멀티미디어 응용을 위한 수신측 중심의 혼잡 제어 알고리즘

준회원 정 기 성*, 박 종 훈*, 정회원 홍 민 철, 유 명 식*

A Receiver-based Congestion Control Algorithm with One-way Trip Time for Multimedia Applications

Gi-Sung Jung*, Jong-Hun Park* *Associate Members*,
Min-Cheol Hong*, Myung-Sik Yoo* *Regular Members*

요 약

멀티미디어 응용 서비스의 요구가 증대됨에 따라 멀티미디어 QoS (Quality of Service) 제공이 중요한 문제로 다루어지고 있다. 이에 따라 양끝단 시스템의 응용계층에서 혼잡 제어를 통하여 QoS를 제공하려는 많은 연구가 진행되고 있다. 본 논문에서는 송신측과 수신측이 적절한 협력을 바탕으로 전송 속도를 조절함으로서 혼잡에 대응하는 유니케스트 응용에 적합한 새로운 혼잡 제어 알고리즘인 RRC-OTT (Receiver-based Rate Control with One-way Trip Time)를 제안하였다. RRC-OTT 알고리즘은 수신측에서 받아들이는 패킷의 도착 분포를 분석하여 네트워크의 혼잡 상황을 파악하고 이에 따라 적절한 시기에 송신측으로 회신 정보 (ACK 패킷)를 보내줌으로서 전송 속도를 조절하도록 설계되었다. 시뮬레이션을 통하여 RRC-OTT 알고리즘이 기존의 송신측 중심 혼잡제어 알고리즘인 RAP (Rate Adaptation Protocol)에 비하여 ACK 패킷 오버헤드를 현저히 줄일 수 있었고 멀티미디어 QoS 측면에서 훨씬 낮은 지터 분포를 나타냄을 확인하였다.

Key Words : 멀티미디어 QoS, 혼잡 제어, OTT, 전송 지터

ABSTRACT

Supporting QoS (Quality of Service) for the multimedia applications becomes an important issue as the demand of multimedia applications increases. Thus, it is necessary for the application layer to have an efficient congestion control algorithm, which can support the multimedia applications' QoS requirements. In this paper, we propose a new application layer congestion control algorithm, called RRC-OTT (Receiver-based Rate Control with One-way Trip Time). RRC-OTT algorithm differs from the previously proposed algorithms in that the receiver takes the responsibility of the network congestion control. Thus, RRC-OTT algorithm can not only precisely estimate the network congestion using OTT (One-way Trip Time), but reduce the work load from the sender (e.g., the web server). Our simulation study shows that RRC-OTT algorithm can maintain the comparable link utilization to the previously proposed algorithms and keep the packet jitter low, which thus can help enhance the quality of multimedia applications.

Ⅰ. 서 론

인터넷이 발전하면서 과거의 일반적인 데이터만 주고받는 응용과 달리 비디오나 오디오 스트리밍

*숭실대학교 정보통신전자공학부 (jsung3310@daum.net, bluearets@hanmail.net, myoo@e.ssu.ac.kr, mhong@e.ssu.ac.kr)
논문번호 : 030173-0424, 접수일자 : 2003년 4월 24일
※ 본연구는 한국과학재단 목적기초연구 R01-2002-000-00073-0(2002)지원으로 수행되었습니다.

같은 실시간 멀티미디어 응용에 대한 요구가 높아지고 있다. 이런 멀티미디어 응용은 일반 데이터 트래픽과 달리 전송 지연에 민감한 반면 작은 손실은 허용되는 특성이 있다. 즉, 실시간 멀티미디어 트래픽은 수신측에서 받는 즉시 실시간으로 재생하여야 하므로 송신측과 수신측의 전송 지연 지터 (jitter)는 작아야 하고 반면에 어느 정도의 데이터 손실은 수신측에서의 에러 복구가 가능하다[1].

멀티미디어 데이터는 제한된 네트워크 자원과 다양한 전송채널 환경 등으로 인해 QoS 제어 메커니즘에 대해 강한 요구가 따른다. 인터넷의 네트워크 계층에서 QoS를 제공하기 위하여 많은 연구가 진행되어 왔으며 대표적인 것으로 IETF의 IntServ와 DiffServ를 들 수 있다. IntServ는 RSVP (Resource ReSerVation Protocol)[2] 시그널링을 통해 인터넷상에 자원을 예약하는 것인데 이것은 네트워크 라우터의 지원을 받아야 하며 확장성과 같은 문제점이 있다. 이런 IntServ의 단점으로 인해 DiffServ가 제안되었다. DiffServ는 많은 플로우를 몇 개의 클래스로 나누어 중간 라우터에서 클래스 별로 처리하게 된다. 그러나 멀티미디어 응용의 QoS에 대한 더욱 효과적인 제어방법은 멀티미디어 응용과 직접적인 관련이 있는 양 끝단 시스템의 응용계층에서 혼잡제어와 에러 복구 등의 알고리즘을 통해 보다 나은 서비스를 제공하는 것이다. 응용 계층의 혼잡제어는 네트워크 계층과 독립적으로 양 끝단 응용 계층에서 네트워크의 상황에 따라 전송속도를 조절하여 혼잡에 대응함으로서 QoS를 제공하는 것이다.

