pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2013.2.3.117

# TFRC Congestion Control for Mobile Streaming Services Based on Guaranteed Minimum Transmission Rate

Kang Seob Lee† · Seung-sik Choi††

## ABSTRACT

In this paper we propose a TFRC(TCP Friendly Rate Control) which guarantees a minimum rate in order to improve the efficiency of the previous TFRC which cannot distinguish congestion losses and wireless losses and decreases throughput both in wired and wireless networks. This TFRC technique is able to guarantee a minimum rate for video by restricting a loss event rate with packet loss probability about existing TFRC and constraining a rate reduction from the feedback timeout. When we experimented both the existing TFRC and the new one with TCP in the same network, we found that the latter is better than the former. Consequently, it shows that the proposed TFRC can improve video streaming quality using a guaranteed minimum transmission rate.

**Keywords : TFRC, TCP-Friendly, Multimedia, Congestion Control**

# 모바일 스트리밍 서비스를 위한 최소전송률 보장 기반 TFRC 혼잡제어

이 강 섭† · 최 승 식††

## 요 약

본 논문에서는 유무선 이종망의 네트워크에서 혼잡손실과 무선손실을 구별하지 못하고 전송률을 감소시키는 TFRC(TCP Friendly Rate Control)의 성능을 개선하기 위해서 최소 전송률을 보장하는 TFRC를 제안한다. 이 TFRC기법은 기존 TFRC기법의 패킷손실률에 따른 손실이벤트 비율(loss event rate)을 제한하고 피드백 타임아웃에 따른 전송률 감소를 제한함으로써 비디오의 최소전송률을 보장 한다. TCP와 기존의 TFRC, TCP와 제안하는 TFRC를 각각 같은 네트워크 망에서 경쟁하는 환경으로 실험 했을 때 기존의 TFRC와 비교해서 제안하는 TFRC의 전송률이 보장되어 더 좋은 성능을 보였다. 결과적으로 제안하는 TFRC기법이 최소전송률을 보장함으로써 비디오 스트리밍 서비스의 품질을 보장할 수 있다는 것 을 알 수 있다.

**키워드 : TFRC, TCP-Friendly, 멀티미디어, 혼잡제어**

## 1. 서 론

IEEE 802.11 무선 랜과 같은 광대역 무선망의 급속한 확대와 3G에 이은 4G 이동통신망이 상용화되기 시작하면서 무선 네트워크의 사용이 크게 증가하고 있다. 더욱이 태블릿 PC와 스마트 폰의 보급 확대가 이에 큰 기여를 하였고 무선 모바일 상에서 인터넷 서비스를 이용하는 사용자가 많아졌다. 인터넷의 큰 비중을 차지하는 멀티미디어 응용 프로그램 서비스, 즉 비디오 스트리밍 서비스 역시 마찬가지다. 그 중에서도 스마트 폰이나 태블릿PC에서 제공하는 유튜브 서비스처럼 last-hop을 무선구간으로 하는 이종 망에서 멀티미디어 전송시스템 연구가 활발하다[1][2], [11-13]. 멀티미디어 실시간 어플리케이션은 전송되는 데이터가 어느 정도 손실이 있더라도 상관없지만 영상이 끊어지게 되는 전송지연에 민감하기 때문이다. 패킷의 일부 손실은 감수할 수 있어도 효율적인 영상 스트리밍을 위해서는 지연에 의한 전송률을 감소시키는 일이 없어야 하는 것이다. 이러한 이유로 대부분의 멀티미디어 스트리밍은 혼잡제어를 수행하지 않는 UDP 프로토콜을 사용한다. 그러나 다중 네트워크에서 TCP와 UDP를 사용하는 어플리케이션들이 같은 혼잡상태의 채널을 이용한다고 생각하면 문제가 생긴다. TCP프로토콜을 이용하는 어플리케이션은 혼잡에 의해서 전송률을 제

