드이면 서비스 슬롯을 소모함으로써 그 대응되는 세션 큐에 있는 패킷들을 전송한다(라인 20). 그렇지 않으면, 즉, *next*가 비잎 노드이면, 라인 7에서 라인 18이 이 경우를 다루는데 그 서비스 슬롯은 소모되지 않고 *next*에 재귀적으로 전달되며 *next*의 가중치 값은 하나 감소한다(라인 10). 그 뒤 만일 *next*가 더 이상 저장 기간 내에 있지 않으면 *next*는 노드 n의 서비스 목록에서 제거된다 (라인 12-13). 그렇지 않은 경우 만일 *next*의 가중치 값이 0이 되면 *next*는 노드 n의 서비스 목록의 끝에 다시 첨부된다(라인 16-17).

그림 6은 패킷이 도착할 때 적용되는 H-DRR의 입력 모듈을 나타낸다. 패킷이 도착할 때 그 패킷은 해당 세션의 큐에 입력된다(라인 2). 만일 이 도착이 그 세션을 새롭게 저장 기간 내에 있게 하면 그 세션 큐를 가리키

```
1  Enqueueing: on arrival of packet p to session s
2    enqueue(s,p);
3    if( session s is newly backlogged)
4      let the node n be the leaf for session s;
5      addToList( n );
6
7  addToList( n )
8   let p be parent(n);
9    if( n is a leaf node ) d_n = 0;
10   w_p = 0;
11   RSR_p = 0;
12   append( list(p), n );
13   NUM_p = NUM_p + 1;
14   if( p != root and NUM_p == 1 )
15     addToList( p );
```

그림 6 H-DRR의 입력 모듈

고 있는 잎 노드 n가 addToList() 함수를 호출함으로써 노드 n의 부모 노드에 서비스 항목을 추가한다. 이 함수의 동작은 라인 7에서 15에 기술되어 있다. 이 함수는 우선 각종 변수들을 초기화 하고(라인 9-11), 부모 노드에 서비스 항목을 추가하고(라인 12-13), 그리고 필요하다면 부모 노드의 부모 노드에 재귀적으로 적용된다.

예를 들어, 그림 7은 그림 2에 나타난 계층 구조를 가정했을 때 서비스 슬롯이 어떻게 적절한 세션에게 분배되고 전달되는 지를 보여준다. 세션들은 S_1, S_4, S_2, S_3, S_5의 순서로 저장 기간 내에 있게 되었다고 가정한다. 또 서비스 등급 2에 속하는 추가적인 세션 S_6이 새롭게 승인되고 t_2 시점에 세션 S_6으로부터 패킷이 도착했다고 가정한다. 또 이 그림은 setWeight() 함수를 사용해서 계산된 비잎 노드의 가중치 값과 가중치 합의 값이 어떻게 변화하고 있는 지도 포함하고 있다. 시간 t_0에 루트(C_1)에 서비스 슬롯이 주어졌을 때 그 슬롯은 노드 C_2, C_3, l_4를 거쳐서 결국 세션 S_4에 전달된다. 비잎 노드의 가중치 합의 값을 설정하는 시간 점들 사이의 슬롯 구간들은 해당 서비스 등급에 대한 각 서비스 라운드 지속 기간으로 간주할 수 있다. 예를 들어, 슬롯 구간 $(t_0,t_6]$은 서비스 등급 3에 대한 한 번의 서비스 라운드 구간으로 간주할 수 있고, 서비스 등급 2에 대해서는 두 번의 서비스 라운드 구간, 즉, $(t_0,t_4]$ 구간 동안 한 번의 라운드와 $(t_4,t_6]$ 구간 동안 또 한 번의 라운드로 간주할 수 있다. 서비스 등급 2에 대한 k-번째 라운드에 속하는 t_2 시간에 세션 S_6로부터 도착하는 패킷들은 그 다음 라운드, 즉 $(k+1)$-번째 라운드에 서비스를 받기 시작한다. 주목할 점은 서비스 등급 1에 속하는 세션 S_1, 서비스 등급 2에 속하는 세션 S_3, 그리고 서비스 등급 3에 속하는 세션 S_4는 시간 구간 $(t_2,t_6]$ 동안 서비스 슬롯을 각각 4 개, 2개, 그리고 1개를 받는다. 이 서비스 등급 간 서비스 비율은 측정하는 시간 구간에 따라서 다소 위반될 수 있지만 그 위반되는 양은 4절에