멀티미디어 응용의 데이터 전송을 위하여 사용되고 있는 전송 계층 프로토콜은 TCP (Transmission Control Protocol)와 UDP (User Datagram Protocol)가 있다. 실시간 멀티미디어 데이터를 TCP를 사용하여 전송하게 되면 TCP 혼잡 제어 알고리즘에 의해 전송속도가 최소 요구 대역폭 이하로 내려 갈 수 있으며 불필요한 재전송이 발생할 수 있고 패킷 지연 및 지터 특성이 저하 될 수 있다. 그러므로 실시간성을 요구하는 멀티미디어 트래픽은 보통 UDP를 사용하여 전송 된다. 그러나 UDP는 단지 전송 기능만을 하게 되며 QoS 제공을 위해 혼잡에 대응하는 메커니즘이 없기 때문에 UDP 상위 계층인 응용 계층에서 혼잡 제어 역할을 담당하게 된다. 응용 계층에서의 QoS 제공을 위한 혼잡 제어 알고리즘은 같은 링크를 사용하는 여러 가지 플로우들이 공정하게 대역폭을 사용하도록 해야 한다. 그러므로 UDP를 사용하는 멀티미디어 스트리밍 응용들의 혼잡 제어 알고리즘은 TCP 기반의 응용들과의 친밀도가 중요한 요소라 하겠다.

이 논문에서는 실시간 멀티미디어 응용을 위한 혼잡 제어 알고리즘인 RRC-OTT (Receiver-based Rate Control with One-way Trip Time)를 제안하였다. 이 알고리즘은 유니케스트 응용에 초점이 맞추어져 있는 ACK 기반의 양 끝단 혼잡 제어 알고리즘으로서 수신측이 주 역할을 담당하게 된다. 다른 응용 계층의 혼잡제어 알고리즘과 같이 기본적으로 AIMD (Additive Increase Multiple Decrease)[3] 알고리즘을 사용하여 전송 속도를 조절한다. 수신측에서 수신하는 패킷의 손실과 패킷 지연 시간 (one-way trip time) 분포를 이용하여 네트워크 상태를 분석하고 그 결과를 능동적으로 송신측에 전달하여 송신측이 효과적으로 전송 속도를 제어하도록 한다.

이 논문에서 제안된 RRC-OTT 알고리즘은 기존 혼잡제어 알고리즘의 단점인 ACK 패킷의 과다한 대역폭 사용을 현저히 감소시켰으며, 송신측 중심이 아닌 멀티미디어 데이터를 수신하는 수신측에서 혼잡제어의 주 역할을 하여 멀티미디어 QoS 제어를 더욱 효율적으로 할 수 있다는 장점이 있다. 또한 기존의 송신측 중심의 혼잡제어 알고리즘과 비교하였을 때 다른 멀티미디어 소스들과 공정하게 대역폭을 공유하고 패킷 지터, 손실 측면에서도 매우 우수한 성능을 보여준다.

이 논문의 구성은 2장에서 최근 연구되고 있는 실시간 멀티미디어 응용의 혼잡제어 알고리즘에 대해 기술하고 3장에서는 RRC-OTT 혼잡제어 알고리즘을 자세히 설명한다. 4장에서는 시뮬레이션 결과를 통해 RRC-OTT의 특성 및 성능을 살펴보고 마지막 5장에서는 결론과 함께 향후 연구 방향을 논의한다.

Ⅱ. 관련 연구

현재 멀티미디어 응용의 혼잡제어 방식 중 가장 활발히 연구가 진행되는 방법이 전송 속도 제어 (Rate Control) 방식이다[1][4]. 전송 속도 제어는 네트워크의 가용한 대역폭을 예측함으로서 멀티미디어 트래픽의 전송 속도를 결정하는 기술이다. 전송 속도 결정 역할을 송신측, 수신측 또는 송신측과 수신측이 같이 맡는지에 따라 송신측 전송 속도 제어 방식과 수신측 전송 속도 제어방식, 혼합 전송 속도

제어 방식으로 나눌 수 있다.

대표적인 송신측 전송 제어 방식에는 TLFC (TCP-like Flow Control Algorithm)[5]와 RAP(Rate Adaptation Protocol)[6]가 있다. TLFC는 RTP/RTCP (Real Time Protocol / Real Time Control Protocol)[7]에 의해 전송되는 패킷 손실 정보와 RTT (Round Trip Time) 정보를 이용하여 네트워크의 상태를 파악하고 이에 따라 전송 속도를 AIMD 방식으로 조절하게 된다. 송신측은 네트워크의 혼잡 상황을 패킷 손실과 RTT 두 가지 파라미터로 판단하여 Congested, Unloaded, Loaded와 Light_congested의 네 가지 상태로 나누게 되며 이에 따라 전송 속도를 상황에 맞게 조절한다.

TLFC는 패킷 손실이 발생할 때까지 계속 전송 속도를 올리는 다른 알고리즘과 달리 패킷 손실이 발생하지 않더라도 RTT가 임계점까지 증가하면 전송 속도를 다소 줄이게 된다. 이처럼 혼잡에 미리 대응함으로 안정된 전송속도를 유지하여 양 끝단 전송 지터가 작아지는 장점이 있다. TLFC는 그러나 Upper_Ratio와 Lower_Ratio 같은 파라미터의 최적화가 이루어지지 않았고 RTP/RTCP에 의해 제공되는 회신 정보가 적어 네트워크 상황에 대한 전송 속도 조절의 대처가 느리다는 단점이 있다.

