碩 士 學 位 論 文
指導敎授 金 南 潤

지연시간에 기반한 멀티미디어 전송율 제어 알고리즘 연구

Study on Delay-based Rate Control Algorithm for Multimedia Streams

2006年 6月 日

漢城大學校 一般大學院

情報시스템工學科

情報시스템專攻

宋 勇 憲

碩 士 學 位 論 文
指導敎授 金 南 潤

지연시간에 기반한 멀티미디어 전송율 제어 알고리즘 연구

Study on Delay-based Rate Control Algorithm for Multimedia Streams

위 論文을 情報시스템工學 碩士學位論文으로 提出함

2006年 6月 日

漢城大學校 一般大學院

情報시스템工學科

情報시스템專攻

宋 勇 憲

宋勇憲의 工學 碩士學位論文을 인정함

2006年 6月 日

심사위원장 이 기 원 

심사위원 방 갑 산(인) 

심사위원 김 남 윤 

< 제목 차례 >

<그림 차례>

제 1 장 서 론

제 1 절 연구 배경

웹 서비스나 FTP 서비스는 오늘날 인터넷 성장의 원동력이 되었다. 인터넷의 폭발적인 성장과 함께 실시간 방송, 원격 화상회의, VoIP와 같은 멀티미디어 서비스가 빠르게 증가하고 있다.

멀티미디어 서비스는 웹 서비스나 FTP 서비스와는 다른 몇 가지 특징을 가지고 있다 [1-4]. 첫째, 제한된 시간 안에 패킷이 전송되어야 한다. 제한된 시간을 초과한 패킷은 패킷 손실로 취급된다. 둘째, 패킷 손실에 민감하다. 잦은 패킷의 손실은 서비스의 품질을 떨어뜨리게 된다. 물론 멀티미디어 서비스가 자체적으로 손실 보상이라는 알고리즘을 사용하여 패킷 손실을 보상한다. 그러나 네트워크 혼잡에 의한 패킷 손실이 많을 시에는 서비스의 품질이 크게 떨어질 수 있다. 셋째, 전송율의 변화가 너무 심해서는 안 된다. 이는 화면의 잦은 변화로 사용자를 피곤하게 만든다.

멀티미디어 서비스의 이러한 특징 때문에 전송 프로토콜로 TCP를 사용하기보다는 UDP를 많이 사용한다. 그러나 UDP는 TCP와는 달리 네트워크 혼잡제어를 수행하지 않는다. 따라서 UDP를 사용하는 서비스는 네트워크에 혼잡이 발생해도 자신이 보내고자 하는 전송율로 데이터를 계속해서 보내게 되어 많은 데이터 손실이 발생한다. 이것은 TCP를 사용하는 서비스와 UDP를 사용하는 서비스간에 자원의 공정한 이용을 실현할 수 없을 뿐만 아니라 네트워크 혼잡도 해결할 수 없는 결과를 야기한다.

UDP를 사용하는 서비스의 이러한 문제를 해결하기 위해 TFRC(TCP Friendly Rate Control) 알고리즘이 IETF 작업 그룹에 의

해 발표되었다 [5]. TFRC는 네트워크 혼잡을 해결하고 다른 서비스들과의 공정한 자원 이용을 위해 설계된 알고리즘이다. 그러나 TFRC 알고리즘은 네트워크 혼잡에 의한 패킷 손실이 발생했을 때 이를 해결하기 위해 제안된 알고리즘이다. 그러므로 TFRC는 네트워크 혼잡에 의한 패킷 손실이 없는 한 계속해서 데이터 전송율을 증가시키기 때문에 지속적으로 네트워크 혼잡에 의한 패킷 손실이 발생할 가능성이 존재한다. 지속적인 패킷의 손실은 멀티미디어 서비스의 QoS(Quality of Service) 저하의 원인이 되므로 TFRC 알고리즘은 멀티미디어 서비스에 적합하지 않다. 따라서 멀티미디어 서비스의 QoS를 개선하기 위해서는 네트워크 혼잡에 의한 패킷 손실이 발생하기 전에 데이터의 전송율을 조절하기 위한 알고리즘이 요구된다.

제 2 절 연구 내용

패킷 손실을 줄이기 위해서는 패킷 손실의 원인을 분석할 필요가 있다. 패킷 손실의 원인으로 네트워크 링크 에러, 네트워크 혼잡 등을 들 수 있다. 유선 네트워크에서는 링크 에러가 거의 발생하지 않는다고 볼 때 네트워크 혼잡이 패킷 손실의 가장 큰 원인으로 볼 수 있다.
네트워크 혼잡은 라우터의 패킷 처리능력의 부족에 의해 발생한다. 라우터에 입력되는 패킷의 양이 라우터가 처리할 수 있는 패킷의 양보다 많을 때 라우터의 큐 길이가 증가하게 되고 이로 인해 패킷의 지연시간이 증가하게 된다. 라우터의 큐 크기는 제한되어 있기 때문에 계속해서 라우터에 들어오는 패킷의 양이 라우터가 처리할 수 있는 패킷의 양보다 많으면 라우터의 큐는 계속해서 쌓이게 되고 큐가 한계에 이르면 라우터는 패킷을 폐기하게 된다. 따라서 패킷 손실을 예방하기 위해서는 라우터가 처리할 수 있는 패킷의 양과 라우터에 들어오는 패킷의

양을 동일하게 해줄 필요가 있다. 이를 위해 지연시간을 이용한 네트워크 혼잡의 징후를 사전에 파악하는 것이 필요하다. 패킷 지연시간은 송신자와 수신자 사이의 패킷 전송 시간을 말한다. 지연시간은 네트워크 혼잡에 의한 패킷 손실과 밀접한 관계가 있다. 지연시간의 증가는 라우터에 들어오는 패킷의 양이 라우터가 처리할 수 있는 패킷의 양보다 많아 라우터 큐의 길이가 증가하고 있다는 것을 의미한다. 라우터는 큐의 길이가 한계에 도달하면 패킷을 폐기하기 때문에 지연시간이 증가하면 패킷 손실의 확률이 증가한다. 본 논문에서는 이러한 사실에 기반해, 라우터에서 패킷 손실이 발생하기 전에 지연시간에 따라 데이터 전송율을 조정하여 라우터에 입력되는 패킷의 양과 라우터가 처리할 수 있는 패킷의 양을 동일하게 해줄 수 있는 DBRC(Delay-Based Rate Control) 알고리즘을 설계하였다.

제 3 절 연구 결과

멀티미디어 서비스의 QoS는 지연시간과 패킷 손실에 크게 영향을 받는다. 멀티미디어 서비스는 실시간성을 요구하므로 높은 지연시간은 패킷 손실로 취급된다. 따라서 QoS를 높이기 위해서는 멀티미디어 서비스가 요구하는 지연시간 내에 패킷이 도착해야 하고 패킷의 손실도 많지 않아야 하며 전송율의 변화도 크지 않아야 한다.

이러한 멀티미디어 서비스의 QoS 특성을 고려해 TFRC와 본 논문에서 제안한 DBRC 알고리즘의 성능을 비교하였다. 먼저 DBRC는 멀티미디어 서비스가 요구하는 지연시간 내에 패킷이 도착하는 결과를 보인 반면 TFRC는 많은 패킷이 멀티미디어 서비스가 요구하는 지연시간을 초과하여 도착하는 결과를 보였다. 전송율의 변화에서도 DBRC는 빠르게 일정한 전송율을 유지하는 데 반해 TFRC는 지속적인 전송율의

변화를 경험하였다. 패킷 손실에서도 DBRC의 경우 초기 패킷 손실이 후 추가적인 패킷 손실이 발생하지 않은데 반해 TFRC는 지속적인 패킷 손실이 발생하였다. 이는 DBRC 알고리즘이 TFRC 알고리즘보다 멀티미디어 서비스의 QoS를 높이는데 적합하다는 것을 말해준다.

제 4 절 논문의 구성

2장 관련 연구에서는 네트워크 혼잡제어 알고리즘으로 많이 사용되고 있는 AIMD [6]와 TFRC 알고리즘에 대해 분석한다. 3장에서는 DBRC 알고리즘에 대해 설명한다. 4장에서는 네트워크 시뮬레이션 도구인 NS-2를 사용하여 여러 네트워크 환경에서 DBRC 알고리즘과 TFRC 알고리즘을 비교 분석한다. 마지막으로 5장에서는 결론 및 향후 연구에서 DBRC 알고리즘의 성능과 향후 개선 사항에 대해 이야기한다.

제 2 장 관련 연구

인터넷의 급격한 성장과 함께 한정된 네트워크 자원을 효율적으로 이용하고 네트워크 혼잡을 제어하기 위한 많은 연구가 진행되어 왔다 [7-16]. 이번 장에서는 네트워크 혼잡제어 알고리즘으로 많이 사용되고 있는 AIMD와 TFRC 알고리즘에 대해 분석한다.

제 1 절 AIMD (Additive Increase Multiplicative Decrease)

1. AIMD 개념

AIMD는 네트워크 통신에서 과도한 트래픽 발생으로 인한 네트워크에 혼잡이 발생했을 때 네트워크 혼잡을 해결하기 위해 설계되었다. AIMD를 사용하는 대표적인 프로토콜로 오늘날 인터넷 성장의 원동력이 된 TCP (Transmission Control Protocol)를 예로 들 수 있다.