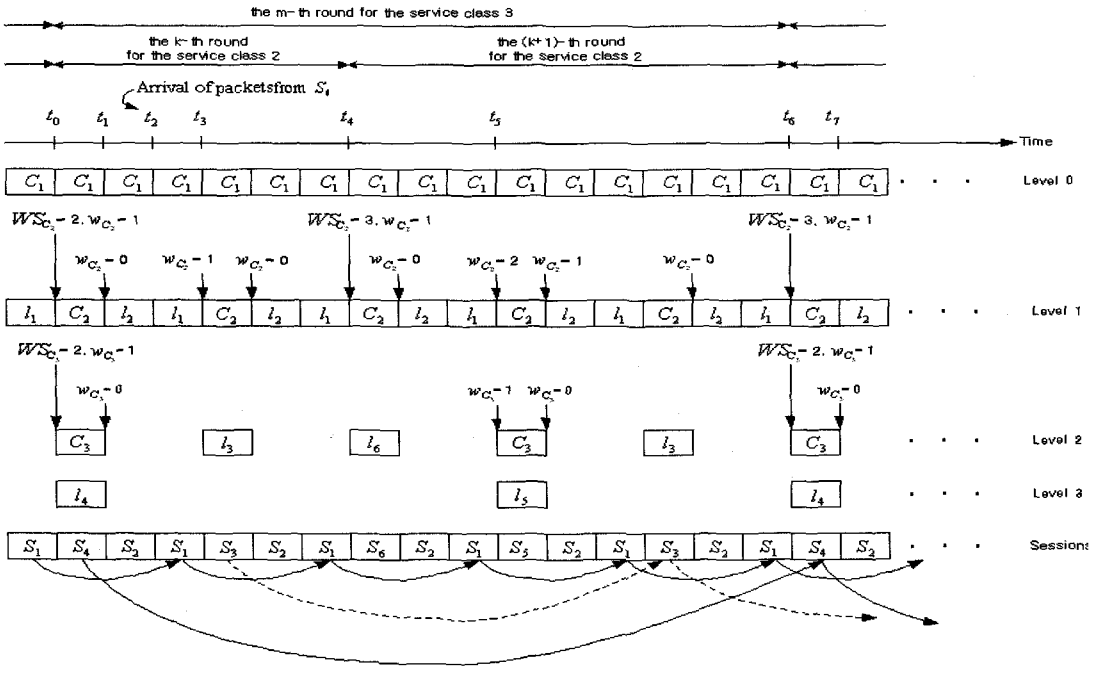

그림 7 서비스 슬롯 분배와 소모 예

보인 바와 같이 단지 작은 상수 값으로 한정된다.

H-DRR 알고리즘의 구현 복잡도는 사실상 O(1)이다. 입력 모듈과 출력 모듈을 분석해 보면 구현 복잡도는 계층의 깊이가 D라고 할 때 O(D)임을 알 수 있다. 그러나 계층의 깊이는 서비스 등급의 개수와 동일하고, 서비스 등급은 미리 2-4개 정도로 고정된다. 따라서 H-DRR 알고리즘의 구현 복잡도는 상수로 간주할 수 있다.

4. 수학적 분석 결과

공평성에 대해서 많이 받아들여지는 정의 중 하나는 [8]에서 제안된 것이다. 서비스 등급 c에 속하는 세션 i에 의해서 임의의 시간 구간 $(s,t]$ 동안에 전송된 패킷의 양을 $W_i^c(s,t)$로 표기하자. [8]에서 제안한 공평성 정의는 할당된 서비스 비율에 의해 정규화 된 서비스 양(normalized service amounts)의 차이는 상수로 한정된다는 직관적인 의미를 갖는다. 구체적으로, 동일한 서비스 등급 c에 속하는 임의의 두 세션 i와 j가 저장 기간 내에 있는 모든 시간 구간 $(t_1,t_2]$에 대해서 스케줄링 알고리즘이 다음 식을 만족하면 우리는 이 알고리즘이 서비스 등급내의 세션들에게 공평하다고 정의한다.

$$|W_i^c(t_1,t_2) - W_j^c(t_1,t_2)| \le C_1. \quad (2)$$

여기서 C_1는 이 스케줄링 알고리즘의 공평도라고 불리는 상수 값이다. C_1의 값이 더 작을수록 더 높은 공평도를 갖는다.

또, 각각 서비스 등급 k과 l $(k \neq l)$에 속하는 임의의 두 세션 i와 j가 저장 기간 내에 있는 모든 시간 구간 $(t_1,t_2]$에 대해서 스케줄링 알고리즘이 다음 식을 만족하면 우리는 이 알고리즘이 서비스 등급 간에 제어성을 제공한다고 정의한다.

$$\left| W_i^k(t_1,t_2) - \frac{Q_k}{Q_l} W_j^l(t_1,t_2) \right| \le C_2. \quad (3)$$

여기서 Q_k와 Q_l은 서비스 등급 k와 l에 설정된 서비스 품질 인자 값이다. 또, C_2는 우리가 알고리즘의 제어도(controllability measure)라 부르는 상수이다. 더 작은 C_2 값을 제공하는 알고리즘이 더 짧은 시간 구간에 대해서도 대역폭을 네트워크 사업자가 원하는 대역폭 비율로 서비스할 수 있다.