※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초 연구사업임(2010-0024926).
† 준 회 원 : 인천대학교 컴퓨터공학과 학부생(주저자)
†† 종신회원 : 인천대학교 컴퓨터공학과 부교수
논문접수 : 2012년 6월 28일
수 정 일 : 1차 2012년 11월 1일, 2차 2013년 1월 11일
심사완료 : 2013년 1월 11일
* Corresponding Author : Seung-sik Choi(sshoi@incheon.ac.kr)

어하게 되는 반면 UDP는 전송률을 유지하게 될 것이다. 따라서 TCP가 UDP에 비해 대역폭을 더 적게 사용하는 불공평성이 발생한다. 이러한 문제점을 보안하기 위해서 TCP Reno의 혼잡제어를 모델링한 TFRC(TCP Friendly Rate Control)기법이 제안되었다. TCP와 유사하게 패킷 손실에 따라 흐름제어를 하여 TCP와 공평하게 대역폭을 유지 할 수 있는 것이다[3][4], [6-8].

하지만 서두에서 언급한 무선모바일을 last-hop으로 하는 환경에서 TFRC기법은 무선구간에서의 패킷손실을 유선구간의 손실과 더불어 같은 패킷 손실로 파악한다. 이로 인해 무선구간에서의 손실 역시 흐름제어를 하게 되고 이에 따른 불필요한 전송률 감소는 무선 모바일에서 원활한 영상 스트리밍 지원에 문제가 생기게 한다.

## 2. 관련 연구

네트워크상에서 TCP와 균등하게 전송률을 유지하기 위해서 TCP 친화적인 프로토콜들이 제안되고 있다. TFRC기법 또한 TCP 친화적인 프로토콜들 가운데 하나로서 S. Floyd 등에 의해 제안되었다. TFRC(TCP Friendly Rate Control) 프로토콜은 TCP 프로토콜과 유사하게 혼잡손실에 따라서 전송률을 높이고 줄임으로써 TCP 친화적으로 동작하는 방법이다. 이러한 TFRC는 식 (1)에 따라 전송률을 계산한다[3],[4].

$$X=\frac{S}{RTT\sqrt{\frac{2p}{3}}+RTO(3\sqrt{\frac{3p}{8}}p(1+32p^2))} \quad (1)$$

이는 TCP Reno를 기반으로 Padhye가 수학적 공식으로 모델링한 식 (2)에 해당하는 TCP의 전송률 방정식을 모델링하였다[3].

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

식 (1)에서 X는 byte/s 단위의 평균 전송률이고 S는 전송되는 패킷의 Byte단위 크기이고 RTT는 왕복시간, RTO는 TFRC의 재전송 타이머의 타임아웃 값이고 p는 수신 측에서 피드백 하는 손실 이벤트 비율(loss event rate) 크기이다. 수신 측은 이 손실 이벤트 비율을 계산하기 위해서 패킷손실률을 구한다. Fig. 1에서 보는 것처럼 수신 측에서 패킷들을 전송받는 중에 손실이 발생하면 이를 기억하고 있다가 현재의 왕복시간 만큼 시간이 지난 뒤부터 추가적인 패킷의 손실이 발생하면 두 패킷 사이의 손실된 패킷 수를 통해 패킷손실률을 계산한다. 가장 최근 패킷이 전송되었을 때까지 8개의 왕복시간 동안 패킷손실률을 수신 버퍼에 유지하고 이들의 평균손실률, 손실 이벤트 비율을 계산한다. 이 계산된 값에 따라서 손실에 따른 손실 이벤트 비율을 송신 측으로 피드백 하게 된다[2-5].

