

# 멀티미디어 데이터를 위한 TCP-friendly Rate Control Protocol

나승구, 김용건  
삼성전자 네트워크 사업부  
{sgnamc, ykkim63}@samsung.com

## TCP-friendly Rate Control Protocol for Multimedia data

Seung-Gu Na, Yong-Keon Kim  
Telecommunication Network, Samsung Electronics Co., Ltd

### 요약

최근 TCP와 유사하게 멀티미디어 데이터를 전송하기 위한 rate control 프로토콜에 관한 연구가 활발하게 진행되고 있다. 본 논문에서는 멀티미디어 전송 방식에 있어서 TCP와 공정성을 유지하며 TCP와 유사하게 동작하도록 하는 TRCP 프로토콜을 제안한다. TRCP는 TCP vegas 이론을 응용하였으며 RTT와 패킷손실율에 의해 네트워크 혼잡 상태를 미리 예측하고 TCP의 AIMD 방식을 사용하여 TCP와 유사하게 전송율을 조절하는 프로토콜이다. 이 프로토콜에 대한 TCP와 공정성을 검증하기 위하여 시뮬레이션을 실시하고 그 결과를 분석한다.

### 1. 서론

인터넷에서 음성이나 영상과 같은 실시간성의 멀티미디어 데이터가 많은 용용 프로그램에서 이용되고 있고 급속도로 증가하고 있는 추세에 있다. 파일전송과 같은 대용량의 데이터 전송과는 달리 멀티미디어 데이터는 전송 지연, 패킷 손실, 네트워크 대역폭의 변화 등에 매우 민감한 특성을 지니고 있다. 그러므로 용용프로그램에서 고수준의 QoS(Quality Of Service)를 달성하기 위하여 데이터의 전송율을 조절하는 것이 매우 중요하다.

데이터의 전송지연이나 패킷 손실 등은 네트워크 혼잡에 의하여 발생되는데 만약 혼잡제어를 하지 않는다면 네트워크 대역폭을 낭비하게 되거나 혼잡을 가중시키는 결과를 초래하게 된다. 혼잡제어를 위한 방식은 크게 송신자 기반 방식, 수신자 기반 방식, 라우터 기반 방식의 세 가지로 분류된다. 첫째, 송신자 기반 방식[1][2]은 수신자의 도움을 받아 네트워크 상태에 따라 전송율을 조절한다. 둘째, 수신자 기반 방식[3]은 여러 개의 세션 중에서 네트워크 상태에 따라 수신자가 세션에 참가하거나 탈퇴하므로써 적당한 수신율을 선택한다. 셋째, 라우터 기반 방식은 혼잡상황에서 라우터가 각 플로우를 식별하여 대역폭을 불공

정하게 사용하는 플로우의 패킷들을 폐기한다. 수신자 기반 방식에서는 송신자가 계층적 코딩과 같은 처리를 해주어야 하며, 라우터 기반 방식에서는 라우터에 들어갈 만한 적절한 혼잡제어 메커니즘이 없고 설령 존재한다고 하더라도 모든 라우터들이 일시적으로 변화되기는 어렵다. 현재의 인터넷 환경에서는 각 종단의 용용프로그램에서 혼잡제어를 하는 것이 가장 손쉬운 방법이라 할 수 있다.

최근에는 용용프로그램에서 고정된 전송율로 멀티미디어 데이터를 전송하는 것이 아니라 네트워크 상태에 따라 가변적으로 전송율을 조절하기 위한 시도들이 진행되고 있다. 또한 TCP 프로토콜을 사용하는 용용프로그램들과 대역폭을 공정하게 나누어 사용할 수 있도록 하기 위하여 TCP와 유사하게 전송할 수 있는 프로토콜에 관한 연구가 활발하게 진행되고 있다. 대표적으로 RAP[4]과 TFRC[5] 프로토콜을 들 수 있는데 RAP은 TCP 프로토콜과 유사하게 동작하도록 하기 위하여 데이터의 전송율을 선형적인 증가와 기하급수적인 감소하는 방식인 AIMD(Additive Increase Multiplicative Decrease)를 사용한다. 이 프로토콜은 원래의 TCP 프로토콜에서 신뢰성을 제공하기 위한 여러 복구 메커니즘 부분을 제외시켰으며 TCP와 유사하게 그리고 공정성을 유지하기 위한 목

적으로 개발되었다. RAP의 AIMD 방식은 전송율의 기복이 심하기 때문에 안정된 전송율을 제공하지 못한다는 문제점이 있다. 이를 개선하기 위하여 TRCP에서는 수신자가 RTT와 패킷손실에 의해 Equation을 근거로 전송율을 산출한 후 송신자에게 피드백하면 송신자는 이 정보를 토대로 적절한 전송율을 결정한다.

