2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

특정 트래픽의 실시간 보장을 위한 대역폭 공유 큐잉 알고리즘의 설계

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

The design of bandwidth sharing queueing algorithm to guarantee realtime of specific traffic

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

요 약

정체(congestion)가 발생한 네트워크에서 자원관리 시스템은 정체를 제어하고 프레임 상실(loss)을 최소화함으로써 리얼타임 트래픽에 대한 요구를 충족시키기 위해 절대적으로 요구된다. 그에 따라 본 논문에서는 외부출력 인터페이스(Outgoing Interface)를 통과한 각종 멀티미디어 프레임들의 정체나 상실을 최소화하는 것을 기본 목적으로 하여 임계작업(mission critical)의 애플리케이션이 요청한 트래픽이 삽입되는 큐에 프레임이 존재하는 한, 부 스케줄러(Child Scheduler)를 생성하여 연속적으로 처리를 해주는 한편, 다른 트래픽들은 주 스케줄러(Parent Scheduler)가 할당된 대역폭만큼 Round-Robin 방식으로 계속 처리를 해주도록 하여 기아(starvation)가 일어나지 않도록 하는 CQ(Custom Queuing) 기반의 대역폭 공유 큐잉 알고리즘을 설계하였다.

1. 서 론

네트워크를 통해 각종 멀티미디어 프레임들의 정체나 상실을 최소화하면서 목적지 호스트로의 효율적인 전송을 위해 서버, 라우터 등에서는 그림 1과 같은 단계를 거치는 메커니즘들이 적용되고 있다. 그림 1에서 식별자(Classifier)는 각각의 프레임이 어느 큐에 삽입되어야 하는지를 결정하는 역할을 하고, 큐 관리자(Queue manager)는 스케줄러(Scheduler)에 의해서 사용되는 여러 개의 큐를 관리하는데 필요한 서비스를 제공하며, 스케줄러는 여러 개의 큐에 존재하는 프레임들 중 다음에 전송해야할 프레임을 선택하는 역할을 한다. 이 세 부분 중에서 가장 핵심이 되는 부분은 스케줄러라고 할 수 있으며, 바로 이 부분이 프레임들의 정체나 상실을 최소화하여 네트워크를 통해 원활하게 전송될 수 있도록 실질적인 QoS(Quality of Service)를 적용하게 된다.

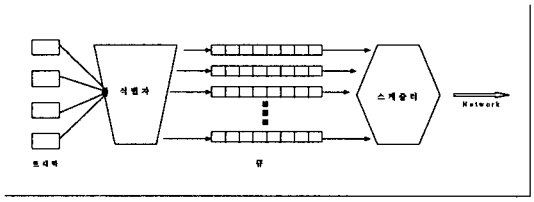

그림 1. 자원관리 메커니즘

이 때, 큐 관리자는 스케줄러 정책에 따라 정책을 결정하므로 큐 관리자와 스케줄러를 통합하여 큐잉

(queueing)으로 정의할 수 있다. 현재 많은 큐잉 알고리즘들이 존재하고 있고 실제로 리눅스 뿐만 아니라 여러 OS에서 이 알고리즘들이 적용되고 있으나 상황에 따라 데이터 상실이나 기아(starvation)과 같은 현상들이 발생할 수 있다.

따라서 본 논문에서는 트래픽들의 데이터 상실이나 기아를 최소화하는 것을 기본 목적으로 하여, 특정 트래픽에 대해서는 계속적인 처리를 해주면서도 다른 트래픽들에는 기아가 일어나지 않도록 하는 CQ 기반의 큐잉 알고리즘을 설계하였다.

본 논문의 전체 구성을 살펴보면 2장에서는 큐잉이 적용되는 시점과 각종 큐잉 알고리즘 및 CQ에 대한 구체적인 내용을 설명하였고, 3장에서는 CQ 알고리즘 기반의 특정 트래픽의 실시간 보장을 위한 대역폭 공유 큐잉 알고리즘을 제안하였다. 마지막 4장에서는 결론 및 향후 과제를 설명하였다.

2. 관련연구

2.1 큐잉의 적용 시점

리눅스를 비롯한 각종 OS에서 큐잉 정책이 적용되는 시점은 프레임이 외부 출력 인터페이스(outgoing interface)로 전송되기 바로 전이다[1, 2, 3]. 예를들어, 리눅스에서는 프레임을 외부로 보내기 전에 datalink layer에 속해있는 dev_queue_xmit()라는 함수가 호출되면서 큐 관리자와 스케줄러가 작업을 시작한다.

즉, dev_queue_xmit()함수가 호출되면 식별자와 큐 관리자와 관련된 enqueue 함수가 불려지고 그런 다음 스케줄러의 정책에 따라 외부로 프레임을 보내기 위해 dequeue 함수가 호출된다. 그러므로 큐잉은 enqueue함수가 불려지는 순간에 적용이 되는 것이라 할 수 있다.