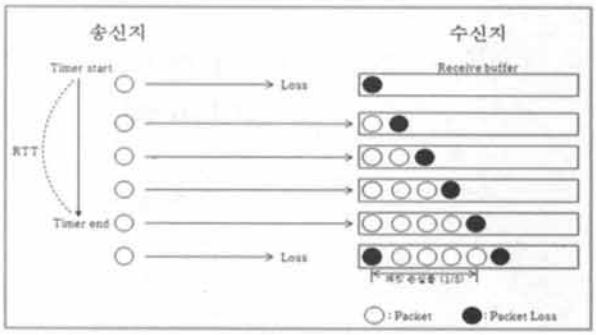


Fig. 1. Packet Loss Ratio in TFRC

이와 더불어 피드백 되는 패킷들이 네트워크의 혼잡으로 손실되는 경우도 발생한다. 이 경우에 피드백 된 정보가 없어 전송률을 조절할 수 없게 된다. TFRC 프로토콜에서 트래픽 전송을 시작할 때 왕복시간 타이머(RTT Timer)와 함께 피드백 타이머(Feedback Timer)를 시작한다. 패킷을 전송하고 피드백을 받는 동작 중에 혼잡으로 인해 피드백에 대한 손실이 발생하게 되고 피드백 타이머의 시간 동안 피드백이 수신되지 않으면 TFRC기법은 전송률을 절반으로 감소시키는 동작을 수행한다. 이는 TFRC 프로토콜이 TCP Reno의 동작을 모델링 한 것으로 TCP Reno 프로토콜에서 세 개의 중복 Ack가 발생할 경우 혼잡윈도를 절반으로 줄이는 동작을 모델링 한 것이다[6].

이러한 TFRC 프로토콜은 수신측이 전송 받은 패킷으로부터 손실을 발견했을 때 손실 이벤트 비율을 계산하고 송신 측으로 피드백 한다. 송신측은 이를 토대로 전송률을 조절한다[7-9].

만약 전송한 패킷에 대한 피드백이 돌아오지 않을 경우 피드백 타이머에 의해서 타임아웃 이벤트(event)가 발생하고 비율(rate)을 현재의 1/2로 감소시킨 뒤 재전송한다. 현재의 네트워크 상태가 혼잡하여 전송한 패킷이 손실되었고 피드백 되지 않는다고 판단하기 때문이다. 그런데 이러한 과정에서 TFRC는 혼잡에 대한 손실만 있다고 판단하기 때문에 무선 구간에서 동작할 경우 무선 손실마저도 혼잡 손실로 판단하고 전송률을 감소시킨다[3][4], [10][11].

기존에 유무선 이종망에서 TFRC트래픽의 비율을 보장하는 방법은 손실 원인을 구별하는 알고리즘을 이용하여 위와 같은 무선 네트워크에서 TFRC기법이 갖는 문제점을 해결하려는 연구가 많았다.

시간적인 정보를 이용하여 손실원인을 구별하는 알고리즘으로 TFRC-Biaz와 TFRC - Spike가 있다. Biaz는 패킷간의 간격을 측정하여 손실 원인을 구별하고 Spike는 패킷의 지연시간을 측정하여 값의 평균적인 수치에 따라 혼잡 손실(유선구간에서의 손실) 무선손실을 구별한다. 이러한 방법은 시간적인 정보를 토대로 확률적인 손실원인을 찾는 것으로 정확도가 에러복구 기법이 사용될 때 저하되는 경향이 있다[7].

SLD 기법은 도착한 패킷들의 상태를 구분하여 혼잡의 정도를 파악하고 원일을 구별하는 방법이다. 정상적으로 도

착한 패킷과 손실된 패킷, 혼잡상태에서 마킹된 패킷의 비율로 혼잡의 정도를 결정하여 손실 원인 별로 전송률을 제어하는 방법이다[1].

WM-TFRC는 무선 적응 레이어(WAL)를 추가하여 이 무선구간에서 전송되는 패킷으로부터 패킷손실 정보를 피드백 받아 평균적인 무선채널의 손실을 구하여 혼잡손실과 구분하는 방법이다[7].