한 가지 지적할 점은 식 (3)은 예측성도 포함하고 있다는 점이다. 여기서 예측성은 상위 등급에 속한 세션이 하위 등급에 속한 세션보다 더 낫거나 혹은 최소한 더 나쁘지 않은 서비스를 받음을 의미한다. 구체적으로 우리는 각각 서비스 등급 k와l $(k \neq l)$에 속하는 임의의 두 세션 i와 j가 저장 기간 내에 있는 모든 시간 구간 $(t_1,t_2]$에 대해서 스케줄링 알고리즘이 다음 식을 만족하면 우리는 이 알고리즘이 서비스 등급간에 예측성을 제공한다고 정의한다.

$$W_j^l(t_1,t_2) - W_i^k(t_1,t_2) \le C_3. \quad (4)$$

여기서 C_3는 우리가 알고리즘의 예측도 (predictability measure)라 부르는 상수이다. 식 (4)는 낮은 우선순위를 갖는 서비스 등급 l에 속한 세션 j의 여분의 서비스 양이 높은 우선순위를 갖는 서비스 등급 k에 속한 세션 i의 여분의 서비스 양보다 기껏해야 C_3 바이트를 초과할 수 없음을 의미한다. 따라서 더 작은 C_3 값을 제공하는 알고리즘이 더 예측 가능하다. 주목할 점은 식 (3)과 식 (4)로부터, 스케줄링 알고리즘이 제어성을 가지면 예측성도 가지게 된다는 점이다. 그러나 그 역은 성립하지 않는다.

제안하는 H-DRR 알고리즘은 동일 서비스 등급에 속한 세션들에게 공평하다. 우선 다음 보조정리 1은 저장 기간 동안 잎 노드에 의해서 전송된 패킷의 최소량과 최대량을 추출한다.

- **보조정리 1** H-DRR 알고리즘에서 연속적으로 저장 기간 내에 있는 잎 노드 n을 고려하자. 퀀텀 크기를 Q, 그리고, 최대 패킷 크기를 $L^{\max}$로 표기하자. 또 노드 n과 형제 노드들의 k-번째 서비스 라운드의 종료 시간을 r_k로 표기하고, 첫 번째 라운드의 시작 시간을 r_0으로 표기하자. 만일 K번째까지 서비스 라운드 동안, 즉, 시간 구간 $(r_0,r_K]$ 동안 노드 n이 가리키는 세션 큐에서 전송된 패킷 양을 $W_n(r_0,r_K)$으로 표기하면,

$$KQ - L^{\max} \le W_n(r_0,r_K) \le KQ + L^{\max}. \quad (5)$$

증명. 부록을 참조할 것. □

- **정리 1** H-DRR 알고리즘에서 동일한 서비스 등급 c에 속하고, 시간 구간 $(\tau,t]$ 동안 계속해서 저장 기간 내에 있는 두 세션 i와 j를 고려하자. 퀀텀 크기를 Q, 그리고 최대 패킷 크기를 $L^{\max}$로 표기하자. 그러면,

$$|W_i(\tau,t) - W_j(\tau,t)| \le (2L^{\max} + Q). \quad (6)$$

증명. 부록을 참조할 것. □

정리 1은 높은 공평도를 갖기 위해서는 Q가 작은 값을 가져야 함을 보여 준다. 가장 작은 Q 값은 최대 패킷 크기까지로 제한되므로, H-DRR 알고리즘은 Q 값을 최대 패킷 크기로 설정할 때 가장 높은 공평도를 갖는다.

또, H-DRR 알고리즘은 서비스 등급 간 제어성을 제공한다.

- **보조정리 2** H-DRR 알고리즘에서 시간 구간 $(\tau,t]$ 동안 계속해서 저장 기간 내에 있는, 서비스 등급 k에 속하는 세션 i와 서비스 등급 l에 속하는 세션 j를 고려하자. 일반성을 잃지 않고, $k<l$이라고 가정하자. 서비스 등급 c에 대한 서비스 품질 인자 값을 Q_c로 표기하고, 시간 구간 $(\tau,t]$ 동안 세션 m에게 서비스 완료된 라운드 수를 K_m으로 표기하자. 그러면,

$$\left\lfloor \frac{Q_l}{Q_k}(K_i-1) \right\rfloor \le K_j \le \left\lceil \frac{Q_l}{Q_k}(K_i+1) \right\rceil . \quad (7)$$