AIMD의 동작 방법은 다음 두 단계로 이루어져 있다. 첫째, 네트워크에 혼잡이 없을 때 AIMD는 점진적으로 윈도우 크기를 증가시킨다. 만약 수신자로부터 전송된 패킷에 대한 응답이 도착하면 네트워크에 혼잡이 없다고 판단하여 계속해서 윈도우 크기를 증가시킨다. 이러한 점진적인 전송율의 증가는 네트워크의 혼잡으로 인한 패킷 손실이 발생하기 전까지 계속된다. 둘째, 네트워크에 혼잡이 발생하여 패킷 손실이 발생하면 AIMD는 전송율을 반으로 감소시킨다. 이는 AIMD가 네트워크에 혼잡이 발생했을 때 빠르게 혼잡을 해결하기 위해 설계되었기 때문이다. 아래는 AIMD의 전송율 제어 의사코드를 보여준다.

$$W_{i+1} \leftarrow W_i + \alpha \text{ when } f = 0$$

$$W_{i+1} \leftarrow W_i * (1-\beta) \text{ when } f > 0$$

W_i는 i번째 주기의 윈도우 크기, α는 윈도우 크기 증가 상수, β는 윈도우 크기 감소 상수, f는 패킷 손실율을 나타낸다. TCP에서는 일반적으로 $\alpha = 1$, $\beta = 0.5$의 값을 가진다.

2. AIMD의 장점/단점

AIMD는 패킷 손실이 발생하지 않는 한 계속해서 윈도우 크기를 증가시키므로 자원의 효율적인 이용이 가능하다. 또한 네트워크에 혼잡이 발생했을 때 빠르게 혼잡을 해결할 수 있는 장점도 가지고 있다.

그러나 AIMD는 패킷 손실과 잦은 전송율의 변화에 민감한 멀티미디어 스트림의 전송율 제어 알고리즘으로 사용하기에는 몇 가지 문제가 있다. 첫째, AIMD는 지속적으로 패킷 손실을 경험한다. 둘째, AIMD는 패킷 손실 시 전송율을 크게 감소시킨다. 아래의 그림 1은 AIMD의 전송율 그래프를 보여준다. 그림에서 패킷 손실 시 전송율이 반으로 줄어드는 것을 볼 수 있다. 또한 지속적인 패킷 손실로 인해 전송율의 변화가 잦음을 확인 할 수 있다. 이것은 멀티미디어 서비스의 QoS 저하의 원인이 되기 때문에 AIMD를 멀티미디어 서비스에 적용하는 것은 바람직하지 않다.

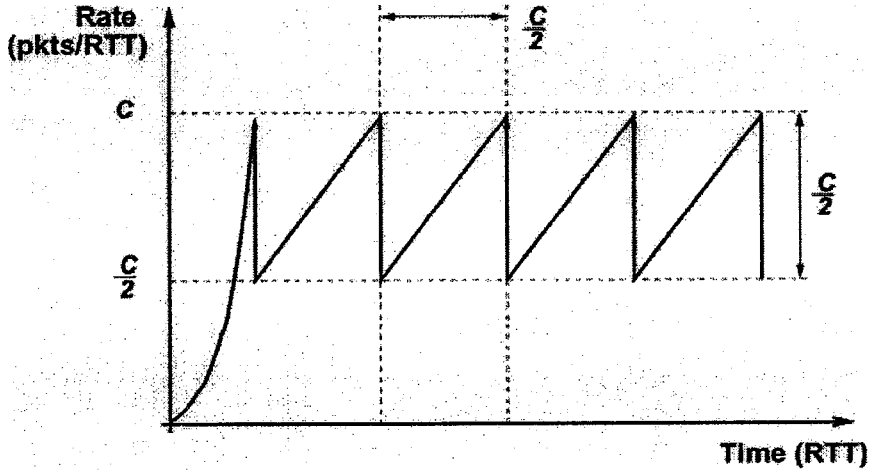


그림 1. AIMD의 전송율 그래프

제 2 절 TFRC (TCP Friendly Rate Control)

1. TFRC 개념

TFRC는 통신 네트워크 환경에서 AIMD 알고리즘을 사용하는 TCP 트래픽과 경쟁하는 유니 캐스트 플로우를 위해 설계되었다. TFRC는 AIMD와 마찬가지로 네트워크 상에 혼잡이 발생 시 네트워크 혼잡을 해결하기 위해 설계된 알고리즘이다. 또한 동일한 연결 상에서 TCP를 사용하는 다른 트래픽과의 네트워크 자원의 공정한 이용과 자원의 효율적인 이용을 위해 설계되었다.

TFRC는 패킷 손실율, RTT(Round Trip Time), 패킷의 크기 등을 사용하여 전송율을 결정한다. 패킷 손실율은 수신자가 경험하는 패킷의 손실율을 말하며 RTT는 송신자가 전송한 패킷이 수신자를 거쳐 돌아오는 시간을 말한다. TFRC는 패킷 손실율의 크기와 RTT에 기반해 전송율을 결정하므로 단순히 패킷 손실에 기반해 전송율을 조정하는 AIMD보다 전송율의 변화가 작고 패킷 손실도 줄일 수 있다. 다음은

TFRC의 혼잡제어 메카니즘을 보여준다.

- 수신자는 패킷 손실율을 측정하여 송신자에게 패킷 손실율이 담긴 피드백 메시지를 전송한다.
- 송신자는 수신자가 전송한 피드백 메시지를 사용하여 RTT를 측정한다.
- 송신자는 피드백 메시지의 패킷 손실율과 측정한 RTT를 TFRC의 전송율 조정 수식을 사용하여 새로운 전송율을 계산한다.
- 송신자는 TFRC의 전송율 수식에 의해 계산된 전송율로 데이터를 전송한다.

다음은 TFRC의 전송율 수식을 보여준다.

$$T = \frac{S}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

T는 데이터 전송율, S는 패킷의 크기, R은 RTT(Round Trip Time), p는 패킷 손실율, t_{RTO}는 재 전송 타임 아웃을 나타낸다. 수식에서 RTT가 전송율의 변화에 큰 영향을 준다. 반면 패킷 손실율은 전송율의 선형적인 변화에 영향을 준다. 이는 TFRC가 AIMD와는 달리 패킷 손실 시 전송율의 변화가 크지 않다는 것을 말해준다.

2. TFRC의 장점/단점

TFRC는 TCP와 비교하여 더 낮은 전송율의 변화를 가지기 때문에 전송율의 변화에 민감한 멀티미디어 서비스에 적합한 알고리즘이다. 또

한 인터넷의 지배적인 TCP 트래픽과의 공정한 자원이용과 네트워크에 혼잡이 발생하였을 때 빠르게 네트워크 혼잡을 해결할 수 있는 장점이 있다.

그러나 TFRC는 패킷 손실이 없는 한 계속해서 전송율을 증가시키기 때문에 지속적인 패킷 손실이 발생한다. TFRC가 TCP에 비해 낮은 전송율의 변화를 보이기는 하지만 지속적인 패킷 손실은 멀티미디어 서비스의 QoS 저하라는 결과를 야기하는 단점이 있다.

제 3 장 DBRC 알고리즘

제 1 절 시스템 모델

본 논문은 인터넷 환경에서 스트리밍 서버와 클라이언트 사이의 멀티미디어 스트림을 전송하는 시스템 모델을 사용한다. 스트리밍 서버는 저장된 비디오 파일을 클라이언트에게 전송하거나 실시간으로 캡쳐한 비디오를 클라이언트에게 전송한다. 클라이언트는 멀티미디어 스트림을 수신하는 동안 패킷 손실율과 지연시간의 정보를 이용하여 멀티미디어 스트림의 전송율을 계산한다. 이렇게 계산된 전송율의 정보는 Receiver Report를 통해 스트리밍 서버에 보내어지고 스트리밍 서버는 전송율을 조정하여 전송한다. 그림 2는 본 논문의 시스템 모델을 보여준다.

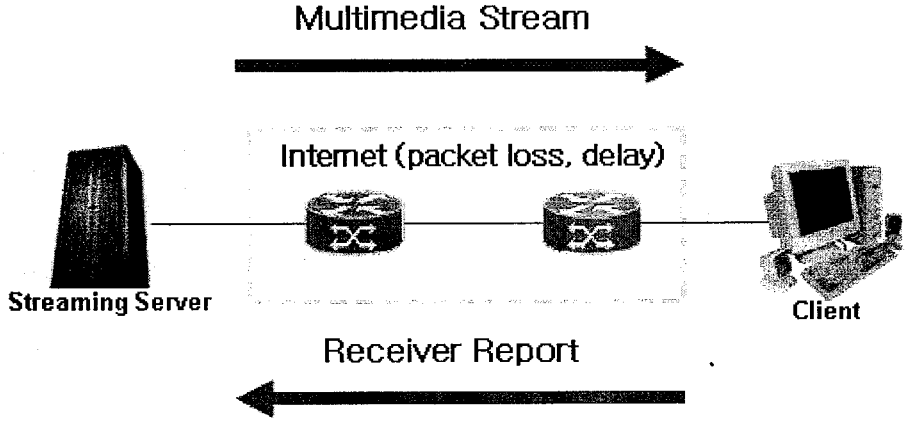


그림 2. DBRC 시스템 모델

제 2 절 용어 정의

표 1은 DBRC 알고리즘의 용어를 정의를 보여준다.