RAP는 AIMD를 적용하며 유니케스트에 적합한 송신측 전송 속도 제어 알고리즘이다. 매 패킷마다 ACK 정보를 받으며 패킷 손실이 생기면 즉시 전송속도를 반으로 줄이고 손실이 없으면 매 SRTT 마다 속도를 증가시킨다.

RAP는 TCP 트래픽과 대역폭을 공정하게 사용하지만 매 패킷마다 ACK 정보를 요구함으로서 ACK 정보가 대역폭을 과다하게 사용하게 되는 단점이 있다. 또, 패킷 손실이 있을 때까지 전송속도를 계속 증가하고 손실이 생기면 즉시 속도를 반으로 떨어트림으로서 결국 전송 속도의 심한 기복을 가져오게 되어 양끝단의 전송 지연 지터 (jitter)가 심해져 멀티미디어 QoS를 저하시키는 단점이 있다.

TLFC와 RAP와 같은 송신측 전송 속도 제어 알고리즘은 유니케스트 응용과 멀티케스트 응용 둘 다 사용이 가능하며 ACK 정보를 기반으로 네트워크의 가용 대역폭에 맞게 전송 속도를 조절하는 방식이다. 이에 따라 패킷 손실을 최소화 할 수 있는 장점이 있는 반면 수신측의 수가 많아질수록 송신측 서버의 부하가 증가하게 되는 단점이 있다. 또한 ACK 정보의 RTT에는 멀티미디어 데이터 패킷의 경로의 혼잡 상황뿐만 아니라 ACK 패킷 경로의 혼잡상황까지 반영되기 때문에 정확한 멀티미디어 데이터 패킷 경로의 혼잡 상황을 반영할 수 없다.

수신측 전송 속도 제어 방식은 계층적 멀티케스트 응용에서 주로 사용되며 수신측에서 네트워크의 상태에 따라 채널을 더하거나 뺌으로서 수신 속도를 조절하는 방식이다[8]. 송수신 혼합 제어 방식은 멀티게스트 응용에 주로 사용되고 있으며 DSG (Destination Set Grouping) 프로토콜[9] 같이 송신측에서는 AIMD 방식으로 각 채널의 전송 속도를 조절하고 수신측에서도 네트워크 상태에 따라 채널을 더하거나 뺌으로서 수신 속도를 조절하게 된다.

Ⅲ. RRC-OTT 알고리즘

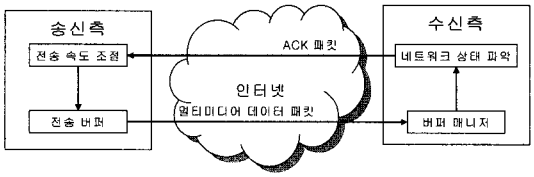


그림 1. RRC-OTT 알고리즘 구조

지금까지 제안된 유니케스트 응용의 혼잡제어 알고리즘이 주로 수신측에서 보낸 ACK 정보를 송신측에서 분석하여 전송 속도를 결정하는 것과 달리 RRC-OTT는 그림 1에서처럼 수신측에서 수신하는 패킷의 도착 분포 특성 분석을 통해 네트워크의 상태를 파악하고 이 정보를 송신측으로 보냄으로서 송신측이 전송 속도를 조절하도록 설계되었다. 이는 송신측에 있던 네트워크 상태 분석 기능을 수신측이 갖게 한다는 것을 의미한다. 이렇게 함으로서 송신측 (예: 웹 서버, 멀티미디어 서버)이 많은 수의 수신측과 연결되어 있을 때 송신측이 각각의 수신측에 대해 전송 속도 조절시 필요한 계산 (예: 타이머, RTT 계산 등)을 송신측이 아닌 수신측이 하게 되어 송신측의 부하를 현저히 줄일 수 있게 하였다. 또한 매번 ACK 정보를 보내지 않고 적절한 시기에 보내도록 하여 ACK 정보에 의한 대역폭 소비도 줄일 수 있다.

네트워크의 상태를 파악하는 파라미터로 패

킷 손실과 OTT (One-way Trip Time)를 사용하였다. RRC-OTT 알고리즘에서 RTT대신 OTT를 사용하여 네트워크의 상태를 분석하는 것은 다음과 같은 장점이 있다. 그림 1에 보이는 것처럼 멀티미디어 데이터 패킷과 ACK 패킷은 서로 다른 경로 (링크 대역폭, 버퍼 용량)를 따라 전송된다. 따라서 RTT를 사용하게 되면 ACK 패킷 경로의 혼잡 상황도 송신측의 전송 속도 조절에 반영될 수 있다. 그러나, RRC-OTT에서 사용하는 OTT는 송신측과 수신측사이의 패킷 지연만 반영하기 때문에 좀 더 정확한 네트워크 상태 분석이 이루어진다.

1. 송신측 알고리즘

송신측에서는 수신측으로부터 전송되는 ACK 종류에 따라 적절하게 전송 속도를 조절하게 된다. 기본적으로 AIMD 방식을 적용하기 때문에 일정한 양만큼 속도를 증가시키는 ACK (Slow Up ACK)와 네트워크 혼잡으로 인해 속도를 반으로 줄이는 ACK (Fast Down ACK)의 두 가지가 존재한다. 또한 수신측으로부터 전송되는 ACK 정보의 연속적인 손실을 대비한 Safety 타이머를 두게 된다.