2.2 일반적인 큐잉 알고리즘

■ FIFO (First In, First Out Queueing)[1, 2]

FIFO는 가장 단순하면서도 보편적인 큐잉 알고리즘으로, 출력 인터페이스에 먼저 도착한 프레임을 먼저 네트워크로 우선적으로 전송한다. FIFO의 문제점은 특정 트래픽이 대단위 파일 전송에 있어서 모든 대역폭을 다 써버리는 경향이 있기 때문에 다른 종류의 트래픽의 대역폭까지도 다 써버릴 우려가 있다. 그렇기 때문에 FIFO는 정체가 거의 없고 대역폭이 큰 링크에서 효율적이다.

■ PQ (Priority Queueing)[1, 2]

PQ는 가장 기본적인 트래픽 우선순위화 알고리즘으로, 트래픽의 특성에 따라 high, medium, normal, 그리고 low 큐에 프레임이 삽입된 다음 가장 높은 우선순위를 가진 큐부터 먼저 처리되도록 하는 방식이다. PQ는 임계작업(mission critical)의 애플리케이션이 요청한 트래픽에 우선순위를 높게 하여 실시간 전송을 절대적으로 보장해 줄 수 있지만, 낮은 우선순위를 할당받은 큐는 높은 우선순위를 할당받은 큐가 계속 트래픽을 제공받는다면 가아가 발생할 우려가 있다. 따라서 PQ는 가장 높은 우선 순위를 부여받은 트패픽의 대역폭인 낮은 상황에서 효율적인 알고리즘이다.

■ WFQ (Weighted Fair Queueing)[1, 2, 3, 4]

WFQ는 트래픽들을 양방향 트래픽과 단방향 트래픽으로 나누어 양방향 트래픽에 대해서는 처리 시간을 줄이기 위해 큐의 앞부분에 위치시키고 단방향 트래픽에 대해서는 큐의 뒤쪽에 위치시키는데, 상대적으로 우선순위가 높으나 적은 대역폭을 사용하는 양방향 트래픽이 큐에서 빠른 속도로 처리되기 때문에 남은 대역폭을 사용하는 낮은 우선순위의 단방향 트래픽의 기아를 방지할 수 있다. 그러나 WFQ는 다음에서 설명할 CQ에 비해 대역폭 할당에 있어서 복잡한 알고리즘을 사용할 뿐만 아니라 정확한 제어를 하지 못한다. 이러한 WFQ은 양방향의 Frame Relay 상의 Voice 데이터를 전송하는데 효율적이다.

■ CQ (Custom Queueing)[1, 2, 3, 4]

CQ는 그림 2와 같이 외부로 나가는 여러 종류의 트래픽들에 대하여 링크에 대한 최소한의 대역폭을 할당해 준 다음, 가장 우선순위가 높은 트래픽부터 가장 우선순위가 낮은 트래픽까지 할당된 대역폭 만큼을 Round-Robin 방식으로 처리해 주기 때문에 데이터 상실이나 기아를 최소화시킬 수 있는 효율적인 자원 관리 알고리즘이다. 이러한 CQ는 다양한 멀티미디어 데이터 전송에 있어서 효율적이다.

이런 일반적인 큐잉 알고리즘들 중에서 CQ 알고리즘을 개선하여 정체나 데이터 상실을 미연에 방지할 수 있을 뿐만 아니라, 특정 트래픽에 대해서는 계속적인 처리를 해주면서 다른 트래픽들에는 기아가 일어나지 않도록 하는 큐잉 알고리즘을 설계하고자 한다.

2.3 CQ 알고리즘 분석

먼저 기존 CQ에 대해 좀더 상세히 설명하면, 그림 2와 같이 각각의 트래픽이 식별자에 의해 분류되어 적합한 큐에 삽입되고, 각 큐는 Round-Robin 순서로 각 큐에 할당된 대역폭 비율만큼의 프레임을 외부로 보내도록 처리된다. 이 때 CQ에서 생성될 수 있는 큐의 수는 10에서 16개 사이가 가장 적당하며, 전체 트래픽이 링크에 대하여 이용할 수 있는 대역폭 총 이용률을 95퍼센트를 초가 해서는 안되도록 하는데, 이것은 만약 어떤 트래픽이 할당된 대역폭의 이용률을 초가해서 링크를 이용할 경우가 발생할 수 있기 때문에 남은 5퍼센트만큼은 이 트래픽이 사용할 수 있도록 하기 위해서이다. 또한, 특정 트래픽에 의해 사용되지 않은 대역폭을 다른 트래픽에 의해 사용될 수 있도록 할 수 있다.

