2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

# 고속 네트워크에서 실시간 전송을 지원하는 공정 큐잉 알고리즘

윤여훈, 김태윤
고려대학교 컴퓨터학과
e-mail : joy1223@netlab.korea.ac.kr

# Fair Queuing Algorithm Supporting Real Time Transmission in High Speed Network

Yeo-Hoon Youn, Tai-Yun Kim
Dept. of Computer Science & Engineering, Korea University

요 약

네트워크에서 다양한 애플리케이션의 서비스 성능을 저하시키는 불공정 큐잉 문제를 해결하기 위해 현재 공정 큐잉 분야가 활발히 연구중이다. 그 중에서 DRR(Deficit Round Robin)은 작업 복잡도가 낮고 구현이 간단한 기법으로 이전 라운드에서의 서비스 결손량을 다음 라운드에서 서비스하도록 하여 공정한 서비스를 보장하는 기법이다. 그러나 엔터프라이즈 환경과 같은 고속 네트워크 환경에서 최대 수 kbyte 이상의 패킷 사이즈를 가지는 서비스들에 대해 불필요한 SQ(Service Quantum) 재설정 횟수 및 라운드 순회 횟수로 인한 지연시간 증가를 일으킨다.

본 논문에서는 매 라운드마다 전송을 앞둔 패킷의 사이즈를 고려하여 SQ를 동적으로 설정하는 기법을 제안한다. 제안한 기법은 각 큐의 가장 앞쪽에 있는 패킷들 중 사이즈가 아무리 큰 패킷도 현재 라운드에서 서비스될 수 있고, 패킷을 처리하는데 있어서의 작업 복잡도 또한 최소화하는 기법으로 다양한 애플리케이션들에 대한 지연시간을 최소화한다.

## 1. 서 론

오늘날 네트워크의 대역폭이 커지고 동시에 실시간 처리를 요하는 다양한 멀티미디어 애플리케이션들이 생성되고 있다. 특히 최대 수 gigabits의 대역폭을 갖는 고속 전용선을 사용하는 엔터프라이즈 네트워크 등과 같은 환경에서는 광대역 통신과 실시간 처리 및 실시간 전송 메커니즘이 필요하다. 이러한 메커니즘 중의 하나인 패킷 스케줄링 알고리즘은 특정 애플리케이션 트래픽의 대역폭 점령과 그 외의 트래픽의 병목 및 기아(starvation)로 인한 데이터 상실을 막기 위한 주요 메커니즘이다.

대부분의 네트워크에서 사용되는 최선 노력(best-effort) 기반의 FCFS기법은 스케줄러로 입력되는 패킷의 사이즈와 입력되는 속도가 상대적으로 큰 트래픽이 큐를 점령하는 경우가 발생하여 다른 트래픽들에 대한 병목 및 기아(starvation)가 발생한다[2], [7]. 따라서 고속 네트워크 환경에서 최소 수십 bytes에서 최대 수 kbyte 이상의 패킷 사이즈를 갖는 애플리케이션들을 각각 독립적으로 관리하면서 공정한 서비스를 보장하는 패킷 스케줄링 기법이 필요하다. 최근에 공정한 서비스 제공을 위한 많은 패킷 스케줄링 기법들이 개발되고 있고, 그 중에서 DRR(Deficit Round Robin)은 다른 것들에 비해 구현이 쉽고 낮은 작업 복잡도를 가지는 기법으로 정확한 공정성을 보이는 패킷 스케줄링 알고리즘이다[2], [7], [9]. 스케줄러가 첫 번째 큐부터 마지막 큐까지 한번씩 방문하는 것을 라운드라고 정의할 때, DRR은 현재 라운드에서 각 큐에 대한 상대적인 서비스 결손량과 모든 큐에 고정적으로 설정된 기본적인 할당량의 합을 나타내는 SQ(Service Quantum) 만큼을 다음 라운드에서 서비스하여 정확한 공정성을 보장하는 기법이다. 그러나 전용 고속망을 이용하는 엔터프라이즈 네트워크와 같은 환경에서 애플리케이션간의 패킷 사이즈 차가 최대 수 kbyte 이상인 상황에서는 적합하지 않다. 만약 패킷 사이즈가