표 1. DBRC 용어

α	목표 지연시간(M*α)을 위한 상수 ($\frac{1}{2}$ <= α <= 1)
β	전송율 증가/감소 상수
γ	전송율 감소 상수 (패킷 손실 시)
M	패킷 손실 시 최대 지연시간
R_i	epoch i에서의 전송율
F_i	epoch i에서의 패킷 손실율
$d_i(j)$	epoch i에서의 패킷 j의 지연시간
D_i	epoch i에서의 평균 지연시간 $D_i = \frac{1}{n}\sum_{i=1}^{n} d_i(j)$
δ	지연시간 표준편차
ε	M과 D_i 변화에 따른 추가적인 전송율 증가/감소 인자 $\varepsilon = \frac{M*\alpha - D_i}{M*\alpha}$ (-1 < ε < 1)
c	지연시간 증가/감소 상수 c = -1: 지연시간 증가 c = 1: 지연시간 감소 c = 0: 지연시간 변화 없음

제 3 절 전송율과 지연시간 분석

1. DBRC의 기본 개념

DBRC는 패킷의 지연시간에 기반해 멀티미디어 스트림의 전송율을 제어하기 위해 설계되었다. DBRC는 네트워크에 혼잡이 발생하기 전에 네트워크 혼잡의 징후를 미리 감지하여 전송율을 감소시킨다. DBRC의 이런 행위는 네트워크 혼잡에 의한 패킷 손실을 줄이고 패킷의 긴 지연시간을 예방함으로써 멀티미디어 서비스의 QoS를 보장한다.

네트워크 혼잡의 징후를 사전에 감지하기 위해 DBRC는 지연시간을 이용한다. 지연시간은 송신자와 수신자 사이의 패킷 전송 시간을 의미하며 라우터의 패킷 처리능력과 밀접한 관계가 있다. 라우터의 패킷 처리능력은 라우터에 들어오는 패킷을 얼마나 빨리 처리할 수 있는가에 달려있다. 라우터의 패킷 처리능력보다 들어오는 패킷의 양이 많으면 지연시간이 증가하게 된다. 반대로 라우터의 패킷 처리 능력보다 들어오는 패킷의 양이 적다면 지연시간은 감소하게 된다. 따라서 지연시간은 그림 3에서와 같이 패킷의 병목지점이 되는 라우터의 입력 비트율과 출력 비트율에 따라 결정된다.

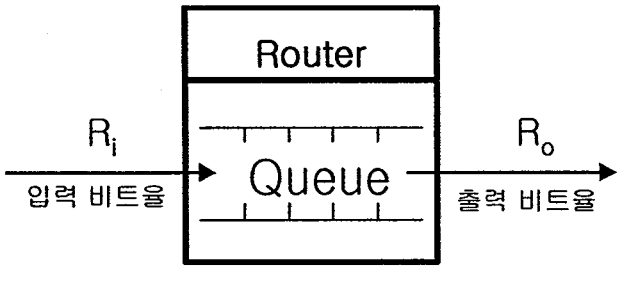


그림 3. 라우터에서의 패킷 처리

라우터의 입력 비트율이 출력 비트율보다 크다면 그림 4의 구간 a에서와 같이 지연시간이 증가하게 된다. 지연시간이 계속 증가하여 지연시간이 M(패킷 손실 시의 최대 지연시간)이 되면 라우터의 큐 길이가 최대가 된다. 이때 라우터는 이후에 들어오는 패킷을 버리게 되어 패킷 손실이 발생하게 된다. 따라서 구간 a에서는 패킷 손실을 방지하기위해 전송율을 감소시키는 것이 바람직하다.

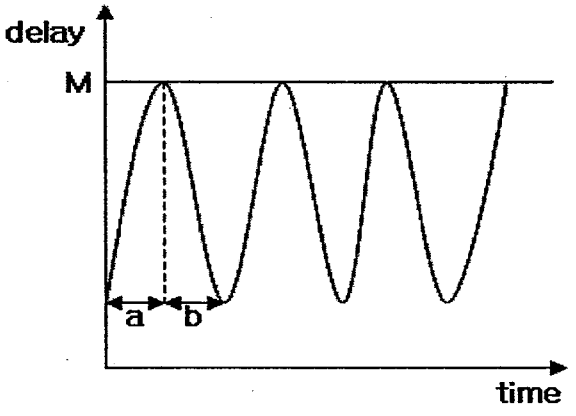


그림 4. 지연시간 변화 그래프

라우터의 입력 비트율이 출력 비트율보다 작다면 그림 4의 구간 b에서와 같이 지연시간이 감소하게 된다. 지연시간의 감소는 라우터의 전송 능력을 최대로 활용하지 못한다는 것이므로 이 때는 전송율을 증가시키는 것이 바람직하다.

지연시간의 크기는 라우터의 큐에 저장된 패킷의 수에 따라 결정된다. 라우터의 큐 길이가 최대에 근접했을 때는 지연시간이 크다. 만약 이 상태에서 전송율이 증가하면 라우터의 큐는 더 이상의 패킷을 받을 수 없게 된다. 이때 라우터는 들어오는 패킷을 폐기하게 되어 패킷 손실이 발생할 수 있으므로 전송율을 증가시키는 것은 바람직하지 않다. 다음으로 큐 길이가 최소에 근접했을 때는 지연시간이 작다. 이 때는

라우터의 전송능력을 최대로 이용하기위해 전송율을 증가시키는 것이 필요하다.

2. DBRC 설계 시 고려사항

DBRC는 전송율을 조절할 때 두 가지 원인에 기반을 두고 전송율을 결정한다. 첫째, 지연시간이 증가하거나 감소중일 때는 전송율을 일정 비율로 증가시키거나 감소시킨다. 이는 지연시간이 급격하게 증가하여 네트워크 혼잡에 의한 패킷 손실을 방지하고 지연시간이 급격하게 감소하여 네트워크 자원의 비효율적인 이용을 방지하기 위함이다. 둘째, 현재 지연시간에 기반해 전송율을 제어한다. 이때는 전송율의 증가/감소 속도를 조절하기위해 목표 지연시간과의 차이에 비례하여 전송율을 조절한다. 이렇게 함으로서 그림 5에서와 같이 지연시간이 DBRC가 목표로 하는 지연시간(α * M)에 수렴하도록 할 수 있다.

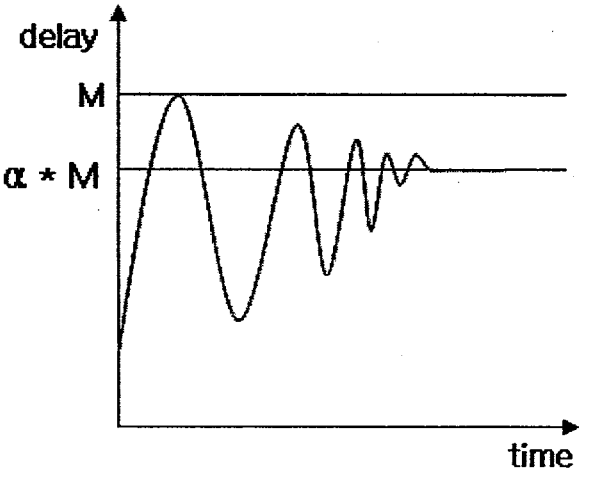


그림 5. DBRC의 목표 지연시간

3. DBRC의 목표

DBRC는 멀티미디어 서비스의 QoS를 보장하기 위해 패킷 손실의 최소화, 지연시간의 일정한 유지, 전송율 변화의 최소화에 목표를 두고 있다.

패킷 손실을 최소화하는 것은 TCP를 사용하는 웹 서비스나 FTP 서비스와는 달리 멀티미디어 서비스의 QoS를 보장하기 위해서 상당히 중요하다. TCP를 사용하는 서비스는 패킷 손실이 발생하면 재 전송을 통하여 손실된 패킷을 복구할 수 있다. 그러나 멀티미디어 서비스는 패킷이 도착해야 하는 시간인 데드라인이 존재하고 이 데드라인을 만족하지 못하면 패킷은 손실로 취급된다.

지연시간의 일정한 유지는 전송율의 변화와 관련이 있다. 지연시간의 큰 변화는 전송율이 크게 변하는 원인을 제공한다. 전송율의 큰 변화는 멀티미디어 서비스의 QoS 저하의 원인이 된다. 따라서 지연시간을 일정하게 유지시켜 줌으로써 QoS를 높일 수 있다.

전송율 변화의 최소화는 멀티미디어 서비스의 QoS에 상당한 영향을 미친다. 사용자가 스트리밍 서버로부터 영화를 실시간으로 감상한다고 했을 때 전송율의 변화가 심하면 화면의 변화가 심하게 되어 사용자가 눈의 피로함을 느끼게 된다. 따라서 다소 화질은 떨어지더라도 일정한 화질을 유지하는 것이 멀티미디어 서비스를 위해 바람직하다.