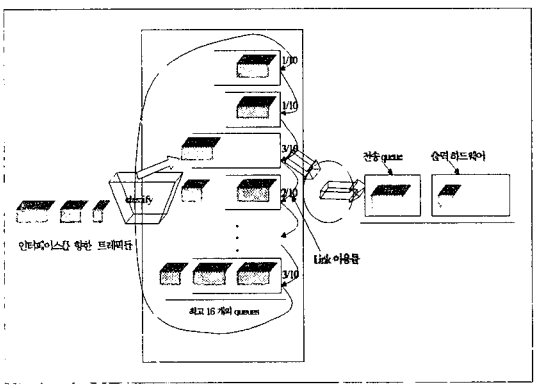

그림 2. CQ(Custom Queueing) 메커니즘

CQ 알고리즘에서 가장 중요한 것은 바로 각 큐에 할당되는 링크 이용률에 따라 출력되는 프레임의 수를 효과적으로 계산하는 방법이다. 다음 2.3.1의 내용에서 각 트래픽의 링크 이용률에 따라 보내지는 프레임 수 및 총 bytes 값을 구하기 위한 효율적인 계산법을 설명하고자 한다.

▣ 링크 이용률에 따른 출력 프레임 수 산출법[2]

i. 먼저, 트래픽들에 따른 각 프레임 크기의 상대적인 비율은 가장 큰 프레임의 크기를 각 프레임의 크기로 나누어서 구한 다음, 각 트래픽이 link의 대역폭에 대하여 가지기를 원하는 각각의 대역폭 퍼센티지를 곱한다.

트래픽 A가 보냈던 프레임 크기가 1086 bytes, 트래픽 B가 보냈던 프레임 크기가 291bytes, 트래픽 C가 보냈던 프레임 크기가 831이었다고 가정하면, 그 비율은 다음과 같다.

$$\frac{1086}{1086}\times 0.2 : \frac{1086}{291}\times 0.6 : \frac{1086}{831}\times 0.2$$

즉 0.2 : 2.239 : 0.261

iii. 위의 결과 값들 중에 가장 작은 값으로 각 값을 나눔으로써 비율의 형태를 좀 더 표준화시킨다.

$$\frac{0.2}{0.2}\times 100 : \frac{2.239}{0.2}\times 100 : \frac{0.261}{0.2}\times 100$$

즉 1 : 11.2 : 1.3

이 때 소수점 이하가 0.5 보다 작아도 반올림해주어 결과적으로 1 : 12 : 2의 비율이 나온다.

이 비율은 각 트래픽이 사용하는 대역폭의 퍼센티지가 대략 20, 60 그리고 20이 되도록 외부로 보내야하는 프레임 수에 대한 비율이다.

3. 특정 트래픽의 실시간 보장을 위한 대역폭 공유 큐잉 알고리즘의 설계

3.1 개선된 CQ 기반 큐잉 메커니즘

CQ에서 생성할 수 있는 큐들은 앞에서도 언급했듯이 10개에서 16개가 적합하다. 만약 어떤 큐가 할당된 비율만큼의 프레임을 서비스한 다음에도 그 큐에 처리되어야 할 프레임들이 남아 있거나, 또는 그 큐에 있는 모든 프레임들을 서비스하자 마자 새로운 프레임이 도착했다면, 남아있는 프레임이나 새로 삽입된 프레임이 처리되기 위해서는 10~16사이의 모든 큐들을 다 거친 다음에야 프레임들이 다시 처리되기 때문에 그만큼의 지연은 감수해야 된다.

그러나 임계작업(mission critical)의 애플리케이션이 요청한 트래픽이 삽입되는 큐에 한해서는 그림 2와 같이 스케줄러에 대한 복사된 부 스케줄러를 생성하여 처리하게 함으로써 지연을 최소화 할 수 있다.

그리고 각 큐의 외부로 전송될 할당된 비율만큼의 프레임 수를 산출하기 위해 그림 3과 같은 상황에서 다음과 같이 2.3절에서 설명한 산출법을 이용하여 계산 할 수 있다.

그림 3에서처럼 임계작업의 애플리케이션이 요청한 트래픽이 삽입된 큐 A를 비롯한 다양한 트래픽이 각 큐에 삽입되어 질 때, A에 삽입된 트래픽의 프레임 크기와 링크 대역폭에 대한 이용률이 각각 960bytes와 40%이고, B가 370bytes와 20%, C가 573bytes와 10%, D가 780bytes와 30%라고 한다면,

다음과 같은 결과 값을 얻을 수 있다.