수 kbyte 이상인 멀티미디어 애플리케이션의 패킷들이 패킷 스케줄러로 입력될 경우 SQ가 패킷 사이즈 이상의 값을 가질 때까지 SQ를 재설정하면서 패킷을 큐에 담아 둔다. 불필요한 SQ 재설정 횟수 및 라운드 횟수로 인해 시간이 지연되기 때문에 기존의 DRR과 같은 알고리즘으로 최적의 성능을 보장할 수 없다. 따라서 패킷 사이즈에 따라 SQ를 동적으로 재설정하는 기법이 요구된다.

본 논문에서는 매 라운드마다 전송을 앞둔 패킷의 사이즈를 고려한 SQ에 대한 동적 설정 기법 및 패킷을 보다 낮은 작업 복잡도(work complexity)를 가지고 처리하는 기법을 제안하여 지연시간을 최소화하는 패킷 스케줄링 기법을 제안한다.

## 2. 관련 연구

### 2.1 고속 네트워크 환경에서의 애플리케이션 서비스

엔터프라이즈 인터넷/인트라넷 환경과 같은 고속 네트워크 환경에서 컴퓨터 및 통신 네트워크를 활용하는 애플리케이션을 속성에 따라 다음과 같이 네 가지 클래스로 분류한다[11], [12].

○ 클래스 1: 임계 작업
- 금융 시스템, 항공 예약 시스템, 전자 상거래의 전자 결제 시스템 등.
- 연성 실시간(soft real time) 또는 경성 실시간(hard real time)처리를 요구한다.
- 패킷 사이즈는 100byte 정도로 크지 않다.

○ 클래스 2: 대화형 작업
- 인터넷 전화, 웹 비디오 전화, 주문형 뉴스, 인터넷 비디오 광고 등.
- 주기적으로 데이터를 보내는 경성 실시간(hard real time) 처리를 요구하며, 재전송 메커니즘이 의미가 없다.
- 음성의 경우 패킷 사이즈가 100byte 정도이며, 비디오의 경우는 수 kbyte 이다.

○ 클래스 3: 고용량 고품질 작업
- 높은 정밀도의 오디오/비디오 스트림 처리를 요구하는 것으로 디지털 방송, 주문 형 비디오 등.
- 경성 실시간 처리를 요구하며, 재전송 메커니즘이 의미가 없다.
- 패킷 사이즈는 매우 크다.

○ 클래스 4: 일반 작업
- 보통의 파일, 텍스트, 그래픽 이미지와 MP3 데이터 등.
- 실시간 처리를 요구하지 않으며, 재전송 메커니즘이 요구된다.
- 패킷 사이즈는 50 ~ 1500bytes 정도이다.

### 2.2 기존의 패킷 스케줄링 알고리즘 및 DRR 알고리즘 분석

ORR[7], [9]은 스케줄러가 각 큐를 방문할 때마다 패킷 하나씩을 전송하는 기법으로 패킷 사이즈가 동일한 네트워크에서는 아주 정확한 공정성을 보장하지만, 패킷 사이즈가 다른 상황에서는 패킷 하나를 전송할 때마다 패킷의 크기가 상대적으로 큰 것에 대한 작은 것의 손실은 계속 누적되는 불공정 큐잉을 일으켜 공정성 저하를 일으킨다. WRR[3]은 각 큐에 대하여 전송할 패킷의 수를 나타내는 가중치(weight)를 패킷의 평균 사이즈를 기반으로 설정하기 때문에 정확한 공정성을 보장하지는 못한다. WFQ[1], [5], [7] 계열의 알고리즘은 들어오는 패킷에 대해 마감시간을 나타내는 인덱스를 부여하여 패킷이 들어올 때마다 패킷들을 인덱스에 따라 큐 안에서 정렬시켜 마감시간이 가장 임박한 패킷부터 먼저 서비스하는 기법이다. 이것은 공정성에서 우수한 성능을 보이지만 패킷의 인덱스를 계산하고 그 것에 따라 큐 안에서 정렬하는데 높은 작업 복잡도가 요구되어 근원지와 목적지 사이에 노드 수가 많을 때 패킷을 전송하는데 많은 시간이 지연된다.

