링크 계층 재전송을 고려한 무선 패킷 스케줄링 알고리즘

정회원 김 남 기*, 윤 현 수**

Wireless Packet Scheduling Algorithms based on Link Level Retransmission

Namgi Kim*, Hyunsoo Yoon** ***Regular Members***

요 약

본 논문에서는 MAC 계층 이하에 존재하는 링크 계층 재전송 기법과 잘 부합할 수 있는 새로운 무선 패킷 스케줄링 알고리즘인 WFQ-R (Wireless Fair Queueing with Retransmission) 알고리즘을 제안한다. WFQ-R 알고리즘에서 링크 계층 재전송에 의해 사용된 자원은 재전송을 수행한 플로우(flow)가 다른 플로우들에게 미리 차용한 자원으로 취급한다. 즉 MAC 계층 이하의 재전송 과정에서 패킷 스케줄러의 허락 없이 사용되어진 자원을 재전송을 수행한 플로우에게 책임 지움으로써 WFQ-R 알고리즘은 무선 공평성(fairness)을 획득할 수 있게 된다. 본 논문에서는 실험을 통하여 WFQ-R 알고리즘이 공평성을 유지함과 동시에 시스템 성능을 최대화 함을 보인다. 또 플로우 구분성(seperation)과 보상성(compensation)도 획득할 수 있음도 보인다.

Key Words : Wireless packet scheduling, Wireless fair queueing, Wireless QoS, Link level retransmission, Wireless networks

ABSTRACT

We propose a new wireless fair queueing algorithm, WFQ-R (Wireless Fair Queueing with Retransmission), which is well matched with the LLR (Link Level Retransmission) algorithm and does not require channel prediction. In the WFQ-R algorithm, the share consumed by retransmission is regarded as a debt of the retransmitted flow to the other flows. Thus, the WFQ-R algorithm achieves wireless fairness with the LLR algorithm by penalizing flows that use wireless resources without permission under the MAC layer. Through simulations, we showed that our WFQ-R algorithm maintains fairness adaptively and maximizes system throughput. Furthermore, our WFQ-R algorithm is able to achieve flow separation and compensation.

Ⅰ. 서 론

최근 컴퓨터 기술의 급속한 발전으로 인해 기존의 텍스트 위주의 사용자 환경에서 벗어나 이미지, 그래픽, 오디오 및 비디오 데이터 등을 제공하는 멀티미디어 사용자 환경으로 변화하고 있다. 네트워크 상에서 다양한 종류의 통신 응용프로그램을 지원하기 위해서는 각각의 패킷 플로우(flow)에 적합한 서비스 품질(Quality of Service)을 제공해 줄 수 있어야 한다. 유선 네트워크에서 공유 링크 상의 패킷 플로우들에게 공평한 서비스 품질을 제공하는 방법으로는 FFQ(Fluid Fair Queueing) 모델[1]이 가장

* 삼성전자 정보통신총괄 통신연구소 (ngkim@camars.kaist.ac.kr), ** 한국과학기술원 전자전산학과 (hyoon@camars.kaist.ac.kr)
논문번호 : KICS2004-08-164, 접수일자 : 1998년 8월 24일

보편적이다. 그래서 이를 기반으로 한 수 많은 변형 알고리즘들이 제안되어져 왔다. 하지만 무선 네트워크에서 무선 채널은 유선 채널과 다르게 사용자의 위치에 따라 그리고 시간에 따라 채널이 변화하는 특성을 지닌다. 따라서 무선 네트워크에서는 이렇듯 버스트(busrt)하고 위치의존(location-dependent)적인 에러를 가지는 무선 채널이 존재하기 때문에 신뢰성 있는 유선 채널을 기반으로 한 FFQ 알고리즘들을 무선 네트워크에 직접 사용할 수 없다.

최근에는 이러한 위치 기반의 에러 특성을 가지는 무선 네트워크 상에서 서비스 품질을 제공하기 위하여 무선 패킷 스케줄링 알고리즘들이 제안되어 오고 있다[2-7]. 이 알고리즘들은 플로우에 해당하는 무선 채널의 에러를 미리 예측하여 동적으로 채널 할당을 재조정함으로써 채널 에러로 인한 성능 저하를 막고 공평한 서비스를 각 플로우에게 보장해 준다. 즉 어떤 플로우에게 패킷을 전송하기 직전에 그 패킷이 에러가 날 것이라고 예측이 되면 그 패킷을 전송하지 않고 다른 플로우에게 채널 사용을 양보함으로써 성능 저하를 막고 양보한 플로우는 추후에 추가적인 자원 할당을 받음으로써 플로우간의 공평성을 유지 하는 방법이다. 그러므로 무선 패킷 스케줄링 알고리즘들은 공유되는 무선 채널을 채널 예측을 통해 다중 사용자들에게 공평하게 나누어 주는 역할을 하게 된다[8].

하지만 이러한 예측 기반 무선 패킷 스케줄링 알고리즘은 너무나 이상적인 상태를 가정하고 실질적인 문제들을 고려하지 않는다는 단점을 지닌다. 즉, 지금까지 제안된 알고리즘들은 모두 패킷 전송 전에 정확한 채널 예측을 바탕으로 하고 있으며 MAC 계층 이하에 보편적으로 존재하는 링크 계층 재전송 기법에 대해서는 전혀 고려하고 있지 않다. 패킷 전송 전에 완벽한 채널 예측은 사실 상 거의 불가능하다. 따라서 이러한 채널 예측 대신 대부분의 무선 네트워크 시스템에서는 채널 에러가 발생했을 때 MAC 계층 이하에서 보상해 주는 링크 계층 재전송 기법을 도입하고 있다. 그러므로 본 논문에서는 채널 예측을 필요로 하지 않으면서 링크 계층 재전송 기법과 잘 부합되는 새로운 무선 패킷 스케줄링 알고리즘을 제안한다.