증명. 부록을 참조할 것. □

- **정리 2** H-DRR 알고리즘에서 시간 구간 $(\tau, t]$ 동안 계속해서 저장 기간 내에 있는, 서비스 등급 k에 속하는 세션 i와 서비스 등급 l에 속하는 세션 j를 고려하자. 일반성을 잃지 않고, $k<l$이라고 가정하자. 서비스 등급 c에 대한 서비스 품질 인자 값을 Q_c로 표기하고, 시간 구간 $(\tau,t]$ 동안 세션 m에게 서비스 완료된 라운드 수를 K_m으로 표기하자. 또한, 시간 구간 $(\tau,t]$ 동안 서비스 등급 c에 속하는 세션 m으로부터 전송된 양을 $W_m^c(\tau,t)$으로 표기하자. 퀀텀 크기를 Q, 그리고 최대 패킷 크기를 $L^{\max}$로 표기하자. 그러면,

$$\left| W_i^k(\tau,t) - \frac{Q_k}{Q_l} W_j^l(\tau,t) \right| \le (Q+L^{\max})\left(1+\frac{Q_k}{Q_l}\right). \quad (8)$$

증명. 부록을 참조할 것. □

5. 모의실험

이 절에서 우리는 모의실험 실험을 통해서 H-DRR 알고리즘과 DRR 알고리즘을 공평성과 제어성 측면에서 비교한다. 실험은 널리 사용되는 ns-2 시뮬레이터 [9]를 사용하였다.

그림 8은 비교를 위한 실험 시나리오 환경을 나타낸다. 총 12개의 세션 s1, ..., s12가 라우터 R1을 거쳐 R2와 통신을 한다. 각 세션과 라우터 R1은 5 Mbps 전송선 대역폭을 가지며 전파 지연은 2 ms이다. 라우터 R1과 R2간 전송선은 10 Mbps 대역폭을 제공하고, 전파 지연은 1 ms이다. 최대 패킷 크기는 1 KB이고 세 개의 서비스 등급, 즉, 서비스 등급 1, 서비스 등급 2, 그리고 서비스 등급 3을 가정하였으며 서비스 품질 인자 값은 각각 4, 2, 1로 설정하였다. 서비스 등급 1은 세 개의 FTP 세션 s1, s2, s3과 한 개의 텔넷 세션 s4, 그리고 한 개의 on/off 세션인 s5를 갖는다. 서비스 등급 2는 세 개의 FTP 세션 s6, s7, s8 과 한 개의 텔넷 세션 s9를 갖는다. 서비스 등급 3은 두 개의 FTP 세션 s10, s11과 한 개의 텔넷 세션 s12를 갖는다. FTP 세션들은 성능 평가시 대역폭이 중요하고, 텔넷 세션들은 패킷 지연이 더욱 중요하다. On/Off 세션은 네트워크 부하를 가변적으로 변화시키기 위한 용도로 사용된다. 즉, 500 ms동안 패킷을 2 Mbps로 일정하게 보내고 500 ms 동안은 쉬는 것을 반복하는데 On 기간과 Off 기간에 대역폭이 공평하게 서비스 비율에 맞게 각 서비스 등급의 세션들에게 분배되는 지를 확인하기 위한 것이다. DRR과 H-DRR 알고리즘에 설정되어야 하는 퀀텀 크기는 둘 다 1 KB로 설정하였다.

그림 9와 그림 10은 각각 DRR과 H-DRR의 경우 서비스 등급 2에 속한 세션들의 성능(throughput)을 나타낸다. 이 그림들은 긴 실험 시간 중에 초기 시간이 지난 10 초부터 13.5 초까지 구간에 대한 성능 결과를 나타낸 것이며, 작은 시간 구간에 대한 대역폭 분배를 자세히 살펴보기 위해 3.5 초 정도의 구간만을 출력하였다. 두 그림 모두 On/Off 구간에 따라 제공되는 여분의 대역폭이 세션들에게 공평하게 분배됨을 확인할 수 있다.