DRR은 매 라운드마다 각 큐의 결손량을 다음 라운드에서 보상하여 정확한 공정성을 보장하는 알고리즘으로, DRR에서 DC(Deficit Counter), Q(Quantum), SQ(Service Quantum)은 각 큐간의 공정한 서비스를 제공하기 위해 사용되는 것들이다. DC는 큐의 상대적인 결손량을 나타내고, Q는 모든 큐에 고정적으로 부여된 동일한 할당량을 나타내며, SQ는 이전 라운드에서의 DC에 Q를 더한 값이다.

DRR은 매 라운드마다 각 큐에 SQ만큼을 서비스하여 정확한 공정성을 보장한다. 또한 현재 패킷을 포함하는 큐들의 인덱스 리스트인 활성리스트를 운영하여 스케줄러가 큐들을 순회하는데 있어서 지연을 최소화한다[2], [7], [9].

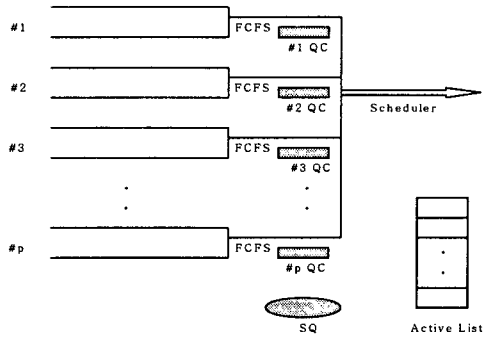
그러나 DRR은 기본적인 서비스 할당량 Q를 고정적으로 설정해 놓는 기법이기 때문에 엔터프라이즈 환경에서 데이터 량이 많고 패킷 사이즈가 상당히 큰 클래스 2, 3에 해당하는 애플리케이션의 서비스를 효율적으로 지원하지 못한다. 패킷 사이즈가 매우 큰 인터넷 방송, 주문형 비디오 등 클래스 2, 3의 멀티미디어 서비스 트래픽들이 큐로 삽입될 경우 스케줄러는 SQ가 패킷 사이즈보다 클 때까지 SQ를 계속 재설정하면서 최대 수십 번의 라운드를 순회한다. 불필요한 SQ 재설정 횟수와 라운드 순회 횟수로 인한 시간이 지연되어 경성 실시간 처리를 요하는 클래스 2, 3에 해당하는 애플리케이션의 서비스 품질을 저하시킨다. 따라서 애플리케이션간의 패킷 사이즈의 차가 매우 큰 엔터프라이즈 환경에서 최적의 성능을 위해 패킷 사이즈에 따라 서비스 할당량 SQ를 동적인 설정하는 기법이 요구된다.

## 3. 제안한 공정 큐잉 알고리즘

### 3.1 제안된 큐잉 알고리즘의 개요

제안한 알고리즘은 엔터프라이즈 환경의 다양한 애플리케이션의 패킷 사이즈와 그 특성을 고려하여 기존의 DRR을 수정한 것으로 매 라운드를 시작하

기 전 전송을 앞둔 패킷들의 사이즈를 고려하여 SQ를 동적으로 설정하는 기법을 제안한다. 이것은 각 큐의 가장 앞쪽에서 전송을 앞둔 패킷들 중 사이즈가 가장 큰 패킷도 현재 라운드에서 서비스될 수 있도록 하여 불필요한 SQ 재설정 횟수 및 라운드 순회 횟수를 줄여 지연시간을 최소화하는 기법이다. 스케줄러의 전체적인 개요는 <그림 1>와 같다.