## 3. 최소전송률을 보장하는 TFRC기법

### 3.1 System Model

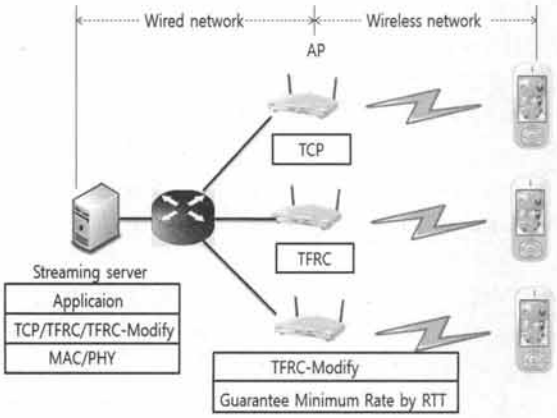


Fig. 2. Streaming services that we propose and System structure served by TCP, TFRC and TFRC-Modify

제안하는 최소전송률을 보장하는 TFRC기법은 TFRC-Modify기법이라고 칭하고 그 시스템은 Fig. 2와 같은 구조를 보이고 있다. 순수한 TFRC 프로토콜을 기반으로 하여 이를 수정한 TFRC-Modify기법으로 수정한 방법은 전송률을 임의의 수치 이하로는 내려가지 않게 하여 최소 전송률을 보장하는 TFRC기법이다. 비디오 스트리밍 패킷은 스트리밍 서버로부터 시작되어 유선구간을 지나 AP를 통해 무선 단말기인 스트리밍 클라이언트로 전송된다. 이때 제안하고자 하는 TFRC-Modify기법을 이용한 스트리밍 서비스는 전송하는 패킷의 최소 전송률을 왕복 시간 타이머에 따라 보장해 주도록 한다. 비디오 스트리밍 서비스의 특성상 특정 전송률 이상을 유지하여 서비스 패킷을 전송한다면 어느 정도의 손실이 발생하는 혼잡한 네트워크에서도 끊김 없이 원활한 스트리밍 서비스를 제공할 수 있다. 이는 제안하는 TFRC-Modify기법을 통해 무선구간과 유선구간에서의 손실을 구분하고 유선구간에서의 혼잡시에 최소전송률을 보장하여 스트리밍 서비스를 제공하는 시스템을 제공한다. 결과적으로 서버가 제공하는 서비스 중에서 영상 스트리밍 서비스의 경우에만 혼잡상황에서 최소전송률을 보장하는 TFRC-Modify기법을 통해 전송률을 제어하고 다른 모든 서비스는 TCP기법을 통해 전송한다(TCP기법은 TFRC와의 결과적인 성능을 알아보기 위한 비교군 으로서 존재한다).

### 3.2 TFRC-Modify기법

제안하고자 하는 TFRC-Modify기법의 전송률을 제어하는 방법을 두 가지 단계로 나누어 정의 하고 이에 따른 알고리즘을 설명한다.

1단계는 피드백 되는 손실 정보를 통해 전송률을 조절하는 방법이다. 이때 전송률을 제어하는 과정에서 제한하고자 하는 최소전송률보다 낮게 측정 된다면 최소전송률로 전송률을 변경한다. 손실의 원인을 구별하여 전송률을 계산하는 방법이 아닌 직접적으로 전송률의 수치를 조정하여 이를 보장하는 방법이다. 이는 기본적으로 TFRC의 전송률 제어 알고리즘을 따르고 그에 수정한 방법을 통해 최소전송률을 보장한다. TFRC의 기본적인 전송률 제어 방법은 송신측이 수신지로부터 피드백을 받았을 때 피드백의 정보에 기반 하여 전송률을 조절하는 것이다. 본문 2장의 식 1에서 언급한 것처럼 TFRC의 평균 전송률을 X라고 할 때 다른 변수들은 다음과 같이 정의 할 수 있다. X는 byte/s 단위의 평균 전송률이고 S는 전송되는 패킷의 Byte단위 크기이고 RTT는 왕복시간, RTO는 TFRC의 재전송 타이머의 타임아웃 값이고 p는 수신 측에서 피드백 하는 손실 이벤트 비율(loss event rate) 크기이다. 이는 전송된 패킷 중 손실이 발생한 이벤트의 비율로 0 에서 1.0 사이의 값을 갖는다. t는 전송률을 계산하기 위한 시간 변수들로 초(s)단위로 나타낸다. t_now는 현재의 시간, t_mbi는 패킷 간에 백오프 할 수 있는 최대시간이고 t_ld는 가장 최근에 전송률이 2배가 된 시간, 즉 slow_start가 종료된 시간을 말하며 t_RTT는 왕복시간을 말한다. 손실 송신 측에서 전송률(X)을 알고 있고 가장 최근에 갱신된 RTT와 타임아웃 값(RTO)를 피드백을 통해 알고 있기 때문에 피드백이 도착했을 때 이 정보들을 통해서 다시 전송할 전송률을 갱신한다. 갱신하는 방법은 Fig. 3에 기술된 알고리즘과 같다.

```
가장 최근에 수신된 패킷에 손실이 발생 했을 경우

손실 이벤트 비율 재설정

RTT 갱신

타임아웃 주기 업데이트

전송률 갱신
if (p > 0)
  calculate R_calc using the TCP throughput
  equation.
  R = max(min(R_calc, 2*R_recv), s/t_mbi);

Else
  If (t_now - t_ld >= t_RTT)
  R = max(min(2*R, 2*R_recv), s/t_RTT);

수정한 기법에 따른 전송률 재설정
if(R < Rmin)
      R = Rmin

다음 패킷 전송
```