$$A는 \frac{960}{960} \times \frac{100}{0.05} = 2개,$$

$$B는 \frac{370}{960} \times \frac{100}{0.05} = 0.4 \Rightarrow 1개,$$

$$C는 \frac{573}{960} \times \frac{100}{0.5} = 0.25 \Rightarrow 1개,$$

$$D는 \frac{780}{960} \times \frac{100}{0.5} = 1.2 \Rightarrow 2개.$$

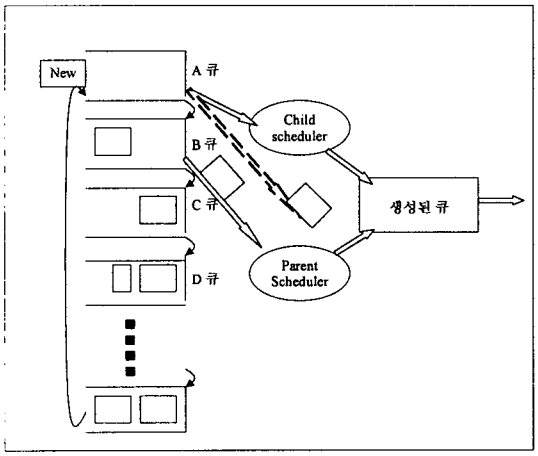

그림 3. 개선된 CQ 메커니즘

3.2 대역폭 공유 큐잉 알고리즘 세부 사항

주 스케줄러에 대해 복사된 부 스케줄러를 생성하기 위하여 커널 속에서 fork()라는 시스템 호출 함수를 사용하도록 한다. fork()의 호출로 자식 프로세스가 생성된 후에는 부모 프로세스와 자식 프로세스가 동시에 수행되기 때문에 주 스케줄러와 부 스케줄러 또한 동시에 수행된다. 또한 기존의 10~16개 사이의 큐들과 전송 큐 사이에 새로운 큐를 하나 생성하여 주 스케줄러와 복사된 부 스케줄러로부터 프레임을 받은 다음 FIFO방식을 적용하여 먼저 도착한 프레임부터 먼저 전송 큐에 전송하도록 함으로써 두 스케줄러에 의해 보내지는 프레임들로 인한 정체를 최소화 할 수 있다.

부 스케줄러가 작업하던 큐에서 작업을 끝내자마자 새로 생성된 큐와 함께 소멸되도록 하고 또 다시 똑같은 경우가 발생하면 다시 부 스케줄러와 큐가 다시 생성되도록 한다.

위와 같이 CQ를 개선함으로써 특정 트래픽에 대해서는 계속적인 처리를 해주면서도 다른 트래픽들에는 기아가 일어나지 않도록 할 수 있다.

4. 결론 및 향후 과제

본 논문에서 설계한 CQ 기반의 특정 트래픽의 실시간 보장을 위한 대역폭 공유 큐잉 알고리즘은 주 스케줄러와 부 스케줄러를 사용함으로써 임계작업(mission critical)의 애플리케이션이 요청한 트래픽에 대해 할당된 대역폭만큼의 프레임을 연속적으로 처리해줄 뿐만 아니라 동시에 다른 트래픽들에 대해서도 할당된 대역폭만큼의 프레임을 Round-Roubin 방식으로 각 큐를 순회하면서 처리해 주도록 하였다.

이러한 CQ 기반의 개선된 알고리즘은 기존의 FIFO와 Priority 알고리즘에 비해 데이터 상실이나 기아와 같은 현상을 현저히 줄일 수 있을 뿐만 아니라, 대역폭 할당에 있어도 WFQ 보다 정확한 제어를 할 수 있고, 기존의 CQ에 비해서도 임계작업의 애플리케이션이 요청한 트래픽에 대한 실시간 전송률을 높일 수 있다.

향후 과제로는, 프레임 산출법을 이용한 대역폭 할당 알고리즘과 부 스케줄러를 보다 효율적으로 관리할 수 있는 알고리즘을 enqueue()와 dequeue()함수에 적용하기 위한 연구를 계속할 것이다.

참고문헌

[1] Saravanan Radhakrishnan, "Linux - Advanced Networking Overview", Internet Draft, August, 1999.

[2] Solution of CISCO , "Bandwidth Management and Queueing". Internet Draft.

[3] Nirdosh J. Shah, "Exploring Stream Rate Control in LAN Environments", Internet Draft, June, 1999

[4] Sally Floyd, Van Jacobson, " Link-Sharing and Resource Management Models for packet Networks", IEEE/ACM Transactions on Networking, Vol. 3, No. 4, 1995.

[5] Solution of CISCO, "Planning for Quality of Service", Internet Draft.

[5] Sally Floyd, "Notes on CBQ and Guaranteed Service", Internet Draft, July, 1995.

[6] William Stalling, "High-Speed Networks: TCP/IP and ATM design principles", 1998.

[7] William Stalling, "High-Speed Networks:TCT/IP and ATM design principles", 1988.

[8] 조유근 역," *UNIX* ™ 시스템 프로그래밍", 1999.