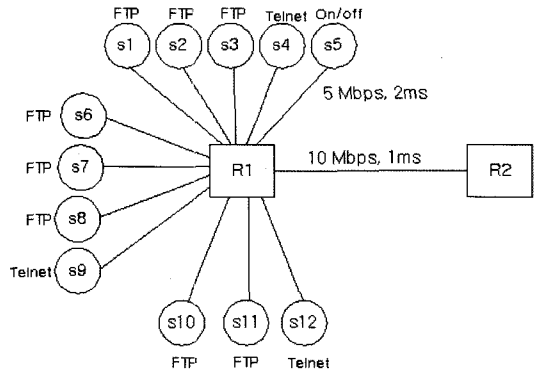

그림 8 실험을 위한 토폴로지

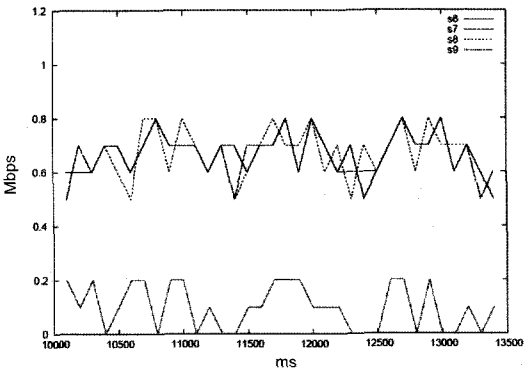

그림 9 DRR의 경우 서비스 등급 2의 성능 측정 결과

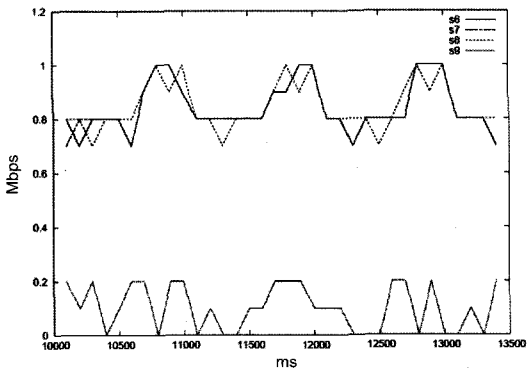

그림 10 H-DRR의 경우 서비스 등급 2의 성능 측정 결과

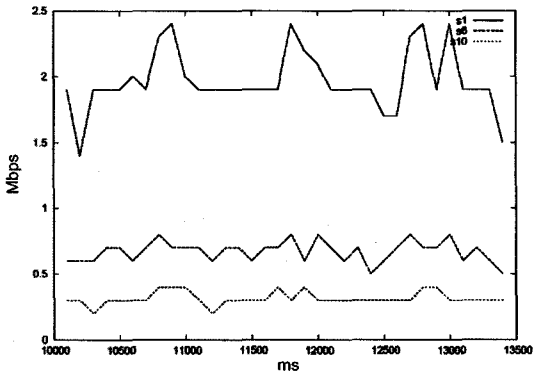

그림 11 DRR의 경우 서비스 등급간 성능측정 결과

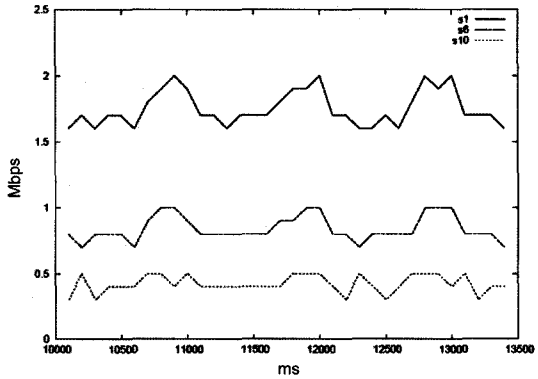

그림 12 H-DRR의 경우 서비스 등급간 성능측정 결과

그러나 H-DRR의 경우가 On/Off 구간에 따라 좀 더 공평하게 대역폭을 분배함을 볼 수 있다. 서비스 등급 1과 서비스 등급 3에 대한 세션들의 성능 결과도 이와 비슷하였고, FTP가 아닌 CBR(constant-bit-rate) 응용을 적용했을 때도 거의 동일한 결과를 얻었으며, 지면 관계상 다른 비슷한 성능 비교 결과 그림은 생략하였다.

서비스 등급간 설정된 서비스 품질 인자 값에 따라 성능(throughput) 관점에서 DRR과 H-DRR이 얼마나 제어성을 갖는 지는 그림 11과 그림 12에 각각 나타내었다. 서비스 등급 1, 2, 3에 대해서 각각 대표 세션으로 s1, s6, s10을 임의로 선택하였다. 이 경우도 DRR과 H-DRR 모두 비슷하게 설정된 서비스 비율대로 서비스 등급 간 성능(throughput) 차별화를 제공함을 확인할 수 있지만 H-DRR의 경우가 조금 더 서비스 비율에 근접해서 성능 차별화를 제공하고 있음을 볼 수 있다.