Fig. 3. Rate control according to the loss in feedback

네트워크 채널에서 손실이 발견되면 이에 의해 TFRC기법의 알고리즘에 따라 패킷손실률을 구하고 손실 이벤트 비율을 계산한다. 그 이후에 계산된 손실 이벤트 비율(loss event rate)에 따라 변경할 전송률을 피드백으로 송신 측에 전송한다. 피드백을 통해서 RTT를 갱신하고 타임아웃 주기를 업데이트 한다. 이후 전송률(R)을 손실 이벤트 비율의 크기에 따라 업데이트 한다. 손실 이벤트 비율이 0보다 크다면 TCP 전송률 공식에 따라 전송률 값 R을 구한다. 이 경우 손실 이벤트가 발생하였다고 판단하고 전송률을 조절한다. 그 값을 최소 전송률로 설정한 *Rmin*값과 비교한다. 만약 변경할 전송률이 제한하는 최소전송률 보다 크다면 다음 패킷을 보내기 위한 전송률을 손실 이벤트 비율에 의해 계산된 전송률로 유지하여 피드백 한다. 이와 다르게 변경할 전송률이 제한하는 최소전송률보다 작은 경우에는 피드백할 전송률을 제한하고자 하는 전송률(*Rmin*)로 변경한다. 그 이후에 패킷을 변경된 전송률로 전송한다.

더불어 패킷이 송신된 뒤에 피드백 타이머가 동작하고 피드백이 돌아오기를 기다린다. 만약 채널에서 피드백이 손실되어 돌아오지 못할 경우 전송률을 재조정하고 패킷을 전송해야 하기 때문이다.

```
TFRC No Feedback Timer가 타임아웃 됐을 경우

발생한 이벤트에 따라 비율을 감소

전송률을 현재의 절반으로 줄임

if(R < Rmin)
      R = Rmin

No Feedback Timer를 재설정
상태를 혼잡회피로 변경

다음 패킷 전송
```

Fig. 4. Rate control algorithm of TFRC when feedback timer timeout

이렇게 피드백이 손실되어 돌아오지 않고 피드백 타이머가 타임아웃 되었을 때 전송률을 절반으로 줄이는 과정에서 Fig. 4와 같이 절반으로 줄인 전송률의 값을 제한하고자 하는 최소전송률(*Rmin*)과 비교한다. 절반으로 줄인 전송률의 값이 최소 전송률보다 크다면 제한 없이 전송률을 변경하여 다시 패킷을 보내게 된다. 만약 절반으로 줄인 전송률이 제한한 최소전송률 보다 작다면 다음 패킷을 보내기 위한 전송률을 *Rmin*으로 조정 한 뒤에 다시 전송한다.