제 4 절 DBRC 알고리즘

1. 전송율 제어

DBRC는 지연시간에 기반해 전송율을 제어하기 위해 설계된 알고리

즘이다. DBRC는 기본적으로 지연시간이 증가하면 전송율을 감소시키고 지연시간이 감소하면 전송율을 증가시킨다. 여기에 추가적으로 현재 지연시간에 따라 전송율의 변화량을 조정한다. 그림 6은 지연시간에 따른 전송율의 변화를 보여준다. 그림 6에서 지연시간이 감소상태에 있다고 했을 때, 현재 지연시간이 DBRC가 목표로 하는 지연시간(M*α)보다 크면 전송율의 증가량이 작아지고 현재 지연시간이 DBRC가 목표로 하는 지연시간 보다 작으면 전송율의 증가량이 많아진다. 반대로 지연시간이 증가 상태에 있다고 한다면, 현재 지연시간이 DBRC가 목표로 하는 지연시간 보다 크면 전송율의 감소량이 많아지고 현재 지연시간이 DBRC가 목표로 하는 지연시간보다 작으면 전송율의 감소량이 작아진다.

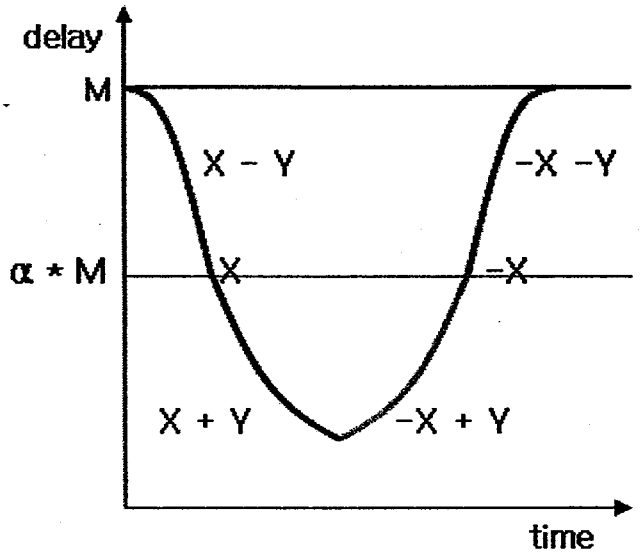


그림 6. 지연시간에 따른 전송율 변화

위에서 설명한 DBRC의 두 가지 측면에서의 전송율 변화를 수식으로 표현하면 다음과 같다.

첫째, 지연시간 증가/감소 시 전송율의 변화는 아래 수식으로 표현 할

수 있다.

$$R_{i+1} = R_i * X, \text{ where } X = c * R_i * \beta$$

R_i는 epoch i에서의 전송율을 나타내고, c는 지연시간이 증가하고 있는지 감소하고 있는지를 나타내는 상수로서 c 값은 지연시간 증가, 지연시간 감소, 지연시간 일정에 따라 각각 -1, 1, 0 값을 가진다. β는 전송율 변화 비율 파라미터이다. 위의 수식은 다음의 세 가지 경우로 해석이 된다. 지연시간이 증가하면(c=-1) 전송율을 $R_i * \beta$ 만큼 증가시킨다. 지연시간이 감소하면(c=1) 전송율을 $R_i * \beta$ 만큼 감소시킨다. 지연시간이 변화가 없으면(c=0) 전송율을 변화시키지 않는다.

둘째, 현재 지연시간에 따른 전송율 변화는 아래의 수식과 같이 표현할 수 있다.

$$R_{i+1} = R_i * Y, \text{ where } Y = \varepsilon * R_i * \beta$$

$$\varepsilon = \frac{M * \alpha - D_i}{M * \alpha}$$

$(-1 <= \varepsilon <= 1, \text{ when } 0 <= D_i <= 2 * \alpha * M)$

R_i는 epoch i에서의 전송율을 나타내고, M은 패킷 손실 시 최대 지연시간을 나타낸다. α는 목표 지연시간을 위한 상수이다. D_i는 epoch i에서의 평균 지연시간이다. ε는 목표 지연시간과 현재 지연시간에 따른 전송율 변화량을 위한 상수 이다.

위 수식은 다음의 세 가지 경우로 해석할 수 있다. 현재 지연시간이 DBRC의 목표 지연시간보다 크면 전송율을 $R_i * \beta * \varepsilon$ 만큼 감소시킨다. 현재 지연시간이 목표 지연시간보다 작다면 전송율을 $R_i * \beta * \varepsilon$ 만큼 증가시킨다. 현재 지연시간이 목표 지연시간과 같다면 전송율을 변화시키지 않는다.

2. 전송율 제어 수식

위의 전송율 변화에서 지연시간의 증가/감소에 따른 전송율 변화 수식과 현재 지연시간에 따른 전송율 변화 수식을 하나의 수식으로 표현하면 아래와 같다.

$$R_{i+1} = R_i * X * Y$$
$$= R_i + (c + \varepsilon) * R_i * \beta$$

위의 전송율 제어 수식의 특징은 아래와 같다.

지연시간 감소 시

$0 <= \text{전송율 변화량} <= 2 * R_i * \beta$

지연시간 증가 시

$-2 * R_i * \beta <= \text{전송율 변화량} <= 0$

지연시간 변화 없음

$-R_i * \beta <= \text{전송율 변화량} <= R_i * \beta$

지연시간이 감소할 때는 전송율을 최소 0에서 최대 $2 * R_i * \beta$ 만큼

증가시킨다. 지연시간이 증가할 때는 전송율을 최소 0에서 최대 2 * R_i * β 만큼 감소시킨다. 지연시간이 변화가 없을 때는 전송율을 최소 R_i * β 만큼 감소시키거나 최대 R_i * β 만큼 증가시킨다. 이러한 전송율 변화량을 통해 DBRC는 빠른 시간 내에 지연시간을 DBRC가 목표로 하는 지연시간으로 수렴하도록 한다. 그림 7은 DBRC 알고리즘의 의사 코드를 보여준다.

```
if (F_j > 0) /* 패킷 손실이 발생했을 때 */
  R_i <- R_{i-1} * (1 - γ)
else if (D_j > M) /* 현재 지연시간이 패킷 손실 시 최대 지연시간
                     보다 크면 패킷 손실과 동일하게 취급 */
  R_i <- R_{i-1} * (1 - γ)
else
  if (c == -1) /* 지연시간 증가 */
    R_i <- R_{i-1} + R_{i-1} * (-1 + ε) * β
  else if (c == 1) /* 지연시간 감소 */
    R_i <- R_{i-1} + R_{i-1} * (1 + ε) * β
  else if (c == 0) /* 지연시간 일정 */
    R_i <- R_{i-1} + R_{i-1} * ε * β
  else
    R_i <- R_{i-1}
```

그림 7. DBRC 알고리즘 의사 코드

3. 지연시간 증가/감소(c) 결정 알고리즘

DBRC는 지연시간에 기반해 전송율을 제어하기위해 설계되었다. 따라서 지연시간이 현재 증가하고 있는지, 감소하고 있는지, 변화가 없는지를 정확하게 판단을 해야 정확한 전송율 제어를 수행할 수 있다.

지연시간의 변화 상태를 정확하게 판단하기위해 DBRC는 이전의 지연시간들의 변화를 이용한다. 만약 이전의 지연시간이 증가 상태이고 현재의 지연시간이 이전 지연시간 + 표준 편차보다 크다면 현재의 지연시간은 증가 상태가 된다. 따라서 c = -1(빠른 전송율 감소)이 된다. 이전의 지연시간이 증가 상태이고 현재의 지연시간이 지연시간 - 표준 편차보다 작다면 현재 지연시간이 증가하고 있는지 감소하고 있는지 분명하지 않다. 따라서 c = - ε (전송율을 변화 시키지 않음)가 된다. 이전의 지연시간이 증가 상태이고 현재의 지연시간이 지연시간 + 표준 편차 내에 있다면 현재 지연시간은 변화 없음이 된다. 따라서 c = 0(전송율의 느린 증가)이 된다.

그림 8은 DBRC 알고리즘의 지연시간 증가/감소 결정 알고리즘 의사 코드이다.

```
if (D_i > D_{i-1} + δ)
  if (D_i > D_{i-2} + δ)
    c = -1 /* 지연시간 증가 */
  else
    c = - ε /* 지연시간 증가가 확실하지 않음 */
else if (D_i > D_{i-1} - δ)
  if (D_i > D_{i-2} - δ)
    c = -1 /* 지연시간 감소 */
  else
    c = - ε /* 지연시간 감소가 확실하지 않음 */
else
  c = 0 /* 지연시간 변화 없음 */
```

그림 8. 지연시간 증가/감소 결정 알고리즘 의사 코드

제 4 장 성능 분석

제 1 절 실험 설계

1. 실험 방법

본 논문에서는 DBRC의 성능을 테스트 하기위해 네트워크 시뮬레이션 도구인 NS-2 시뮬레이터를 사용하였다. NS-2는 TCP, 라우팅 프로토콜, 멀티캐스트 프로토콜, RTP(Real Time Protocol), SRM(Scalable Reliable Multicast) 등 다양한 인터넷 프로토콜에 대한 시뮬레이션을 수행하기에 적절한 여러 환경을 제공하고 있어 현재 널리 사용되고 있는 네트워크 시뮬레이션 도구이다.

본 논문에서는 DBRC와 TFRC의 전송율, 패킷 손실율, 지연시간 등을 측정하여 DBRC의 성능을 확인하였다. DBRC의 성능의 정확성을 위해 다음과 같은 여러 네트워크 환경에서 DBRC와 TFRC를 비교하였다.