본 논문에서는 네트워크의 상태에 맞추어 멀티미디어 데이터의 전송량을 동적으로 조절하도록 하고, TCP와 공정하게 대역폭을 사용할 수 있도록 하기 위하여 TCP와 유사하게 동작하도록 하며, 양 종단의 응용프로그램에서 사용할 수 있는 송신자 기반 방식의 Rate Control 방법을 제안한다. 본 논문은 다음과 같이 구성되어 있다. 2장에서는 제안하는 프로토콜의 동작에 관하여 기술하고, 3장에서는 시뮬레이션에 대하여 설명한 후 실험 결과를 분석하고, 마지막으로 결론과 향후 연구 과제를 제시한다.

## 2. TRCP 프로토콜

TCP 프로토콜은 신뢰성을 제공하기 위한 여러 복구 메커니즘과 혼잡시 대이터의 전송량을 조절하기 위한 혼잡제어 메커니즘을 가지고 있다. 그런데 멀티미디어 데이터를 전송하는 시스템에서는 완벽한 신뢰성을 필요로 하지 않으며 데이터의 실시간성을 요구하는 특성 때문에 TCP 프로토콜과 같은 여러 복구 메커니즘은 필요하지 않다.

본 논문에서 제안하는 프로토콜은 수시로 변동하는 네트워크의 상태에 맞추어 멀티미디어 데이터의 전송량을 동적으로 조절하도록 하고, TCP와 공정하게 대역폭을 사용할 수 있도록 하기 위하여 TCP와 유사하게 동작하도록 한다. 송신자가 네트워크 상태를 파악하는 인자는 패킷손실율과 RTT의 두 가지이며, 만약 패킷손실율 또는 RTT가 임계값 이상이면 네트워크 상태를 혼잡상태로 간주하고 임계값 이하이면 비혼잡 상태로 간주한다. 송신자는 TCP 프로토콜이 동작하는 것과 유사하게 AIMD 방식을 사용하여 적절한 전송율을 결정한다. 이하 이 프로토콜의 명칭을 TRCP(TCP-friendly Rate Control Protocol)라 칭하기로 한다.

### 2.1 패킷손실율과 RTT 계산

송신자는 수신자에게 얼마만큼의 데이터를 전송해야 하는지 결정하기 위하여 수신자와 지속적으로 Control 패킷을 교환한다. 이 패킷들에는 RTT를 계산할 수 있는 시간정보와 패킷손실 정보를 포함하고

있기 때문에 송신자는 RTT와 패킷손실율의 정도에 따라 네트워크 상태를 알아낼 수 있다.

상대방의 송수신 상태 정보를 파악하기 위해 RFC1889[6]에서는 일정한 주기(약 5초)로 Control 패킷을 교환하는 방법을 제안하고 있다. 그러나 일정한 주기로는 매 패킷마다 ACK를 발생시키 네트워크 상황을 파악하는 TCP 프로토콜에 비하여 급격하게 변화하는 네트워크 상황을 신속하게 반영하지 못한다는 단점이 있다. 또한 Control 패킷들이 유실되었을 때는 일정한 주기만큼 더 기다려야 하기 때문에 네트워크 상황을 제대로 전달하지 못하는 경우가 발생한다.

TRCP에서는 이러한 문제점을 개선하기 위하여 송신자는 RTT 시간마다 Control 패킷을 송신하고 수신자는 이 패킷을 받자마자 응답 Control 패킷을 전송한다. 송신자가 RTT 시간마다 Control 패킷을 전송하기 때문에 전송 도중에 일부 패킷이 유실되더라도 지속적으로 Control 패킷의 전송이 이루어진다. 이 방법은 일정한 주기에 의한 Control 패킷 전송 방법보다는 많은 패킷을 발생시키지만 Control 패킷이 20바이트 내외의 경량의 패킷이며 TCP의 ACK 패킷보다는 훨씬 적기 때문에 네트워크에 과중한 부하를 주지는 않는다.