Ⅱ. 연구 배경 및 동기

2.1 공평성 기준 (fairness criteria)

무선 네트워크에는 데이타 공평성과 자원 공평성이라는 두 가지 공평성 기준(fairness criteria)이 존재한다. 데이타 공평성이란 각 플로우가 받은 데이타를 기준으로 한 공평성이고 자원 공평성이란 각 플로우가 사용한 자원을 기준으로 한 공평성을 말한다. 유선 네트워크에서는 채널 상에 에러가 거의 존재하지 않기 때문에 이 두 공평성에 차이가 없다. 하지만 무선 네트워크에서는 에러가 자주 발생하는 무선 채널의 특성으로 인해 이 두 가지 공평성이 크게 차이 나게 된다. 무선 네트워크에서 데이타 공평성은 각 플로우가 받은 데이타를 모두 같게 하는 데 목적이 있다. 따라서 각 사용자가 궁극적으로 받는 데이터 량을 모두 같게 보장해 준다. 하지만 이 개념은 무선네트워크에 적합하지 않다. 왜냐하면 무선 네트워크에 데이타 공평성을 적용할 경우, 에러가 심하게 생기는 플로우가 모든 자원을 독식하여 소비해 버려 시스템 전체 성능을 크게 저하 시킬 수 있기 때문이다. 따라서 무선 네트워크에서는 데이타 공평성 보다 자원 공평성이 더 효율적인 기준이 된다. 자원 공평성이란 받은 데이터 량이 아니라 각 플로우가 사용한 자원의 양을 모두 같게 하는 것을 말한다. 따라서 자원 공평성은 특정 플로우의 채널 상태에 영향을 받지 않으면서 모든 자원을 각 플로우에게 균등히 분배 할 수 있는 특징을 지닌다. 그러므로 무선 네트워크에서는 자원 공평성이 데이타 공평성보다 적합하다고 할 수 있으며 본 논문에서도 자원 공평성에 더 큰 의미를 둔다.

2.2 기존의 무선 패킷 스케줄링 알고리즘들

앞서 언급했듯이 유선 네트워크에서는 FFQ 모델[1]을 기반으로 많은 패킷 스케줄링 알고리즘들이 제안되어져 왔다. FFQ 모델에서는 각 플로우를 유기(fuild) 플로우로 취급한다. 그러나 이러한 FFQ 모델은 기본적으로 비트 단위의 전송을 바탕으로 제안되었다. 따라서 패킷 단위의 전송이 이루어지는 실제 네트워크에서 FFQ 모델을 적용하기 위한 많은 변형 알고리즘들도 개발되어져 왔다.

하지만 무선 네트워크에서는 시간과 위치에 따른 채널에러가 생기기 때문에 이러한 FFQ 모델을 직접적으로 적용할 수는 없다. 시간과 위치에 따라 패킷전송 가능 여부가 달라지는 무선 채널은 에러가 없는 플로우와 에러가 발생하는 플로우 사이에 다른 양의 자원을 할당하게 만들고 FFQ 모델을 붕괴시킨다. 따라서 이러한 문제를 극복하기 위해 무선 네트워크를 위한패킷 스케줄링 알고리즘들이 제안되어져 오고 있다. 무선 패킷 스케줄링 알고리즘들은

보상 모델(compensation model)을 근간으로 하여 각 플로우의 공평성을 유지한다. 즉 어떤 플로우가 채널에러로 인해 패킷을 전송하지 못하게 되면 이 때 받은 자원을 다른 플로우에게 양보하고 추후에 보상 받음으로써 각 플로우의 공평성을 유지하는 것이다. 이러한 보상모델을 근간으로 한 무선패킷 스케줄링 알고리즘이 최근까지 많이 제안되어 지고 있다[2-7]. 하지만 이들 알고리즘들도 문제점을 가지고 있다. 즉 전통적인 예측 기반의 무선패킷 스케줄링 알고리즘들[2-5]은 정확한 채널 예측을 기반으로 자원할당 여부를 결정하게 되는데, 실제 무선 네트워크 시스템에서 완벽한 채널 예측은 매우 어렵다. 또 기존 알고리즘들은 실제 무선네트워크 시스템에서 주로 사용되는 MAC 계층 이하의 에러 처리 알고리즘에 대해 거의 고려하고 있지 않다. 아주 최근에는 불완전한 채널상태 예측이나 MAC 계층 이하의 알고리즘도 고려하는 무선 패킷 스케줄링 알고리즘[6,7]이 제안되고 있으나, 이 역시 여전히 무선 채널 예측을 필요로 하거나 실제무선 네트워크 시스템에서 가장 보편적으로 사용되고 있는 패킷 재전송을 통한 에러 보상 기법을 고려하지 않는 약점을 지닌다. 따라서 본 논문에서는 채널 예측을 필요로 하지 않으며 링크 계층 재전송 기법과 잘 부합하는 새로운 무선 패킷 스케줄링 알고리즘을 제안한다.