Case 1 : Simple One-hop Link Topology
Case 2 : Multi-hop Link Topology
Case 3 : Complex Multiple-hop Link Topology

2. DBRC 파라미터 값 설정

DBRC의 성능 테스트를 위해 DBRC의 목표 지연시간을 결정하기 위한 상수인 α는 0.8로 설정하였고 전송율 변화 비율 파라미터인 β는 0.05로 설정하였다. 패킷 손실 시 빠른 전송율 감소를 위한 상수인 γ는

0.8로 설정하였다. DBRC는 전송율은 60KByte/s로 시작을 하고 전송율 제어를 위한 주기(epoch)는 1초로 설정 하였다.

3. Case 1 : Simple One-hop Link Topology

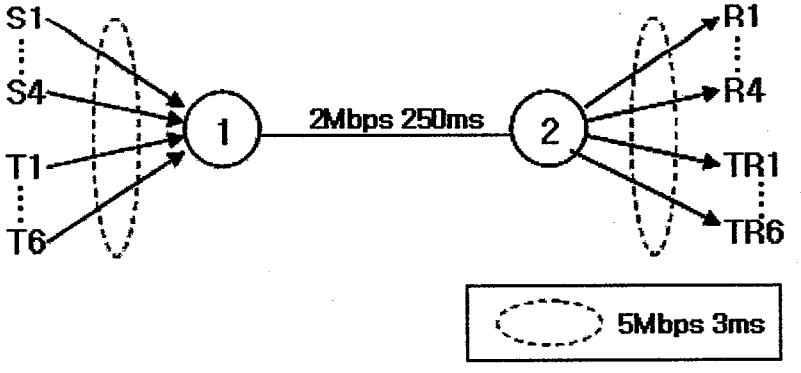


그림 9. Simple One-hop Link Topology

그림 9는 Simple One-hop Link Topology를 보여준다. S1~S4는 멀티미디어 스트림(DBRC, TFRC)을 전송하는 송신자를 나타내고 R1~R4는 송신자가 전송한 멀티미디어 스트림을 수신하는 수신자를 나타낸다. T1~T6은 TCP 패킷을 전송하는 송신자를 나타내고 TR1~TR6는 TCP 패킷을 수신하는 수신자를 나타낸다. 1과 2는 수신한 패킷을 적절한 노드로 전송하는 라우터를 나타낸다. 실험 방법은 S1~S4가 0.1초 간격으로 시작한다. 동시에 T1~T6이 0.1초 간격으로 시작한다. 300초 후에 S1~S2를 링크에서 제거한다.

4. Case 2 - Multi-hop Link Topology

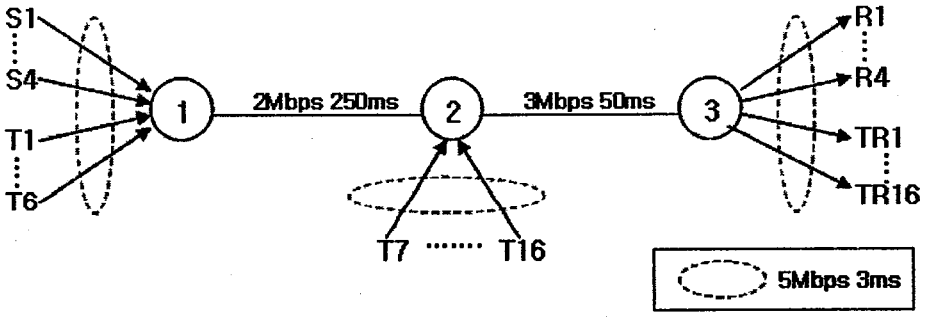


그림 10. Multi-hop Link Topology

그림 10은 Multi-hop Link Topology를 보여준다. 실험 방법은 S1~S4가 0.1초 간격으로 시작한다. 동시에 T1~T6이 0.1초 간격으로 시작한다. 200초 후에 T7~T16이 링크에 연결되어 2와 3사이에 병목현상 구간이 발생한다. 300초 후에 S1~S2를 링크에서 제거한다.

5. Case 3 - Complex Multiple-hop Link Topology

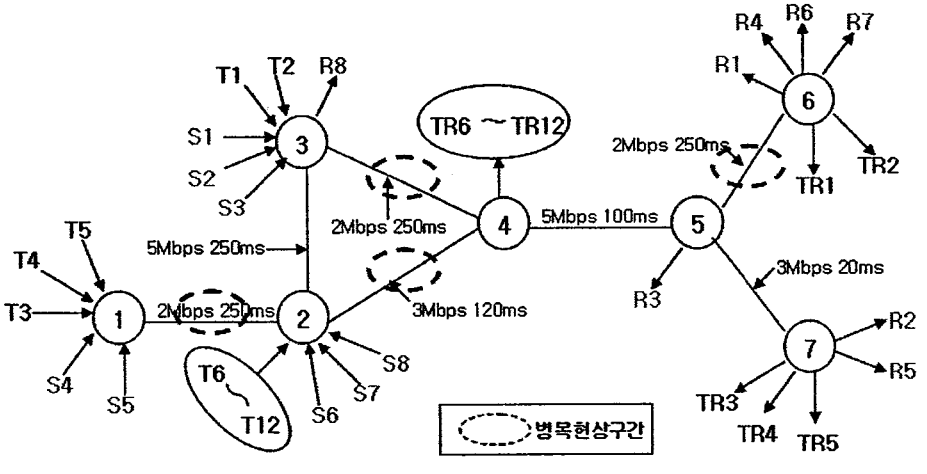


그림 11. Complex Multiple-hop Link Topology

그림 11은 Complex Multiple-hop Link Topology를 보여준다. 이번 실험에서는 실제 인터넷 환경과 유사한 환경에서 DBRC와 TFRC의 패킷 손실, 지연시간 변화, 전송율의 변화를 측정한다.

제 2 절 실험 결과

1. Case 1 : Simple One-hop Link Topology

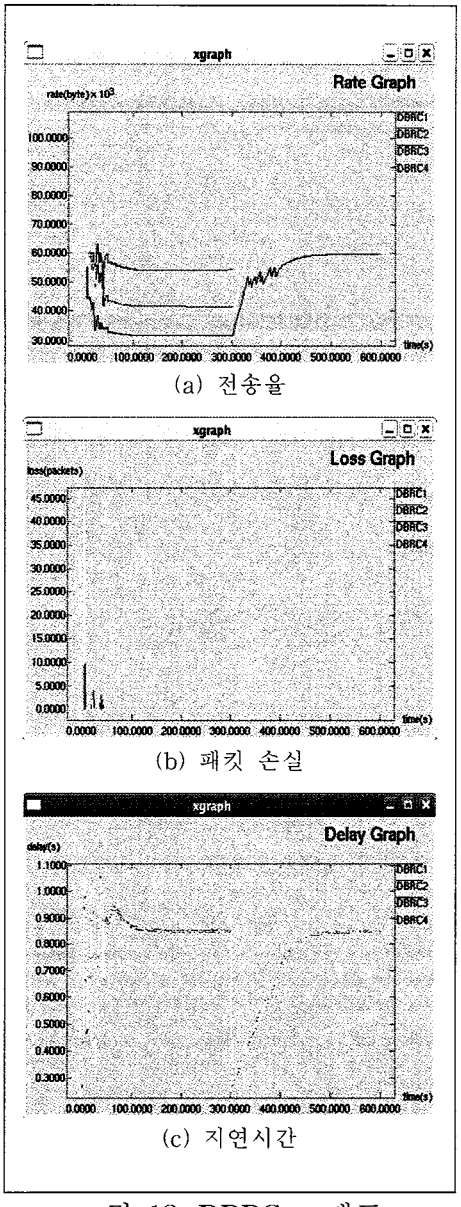


그림 12. DBRC 그래프

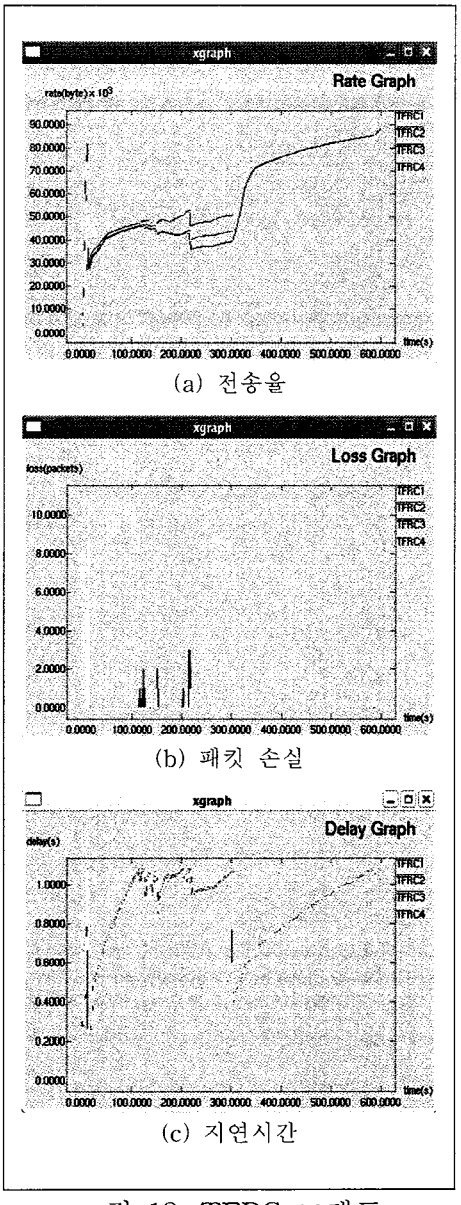


그림 13. TFRC 그래프

그림 12의 (a)와 그림 13의 (a)는 각각 DBRC와 TFRC의 전송율 시뮬레이션 결과를 보여준다. 시뮬레이션 결과는 300초 이전과 300초 이후로 나누어 분석할 수 있다.