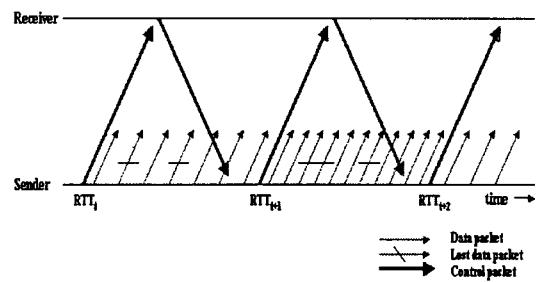


그림 1 data packet과 control packet 전송 모습

네트워크 상태를 측정하기 위한 RTT의 계산은 다음과 같은 방법으로 한다.

1. 송신자는 Control 패킷을 수신자에게 전송할 때마다 시간인 St를 기록한다.
2. 수신자는 송신자로부터 받은 Control 패킷에 대한 응답 패킷을 생성하여 RTT 계산에 필요한 시간 정보들을 기록한 후 전송한다. 시간정보는 수신자가 받은 시간 Rr과 전송한 시간 Rt의 두 가지이다.
3. 송신자는 수신자로부터 Control 패킷을 받은 시간인 Sr을 기록한다.
4. 송신자는  $RTT = Sr - St - Rt - Rr$ 로 계산한다.

이 값에 의해 평균 RTT를 계산한다.

패킷 손실율도 네트워크 상태를 파악하는데 중요한 요소가 되기 때문에 수신자는 이전에 Control 패킷을 보낸 시간으로부터 현재까지 발생한 패킷 손실율을 계산하여 Control 패킷에 실어 보낸다. 패킷 손실율은 송신자가 보낸 패킷의 수와 수신자가 받은 패킷의 수에 의하여 결정한다. 수신 측에서 데이터 패킷에 기록된 순서번호 필드의 차(difference)에 의해 송신 패킷 수를 구할 수 있고, 수신자가 데이터 패킷을 받을 때마다 개수를 카운트하기 때문에 이 값의 차에 의해 수신 패킷 수를 구할 수 있다.

$$\text{패킷손실율} = (\text{송신 패킷} - \text{수신 패킷}) / \text{송신 패킷}$$

RTT와 패킷손실율의 계산에 사용되는 Control 정보의 데이터 구조는 그림2와 같다.

```
struct ctrl_info {
    double send_time_; // 패킷 전송 시간
    double diff_time_; // 수신측에서 지연된 시간
    double loss_rate_; // 패킷 손실율(수신측에서 계산)
};
```

그림 2 RTT와 패킷손실율 계산을 위한 Control 정보의 데이터 구조

## 2.2 전송량 결정

송신자는 RTT와 패킷손실율 정보가 포함되어 있는 Control 패킷이 들어올 때마다 두 가지 정보를 이용하여 적당한 전송율을 결정한다. 본 논문에서는 Window의 크기에 의하여 전송율을 결정하는 TCP vegas[7]의 이론을 응용하여 적용하였다. TCP Vegas에서는 전송 중인 데이터의 양을 측정하는 척도로 RTT를 사용한다. TCP Vegas는 패킷을 보낸 후 ACK가 도착할 때까지 시간을 측정해 그 최소값을 BaseRTT로 저장해 혼잡이 발생하지 않을 때의 RTT 값으로 생각한다. 그리고 윈도우의 크기를 BaseRTT로 나눈 값을 기대값(Expected rate)으로 계산하고, 실제 보내는 현재값(current rate)와의 차이를 계산한다. 만약 그 값이 임계치  $\alpha$ 보다 작은 경우는 망 자원이 많이 남는 것으로 생각해 윈도우의 크기를 늘리고 임계치  $\beta$ 보다 큰 경우는 혼잡이 발생한 것으로 생각해 윈도우의 크기를 줄이게 된다.