<그림 1> 패킷 스케줄러 구조

기존의 DRR에서의 DC 대신 지금까지 전송한 총 데이터 량의 계측자인 QC(Quantum Counter)를 사용한다. 그리고 기존의 DRR과 같이 스케줄러는 현재 패킷을 하나이상 포함하는 큐의 인덱스 노드들의 링크드 리스트인 활성리스트의 순서에 따라 큐들을 방문한다.

제안된 패킷 스케줄러는 enqueue와 dequeue 모듈로 구성되며, enqueue와 dequeue 모듈은 서로 비동기적으로 동작한다.

최초에 enqueue 모듈이 동작하면서 모든 큐들의 QC를 0으로 초기화시킨다. 패킷들이 입력됨에 따라 2.1절에서와 같이 엔터프라이즈 환경에서의 4가지 클래스로 패킷을 구분하여 클래스의 번호와 실시간 처리 요구도에 따라 해당 클래스 큐에 삽입시킨다.

패킷이 계속 입력됨에 따라 그것이 현재 활성 리스트에 속하는 큐로 분류되는지 아닌지를 판단한다. 활성리스트에 속하지 않는다면 큐에 삽입시킨 다음 활성리스트의 끝부분에 큐를 등록시키고, 활성리스트에 속해있다면 활성리스트에 등록할 필요 없이 큐에만 삽입시킨다.

dequeue 모듈은 각 큐에 삽입된 패킷들을 SQ에 따라 실제 출력 링크로 전송하는 부분이다.

SQ(m)(m = 1, 2, 3, ...)를 m번째 라운드에서의 서비스 할당량, $QC(m)_i$ (i = 1, 2, 3, ..., P)를 i번째 큐가 m번째 라운드를 끝낸 후 지금까지 전송한 총 데이터 량, $A(0)_i$을 첫 라운드가 시작될 때 각 큐의 가장 앞쪽에 있는 패킷의 사이즈라고 할 때, 첫 라운드에서의 SQ(1)는 (1)과 같다. 즉, 각 큐의 $A(0)_i$중에 가장 큰 값이SQ(1)가 된다.

$$SQ(1)=Max(\ A(0)_i\ ) \qquad (1)$$

각 큐마다 enqueue 모듈에서 0으로 설정된 $QC(1)_i$에서 시작하여 최대 SQ(1)만큼 패킷을 서비스한 후 $QC(1)_i$에 전송한 데이터 량을 더한다. 활성리스트의 순서에 따라 큐들의 서비스가 끝나고 두 번째 라운드가 시작되기 전에 SQ(2)값이 다시 설정된다. $A(1)_i$를 첫 번째 라운드에서 i번째 큐를 서비스한 다음 i번째 큐의 가장 앞쪽에 위치한 패킷의 사이즈라고 할 때, 두 번째 라운드에서의 SQ(2)는 (2)과 같다. 즉, 모든 큐에 대하여 $QC(1)_i$ 와 $A(1)_i$ 를 합한 값들 중 가장 큰 값으로 두 번째 라운드의 SQ(2)를 재설정 한다.

$$SQ(2) = Max(\ QC(1)_i + A(1)_i) \qquad (2)$$

각 큐마다 이전 라운드의 $QC(1)_i$에서 시작하여 최대 SQ(2)만큼 패킷을 서비스한 후 $QC(2)_i$에는 이전 라운드의 $QC(1)_i$ 값과 현 라운드에서 전송한 데이터 량의 더한 값을 설정한다. 활성리스트의 순서에 따라 큐들의 서비스가 끝나고 세 번째 라운드부터 두 번째 라운드에서와 동일한 방법으로 SQ(m)와 $QC(m)_i$가 재설정 되어 서비스된다.