그러나 서비스 등급 간 패킷 지연의 경우는 DRR과 H-DRR은 큰 차이를 보인다. 그림 13과 그림 14는 각 서비스 등급의 텔넷 세션의 패킷 지연을 DRR의 경우와 H-DRR의 경우에 각각 나타낸 것이다. DRR의 경우 서

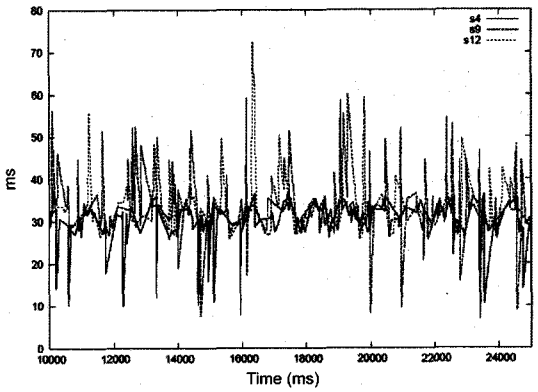

그림 13 DRR의 경우 서비스 등급 간 패킷 지연 측정 결과

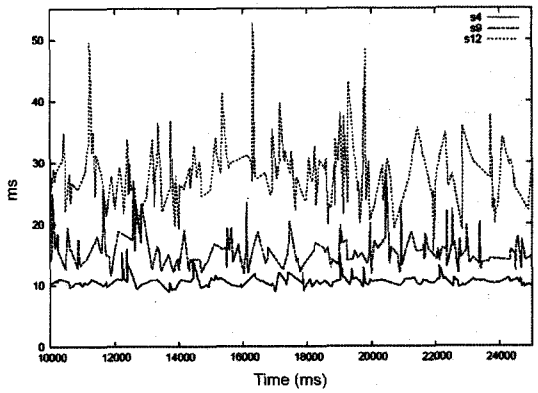

그림 14 H-DRR의 경우 서비스 등급 간 패킷 지연 측정 결과

비스 등급 간 패킷 지연 차별화가 전혀 이루어지지 않는 반면, H-DRR의 경우 대체적으로 서비스 등급에 따라 4:2:1의 비율에 근접해서 차별화된 패킷 지연이 발생하고 있음을 확인할 수 있다.

6. 결 론

이 논문은 사용자 기반 상대적 차별화 서비스를 제공하는 계층적 결손 보완 라운드-로빈 알고리즘(H-DRR)을 제안하였다. 이 알고리즘은 동일 등급에 속한 세션들 간에 대역폭을 공평하게 분배하며, 서로 다른 등급에 속한 세션들 간에는 설정된 서비스 품질 인자 값에 비례해서 차별화된 성능을 제공할 수 있음을 수학적 분석뿐만 아니라 모의실험을 통해서도 보여주었다. 특히 기존 DRR 알고리즘과 비교했을 때 패킷 지연 품질 차별화 측면에서 월등히 뛰어난 제어성을 가지고 있음을 보였다.

향후 과제로는 차별화 서비스 프레임워크에서 본 논문에서 제안하는 방식을 에지 라우터에 적용하고 코어

라우터에는 현존하는 PHB를 채택한 후 상호 보완적으로 동작하는 방법에 관한 연구이다. 즉, 에지 라우터에서 세션 상태를 유지하지만 더 세분화된 품질 차별화 서비스를 제공하고, 또 코어 라우터에 대한 입력 양을 조절하는 역할을 수행하는 것이다. 그러면, 코어 라우터는 세션 상태를 유지하지 않음으로써 확장성을 높이면서도 에지 라우터의 입력 조절 효과를 통해서 서비스 차별화의 품질을 향상 시킬 것으로 기대된다.

참 고 문 헌

[1] M. Shreedhar and George Varghese. Efficient Fair Queuing Using Deficit Round-Robin. IEEE/ACM Tran. Networking, 4(3):375-385, June 1996.

[2] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. In ACM SIGCOMM, pp. 109-120, 1999.

[3] S.Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, December 1998. RFC 2475.

[4] B. Davie, A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB, March 2002. RFC 3246.

[5] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group, June 1999. RFC 2597.

[6] Ion Stoica, Hui Zhang, and T. S. Eugene Ng. A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Services. IEEE/ACM Tran. Networking, 8(2):185-199, 2000.

[7] J.C.R. Bennett and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. IEEE/ACM Tran. Networking, 5(5):675-689, October 1997.

[8] S. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. In INFOCOM, pp. 636-646, 1994.

[9] http://www.isi.edu/nsnam/ns/

부 록

• **보조정리 1의 증명**. k번째 서비스 라운드를 끝냈을 때 노드 n의 결손 보완 카운터 값을 D_n^k로 표기하자. k번째 라운드 동안, 앞 노드 n이 가리키는 세션 큐로부터 전송된 패킷양은$(Q+D_n^{k-1}-D_n^k)$이다. 따라서 만일 첫 번째 라운드부터 K번째 라운드까지 그 세션 큐로부터 전송된 패킷 양을 합하면,

$$W_n(r_0,r_K)=KQ+D_n^0-D_n^K. \tag{9}$$

모든 결손 보완 카운터 값들은 항상 0보다 같거나 크고 최대 패킷 크기 $L^{\max}$보다 작다. 즉,

$$D_n^0 \geq 0 \text{ and } D_n^K < L^{\max}. \tag{10}$$

식 (9)와 식 (10)로부터 이 보조정리는 증명된다. □

• **정리 1의 증명**. 세션 i와 j를 가리키는 앞 노드를 각각 노드 i와 노드 j로 표기하자.