2.3 패킷 재전송에 의한 기존 알고리즘의 문제점

그림 1은 링크 계층 재전송 기법이 존재하는 시스템에 변화 없이 FFQ 모델을 적용했을 경우 발생하는 공평성 붕괴를 보여 주고 있다. 이 그림에서 플로우 1, 2, 3은 각각 1, 1, 2의 비중(weight)을 가진다. 에러가 없는 상황에서는 각 플로우가 비중에 맞게 2, 2, 4의 패킷을 두 라운드(round)에 걸쳐 할당 받게 된다. 하지만 플로우 1의 첫 패킷이 채널 에러로 인해 재전송 되었고 이로 인해 3 패킷을 전송할 수 있는 자원이 재전송에 의해 사용되었다고 하자. 그러면 결국 각 플로우는 두 라운드가 끝났을 때 2, 1, 2개의 패킷만을 전송 받게 된다. 더구나 자원 할당 측면에서 보면 각 플로우는 5, 1, 2개의 자원을 사용한 꼴이 된다. 이는 명백히 공평성이 깨어진 것이며, 기존의 예측 기반 패킷스케줄링 알고리즘이 링크 계층의 패킷 재전송 기법을 고려하지 않는다면 실제 패킷 재전송 기법들 도입한 보편적인 무선 네트워크 시스템에서 공평성을 유지 할 수 없음을 증명해 주고 있다. 따라서 본 논문에서는 이러한 상황에서도 공평성을 유지할 수 있는 새로운 무선 패킷 스케줄링 알고리즘을 제안한다.

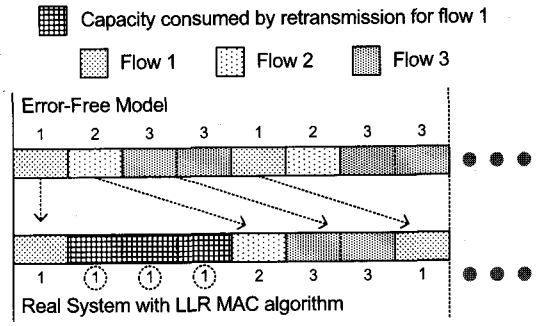


그림 1. 링크 계층 재전송에 의한 FFQ 모델의 공평성 붕괴의 예

Ⅲ. 패킷 재전송을 고려한 무선 패킷 스케줄링 알고리즘

본 절에서는 새로이 제안하는 WFQ-R (Wireless Fair Queueing with Retransmission) 알고리즘을 소개한다. WFQ-R 알고리즘의 기본적인 개념은 패킷 재전송에 의해 사용되어진 자원은 재전송을 수행한 플로우가 다른 플로우들에게 미리 차용한 자원으로 보는 것이다. 즉 링크 계층에서 재전송이 일어나면 재전송이 발생한 플로우는 재전송에서 사용한 자원 중 필요한 만큼을 후에 다른 플로우들에게 양보하게 된다. 따라서 재전송된 플로우는 서비스를 앞서서 받은 리딩(leading) 플로우가 되고 나머지 플로우들은 서비스를 다 받지 못한 래깅(lagging) 플로우가 되는 것이다. 그리고 패킷 스케줄러는 래깅 플로우가 리딩 플로우에 비해 채널을 점유하는데 높은 우선순위를 준다.

WFQ-R 알고리즘에는 FIC (Flow-In- Charge)와 SIC(Server-In-Charge)라는 두 가지의 보상 타입(compensation type)이 있다. FIC 타입은 재전송에 의해 사용되어진 자원 모두를 재전송 플로우에게 책임 지우는 것이다. 즉 재전송이 일어난 플로우가 자신의 채널에 대해 전적으로 책임지는 타입이다. 예를 들어, 그림 2에서처럼 어떤 플로우가 패킷 재전송에 의해 2개의 무선 자원을 더 사용했다고 하자. 그러면 FIC 타입은 이 때 사용된 2개의 무선 자원에 대해 재전송을 수행한 플로우가 모두 책임을 지게 된다. 따라서 다음 두 라운드 동안 재전송 플로우는 전송을 포기하고 자원을 다른 플로우들에게 양보한다.

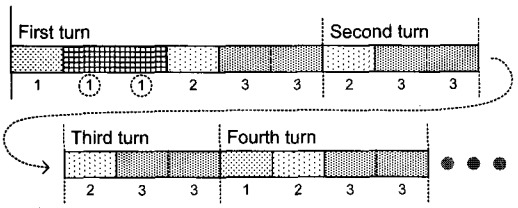


그림 2. FIC 타입 예제

FIC 타입은 무선 자원 할당량 측면에서 보았을 때 매우 공정한 알고리즘이다. 하지만 FIC 알고리즘은 너무 엄격하게 자원을 할당하기 때문에 자주 채널 에러를 겪는 플로우의 서비스가 급격히 떨어지는 면이 있다. 그래서 SIC 타입도 제안한다. SIC 타입은 채널 에러를 플로우가 아닌 채널 서버의 책임이라고 생각하고 패킷 재전송에 사용된 자원에 대한책임을 모든 플로우들이 공동으로 분배하여 지는 방식이다. 따라서 패킷재전송이 발생하면 각 플로우들은 자신의 비중에 비례하여, 재전송에 의해 사용된 자원에 대한 책임을 지며, 재전송에 의해 패킷을 전송받은 플로우도 자신의 비중만큼 만의 책임만을 지게 되므로, 서비스의 품질이 급격히 떨어지지 않게 된다.