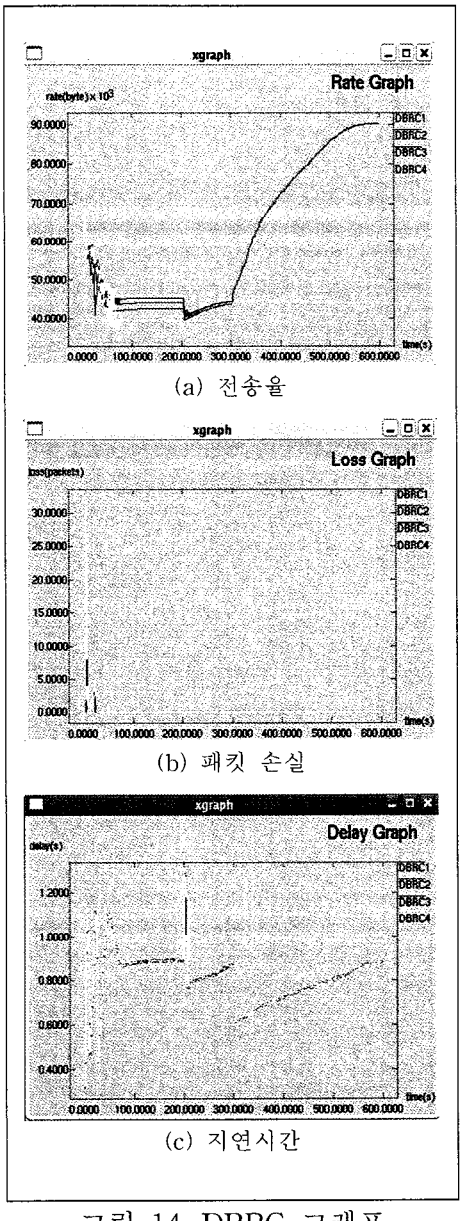
0~300초 사이의 DBRC는 70초 이전에 전송율이 일정하게 유지되는 것을 볼 수 있다. 반면 TFRC는 300초 이전까지 전송율의 변화가 계속되는 것을 볼 수 있다. TFRC의 전송율의 변화가 심한 것은 TFRC는 패킷 손실이 없으면 계속해서 전송율을 증가시켜 지속적으로 패킷 손실을 경험하기 때문이다.

300초 이후에는 1, 2번 스트림이 링크에서 제거된다. 따라서 300초 이전까지 일정한 전송율을 유지하던 DBRC와 TFRC 스트림 3, 4번이 빠르게 전송율이 증가되는 것을 볼 수 있다. 3, 4번 스트림이 빠르게 증가하는 이유는 1, 2번 스트림이 300초에 링크에서 제거되어 1, 2번 스트림이 사용하던 대역폭을 3, 4번 스트림이 확보하기위해 전송율을 증가시키기 때문이다. DBRC의 전송율 증가는 300초에서 빠르게 증가하여 400초 이전에 일정한 전송율을 유지하고 있다. 그러나 TFRC는 전송율의 변화가 계속되는 것을 확인할 수 있다.

그림 12의 (b)와 그림 13의 (b)는 각각 DBRC와 TFRC의 패킷 손실율 시뮬레이션 결과를 보여준다. 그림 12의 (b)에서 DBRC는 초기에 패킷 손실을 경험한 이후 패킷 손실이 발생하지 않는 것을 볼 수 있다. DBRC가 초기에 패킷 손실을 경험하는 이유는 DBRC는 초기에 멀티미디어 서비스가 요구하는 최대 전송율(실험에서는 각 스트림 당 60Kbyte/s)로 시작하므로 라우터 1과 2사이의 대역폭 2Mbps를 초과하기 때문이다. 그림 13의 (b)에서 TFRC는 초기 패킷 손실을 경험한 이후 계속해서 패킷 손실이 발생하고 있다. 이는 TFRC는 패킷 손실이 없을 시 계속해서 전송율을 증가시키기 때문이다.

그림 12의 (c)와 그림 13의 (c)는 각각 DBRC와 TFRC의 지연시간 시뮬레이션 결과를 보여준다. 그림 12의 (c)에서 DBRC는 80초 이후에 지연시간이 0.86초로 수렴하고 있는데, 그 이유는 DBRC는 지연시간이 패킷 손실 시 최대 지연시간(M)의 α%가 되도록 설계되어져 있기 때문이다. 따라서 목표 지연시간(M * α) = 1.08 * 0. 8 = 0.86이 된다. 300초에 스트림 1, 2가 링크에서 제거되어 지연시간이 급격히 떨어지자 3, 4번 스트림은 전송율을 증가시켜 지연시간이 DBRC의 목표 지연시간으로 수렴하도록 하고 있다. 그림 13의 (c)에서 TFRC는 지연시간이 계속해서 변하고 있음을 볼 수 있다. 그 이유는 초기 패킷 손실을 경험한 후에도 패킷 손실이 발생하지 않는 한 계속해서 전송율을 증가시키기 때문이다. 그림 16에서와 같이 TFRC는 패킷 손실이 지속적으로 발생하기 때문에 계속해서 지연시간의 증가와 감소가 반복된다.

2. Case 2 : Multi-hop Link Topology

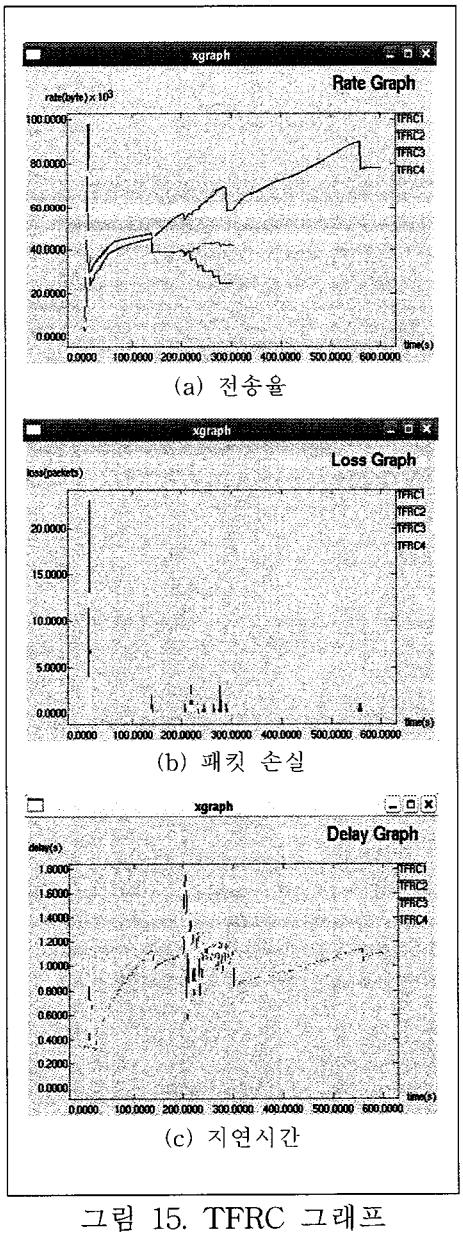


(a) 전송율

(b) 패킷 손실

(c) 지연시간

그림 14. DBRC 그래프



(a) 전송율

(b) 패킷 손실

(c) 지연시간

그림 15. TFRC 그래프

그림 14의 (a)와 그림 15의 (a)는 각각 DBRC와 TFRC의 전송율 시뮬레이션 결과를 보여준다. 그림 14의 (a)에서 DBRC는 80초 이전에

전송율이 일정하게 유지되고 있다. 200초에 전송율이 감소하는 이유는 라우터 2와 라우터 3사이에 10개의 새로운 TCP 플로우가 추가되었기 때문이다. 새로운 플로우 추가에 따라 지연시간이 증가하고 DBRC는 목표 지연시간을 유지하기 위해 전송율을 감소시킨다. 300초에 스트림 1, 2가 링크에서 제거된 후 3, 4번 스트림은 1, 2번의 대역폭을 확보하기 위해 전송율을 빠르게 증가시킨다. 400초 이후에 DBRC의 전송율이 일정하게 유지되는 것을 볼 수 있다. 그림 15의 (a)에서 TFRC의 경우 170초까지 전송율이 꾸준히 증가하다 스트림 4번이 패킷 손실로 전송율이 감소하고 있다. 200초 이후에 1번과 3번 스트림의 전송율이 크게 차이가 나는데 그 이유는 1번 스트림은 패킷 손실을 경험한데 반해 3번은 패킷 손실을 경험하지 않았기 때문이다. TFRC는 이후에도 전송율이 안정되지 못하고 계속해서 변하는 것을 볼 수 있다.