H-DRR 알고리즘의 변하지 않는 기본 사항 중 하나는, 만일 시간 구간 $(\tau,t]$ 동안 양쪽 노드 i와j가 저장 기간 내에 있다면, 이 두 노드들 간에 완결된 서비스 라운드 수의 차이는 결코 1을 넘지 않는 다는 것이다. 따라서 만일 시간 구간 $(\tau,t]$ 동안 노드 i와j에게 주어진 완결된 라운드 수를 각각 K_i와 K_j로 표기하면,

$$|K_i-K_j|\leq 1. \tag{11}$$

식 (11)은 두 가지 경우, $K_i \geq K_j$인 경우와 $K_i \leq K_j$인 경우로 나눌 수 있다.

경우 1: $K_i \geq K_j$. 경우 1은 다시 두 가지 경우, $K_i=K_j$인 경우와 $K_i=K_j+1$인 경우로 나눌 수 있다. 식 (11)과 $K_i \geq K_j$ 때문에 이 두 경우 이외에 다른 경우는 없다. 우선 전자의 경우를 고려하자. 즉,

$$K_i=K_j \tag{12}$$

보조정리 1로부터 ,

$$W_i^c(\tau,t) \leq K_iQ+L^{\max}. \tag{13}$$

이고

$$\begin{aligned} W_j^c(\tau,t) &\geq K_jQ-L^{\max} \\ &= K_iQ-L^{\max} \text{ (식 (12)로부터)}. \end{aligned} \tag{14}$$

식 (13)와 (14)로부터,

$$|W_i^c(\tau,t)-W_j^c(\tau,t)|\leq 2L^{\max}. \tag{15}$$

이제 후자의 경우를 고려하자. 즉,

$$K_j=K_i-1. \tag{16}$$

보조정리 1로부터,

$$\begin{aligned} W_j^c(\tau,t) &\geq K_jQ-L^{\max} \quad \text{(식(16)로부터)} \\ &= (K_i-1)Q-L^{\max} \end{aligned} \tag{17}$$

또 보조정리 1로부터,

$$W_i^c(\tau,t) \leq K_iQ+L^{\max}. \tag{18}$$

식 (17)과 (18)로부터,

$$|W_i^c(\tau,t)-W_j^c(\tau,t)|\leq (2L^{\max}+Q). \tag{19}$$

경우 2: $K_i \leq K_j$. 이 경우 증명은 i와j를 바꾸면 경우 1과 동일하다. 따라서

$$|W_j^c(\tau,t)-W_i^c(\tau,t)|\leq (2L^{\max}+Q). \tag{20}$$

□

• **보조 정리 2의 증명**. 우리는 k단계에서부터 $(l-1)$단계에 위치한 비잎 노드들을 노드 $k,\cdots,$노드 $(l-1)$로 각각 나타낸다. 만일 이 비잎 노드들의 이웃한 서비스 등급 간 상대적 서비스 비율을 각각 $R_k,\cdots, R_{l-1}$로 나타내면,

$$R_n=\frac{Q_n}{Q_{n+1}}, n=k,\cdots,l-1. \tag{21}$$

H-DRR은 이웃한 서비스 등급 간 완료된 서비스 라

운드의 관점에서, 변하지 않는 중요한 기본 사항이 하나 있다. 그것은 m 단계에서 한 번의 라운드가 완료될 때 마다 $m-1$ 단계의 모든 노드들에게 정확히 R_{m-1} 번의 서비스 기회를 제공한다는 점이다. 따라서 l 단계에서 한 번의 라운드가 끝나면, $(l-1)$ 단계에 있는 모든 노드들에게 정확히 R_{l-1} 번의 서비스 기회를 제공하고, 이것은 $(l-2)$ 단계에서는 $(R_{l-2} \cdot R_{l-1})$ 번의 서비스 기회를 의미하고, 결국 k 단계에서는 $(R_k \cdots R_{l-1})$ 번의 서비스 기회를 제공한다. 결국 식 (21)로부터,