TRCP에서는 TCP vegas의 이론을 응용하여 RTT를 측정하여 네트워크 혼잡상태를 예측하도록 하였고 RTT의 변화에 따라 전송율을 조절하도록 하였다. 아래의 공식에서 BaseRTT는 혼잡상태가 아닐 때의

RTT로서 Connection 동안에 측정한 RTT 중에서 가장 작은 RTT 값을 의미한다. Expected는 혼잡상태가 아닌 상태에서 전송 가능한 패킷의 갯수이고, Actual은 현재 상태에서 초당 전송 가능한 패킷의 갯수를 의미한다.

$$\text{Expected} = \text{CurrentRate} / \text{BaseRTT}$$

$$\text{Actual} = \text{CurrentRate} / \text{CurrentRTT}$$

$$\text{Diff} = \text{ExpectedRate} - \text{ActualRate}$$

이 두 값의 차이(difference)를 구하여 임계치의 상한값( $\beta$ ) 이상이면 혼잡상태이므로 전송율을 일정량씩 감소시키고, 하한 값( $\alpha$ ) 이하이면 비혼잡 상태이므로 일정량씩 증가시키며, 그렇지 않으면( $\alpha \leq \text{diff} \leq \beta$ ) 그대로 유지시킨다. 그리고 비혼잡 상태일 때 AIMD기법을 사용하여 임계치 이상일 경우에는 기하급수적으로, 이하일 경우에는 선형적으로 증가하도록 한다.

```
if Diff <  $\alpha$ , CurrentRate = CurrentRate + MTU;
if Diff >  $\beta$ , CurrentRate = CurrentRate - MTU;
if Diff >=  $\alpha$  and Diff <=  $\beta$ ,
    CurrentRate = CurrentRate;
```

TRCP에서는 RTT 뿐만 아니라 패킷손실율을 이용하여 전송량을 결정하도록 한다. TCP에서 타임아웃이 발생하거나 중복된 ACK가 발생할 경우에 slow start 단계로 진입하는데 다음의 두 가지 경우에 이와 유사하게 동작하도록 한다. 패킷 손실율이 3% 이상일 경우에는 네트워크가 혼잡상태인 것으로 간주하고 전송율을 1/2로 감소시킨다. 그리고 네트워크 혼잡 상태가 극심하여 Control 패킷들의 손실이 발생할 경우가 있는데 이 때는 평균 RTT시간의 4배 시간동안 Control 패킷이 들어오지 않으면 전송율을 현재의 1/2로 감소시킨다.

## 3. 실험 및 분석

제안된 프로토콜의 성능을 검증하기 위하여 시뮬레이션을 통하여 실험하였다. 이 실험에 사용된 네트워크 시뮬레이터는 NS-2(version 2.26)[8]이며 Debian Linux(kernel 2.4.18)를 탑재한 PC에서 실행하였다. TRCP는 NS-2.26에 포함되어 있는 RTP 프로토콜의 일부분을 수정하여 구현하였다. TRCP의 성능 실험을 위하여 그림 3과 같이 간단한 토플로지를 구성하였다. 패킷은 송신자 S1으로부터 중간의 라우터 M1을 거쳐서 수신자 R1까지 전달되며 S1~M1 구간의 네트워크 대역폭은 1Mbps이고 전송지연 시간은 10ms로, 병목구간인 M1~R1 구간의 네트워크 대역폭은 400Kbps이고 전송지연 시간은 30ms로 설정하였다.



그림 3 시뮬레이션 토플로지

이 실험에서 TCP 프로토콜은 Vegas 모델을 적용하여 FTP 전송을 하도록 하였다. TRCP와 TCP의 공정성 실험을 위하여 두 프로토콜을 동시에 실행시켜서 600초 동안 전송율의 변화를 각각 측정하였다. 초기에 두 프로토콜을 동시에 실행시켰다가 200초 지점에서 FTP 전송을 멈추고 400초 지점에서 다시 FTP 전송을 시작하였다. 그림 4는 TRCP와 TCP의 전송율 변화를 그래프로 나타낸 것이다.