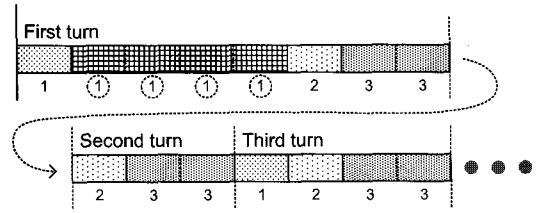


그림 3. SIC 타입 예제

예를 들어, 그림 3에서처럼 SIC 타입을 사용하는 WFQ-R 알고리즘에서 3 개의 플로우가 각각 1, 1, 2의 비중을 가지고 있다고 하자. 그리고 플로우 1의 첫 번째 패킷 전송에서 에러가 발생하여 4 개의 무선 자원이 추가로 재전송에 사용되었다고 하자. 그러면 이 때 전체 플로우의 비중은 4이고 면 플로우 1의 비중은 1이므로, 재전송을 수행한 플로우 1은 전체 재전송 비용 4 개 중에 하나의 무선 자원에 대해서만 책임을 지면된다. 따라서 플로우 1은 두 번째 라운드에서만 자신의 자원을 양보하면 되는 것이다.

3.1 상세 알고리즘 서술

플로우에서 서비스의 손실과 이익을 계산하기 위해에러가 없는 상태의 참조시스템 S^r과 에러가 존재하는 실제 시스템 S를 가정한다. 그리고 플로우는 실제 시스템 S에서 참조 시스템 S^r과 비교하여 리딩(leading), 래깅(lagging), 인싱크(in-sync) 플로우로 구분한다. 리딩 플로우라 함은 현재 자신이 받아야 할 서비스 보다 실제로 더 많은 서비스를 받았음을 나타내고, 래깅 플로우는 실제 받은 서비스가 자신이 받아야 할 서비스에 못 미침을 나타낸다. 그리고 인싱크 플로우는 자신이 받아야 할 서비스와 실제 받는 서비스 양이 일치하는 플로우를 나타낸다. WFQ-R 알고리즘에서는 참조 시스템 S^r로 SFQ (Starttime Fair Queueing) 알고리즘[9]을 사용하고 이를 S^r_{SFQ}로 표현한다. 이는 [4]에서 언급했듯이 채널 에러가 있는 시스템에서는 끝나는 시간 기준이 아니라 패킷의 시작시간을 기준으로 하는 것이 더 간편하기 때문이다.

공평성을 알고리즘에 반영하기 위해 참조 시스템 S^r_{SFQ}과 실제 시스템 S에서 받은 서비스의 차이를 나타내는 변수 lag를 도입한다. 즉 lag_i는 양수이면 플로우 i 가 래깅임을, lag_i 가 음수이면 리딩, 0 이면 인싱크임을 나타낸다. 만약 에러가 전혀 없다면 모든 플로우의 lag_i는 0 이 된다.

래깅 플로우는 자신의 서비스를 보상 받기위해 다른 플로우들에 비해 채널을 할당 받는데 높은 우선순위를 지닌다. 보상 서비스(compensation service)는 래깅 플로우에게 제공되는데 이 서비스는 리딩 플로우가 자원 할당을 양보할 때 생기는 자원으로 가능하게 된다. 그리고 짧은 기간 공평성을 제공하기 위해 우리는 이 보상 서비스를 래깅의 정도에 따라 조금씩 모든 래깅 플로우에게 돌아가면서 분배한다. 이를 위해 보상 가상 시간(compensation virtual time) c_i를 두어, 플로우 i가 래깅인 동안 받은 보상서비스의 양을 정량화 하여 기록한다.

마지막으로 리딩 플로우의 서비스 저하를 천천히 또 자연스럽게 유도하기 위해서 (graceful degradation) 시스템 변수 α ($0 \le \alpha \le 1$)와 s_i를 도입한다. α는 리딩 플로우가 한번에 포기하는 자원의 양을 나타낸다. 즉 리딩 플로우는 래깅 플로우를 위해 한번에 자신에게 할당된 자원 중 최대 $(1 - \alpha)$만큼의 자원을 포기한다. 이를 위해 리딩 플로우 i 는 리딩 플로우일 때 실제로 받는 서비스 양을 정량화 하여 변수 s_i 에 기록해 놓는다.

상세한 WFQ-R 알고리즘은 그림 4와 5에 나타나 있고 여기서 사용되는 변수들에 대한 정의는 표 1에 나타나 있다. WFQ-R 알고리즘에서 채널 서버가 패킷을 받으면 이 패킷은 **on receiving** 함수에 있

```
1  on session i receiving packet p:
2    if (i ∉ A)
3      v_i ← max (v_i, min_{k∈A}{v_k});
4      lag_i ← 0;
5      A ← A ∪ {i}; /* mark flow active */
6    enqueue(queue_i, p);
7
8  on sending current packet: /* get next packet to send */
9    i ← min_{v_i}{i ∈ A};
10   if (lag_i ≥ 0 or (lag_i < 0 and s_i ≤ α v_i))
11     /* flow i non-leading or leading with graceful degradation */
12     send_pkt(i, i); /* flow i served through v_i selection */
13   else /* flow i is leading and not allowed to send */
14     /* select lagging flow j to compensate */
15     j ← min_{c_k}{k ∈ A | lag_k > 0};
16     if (j exists)
17       send_pkt(j, i); /* serve flow j but charge to i */
18       if (i ≠ j and empty(queue_j) and lag_j ≥ 0)
19         leave(j); /* j becomes inactive */
20     else /* there is no lagging flow */
21       send_pkt(i, i); /* serve given back to flow i */
22   if (empty(queue_i) and lag_i ≥ 0)
23     leave(i); /* i becomes inactive */
24
25 send_pkt(j, i) /* serve flow j but charge to i */
26   p ← dequeue(queue_j);
27   v_i ← v_i + p.length / r_i;
28   if (i = j and lag_i < 0 and s_i ≤ α v_i)
29     /* flow i is leading and served through v_i selection */
30     s_i ← s_i + p.length / r_i;
31   if (i ≠ j)
32     lag_j ← lag_j − p.length; /* flow j has gain extra service */
33     if (lag_j > 0)          /* j continues to be lagging */
34       c_j ← c_j + p.length / r_j;
35     if (lag_j + p.length ≥ 0 and lag_j < 0)
36       /* j just becomes leading */
37       s_j ← α v_j;
38     lag_i ← lag_i + p.length; /* flow i has lost service */
39     if (lag_i − p.length ≤ 0 and lag_i > 0)
40       /* i just becomes lagging */
41       c_i = max (c_i, min_{k∈A}{c_k | lag_k > 0});
42   send_and_charge(p, j, i);
43     /* send pkt with retransmission and charge overhead */
```