그림 14의 (b)와 그림 15의 (b)는 각각 DBRC와 TFRC의 패킷 손실율 시뮬레이션 결과를 보여준다. 그림 14의 (b)에서 DBRC는 초기 패킷 손실을 경험한 이후 추가적인 패킷 손실이 발생하지 않았다. 200초에 10개의 새로운 TCP 플로우가 추가 되었는데도 패킷 손실이 발생하지 않은 이유는 DBRC가 목표 지연시간을 유지하고 있어 새로운 플로우의 추가에 유동적으로 대처할 수 있기 때문이다. 그림 15의 (b)에서 TFRC는 200초에 새로운 플로우 추가 이후 많은 패킷 손실을 경험하고 있다. 그 이유는 TFRC가 네트워크 자원을 최대로 이용하려고 하기 때문이다. 따라서 TFRC는 지연시간이 패킷 손실 시의 최대 지연시간에 가깝게 유지된다. 결국 TFRC는 새로운 플로우에 유동적으로 대처하지 못하고 많은 패킷 손실을 경험하게 된다.

그림 14의 (c)와 그림 15의 (c)는 각각 DBRC와 TFRC의 지연시간 시뮬레이션 결과를 보여준다. 그림 14의 (c)에서 DBRC는 80초 이후에 목표 지연시간을 유지하다 200초에 지연시간이 큰 폭으로 증가하였다. 200초에서 지연시간이 패킷 손실 시 최대 지연시간보다 크다는 것을

알 수 있다. 그런데 그림 14의 (b)에서 DBRC는 200초 이후에 패킷 손실이 발생하지 않았다. 그 이유는 첫째, 라우터 2와 라우터 3 사이의 대역폭이 라우터 1과 라우터 2 사이의 대역폭보다 크기 때문이다. 둘째, DBRC 지연시간이 패킷 손실 시 최대 지연시간보다 커지면 패킷 손실을 줄이기 위해 패킷 손실 시와 같이 전송율을 빠르게 감소시키기 때문이다. 셋째, DBRC는 지연시간이 패킷 손실 시 최대 지연시간(M)의 α%가 되도록 설계되어져 있기 때문이다. 따라서 M * (1 - α)%의 지연시간의 여유가 있어 갑작스러운 플로우 유입에 유동적으로 대처할 수 있다. 300초 이후 1, 2번 스트림의 제거로 인해 지연시간이 감소하고 3, 4번 스트림은 전송율을 증가시켜 지연시간을 DBRC의 목표 지연시간으로 수렴시키고 있다. 그림 15의 (c)에서 TFRC는 200초에서 300초까지 큰 지연시간의 변화를 보이고 있다. TFRC는 지연시간이 크게 증가함에도 불구하고 패킷 손실이 발생하지 않으면 전송율을 계속해서 증가시키기 때문에 200초에서 300초 사이에 많은 패킷 손실이 발생하는 것을 볼 수 있다.

3. Case 3 : Complex Multi-hop Link Topology

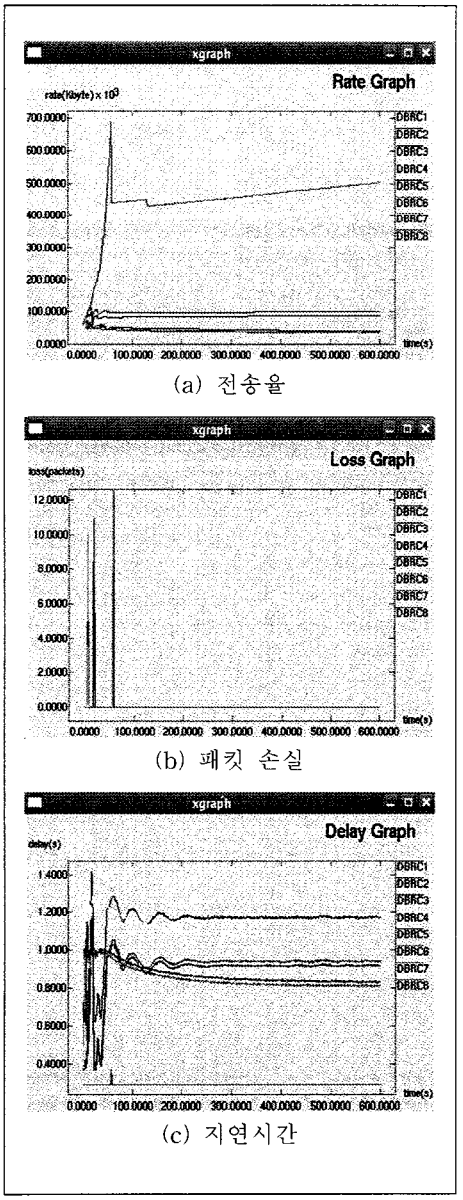


그림 16. DBRC 그래프

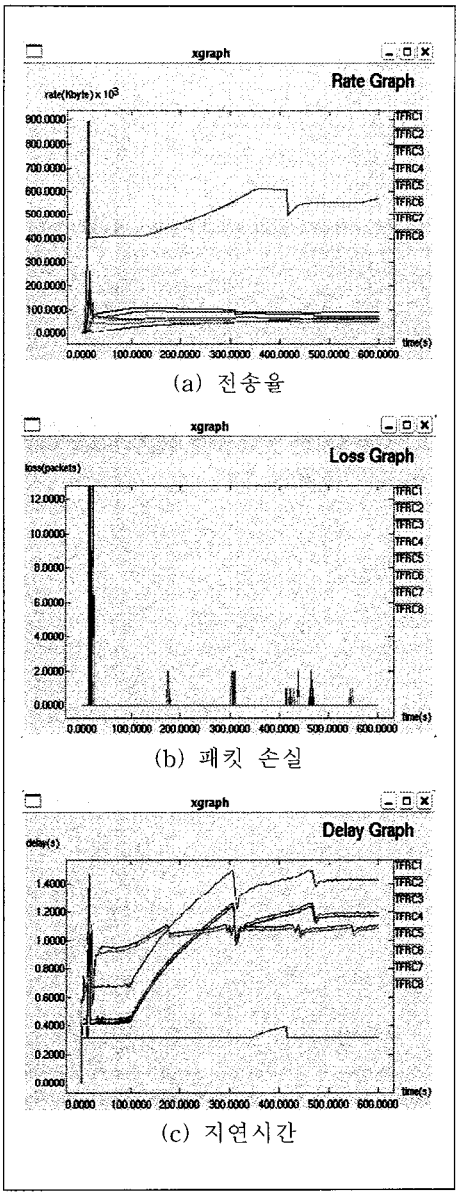


그림 17. TFRC 그래프

그림 16의 (a)와 그림 17의 (a)는 각각 DBRC와 TFRC의 전송율 시뮬레이션 결과를 보여준다. 그림 16의 (a)와 그림 17의 (a)에서 DBRC

와 TFRC 양쪽 모두 일정 시간 후 일정한 전송율을 유지하는 것을 볼 수 있다. 특별히 DBRC의 경우 TFRC 보다 빠른 시간내에 일정한 전송율을 유지하고 전송율의 변화도 거의 없다는 것을 알 수 있다. 스트림 8의 전송율이 다른 스트림에 비해 월등히 높은 이유는 스트림 8이 라우터 2와 라우터 3 사이의 링크를 단독으로 사용하고 있기 때문이다. TFRC의 스트림 8도 같은 이유이다. 또한 두 스트림 모두 초반에 패킷 손실이 발생할 때까지 빠르게 전송율을 증가시키는 것을 볼 수 있다. 일단 패킷 손실을 경험한 후 라우터 2와 라우터 3사이의 대역폭 5Mbps에 근접한 4Mbps의 전송율을 유지하는 것을 볼때 DBRC와 TFRC의 8번 스트림 모두 링크 대역폭을 효율적으로 이용한다고 볼 수 있다. 그러나 TFRC는 DBRC에 비해 전송율의 변화가 심한 것을 볼 수 있다.

그림 16의 (b)와 그림 17의 (b)는 각각 DBRC와 TFRC의 패킷 손실율 시뮬레이션 결과를 보여준다. 그림 16의 (b)에서 DBRC는 초기 패킷 손실이후 스트림 8이 패킷 손실을 경험하는데 이는 DBRC가 초기 패킷 손실 시 최대 지연시간에 기반해 전송율 제어를 수행하기 때문이다. 스트림 8의 경우 라우터2와 라우터 3사이의 대역폭 5Mbps를 단독으로 사용하므로 다른 스트림과는 달리 초기 패킷 손실을 늦게 경험하고 있다. 60초 이후에는 모든 스트림이 패킷 손실을 경험하지 않고 있는 것을 볼 수 있다. 그림 17의 (b)에서 TFRC는 초기 패킷 손실을 경험한 이후 계속해서 패킷 손실이 발생하고 있는데, 이는 TFRC가 이용가능한 대역폭을 최대한 사용하기 위해 패킷 손실이 발생할 때까지 계속해서 전송율을 증가시키기 때문이다.