1) ACK에 따른 전송 속도 조절

송신측에서 Fast Down ACK를 받게 되면 패킷 손실이 일어나거나 네트워크의 심각한 혼잡이 발생한 것으로 간주하여 그 즉시 전송 속도를 반으로 줄이게 된다. 전송 속도의 조절은 식 (1)을 사용하여 전송되는 패킷 사이의 간격, 즉 IPG (Inter-Packet Gap)를 조정함으로서 이루어진다.

$$IPG_{i+1} = 2 \times IPG_i \qquad (1)$$

반면, 송신측에서 Slow Up ACK를 받게 되면 네트워크의 상황이 양호한 것으로 판단하게 되며 식 (2)를 이용하여 수신 즉시 전송 속도를 올리게 된다.

$$IPG_{i+1} = \frac{IPG_i \times (\alpha \times SOTT)}{IPG_i + (\alpha \times SOTT)} \qquad (2)$$

파라미터 α와 SOTT (Smoothed OTT)는 송신측에서 ACK를 통해 보내는 회신 정보로서 전송 속도 계산에 이용된다. 파라미터 α는 식 (2)에서 보듯이 SOTT의 몇 배를 전송 속도 계산에 이용할지를 나타내는 파라미터이다.

송신측은 앞에 나열한 ACK들을 수신할 때마다 자신의 전송 속도를 조절해간다. 보내는 패킷마다 타이머를 사용하여 타임아웃을 검사하는 RAP와 달리 RRC-OTT 알고리즘은 송신측에서 ACK의 손실에 대한 타이머만 사용하게 되고 RTT 계산을 하지 않으므로 송신측의 부하를 상당히 감소시킬 수 있다.

2) Safety 타이머

수신측에서 받아들이는 매 패킷마다 ACK 패킷을 보내지 않기 때문에 수신측에서 전송된 ACK 패킷이 연속적으로 손실이 발생하는 경우를 대비하여 송신측에서는 ACK에 대한 Safety 타이머를 두게 된다. 송신측이 ACK를 받은 시점에 대한 평균치를 계산하고 이 값의 정수배만큼 기다려도 ACK가 오지 않게 되면 ACK의 연속적인 손실로 판단하게 되어 전송 속도를 반으로 줄이게 된다. 이 후 매 평균 ACK 수신 간격만큼 기다리며 속도를 계속 줄여 나간다.

2. 수신측 알고리즘

수신측은 패킷을 수신하게 되면 네트워크의 혼잡을 판단하기 위해 OTT와 SOTT를 계산하게 된다. 계산된 SOTT값과 OTT의 SDEV (Smoothed DEV) 값은 후에 수신되는 패킷의 OTT와 비교하여 혼잡 판단에 사용하게 된다. 패킷 손실이 발생하거나 혼잡 판단 메커니즘을 통해 혼잡을 인식하게 되면 수신측으로 Fast Down ACK를 전송하여 송신측이 전송 속도를 반으로 줄이게 하고 그렇지 않은 경우는 ACK 패킷 전송 간격을 정하여 Slow Up ACK를 전송하여 송신측이 전송 속도를 높일 수 있도록 한다.

1) 패킷 수신

수신측에서는 매 패킷을 받을 때마다 수신된 패킷의 OTT를 계산하여 네트워크 혼잡 상황을 파악한다. 혼잡 상황 판단에 대한 메커니즘은 이 후 그림 2에서 자세히 설명한다. OTT는 패킷을 수신한 수신측의 현재 시간과 송신측이 보내는 패킷에 실어 보내는 타임스탬프 시간과의 차이로 계산되고 이에 따라 SOTT는 다음과 같이 계산되어 진다.

$$SOTT_i = \frac{7}{8} SOTT_{i-1} + \frac{1}{8} OTT_i \quad (3)$$

계산된 SOTT 값과 OTT의 SDEV 값은 식 (4)와 같이 계산되어 저장이 되고 일정 시간 이후 (식 (4)의 허용 OTT 이후) 도착할 패킷의 OTT와 비교하여 네트워크 혼잡 상황 판단에 사용된다.

2) 혼잡 판단 메커니즘

RAP가 송신측에서 패킷에 대한 타이머를 관리하는 반면 RRC-OTT에서는 수신측에 타이머를 대신한 혼잡 판단 메커니즘이 수행된다.

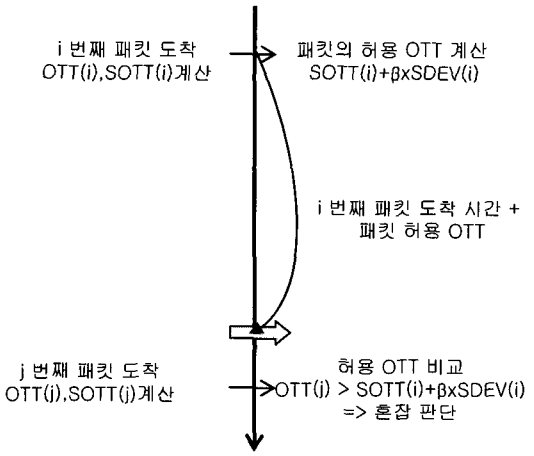


그림 2. 혼잡 판단 메커니즘

혼잡 판단은 그림 2에서 보는 것처럼 i번째 패킷이 도착하면 OTT(i)와 SOTT(i)가 계산되고 이 SOTT(i)와 β×SDEV(i)값을 더한 패킷의 허용 OTT를 식 (4)와 같이 계산한다.