$$R_k \cdots R_{l-1} = \frac{Q_k}{Q_l}. \tag{22}$$

식 (22)는 l 단계에서 한 번의 서비스 라운드가 완료되면 k 단계에서 모든 노드들에게 정확히 Q_k/Q_l번의 서비스 기회 제공을 완료함을 의미한다. 만일 k 단계에서 완료된 라운드 수가 Q_k/Q_l보다 적다면 l 단계에서 한 번의 라운드가 완료된 것이 아니라 아직 진행중임을 의미한다. 이 경우 l 단계에 위치한 노드는 한 번의 서비스를 받았을 수도 있고 아직 받지 못했을 수도 있다. 그러므로 k 단계에서 노드 k에게 K_k 번의 서비스가 주어지면 l 단계에 있는 임의의 노드는 적어도 $\lfloor K_k/(Q_k/Q_l) \rfloor$, 즉, $\lfloor (Q_l/Q_k)K_k \rfloor$ 번의 서비스 기회를 받게 되며 $\lceil (Q_l/Q_k)K_k \rceil$ 번의 서비스 기회보다 많이 받을 수는 없다. 다시 말해,

$$\left\lfloor \frac{Q_l}{Q_k} K_k \right\rfloor \le K_l \le \left\lceil \frac{Q_l}{Q_k} K_k \right\rceil. \tag{23}$$

H-DRR의 변하지 않는 다른 기본 사항은 시간 구간 $(\tau, t]$ 동안 저장 기간 내에 있는, 동일한 단계에 존재하는 모든 노드들에 대해서 임의의 두 노드에게 제공된 서비스 횟수의 차이는 1을 넘지 않는다는 것이다. 즉,

$$|K_i - K_j| \le 1. \tag{24}$$

식 (23)과 식 (24)로부터,

$$\left\lfloor \frac{Q_l}{Q_k}(K_i - 1) \right\rfloor \le K_j \le \left\lceil \frac{Q_l}{Q_k}(K_i + 1) \right\rceil. \tag{25}$$

□

- **정리 2의 증명**. 세션 i를 위한 잎 노드와 세션 j를 위한 잎 노드를 각각 노드 i와 노드 j로 나타내자. $\left| W_i^{fq}(\tau,t) - \frac{Q_k}{Q_l} W_j^{l}(\tau,t) \right|$의 위쪽 한계 값은 다음 두 경우의 최대값이 된다.

경우 1: $\left(W_i^{fq}(\tau,t) - \frac{Q_k}{Q_l} W_j^{l}(\tau,t) \right)$의 최대값을 추출하자. 이 값은 $W_i^{fq}(\tau,t)$의 최대값과 $\frac{Q_k}{Q_l} W_j^{l}(\tau,t)$의 최소값의 차이를 구하면 된다. 먼저 $W_i^{fq}(\tau,t)$의 최대값을 계산하자. 노드 i에게 K_i 번의 서비스 라운드가 주어지면, 보조정리 1로부터,

$$W_i^{fq}(\tau, t) \le K_i Q + L^{\max}. \tag{26}$$

이제 $\frac{Q_k}{Q_l} W_j^{l}(\tau,t)$의 최소값을 추출하자. 보조정리 2로부터,

$$K_j \ge \left\lfloor \frac{Q_l}{Q_k}(K_i - 1) \right\rfloor. \tag{27}$$

보조정리 1로부터,

$$\begin{aligned} & W_j^{l}(\tau,t) && \text{(식(27)으로부터)} \\ & \ge K_j Q - L^{\max} \\ & \ge \left\lfloor \frac{Q_l}{Q_k}(K_i - 1) \right\rfloor Q - L^{\max} \\ & \ge \frac{Q_l}{Q_k}(K_i - 1) Q - Q - L^{\max}. \end{aligned} \tag{28}$$

식 (28)는 다음과 같이 다시 쓸 수 있다.

$$\frac{Q_k}{Q_l} W_j^{l}(\tau,t) \ge (K_i - 1) Q - \frac{Q_k}{Q_l}(Q + L^{\max}). \tag{29}$$

따라서, 식 (26)와 (29)로부터,

$$\begin{aligned} & W_i^{fq}(\tau,t) - \frac{Q_k}{Q_l} W_j^{l}(\tau,t) \\ & \le K_i Q + L^{\max} - (K_i - 1) Q + \frac{Q_k}{Q_l}(Q + L^{\max}) \\ & \le (L^{\max} + Q)\left(1 + \frac{Q_k}{Q_l}\right). \end{aligned} \tag{30}$$

경우 2: $\left(\frac{Q_k}{Q_l} W_j^{l}(\tau,t) - W_i^{fq}(\tau,t) \right)$의 최대값을 추출하자. 이 값은 $W_i^{fq}(\tau,t)$의 최소값과 $\frac{Q_k}{Q_l} W_j^{l}(\tau,t)$의 최대값의 차이를 구하면 된다. 방법은 경우 1과 사실상 동일하므로 지면 공간 관계상 생략한다.

□

편 기 현
정보과학회논문지 : 정보통신
제 32 권 제 2 호 참조

이 종 열
정보과학회논문지 : 정보통신
제 32 권 제 2 호 참조

조 성 익
정보과학회논문지 : 정보통신
제 32 권 제 2 호 참조