그림 4. WFQ-R 알고리즘 PART1

```
44 send_and_charge(p, j, i); /* send pkt p with retransmission */
45                           /* and charge overhead */
46   used_ret ← send(p);    /* send pkt p through MAC layer and */
47     /* return used wireless resources due to retransmission */
48   if (used_ret ≤ 0 or A − {j} = ∅) /* if no retransmission or */
49     return;                        /* no other flows, then return */
50   charged ← charge(used_ret, j); /* charging overhead depending */
51                                  /* on compensation type */
52   if (i = j and lag_j < 0) /* flow i is leading and served through v_i */
53     s_j ← s_j + charged / r_j;
54   else /* in-sync or lagging in i = j, or i ≠ j */
55     lag_j ← lag_j − charged; /* flow j has gain extra service */
56     if (lag_j > 0)           /* j continues to be lagging */
57       c_j ← c_j + charged / r_j;
58     if (lag_j + p.length ≥ 0 and lag_j < 0) /* j just becomes leading */
59       s_j ← α v_j;
60     for (l ∈ A − {j}) /* other flows ditributively get share */
61       lag_l_before ← lag_l;
62       lag_l ← lag_l + charged × η / Σ_{k∈A−{j}} r_k;
63       if (lag_l_before ≤ 0 and lag_l > 0)
64         /* i just becomes lagging */
65         c_l = max (c_l, min_{k∈A}{c_k | lag_k > 0});
66
67 charge(used_ret, j) /* calculate amount of charging overhead */
68   switch (COMPENSATION_TYPE)
69     case FLOW_IN_CHARGE :
70       /* entire overhead is charged to the retransmitted flow */
71       return used_ret × (1 − r_j / Σ_{k∈A} r_k);
72     case SERVER_IN_CHARGE :
73       /* overhead is distributively charged to flows in the server */
74       return used_ret × (1 − r_j / Σ_{k∈A} r_k) × r_j / Σ_{k∈A} r_k;
75
76 leave(i) /* flow i leaves */
77   A ← A \ {i};
78   for (j ∈ A) /* update lags of all active flow */
79     if (lag_j ≤ 0 and lag_j + lag_i × r_j / Σ_{k∈A} r_k > 0)
80       /* j just becomes lagging */
81       c_i ← max (c_i, min_{k∈A}{c_k | lag_k > 0});
82     lag_j ← lag_j + lag_i × r_j / Σ_{k∈A} r_k;
83     if (∃j ∈ A s.t. empty(queue_i) ∧ lag_i ≥ 0)
84       leave(j);
```

그림 5. WFQ-R 알고리즘 PART2

표 1. 변수 정의

Parameters	Definitions
v_i	Virtual time of flow i
l_{agi}	The difference between the service that flow i should receive in a reference error-free packet system and the service it has received in the real system (lagging if positive, leading if negative, and in-sync otherwise)
A	The set of active flows
α	Minimal fraction of service retained by any leading flow
s_i	Normalized amount of service actually received by a leading flow i since it became leading
c_i	Normalized amount of compensation service received by a lagging flow i
rs	The rate of flow i

는 버퍼에 저장된다. 그리고 채널서버는 버퍼에서 적절한 패킷을 선택하여 이를 on **sending** 함수로 보낸다. 그리고 어떤 플로우가 더 이상 보내고자 하는 패킷이 없으면 그 플로우는 **leave** 함수에 의해 스케줄에서 제외된다. **on sending** 함수에서 패킷이 선정되어지면 이 패킷은 **send_pkt()** 함수로 전달된다. **send_pkt()** 함수는 실제로 패킷을 버퍼에서 꺼내어 그 패킷에 해당하는 플로우의 변수를 적절히 조정한다. 그리고 **send_and_charge()** 함수를 호출하고, **send_and_charge()** 함수는 **send()** 함수를 통해 패킷을 실제무선 채널을 통해 전송한다. 패킷을 전송한 후 **send()** 함수는 패킷 재전송에 의해 사용되어진 책임량(charging overhead)을 리턴한다. 이 책임량은 보상 타입에 따라 재전송 플로우가 책임져야 할 자원을 양을 **charge()** 함수를 통해 계산해 낸 값이다. 그리고 마지막으로 이 책임량으로 생기는 이득은 모든 플로우들에게 분배된다.

Ⅳ. 실험 및 결과

본 절에서는 WFQ-R 알고리즘이 어떻게 공평성을 보장하는지에 대한 실험을 수행하고 그 결과를 분석한다. 실험에서 사용된 평가 기준은 다음과 같다.

- 무선 할당 자원(allocated resources): 각 플로우에 할당되어진 총 무선 자원의 양을 나타낸다. 이 무선 할당자원은 자원 공평성을 직접적으로 나타

내는 지표가 된다.