$bytes_k(i)$ (k = 1, 2, 3, ...)를 스케줄러가 k번째 라운드에서 i번째 큐에 대하여 실제 전송한 데이터 값이라고 할 때, $bytes_k(i)$에 대해서 (3)의 관계가 성립하고, m번째 라운드에서 전송한 실제 데이터 값 $bytes_m(i)$에 대하여 스케줄링이 가능하기 위해서 (4)을 만족해야 한다.

$$QC(m)_i = \sum_{k=1}^{m} bytes(k)_i \qquad (3)$$

$$SQ(m) - QC(m-1)_i \geq bytes(m)_i \qquad (4)$$

스케줄러가 $bytes(m)_i$에 대해 (4)을 만족하면서 큐들을 순회하기 때문에 큐를 방문할 때마다 최소한 하나 이상의 패킷 전송을 보장한다. 그러므로 큐들의 가장 앞쪽에 위치한 패킷들 중 패킷 사이즈가 아무리 큰 것이라도 현재 라운드에서 전송됨으로 불필요한 SQ 재설정 횟수 및 라운드 순회 횟수를 줄인다.

매 라운드마다 스케줄러가 각 큐를 방문하는 동안 또는 방문한 후 큐에 패킷이 더 이상 존재하지 않는다면 해당 큐의 인덱스 노드는 활성 리스트에서 삭제되고 $QC(m)_i$는 0으로 설정된다. 활성 리스트에서 삭제되었던 큐에 패킷이 다시 들어왔을 때 또는 새로운 트래픽의 패킷이 들어왔을 때 SQ(m)는 앞에서 설명한 방식으로 설정되고 $QC(m)_i$은SQ(m)와 현재 0으로 설정되어 있는 $QC(m)_i$의 합한 값으로 설정한다.

### 3.2 작업 복잡도 비교

[표 2]은 제안한 알고리즘의 작업 복잡도를 분석하기 위해 DRR과 제안한 알고리즘을 각 단계적으로 연산처리 과정을 비교한 것이다.

[표 2] DRR과 제안한 알고리즘의 단계적 비교

| | DRR | 제안된 스케줄링 알고리즘 |
|---|---|---|
| 1 단계 | 패킷을 분류 | 패킷을 분류 |
| 2 단계 | 매 라운드가 시작되기전에 각 큐마다 $SQ(m)_i$를 계산 ( $SQ(m)_i$ = Q + $DC(m-1)_i$) | 각 큐를 서비스한 다음 각 큐에 대해 $QC(m)_i$ 와 각 큐의 가장 앞쪽에 있는 패킷 사이즈( $A(m)_i$)의 합을 계산 |
| 3 단계 | 각 큐를 서비스한 다음 각 큐에 대해 $DC(m)_i$를 계산 ( $DC(m)_i$ = $SQ(m)_i$ - $bytes(m)_i$) | 2 단계에서 구한 값들 중 가장 큰 값을 추출하여 SQ(m)로 설정 |

1 단계의 패킷 분류에서 DRR과 제안한 알고리즘은 단순히 패킷 헤더의 정보를 보고 패킷을 분류하므로 작업 복잡도는 둘다 작다. 2 단계에서는 큐의 개수가 n 개일 경우 DRR과 제안한 알고리즘은 각각 SQ와 QC+A를 동일하게 n번씩 계산해야 한다. 그러나 3 단계에서는 DRR은 큐의 개수가 n 개일 경우 DC를 n번 계산해야 하지만 제안한 알고리즘에서는 단지 2 단계에서 계산된 각 큐의 QC+A 값들 중 가장 큰 값을 추출하여 SQ로 설정하는 한 번의 작업만 필요하다. 패킷 사이즈를 누적시킨 값인 $bytes(m)_i$와 $QC(m)_i$를 계산하는데 있어서의 작업 복잡도가 동일하다는 것을 감안할 때 두 알고리즘의 작업 복잡도는 (5)의 관계를 만족한다.