이러한 최소전송률을 보장하는 TFRC기법에 RTT에 따라 전송률을 제어하는 방법을 추가한 것이 2단계 방법이다. 2단계에서는 1단계의 최소전송률을 제어하는 방법이 혼잡손실과 무선손실을 구별하는 방법이 아닌 직접적으로 제한하는 방법인 것의 한계를 보완 하고자 한다. 현재의 알고리즘에 패킷들이 송신되고 수신되었을 때 RTT의 수치로 평균값을 계산한다. 모든 피드백되는 패킷들의 전송RTT $RTT_{estimated}$를 측정하고 $RTT_{estimated}$값의 평균 $RTT_{average}$를 유지하고 RTT을 획득할 때 마다 아래의 식 (3) 에 의해 $RTT_{average}$를 갱신한다[16].

$$RTT_{average} = (1-\alpha) \cdot RTT_{average} + \alpha \cdot RTT_{estimated} \quad (3)$$

이에 권장되는 α의 값은 α = 0.125[RFC 2988]이다[9]. 이 식을 이용하여 지속적으로 패킷이 전송될 때 마다 갱신되는 $RTT_{average}$값을 계산한다.

식 (3)에 의해 갱신되는 $RTT_{average}$값에 따라 갱신 직후 수신 되는 피드백의 RTT에 따른 $RTT_{average}$의 평균이 상위로 변하여 Fig. 6과 같이 정규분포의 그래프가 높이가 낮아지고 너비가 넓어지는 경우 혼잡에 따른 RTT의 증가로 판단한다. 혼잡이 발생하는 경우에는 큐잉 지연이 커지기 때문에 수신된 패킷이 도착하는 시간이 증가하고 그에 따른 RTT의 표준편차는 변하지 않지만 RTT의 평균은 증가 한다. 가장 최근에 수신된 RTT의 값에 따른 $RTT_{average}$값이 증가하는 추세일 경우 최소전송률을 보장하도록 한다. RTT가 증가하는 상태인지 알기 위해서 이전의 RTT들의 변화를 이용한다.

$RTT_{average}$가 증가하는 상태이고 현재의 RTT(RTT-estimated)가 이전의 RTT + 표준편차보다 크다면 현재의 RTT은 증가하는 상태라고 판단한다.

반면 Fig. 7과 같이 피드백 되는 RTT의 정규분포의 $RTT_{average}$값은 변하지 않고 표준편차만 변할 경우 이는 무선구간에서의 손실에 따라 수신 되는 RTT의 평균은 변화가 없지만 그 분포가 커진 상황이다. 이 경우 혼잡에 따른 손실과 구별하여 최소전송률을 보장하지 않는다. 식 (4)는 정규분포에서 평균이 $RTT_{average}$일 때 확률밀도함수이다.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-RTTaverage)^2}{2\sigma^2}} \quad (4)$$

식 (4)의 확률밀도함수에 따라 RTT값의 표준편차가 증가하는 경우에는 무선손실에 따른 RTT의 분산이 증가한 상황으로 판단하고 이 경우에는 최소전송률을 보장하지 않는다. 표준편차의 값은 확률밀도함수의 2차 모멘트(Moment)인 식 (5)에 의해 분산 값을 구한 뒤 측정하는 RTT의 집합을 $RTT_{estimated1}$, $RTT_{estimated2}$, $RTT_{estimated3}$ … $RTT_{estimatedn}$이라 할 때 표준 편차는 식(6)에 의해 구할 수 있다.

$$\sigma^2 = E[x^2] = \int_{-\infty}^{\infty} x^2 P(x) dx \quad (5)$$

$$\delta = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(RTT_{estimated_i} - RTT_{average})^2} \quad (6)$$

기존의 전송률을 제한 방법과 더불어 앞서 설명한 방법을 적용하여 혼잡상황을 판단한다.