- 대기 지연 시간(queueing delay): 큐에서 경험되는 지연시간을 나타낸다.
- 받은 데이타(received data): 서버로부터 무선 단말 수신자가 실제로 받은 데이타 양을 나타낸다. 받은 데이타는 데이타 공평성을 나타낸다. 하지만 받은 데이타는 무선할당 자원에 비해 그 중요성이 떨어진다. 그 이유는 앞서 설명했듯이 무선 네트워크에서는 데이타 공평성을 엄격히 유지하지 못하기 때문이다.

4.1 실험 환경

본 논문에서는 실험을 위하여 NS 시뮬레이터[10]를 사용하였다. 실험에 사용된 노드 구조는 그림 6과 같다. 그림에서 알 수 있듯이 실험에는 에러 없는 플로우 2 개, 에러 있는 플로우 1개, 이렇게 총 3 개의 플로우가 사용되었다. 각 플로우의 전송시작 시간은 0초, 0.4초, 1.3초였으며 이 세 플로우의 비중은 1 로 모두 같다. 실험에서 패킷의 길이는 1KByte였으며 패킷 생성 주기는 8 ms였다. 각 플로우는 5KByte의 큐를 서로 독립적으로 가지고 있다. 무선 채널의 대역폭은 1.5Mbps이고 지연은 10ms였다. 실험에서 패킷 재전송은 한 패킷에 대해 최대8번까지 일어날 수 있으며 한 번의 재전송 때마다 한 패킷을 전송할 수 있는 자원이 추가적으로 사용되었다. 기존의 알고리즘과 제안하는 알고리즘을 비교하기 위해 링크 계층 재전송 기법이 존재하는 시스템에서 CIF-Q 알고리즘[4]을 구현하였다.

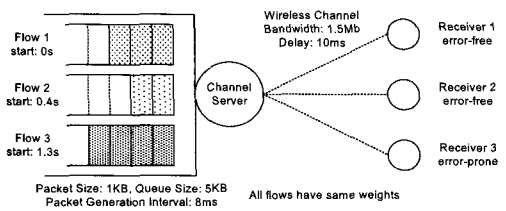


그림 6. 실험 환경

4.2 실험 결과

첫 번째 실험으로 에러 있는 플로우의 재전송 확률을 0.2로 고정해 놓고 실험을 수행하였다. 물론 에러 없는 플로우들의 재전송 확률은 0이다.

표 2는 이 때 할당 받은 자원 량, 큐 대기 시간, 받은 데이터 량을 평균을 내어 구한 실험 결과 값이다. 표 2에서 우리는 CIF-Q 알고리즘이 패킷 재전송 기법 상에서 (CIF-Q_LLR) 올바르게 공평성을 보장해 주지 못함을 알 수 있다. CIF-Q_LLR 알고리즘은 너무나 많은 자원 에러있는 플로우에게 할당하고 있다. 그리고 이 때문에 에러 없는 플로우들의 대기 시간이 에러 있는 플로우와 차별화 되지못하고 함께 증가하는 모습을 볼 수 있다. CIF-Q_LLR 알고리즘은 받은 데이터 량 측면에서 보면 공평한 듯 보인다. 하지만 이는 에러 없는 플로우들의 자원을 빼앗았기 때문에 유지된 것으로, 에러 있는 플로우의 채널상태가 급격히 악화되면 에러있는 플로우가 대부분의 무선자원을 모두 차지하게 되어 궁극적으로 유지 될 수 없는 공평성이다.

표 2. 에러 있는 플로우 3의 재전송 확률이 0.2 일 때의 실험 결과

		플로우 1	플로우 2	플로우 3
무선 할당 자원	CIF-Q_LLR	459.5	459.6	581.0
	WFQ-R_FIC	500.5	499.5	500.0
	WFQ-R_SIC	476.5	476.0	547.5
대기 시간	CIF-Q_LLR	82.0	82.1	83.1
	WFQ-R_FIC	75.7	74.6	97.8
	WFQ-R_SIC	78.9	78.5	86.2
받은 데이타	CIF-Q_LLR	459.5	459.6	460.0
	WFQ-R_FIC	500.5	499.5	397.0
	WFQ-R_SIC	476.5	476.0	441.0

실험 결과에서 CIF-Q_LLR과는 달리 WFQ-R 알고리즘은 무선자원을 각 플로우에게 공평하게 분배함을 알 수 있다. WFQ-R_FIC 알고리즘은 각 플로우에게 정확히 같은 무선 자원을 할당하고 있으며, WFQ-R_SIC 알고리즘은 에러 복구를 위해에러 있는 플로우에게 조금더 많은 자원을 할당해 줌을 알 수 있다. 그리고 WFQ-R_FIC는 에러 있는 플로우와 에러 없는 플로우 사이의 큐 대기 시간을 철저히 분리함에 비해, WFQ-R_SIC는 부드럽게 분리함을 알 수 있다.