$$\text{허용 } OTT_j = SOTT_i + \beta \times SDEV_i \quad (4)$$

$$SERR_i = OTT_i - SOTT_{i-1} \quad (5)$$

$$SDEV_i = \frac{3}{4} SDEV_{i-1} + \frac{1}{4} | SERR_i | \quad (6)$$

여기에서 허용 OTT의 의미는 허용 OTT 이후에 도착하는 패킷(패킷 j)은 i 번째 패킷에서 측정한 네트워크 혼잡 상황으로 예측할 때 허용 OTT 이내로 도착해야함을 말한다. 따라서, i번째 패킷이 도착한 시간과 계산된 패킷 허용 OTT를 더한 시점보다 뒤에 오는 j번째 패킷의 OTT(j)를 계산하여 (7)와 같이 OTT(j)값과 허용 OTT를 비교하고 OTT(j)가 크면 혼잡이 발생한 것으로 판단하게 된다.

$$OTT_j > SOTT_i + \beta \times SDEV_i \quad (7)$$

반면 (8)처럼 받은 패킷의 OTT(j)가 허용 OTT보다 같거나 작으면 혼잡이 발생하지 않은 것으로 판단하게 된다.

$$OTT_j \leq SOTT_i + \beta \times SDEV_i \quad (8)$$

앞서 설명한 바와 같이 OTT는 송신측과 수신측의 타임스탬프의 차로 계산되며 SOTT 역시 이 OTT를 이용하여 계산된다. 혼잡 판단 메커니즘에서 사용되는 OTT와 SOTT가 모두 송신측과 수신측 시스템의 시간 차이를 포함하고 있으므로 만약 송신측과 수신측의 동기가 맞지 않더라도 RRC-OTT의 혼잡 판단 메커니즘에는 문제가 없다.

식 (4)의 β 파라미터는 OTT의 SDEV값을 몇 배로 하여 허용 OTT를 계산하는지를 나타낸다. β값이 작을 경우 혼잡 판단이 빈번하게 발생하게 되어 전송 속도가 크게 올라가기 전에 전송 속도를 내려줌으로서 패킷 지연이 감소하는 반면 대역폭 활용도는 낮아지는 특성이 있다. 반면 큰 β값을 적용하게 되면 네트워크의 혼잡에 대한 판단이 느려져 대역폭 활용도는 좋아지나 패킷 손실이나 지연이 증가하는 단점이 있게 된다.

3) ACK 패킷 전송

패킷의 손실이나 혼잡 판단 메커니즘을 통해 혼잡이 발생한 것으로 판단되면 즉시 Fast Down ACK를 송신측으로 보내 전송 속도를 반으로 줄인다. 반면 패킷의 손실이 없고 혼잡 판단 메커니즘을 통해 혼잡이 없는 것으로 판단되면 ACK전송 시점을 기다린 후에 Slow Up ACK를 보내게 된다. Slow Up ACK 전송 간격은 γ×SOTT이며 여기서 γ는 Slow Up ACK 전송 간격 파라미터이다. 즉, 매 γ×SOTT마다 한 개의 ACK가 전송 되므로 파라미터 γ의 값에 따라 RAP 알고리즘에 비해 줄일 수 있는

ACK 수가 결정된다. 매 γ×SOTT마다 전송되는 Slow Up ACK에 파라미터 α와 SOTT 값을 실어 송신측으로 전송하여 송신측이 전송속도를 식 (2)를 사용하여 올릴 수 있도록 한다.

Ⅳ. 성능평가 및 비교분석

1. 시뮬레이션 환경

제안된 RRC-OTT 알고리즘의 성능을 RAP와 비교하여 평가하였다. 시뮬레이션에 사용된 토플로지는 그림 3처럼 간단한 네트워크로 구성된다. 여러 개의 송신노드가 SW1에 연결 되어 있고 수신노드는 SW2로 연결되어 있다. 송수신 노드와 스위치 사이의 링크는 SW1과 SW2 사이의 링크보다 큰 대역폭을 가지게 되므로 SW1과 SW2 사이의 링크는 병목링크가 되고 SW1은 병목점이 된다.

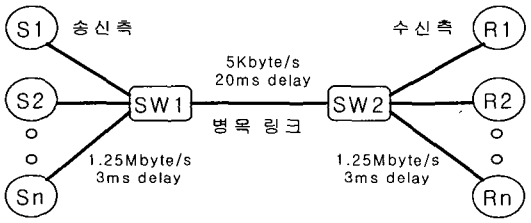


그림 3. 시뮬레이션 토플로지

표 1에는 시뮬레이션에 사용된 여러 파라미터들의 값을 보여주고 있다.

표 1. 시뮬레이션 파라미터

파라미터	값
데이터 패킷 크기	100Byte
ACK 패킷 크기	40Byte
병목 링크 전송 지연	20ms
병목 링크 B/W	5KByte/s
옆 링크 전송 지연	3ms, total 6ms
옆 링크 B/W	1.25MByte/s
병목 버퍼 크기	8KByte
시뮬레이션 시간	60 ~ 100s
β 파라미터	1

2. 시뮬레이션 결과

앞서 설명한 시뮬레이션 환경에서 RAP와 RRC-OTT의 성능평가 결과를 비교 분석한다.