이에 대한 전송률을 제어하는 방법은 Fig. 5와 같다. 손실이 발생 한 뒤 피드백 받은 정보를 통해 손실 이벤트 비율로 전송률을 줄이는 동작에서 감소시킨 전송률이 *Rmin*값보다 작고 동시에 전송받은 피드백의 RTT($_{Destimated}$)이 가장 최근에 전송 받은 평균 RTT($RTT_{average}$)의 정규분포에서 상위와 하위로 벗어나는 경우에 전송률을 *Rmin*으로 변경하는 동작을 한다.

즉, RTT이 평균보다 과도하게 높은 경우와 과도하게 낮은 경우에만 혼잡에 따라 손실이 발생한 상황이라고 판단하고 전송률을 하위 제한한다.

```
가장 최근에 수신된 패킷에 손실이 발생 했을 경우

손실 이벤트 비율 재설정

RTT 갱신

타임아웃 주기 업데이트

전송률 갱신
if (p > 0)
  calculate R_calc using the TCP throughput
  equation.
  R = max(min(R_calc, 2*R_recv), s/t_mbi);

Else
  If (t_now - t_ld >= t_RTT)
  R = max(min(2*R, 2*R_recv), s/t_RTT);

수정한 기법에 따른 전송률 재설정
if(R<Rmin && RTTnow >= RTTaverage+δ)
          R = Rmin

다음 패킷 전송
```

Fig. 5. Rate control based on RTT and information of loss

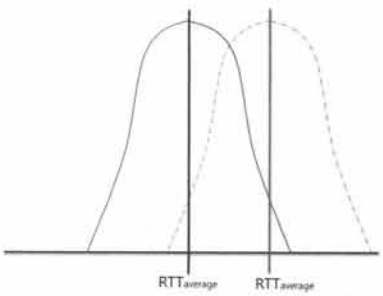


Fig. 6. Average change of RTT nomal distribution feedbacked

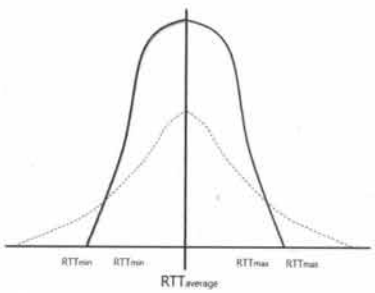


Fig. 7. Standard deviation of RTT nomal distribution feedbacked

## 4. 실험 성능 및 평가

본 논문에서 제안하는 TFRC기법의 성능을 알아보기 위해서 네트워크 시뮬레이터 NS2를 사용하였다[17]. 우선 시뮬레이션을 위한 시나리오는 Fig. 8과 같이 환경을 설정 하였다.

각각의 무선 노드 3, 2에서 TFRC-Modify 프로토콜과 TCP프로토콜에 의해 1000byte의 패킷을 노드 1에 걸쳐 노드 0까지 전송한다. 실험을 시작한 뒤 160초 이후부터 TCP 패킷의 전송을 시작하고 180초 이후에 TFRC-Modify의 전송을 시작한다. 패킷은 CBR 트래픽을 5ms의 지연으로 전송한다. 시뮬레이션은 시작 후 250초에 종료 된다. 이를 통해 TFRC-Modify의 성능을 TCP와 비교한다.