그림 16의 (c)와 그림 17의 (c)는 각각 DBRC와 TFRC의 지연시간 시뮬레이션 결과를 보여준다. 그림 16의 (c)에서 스트림마다 DBRC의 목표 지연시간이 다른 이유는 첫째, 라우터 1과 라우터 7사이에 여러 지점의 병목 구간이 존재하고 둘째, 각 링크마다 지연시간이 다르고 셋

째, 각 스트림의 송신자와 수신자의 전송 경로가 Case 1과 Case 2와는 다르기 때문이다. 스트림 8번의 경우는 라우터 3과 라우터 4사이의 전송 대역폭이 크고 스트림 8번이 단독으로 사용하기 때문에 지연시간의 변화가 거의 없는 것을 알 수 있다. 그림 17의 (c)에서 TFRC의 경우 8번을 제외하고 지연시간의 증가/감소가 주기적으로 반복되는 것을 볼 수 있다. 그 이유는 TFRC는 패킷 손실이 발생하지 않는 한 전송율을 계속 증가시킴으로써 네트워크 혼잡으로 인한 패킷 손실이 발생하기 때문이다.

제 5 장 결론 및 향후 연구

오늘날 실시간 방송, 원격 화상회의, VoIP와 같은 실시간성을 요구하는 멀티미디어 서비스가 빠르게 증가하고 있다. 멀티미디어 서비스는 TCP를 사용하는 웹 서비스나 FTP 서비스와는 다른 특징을 가지고 있다. 첫째, 제한된 시간 안에 패킷이 전송되어야 한다. 둘째, 패킷 손실에 민감하다. 셋째, 전송율의 변화가 심해서는 안 된다. 멀티미디어 서비스의 이러한 특징 때문에 멀티미디어 스트림의 전송 프로토콜로 TCP를 사용하지 않고 UDP를 사용한다. 그러나 UDP는 네트워크 혼잡 제어를 수행하지 않으므로 기존의 TCP를 사용하는 다른 서비스들과의 공정한 자원 이용과 네트워크 혼잡이 발생했을 때 이를 해결하지 못하는 문제가 발생한다. 멀티미디어 서비스의 이러한 문제를 해결하기 위해 TFRC 알고리즘이 제안되었다. TFRC는 네트워크 혼잡을 빠르게 해결하고, AIMD를 사용하는 TCP와는 달리 부드러운 전송율의 변화를 가진다. 그러나 TFRC는 패킷 손실이 발생하지 않는 한 계속해서 전송율을 증가시키므로 지속적인 패킷 손실을 경험하게 된다. 위에서 언급했듯이 멀티미디어 서비스는 패킷 손실에 민감하므로 지속적인 패킷 손실은 멀티미디어 서비스의 QoS 저하의 원인이 된다. 따라서 멀티미디어 서비스의 QoS를 개선하기 위해서는 네트워크 혼잡에 의한 패킷 손실이 발생하기 전에 데이터의 전송율을 제어하기 위한 알고리즘이 요구된다.

본 논문에서는 패킷 손실을 줄이기 위해 패킷 손실의 원인을 분석하여 네트워크의 혼잡으로 인한 패킷 손실이 발생하기 전에 전송율을 제어함으로서 패킷 손실을 줄이기 위한 DBRC 알고리즘을 설계하였다.

본 논문에서 제안한 DBRC 알고리즘은 기존의 멀티미디어 서비스를 위해 제안된 TFRC 알고리즘보다 멀티미디어 QoS를 향상시키는데 다

음과 같은 좋은 결과를 보였다. 첫째, DBRC는 초기의 패킷 손실을 경험한 이후 추가적인 패킷 손실이 발생하지 않은데 반해 TFRC는 지속적인 패킷 손실이 발생하였다. 둘째, DBRC는 빠른 시간 내에 안정된 전송율을 유지한 반면 TFRC는 지속적인 전송율 증가와 패킷 손실로 인해 전송율의 변화가 심했다. 셋째, DBRC는 지연시간을 빠르게 목표 지연시간으로 유지함으로써 지연시간이 멀티미디어 서비스가 요구하는 데드라인을 만족시켰다. 그러나 TFRC는 지속적인 전송율의 증가에 따른 지연시간의 증가 때문에 데드라인을 제대로 만족시키지 못했다.

추가적으로 지연시간과 패킷 손실은 서로 관련이 있다는 사실을 본 논문의 실험과 기존의 논문들을 통해 확인할 수 있다 [17-21]. TFRC 실험결과에서 볼 수 있듯이 지연시간이 계속해서 증가하면 패킷 손실이 계속해서 발생하고 전송율의 변화도 크게 변한다는 것을 알 수 있었다. 그러나 DBRC의 경우 지연시간을 목표 지연시간으로 유지함으로써 추가적인 패킷 손실이 발생하지 않았고 전송율도 거의 변하지 않는 것을 볼 수 있었다. 이러한 사실은 지연시간에 기반해 전송율을 제어하는 DBRC 알고리즘이 멀티미디어 서비스의 QoS 향상에 크게 기여한다는 것을 증명한다.

향후 연구에서는 DBRC 알고리즘의 수학적인 분석을 통해 알고리즘 성능의 이론적인 증명과 실제 프로토타입 시스템에서 성능 분석을 수행할 계획이다.

참 고 문 헌

[1] O.Festor and A.Pras, "A Methodology for Performance Analysis of Real-Time. Continuous Media Applications," 12th International Workshop on Distributed. Systems: Operations and Management DSOM'2001 Nancy France, October, 15-17, 2001.

[2] W. Jiang and H. Schulzrinne , "QoS Measurement of Internet Real-Time Multimedia Services," tech. report CUCS-015-99m, Columbia Univ., New York, Dec, 1999.

[3] I. Busse, B. Deffner, and Henning Schulzrinne, "Dynamic QoS Control of Multimedia Applications based on RTP," Computer Communications, vol. 19, no. 1, pp. 49-58, January 1996.

[4] J. Shin, J. Kim, and C.-C. Jay Kuo, "Quality-of-service mapping mechanism for packet video in differentiated services network," IEEE Trans. Multimedia, vol. 3, pp. 219-231, June 2001.

[5] M. Handley, S. Floyd, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," Internet proposed standard RFC 3448, January 2003.

[6] A. Lahanas and V. Tsaoussidis, "Additive Increase Multiplicative Decrease - Fast Convergence (AIMD-FC)," In Proccedings of Networks 2002.

[7] V. Dumas, F. Guillemin and P. Robert, "A Markovian Analysis of Additive- Increase Multiplicative-Decrease (AIMD) Algorithms," Advances in Applied Probability, Vol. 34, no. 1, 2002.

[8] D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithm for Congestion Avoidance. in Computer Networks," Computer Networks and ISDN Systems, 17:1-14, 1989.

[9] Y. Yang and S. Lam, "General AIMD congestion control," Tech. Rep. TR-200009, University of Texas at Austin, May 2000.

[10] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "Equation-Based Congestion Control for Unicast Applications: the Extended Version," International Computer Science Institute tech report TR-00-003, March 2000.

[11] J. Mahdavi, S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control," Technical note sent to the end2end-interest mailing list, January 8, 1999.

[12] Jitendra Padhye, Jim Kurose, "Don Towsley and Rajeev Koodli, A TCP-Friendly Rate Adjustment Protocol for Continuous Media Flows over Best Effort Networks," UMass-CMPSCI Technical Report TR 98-04, October 1998.

[13] Floyd, S., and Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet," Under submission,

February 1998.

[14] V. Jacobson, "Congestion avoidance and control," Proc. SIGCOMM'88, ACM.

[15] K. Fall S. Floyd. "Promoting the Use of End-to-End Congestion Control in the Internet," IEEE/ACM. Transactions on Networking, August 1999.

[16] Reza Rejaie, Mark Handley, and Deborah Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," To appear in IEEE INFOCOMM 1999.

[17] J. Bolot, "Characterizing end-to-end packet delay and loss in the Internet," J. High-Speed Networks, vol. 2, no. 3, pp. 289-298, Dec. 1993.

[18] J.-C. Bolot, "End-to-end packet delay and loss behaviour in the Internet," in Proc. SIG-. COMM '93, Sept. 1993.

[19] TJ Kostas, MS Borella, and I. Sidhu, "Analysis of Internet Delay and Loss Measurements," Proceedings, Networld+Interop '99 Engineer's Conference, May 1999.

[20] M. Baldi and Y. Ofek, "End-to-end Delay Analysis of Videoconferencing over Packet Switched Networks," in IEEE INFOCOM 1998.

[21] S. Moon, J. Kurose, P. Skelly and D. Towsley, "Correlation of Packet Delay and Loss in the Internet," Technical Report University of Massachussets at Amherst,1999.

ABSTRACT

Study on Delay-based Rate Control Algorithm for Multimedia Streams

Song, Yong-Hon
Major in Information System Engineering
Dept. of Information System Engineering
Graduate School of Hansung University

Recently, the study to improve the QoS(Quality of Service) of multimedia service has been done actively along with the increase of multimedia service. TFRC(TCP-Friendly Rate Control) is now noted for the congestion control algorithm for multimedia service. But TFRC has continuous loss of packets as it performs the congestion control after packet loss. This causes low QoS of multimedia service. Therefore the algorithm to control the transmission rate before packet loss occurs is required.

The packet loss occurs mostly due to the network congestion. The network congestion happens when the number of packets incoming into router is bigger than that of packets which router can process. Thus, delay rapidly increases before packet loss. This thesis adjusts the transmission rate based on delay experienced by a receiver. Specifically, DBRC(Delay Based Rate Control) uses the

current delay and information about whether delay increases or not.

To test the performance of DBRC, NS-2 simulation tool was used for measuring the packet loss, delay, variance of transmission rate of DBRC and TFRC respectively. The testing result shows that DBRC had less packet loss and stable transmission rate and delay variation than TFRC.