그림 4는 파라미터 α와 γ를 2로 하였을 때 단일

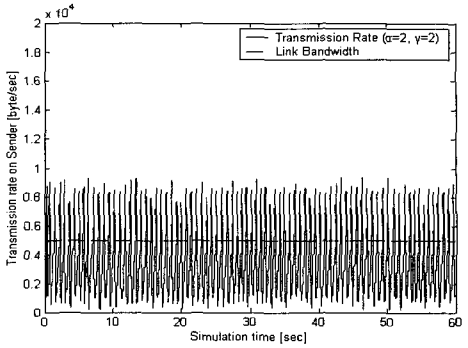


그림 4. 단일 RRC-OTT 전송 속도

소스의 RRC-OTT 전송 속도를 나타내고 있다. 병목 링크의 용량인 5KByte/s를 중심으로 전송 속도가 계속 변하는 것으로 RAP와 매우 유사한 전송 속도 제어 결과를 보여준다.

1) 파라미터 값의 변화에 따른 영향

RRC-OTT의 성능에 영향을 줄 수 있는 파라미터로 α와 γ가 있다. α는 송신측에서 식 (2)를 적용하여 전송 속도를 결정할 때 몇 배의 SOTT값을 적용하는지 나타내는 파라미터이다.

그림 5는 파라미터 α에 따른 큐 지연의 변화를 나타내고 있다. α값이 커지면 식 (2)에 따라 전송 속도가 증가폭이 감소하게 되어 큐에 쌓이는 패킷의 수도 천천히 증가하게 된다. 반면 α값이 작아질 때 전송 속도 증가폭이 커지게 되어 큐에 쌓이는 패킷의 수가 급격하게 증가한다. 따라서 파라미터 α값이 1,2,4로 증가할수록 패킷이 큐에서 지연되는 시간이 현저히 감소하게 됨을 알 수 있다.

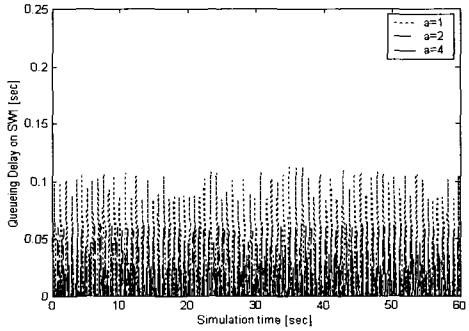


그림 5. 파라미터 α에 따른 큐 지연 변화

그림 6에서는 ACK 패킷의 전송 간격 파라미터 γ에 따른 큐 지연 변화를 나타내고 있다. γ 파라미터는 ACK 패킷이 몇 배의 SOTT 간격 마다 수신측에서 송신측으로 전송되는지를 나타내는 파라미터이다. 파라미터 γ가 커질수록 ACK 패킷의 전송 간격이 커져 송신 속도를 증가시키는 시간 간격이 증가하게 되며 이에 따라 큐에 쌓이는 패킷의 수가 천천히 증가하게 된다. 반면 γ가 작게 되면 보다 자주 ACK를 보내게 되어 전송 속도를 증가시키는 시간 간격이 짧아지기 때문에 큐에 쌓이는 패킷의 수가 급격히 증가한다. 이에 따라 파라미터 α와 마찬가지로 ACK 전송 간격 파라미터 γ가 증가할수록 큐 지연이 줄어들게 된다.

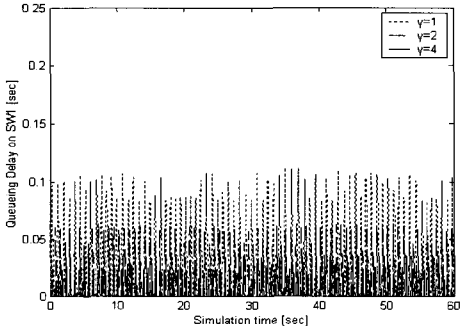


그림 6. 파라미터 γ에 따른 큐 지연 변화

파라미터 값의 증가는 큐 지연측면에서는 좋은 효과를 나타내지만 파라미터 값의 증가에 따라 병목 링크의 활용도는 다소 감소하게 된다. 표 2에서는 파라미터 α와 γ 모두 1,2,4로 증가할 때 68%대를 나타내던 병목 링크 활용도가 57~60%대로 감소하는 것을 보여주고 있다.

표 2. 파라미터에 따른 병목 링크 활용도

파라미터 α	파라미터 γ	병목 링크 활용도
2	1	68.14%
2	2	63.48%
2	4	57.84%
1	2	67.28%
2	2	63.48%
4	2	60.92%

2) RRC-OTT와 RAP의 큐 지연 비교

앞서 본 바와 같이 RRC-OTT의 큐 지연은 파라미터 값에 따라 변하게 된다. RAP와 공정하게 큐 지연을 비교하기 위해 RAP와 비슷한 수준의 파라미터 값을 설정한다. RAP는 SRTT를 사용하여 전송 속도를 증가시키고 RRC-OTT는 SOTT를 사용하게 전송 속도를 증가하게 된다. SRTT가 SOTT의 대략 2배의 값을 가지게 되므로 RRC-OTT의 파라미터 α 값을 2로 설정하였다. 또한 RAP가 매 SRTT마다 전송 속도를 증가시키기 때문에 RRC-OTT의 ACK 전송 간격도 2배의 SOTT마다 보내도록 하기 위하여 파라미터 γ값을 2로 설정하였다.

그림 7은 RAP 단일 소스의 큐 지연을 보여준다. 그림에서 보듯 RAP는 최대 0.2초 이상의 큐 지연이 발생함을 알 수 있다.