두 번째로 에러 있는 플로우의 재전송 확률을 0에서 1.0 으로 변화시켜 가면서 실험을 수행하였다. 이 실험결과는 그림 7, 8, 9에 나타나 있다. 그림 7은 시스템 총 성능으로 모든 플로우의 성능을 합친 결과이다. 그림 8은 에러 없는 플로우와 에러 있는 플로우의 성능차이를 나타낸 것이고 그림 9는 에러 없는 플로우와 에러 있는 플로우 사이의 무선 자원 할당량 차이를 나타낸 그래프이다. 그림 8에서 성능 차이는 에러 없는 플로우의 성능에서 에러 없는 플로우의 성능을 뺀 것이고, 그림 9에서 무선 자원 할

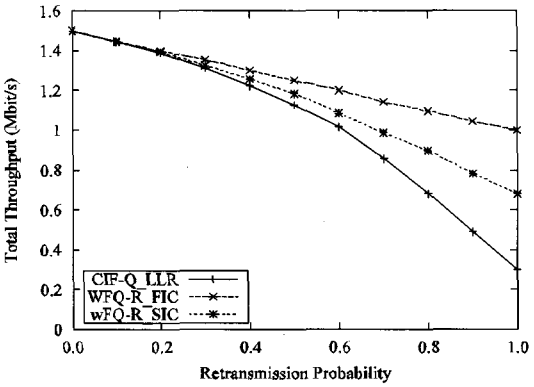


그림 7. 시스템 총 성능

당량 차이는 에러있는 플로우의 무선 자원할당량에서 에러 없는 플로우의 무선 자원 할당량을 뺀 것이다.

그림 7에서 알 수 있듯이 CIF-Q_LLR 알고리즘은 세 알고리즘 중 가장 낮은 성능을 보인다. 그리고 CIF-Q_LLR은 채널 상태가 나빠 재전송 확률이 증가하면 시스템 전체성능이 크게 감소한다. 이 것은 CIF-Q_LLR이 링크 계층의 패킷 재전송 기법을 고려하지 않아, 에러 있는 플로우에게 지나치게 많은 무선 자원을 편중하여 할당하기 때문에 나타난 결과이다. 전체 시스템 성능 면에서는 WFQ-R_FIC 알고리즘이 가장 좋은 성능을 보인다(그림 7). WFQ-R_FIC는 재전송 확률이 높아지더라도 전체 시스템 성능의 저하가 크게 일어나지 않음을 알 수 있다. 이는 WFQ-R_FIC가 재전송에 사용된 자원을 패킷스케줄링에 포함시켜 에러 있는 플로우와 에러 없는 플로우 사이의 무선 자원 할당량의 차이를 없앴기 때문이다(그림 9). 하지만 WFQ-R_FIC는 엄격히 무선 자원을 할당하기 때문에 에러있는 플로우의 재전송 확률이 높은 경우, 에러 있는 플로우와 없는 플로우 사이의 성능 차가 크게 나는 특성이 있다(그림 8). 이를 보완하기 위해WFQ-R_SIC 알고리즘은 에러 있는 플로우에게 적절할 만큼 무선 자원을 더 할당한다(그림 9). 그래서 두 플로우 사이의 성능 차이를 줄이면서도 (그림 8), 전체 시스템 성능을 어느 정도 유지하는 특성을 지니게 된다(그림 7).

지금까지의 실험 결과를 살펴보면 WFQ-R 알고리즘은 링크 계층 재전송 기법을 수용하면서 공평성을 적절히 잘 유지함을 알 수 있다. 더욱이 WFQ-R은 특정 무선 채널이 심각하게 악화되더라도 언제나 전체 성능을 일정수준으로 유지하는 특성을 가지는 모습도 볼 수 있다. WFQ-R 알고리즘에서 FIC 타입과 SIC 타입의 차이는 플로우 분리성(flow separation)과 플로우 보상성(flow compensation)에 있다. 플로우 분리성이란 에러 없는 플로우가 다른 에러 있는 플로우에 의해 성능 면에서 영향을 받지 않는 것을 의미한다. 하지만 지나치게 엄격한 플로우 분리는, 잦은 에러를 겪는 무선 채널을 지닌 플로우의 성능을 지나치게 악화시킨다. 따라서 우리는 이러한 플로우 분리성과 보상성을 적절히 수용하기 위해 FIC 타입과 SIC 타입을 제안하였다. 즉, FIC 타입은 자원 공평성을 엄격히 지켜 플로우 분리성을 보장해 주는 타입이고, SIC 타입은 데이타 공평성도 고려해 에러 있는 플로우에 적절한 무선자원을 더 할당해 주어플로우 보상성 측면도 고려한 타입이다. 따라서 WFQ-R 알고리즘은 대기 시간 및 성능 면에서 플로우간에 분리와 보상을 적응적으로 획득할 수 있다.