DRR의 작업 복잡도 ≥ Propose의 작업 복잡도 (5)

DRR의 작업 복잡도가 O(1) 이므로[2], [9] 제안한 알고리즘의 작업 복잡도 또한 최대 O(1) 이다.

## 5. 결론 및 향후 과제

본 논문에서는 공정한 서비스 제공을 위한 많은 패킷 스케줄링 기법들 중에서 다른 것들에 비해 구현이 쉽고 낮은 작업 복잡도를 가지면서 정확한 공정성을 보이는 패킷 스케줄링 알고리즘인 DRR을 수정한 기법을 제안한다. 본 기법은 엔터프라이즈 네트워크 환경과 같은 고속 네트워크 환경에서 패킷 사이즈가 수 kbytes에 이르고 경성 실시간 처리를 요하는 멀티미디어 패킷들에 대해 서비스 할당량 SQ에 대한 재설정 횟수 및 라운드 순회 횟수를 줄여 지연시간을 시간을 줄일 수 있었다. 향후 연구 과제로는 서비스 지연을 더욱 최소화하기 위해 활성 리스트에 대한 더욱 정교한 운영 방법을 연구하는 것이다.

## 참고문헌


[1] Hemant M. Chaskar, U. Madhow, "Fair scheduling with tunable latency: a round robin approach", GlOBECOM(Global Telecommunications Conference), vol. 2, pp.1328-1333, 1999.

[2] Salil S.kanhere, Alpa B. Parekh and Harish Sethu, "Fair and Efficient Packet Scheduling in Wormhole Networks", Proc. of IPDPS(International Parallel and Distributed Processing Symposium), pp.623-631, 2000.

[3] O. Altintas, Y. Atsumi, T Yoshida, "Urgency -based round robin: a new scheduling discipline for packet switching networks", ICC(IEEE International Conference Communication), vol, 2, pp.1179-1184, 1998.

[4] Hakyong Kim, Yongtak Lee and Kiseon Kim, "Fairness concept in terms of utilisation", IEEE Electionics Letters, vol. 36, No. 4, 17th Feb 2000.

[5] Onur Altintas, Yukio Atsumi and Teruaki Yoshida, "On a Packet Scheduling Mechanism for Supporting Delay Sensitive Applications on High Speed Networks", ICICS'97, Singapore, 9-12 September 1997.

[6] M.H. MacGregor, W. Shi, "Deficits for bursty latency-critical flows: DRR++", Proc. of ICON(IEEE International Conference), pp.287-293, 2000.

[7] Onur Altintas, Yukio Atsumi, Teruaki Yoshida, "A note on fair queueing and best-effort service in the Internet", IWS(Internet Workshop), pp.145-150, 1999.

[8] G. Hasegawa, T. Murata, H. Miyahara, "Comparisons of packet scheduling algorithms for fair service among connections on the Internet", Proc. of INFOCOM(Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies), vol. 3, pp.1253-1262, 2000.

[9] M. Shreedhar, G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin", IEEE/ACM Transaction, vol. 4. No. 3. pp.375-385, June 1996.

[10] Hui Zhang, "Service disciplines for guaranteed performance service in packet-switching networks ", Proc. of IEEE, vol. 83, Issue 10. pp.1374-1396, June 1996.

[11] 유항재, 김태윤, "엔터프라이즈 환경에서의 멀티미디어 QoS 프레임워크", 고려대학교 석사학위논문, 1999.

[12] 김병철, 김태윤, "엔터프라이즈 환경에서 서비스 요구 사항을 지원하는 패킷 스케줄링 알고리즘", 정보과학회논문지: 정보통신, 제 27권, 제 3호, 2000, 9.

[13] Erik. Nordmark, "Contribution of packet sizes to packet and byte volumes", Internet Draft, Jun 1997.