RRC-OTT 단일 소스의 큐 지연은 그림 8과 같다. RAP와 비슷한 수준의 파라미터 값인 α와 γ가 모두 2인 경우에 0.05초의 지연을 가지게 됨을 알 수 있다. 이는 RAP의 1/4 수준으로서 패킷 지연의 상당한 감소를 나타낸다.

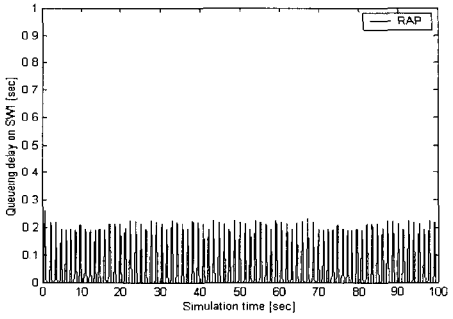


그림 7. RAP 단일 소스의 큐 지연

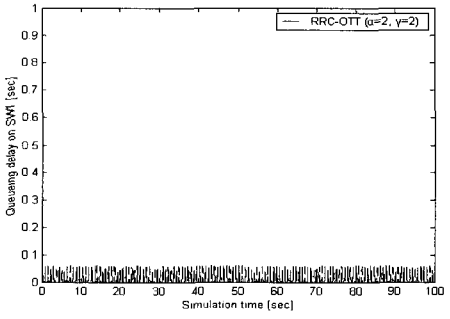


그림 8. RRC-OTT 단일 소스의 큐 지연

3) 여러 개의 RRC-OTT 소스

지금까지 하나의 RRC-OTT 소스의 동작을 살펴보았고 여기서는 여러 개의 RRC-OTT 소스의 동작을 살펴본다.

그림 9에서 RAP의 첫 번째 소스는 시뮬레이션 시작과 함께 패킷 송신을 시작하고 두 번째 소스는 20초에 세 번째 소스는 40초에 시작할 때의 병목 버퍼의 큐 지연을 보여준다. 20초까지 최대 0.2초의 지연을 보이다가 20초 후 두 번째 소스가 동작을 시작하면서 0.3초가 넘는 지연을 보이고 있고 40초 이후에는 최대 0.6초까지의 지연을 나타낸다.

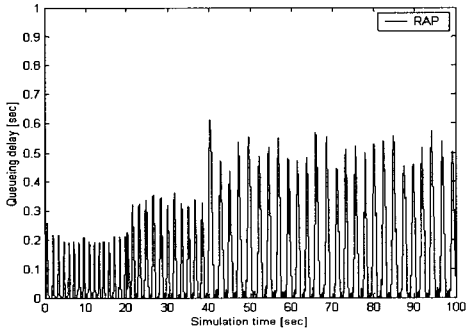


그림 9. RAP 소스의 증가에 따른 큐 변화

반면, 그림 10는 파라미터 α와 γ가 2인 상태에서 RRC-OTT 소스 증가에 따른 큐 지연을 나타낸다. 위와 같이 두 번째 소스가 20초에 세 번째 소스가 40초에 각각 동작을 시작한다. RAP와 달리 20초까지는 0.05초의 지연이 발생하고 40초까지 0.17초, 40초 이후에는 최대 0.3초의 지연이 발생함을 보여준다. 단일 소스의 경우뿐만 아니라 여러 소스의 동작에서도 RRC-OTT는 RAP보다 상당히 낮은 큐 지연 및 지터를 가지는 것을 확인할 수 있다.

그림 11은 소스의 수가 증가할 때 하나의 소스가 사용하는 평균 대역폭을 나타내고 있다. 소스의 수가 1에서 9까지 증가할수록 사용하는 대역폭은 RRC-OTT의 파라미터 값에 따라 조금씩 변할 수 있다. RRC-OTT와 RAP의 차이는 크지 않으며 소스의 수가 많아질수록 거의 같아지는 것을 보여준다.

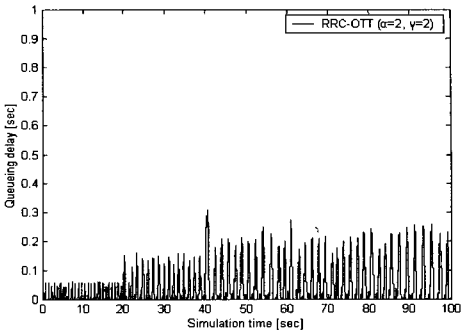


그림 10. RRC-OTT 소스의 증가에 따른 큐 지연 변화

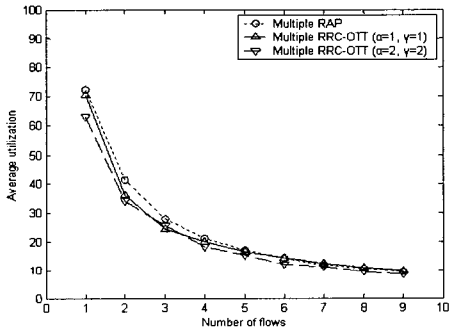


그림 11. 소스의 증가에 따른 각 소스의 평균 대역폭 사용

그림 12는 소스의 수가 증가할 때의 병목 링크의 활용도를 보여준다. 전반적으로 소스의 수가 증가할수록 RRC-OTT와 RAP의 차이가 감소함을 보여주고 있으며 RRC-OTT의 파라미터가 모두 1인 경우는 소스의 수가 6 이상으로 증가하게 되면 오히려 RAP보다 더 높은 링크 활용도를 가짐을 보여준다.