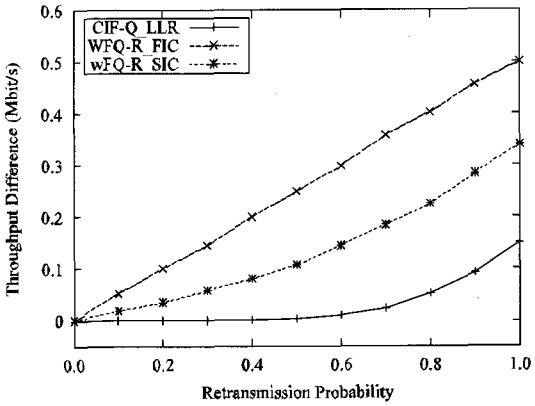


그림 8. 에러 있는 플로우와 없는 플로우 간의 성능 차이

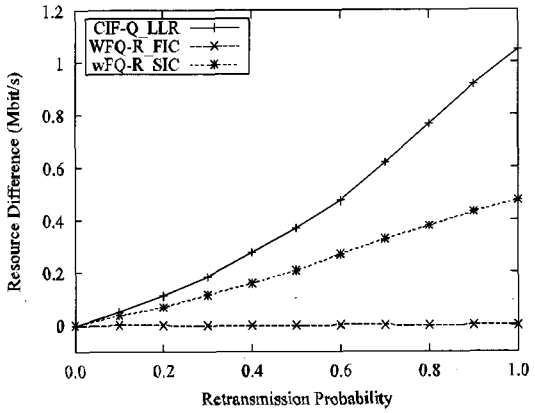


그림 9. 에러 있는 플로우와 없는 플로우 간의 무선 자원 할당 량 차이

V. 결 론

본 논문에서는 기존의 예측 기반공평 대기 알고리즘을 링크계층 재전송 기법 위에서 운영될 때 그 성능을 평가했고, 또 새로운 무선 패킷 스케줄링 알고리즘을 제안하였다. 기존의 예측 기반 알고리즘들은 MAC 계층 이상에서 링크 계층 재전송 기법을 고려하지 않고 수행되어 지기 때문에 재전송 기법을 적용한 시스템에서 공평성을 유지 하지 못함을 실험을 통해 알 수 있었다. 더구나 기존 알고리즘은 심한 에러가 있는 플로우가 존재할 경우, 이 플로우에게 대부분의 무선 자원을 할당하기 때문에 시스템 전체 성능을 크게 저하시킨다. 따라서 우리는 이러한 상황에서 공평성을 유지할 수 있는 새로운 WFQ-R 패킷 스케줄링 알고리즘을 제안하였다. WFQ-R 알고리즘은 링크계층 이하에서 패킷 재전송이 이루어진 플로우에 대해 자원 할당을 차후에 줄임으로서 플로우 간의 전체공평성을 유지한다. 그러므로 WFQ-R 알고리즘은 링크계층 재전송 기법을 패킷스케줄링에 적절히 반영하면서 채널예측을 필요로 하지 않게 된다. 분석을 통해 우리는 제안하는 알고리즘이 무선 공평 대기 알고리즘이 지녀야 할 성질들을 충분히 만족함을 증명하였다. 또 실험을 통해 WFQ-R 알고리즘이 공평성을 적응적으로 잘 보장하면서 동시에 시스템 전체 성능을 최대화함을 알 수 있었다. 그리고 제안하는 알고리즘이 플로우 구분성과 보상성도 모두 만족시킴을 알 수 있었다.

앞으로는 제안된 WFQ-R 알고리즘을 실제로 구현되어 있는cdma2000[11], UMTS[12] 시스템이나, 앞으로 구현되어질 미래 시스템[13-15]을 대상으로 설계하여 구체적인 성능평가를 진행해 볼 예정이다. 그리고 본 논문에서 제안하는 알고리즘과 무선 TCP의 성능 상관관계에 대해서도 연구해 볼 계획이다.

참 고 문 헌

[1] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *ACM SIGCOM'89*, 1989.

[2] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, Aug, 1999.

[3] P. Ramanathan and P. Agrawal, "Adapting Packet Fair Queueing Algorithms to Wireless Networks," *ACM MOBICOM'98*, Oct. 1998.

[4] T. S. E. Ng, I. Stoica and H. Zhang, "Packet Fair Queueing Algorithms for Wireless Networks with Location-Dependent Errors," *IEEE INFOCOM'98*, 1998.

[5] S. Lu, T. Nandagopal and V. Bharghavan, "Design and Analysis of an Algorithm for Fair Service in Error-Prone Wireless Channels," *ACM Wireless Networks Journal*, vol. 6, no. 4, pp. 232-343, 2000.

[6] Y. Liu, S. Gruhl, and E. W. Knightly, "WCFQ: An Opportunistic Wireless Scheduler with Statistical Fairness Bounds," *IEEE Transactions on Wireless Communications*, vol. 2, no. 5, Sep. 2003.

[7] M. Liu, L. B. Milstein, and R. Rao, "Wireless Random Scheduling Protocol with Realistic Channel Conditions," *IEEE MILCOM'03*, Boston, Sep. 2003.

[8] V. Bharghavan, S. Lu and T. Nandagopal, "Fair Queuing in Wireless Networks: Issues and Approaches," *IEEE Personal Communications*, Feb. 1999.

[9] P. Goyal, H. M. Vin, and H. Cheng, "Start-Time Fair Queueing: A scheduling Algorithm for Integrated Services Packet Switching Networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, Oct. 1997.

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

[11] TIA/EIA/cdma2000, "Mobile Station - Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular Systems," *TIA/EIAI Standard*, 1999.

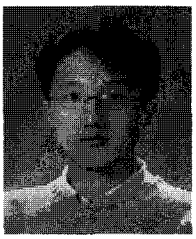
[12] 3GPP, "RLR Protocol Specification," 3GPP TS 25.322 version 3.0, Oct. 1999.

[13] 3GPP2, "Physical Layer Standard for cdma2000 Spread Spectrum Systems Release C," *3GPP2 C.S0002-C V1.0*, May 2002.

[14] IEEE 802.16 Task Group e (Mobile Wireless MAN), http://www.ieee802.org/16/

tge/
〔15〕 IEEE 802.20 Mobile Broadband Wireless Access, http://grouper.ieee.org/groups/802/20/

김 남 기(Namgi Kim) 정회원



1997년 2월 서강대학교 컴퓨터학과 학사
2000년 2월 한국과학기술원 전자전산학과 석사
2005년 2월 한국과학기술원 전자전산학과 박사
2005년 3월~현재 삼성전자 책임연구원

<관심분야> 이동 통신 시스템, 패킷 네트워크

윤 현 수(Hyunsoo Yoon) 정회원

1979년 2월 서울대학교 전자공학과 학사
1981년 2월 한국과학기술원 전산학과 석사
1988년 2월 Ohio 주립대학 전산학과 박사
1989년~ 한국과학기술원 전산학과 교수

<관심분야> 상호연결 네트워크, 병렬 컴퓨터 구조, 컴퓨터 보안