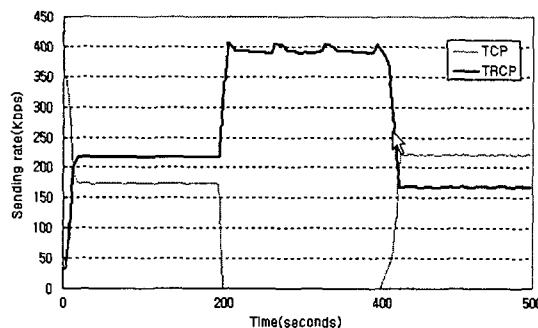


그림 4 TRCP와 TCP의 전송율 변화

그림 4에서 두 프로토콜을 동시에 시작시켰을 때 TCP의 전송율이 우세하다가 시간이 지남에 따라 TRCP가 점차 회복하여 20초 지점에서는 병목 구간 대역폭(400Kbps)의 중간인 220Kbps를 정도로 안정된 모습을 찾는다. 초기에 TRCP의 전송율이 열세인 이유는 control 정보 주기의 초기값이 5초로 설정되어 있어서 5초만에 첫 control 정보를 교환하기 때문이다. 200초 지점에서는 FTP 전송을 멈추었기 때문에 TRCP가 400Kbps의 대역폭을 모두 사용하게 되며 400초 지점에서 FTP 전송이 다시 시작되었기 때문에 대역폭을 서로 나누어 갖게 되는데 이 때는 TRCP가 TCP보다 약간 전송율의 열세를 나타낸다. 서로 다른 프로토콜이 동작하기 때문에 정확하게 대역폭을 공유하기는 어렵지만 위의 결과로 보면 두 개의 프로토콜이 비교적 대역폭을 공정하게 사용하고 있다고 볼 수 있다. 그리고 패킷손실은 실험시간 동안 한 번도 발생하지 않았다. 이는 두 프로토콜이 RTT에 의해 네트워크 혼잡 상태를 미리 예측하여 전송율을 조절한다는 것을 의미한다.

#### 4. 결론

본 논문에서는 수시로 변동하는 네트워크의 상태에 맞추어 멀티미디어 데이터의 전송량을 동적으로 조절하도록 하고, TCP와 공정하게 대역폭을 사용할 수 있도록 하기 위하여 TCP와 유사하게 동작하도록 하는 TRCP 프로토콜을 제안하였다. TCP vegas 이론을 응용하여 RTT에 의하여 네트워크 혼잡상태를 예측하도록 하였으며 TCP 프로토콜과 유사하게 동작하도록 하기 위하여 AIMD 방식을 사용하여 전송율을 조절하도록 하였다. 그리고 이 프로토콜의 성능시험을 위해 간단한 시뮬레이션을 통하여 TRCP와 TCP 트래픽이 서로 공정성을 유지하는 모습을 보였다.

향후 연구과제로는 TRCP와 TCP의 공정성에 관한 실험을 위하여 좀 더 다양한 네트워크 토플로지로 확장하여 하고자 한다. 그리고 TCP vegas 뿐만 아니라 Tahoe, Reno와 같은 모델을 적용하여 이들파도 공정성을 유지하는지 살펴보고, RAP와 TFRC와 같은 프로토콜과의 실험을 통하여 성능비교도 진행하고자 한다.

#### [참고문헌]

- [1] T. Turletti, "The INRIA Videoconferencing System (IVS)", *conneXions – The Interoperability Report*, Vol.8 , No. 10, pp. 20–24, October 1994.
- [2] Seung-Gu Na and Jong-Suk Ahn, "TCP-like Flow Control Algorithm for Real-time Applications", IEEE ICON2000, September 2000.
- [3] McCanne, S. and Jacobson, V., "Receiver-driven Layered Multicast", Sigcomm, pp.117–130, October 1996.
- [4] RazaRejaie, Mark Handley, Deborah Estrin, "RAP:An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", IEEE INFOCOMM, pp.1337–1345, March 1999.
- [5] M. Handley, S. Floyd, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC3448, January 2003.
- [6] Schulzrinne, Casner, Frederick, and Jacobson, V., "RTP: A transport protocol for real-time applications.", RFC1889, January 1996.
- [7] Brakmo, L.S., O'Malley, S.W., and Peterson, L.L., "TCP:Vegas: New Techniques for Congestion Detection and Avoidance", Sigcomm, pp. 24–35, 1994.
- [8] McCanne, S., and Floyd, S. "The LBNL Network Simulator", Lawrence Berkely Laboratory. Software on-line(<http://www-nrg.ee.lbl.gov/ns/>)