V. 결 론

멀티미디어 응용의 혼잡 제어를 위해 많은 알고리즘이 제안되었다. 그러나 실시간 응용이 요구하는 낮은 전송 지터와 적은 손실, 그리고 다른 트래픽과의 공정한 대역폭 공유 등의 요구조건을 만족 시키지 못하였다. 이 논문에서는 수신측이 주된 역할을 담당함으로서 이루어지는 혼잡 제어 알고리즘인 RRC-OTT를 제안하

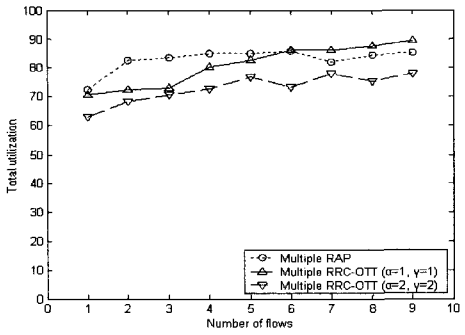


그림 12. 소스의 증가에 따른 병목 링크 활용도

였다. 시뮬레이션 결과를 통해 이 논문에서 제안한 RRC-OTT 알고리즘은 일정한 링크 활용도를 유지하면서도 낮은 전송 지터를 유지하여 멀티미디어 응용 QoS에 적합함을 증명하였다. 또한 송신측에 집중된 기능을 수신측에 분산하여 송신측의 부하를 줄이고 ACK 정보가 많은 대역폭을 소비하는 단점도 보완하였다.

앞으로의 연구는 QoS 제공의 필수 요소인 전송 지터 (jitter) 제어에 대한 연구와 멀티미디어 데이터 코덱과의 최적화 연구가 필요하다.

참 고 문 헌

[1] Dapeng Wu, Yiwei Thomas Hou, Ya-Qin Zhang, "Transporting Real-Time Video over Internet: Challenges and Approaches", Proceedings of the IEEE, 88, pp. 1855-1875, Dec. 2000

[2] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP : A new resource ReSerVation Protocol", IEEE Network, 7, pp. 8-18, Sep. 1993

[3] D. Chiu, R. Jain, "Analysis fo the increase and decrease algorithm for congestion avoidance in computer networks", Journal of Computer Networks and ISDN, 17(1), pp. 1-14, June 1989

[4] Dapeng Wu, Yiwei Thomas Hou, Wenwu Zhu, Ya-Qin Zhang, Jon M. Peha, "Streaming Video over the Internet: Approaches and Directions ", IEEE Transaction On Circuits And System For Video Technology, 11, pp. 282-300, Mar. 2001

[5] Seung-Gu, Jong-Suk Ahn, "TCP-like Flow Control Algorithm for Real-Time Applications", in Proc. IEEE International Conference on Networks, pp. 99-104, 2000

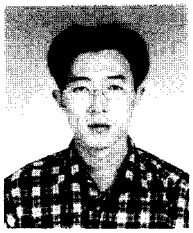
[6] Reza Rejaie, Mark Handly, Deborah Estirn, "RAP:An End-to-End Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", IEEE INFOCOM, 3,pp 1137-1145, Mar. 1999

[7] H. Schulzrinne, S. Casner, R. Frederick, Van Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, Jan. 1996

[8] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven layered multicast", in Proc. ACM SIGCOM'96, pp. 117-130, Aug.1996

[9] S. Y. Cheung, M Ammar, and X. Li, "On the use of destination set grouping to improve fairness in multicast video distribution", in Proc. IEEE INFOCOM'96, pp. 553-560, Mar. 1996

정 기 성(Gi-Sung Jung) 준회원



2002년 2월 : 숭실대학교 전자공학과 학사

2002년 3월~현재 : 숭실대학교 정보통신공학과 석사과정

<주관심분야> 광 네트워크, 네트워크 QoS,

박 종 훈(Jong-Hun Park) 준회원

2003년 2월 : 숭실대학교 전자공학과 학사

2003년 3월~현재 : 숭실대학교 정보통신공학과 석사과정

<주관심분야> 멀티미디어 QoS, IP 멀티케스트, 무선 QoS

홍 민 철(Min-Cheol Hong) 정회원

1988년 2월 : 연세대학교 전자공학과학사
1990년 8월 : 연세대학교 전자공학과 석사
1997년 9월 : Northwestern Univ. Electrical Engineering 박사
1990. 7 - 1991. 8 : LG 정보통신 (연구원)
1997. 9 - 1998. 8 : Northwestern Univ. (Research Fellow)
1998. 9 - 2002. 2 : LG 전자 (선임연구원)
2000. 3 - 현재 : 숭실대학교 정보통신전자공학부 조교수

<주관심분야> 영상 복원, 비선형 영상처리 및 필터링, 영상 압축, Visual Communication

유 명 식(Myung-Sik Yoo) 정회원

1989년 2월 : 고려대학교 전자전산공학과 학사
1991년 2월 : 고려대학교 전자공학과 석사
2000년 : Dept. of Electrical Engineering, SUNY at Buffalo 박사
2000. 1 - 2000. 8 : Nokia Research Center/Bosoton, Senior Research Engineer
2000. 9 - 현재 : 숭실대학교 정보통신전자공학부 조교수

<주관심분야> 광네트워크, OBS, 네트워크 QoS, 유무선 통신망