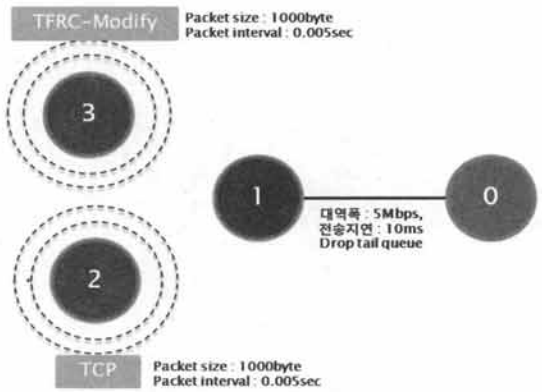


Fig. 8. Simulation environment

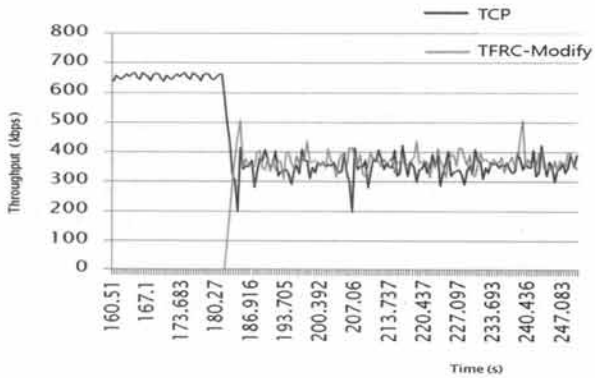


Fig. 9. Comparing TCP with TFRC-Modify that minimum rate is 350kbps when there is no error in the wireless section

Fig. 9는 TCP와 TFRC-Modify의 전송률을 비교한 그래프이다. 수정된 TFRC의 전송률이 TCP에 비해 평균적으로 높은 수치를 보인다.

TFRC-Modify 패킷의 최소 전송률 값을 350kbps 으로 설정해 주었기 때문에 전송률이 그 이하로 내려가지 않는다. 같은 조건에서 TCP와 수정하지 않은 TFRC의 성능을 비교한 Fig. 10의 그래프를 보면 TFRC의 전송률도 TCP와 동등하게 유지되는 모습을 보인다.

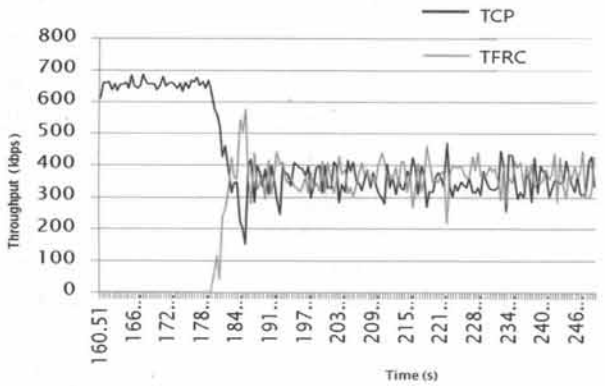


Fig. 10. Comparing TCP with TFRC when there is no error in the wireless section

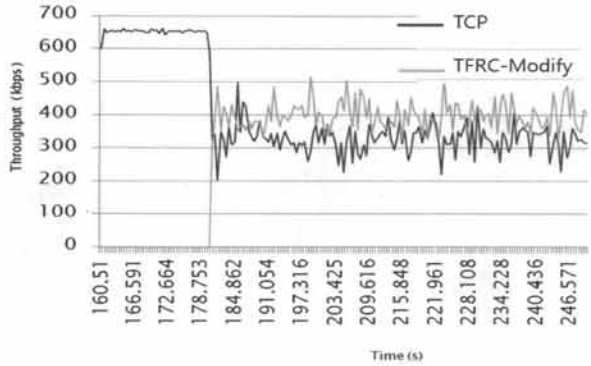


Fig. 11. Comparing TCP with TFRC-Modify that minimum rate is 450kbps when there is no error in the wireless section

이 둘을 비교했을 때 수정된 TFRC-Modify의 전송률이 TFRC보다 높게 유지 된다. 수치상으로는 수정하지 않은 경우 보다 전송률이 약 9% 향상되었다.

동일한 환경에서 최소전송률 값을 450kbps로 설정하여 실험을 한 결과 Fig. 11과 같은 그래프가 나왔다. 최소전송률의 수치를 Fig. 9에 대한 시뮬레이션보다 높게 설정한 결과인 Fig. 11은 Fig. 9와 비교해서 TFRC-Modify의 전송률이 보다 더 높게 유지됐다. 수치상으로는 수정하지 않은 경우인 Fig. 10의 TFRC와 비교하여 약 17% 향상된 전송률을 보였다.

다음으로 시뮬레이션의 환경에서 무선구간에 에러가 있는 경우를 실험했다. TCP와 TFRC패킷을 발생시키는 노드 2, 3과 중계노드인 노드 1 사이에 1%의 에러가 발생하도록 하고 최소전송률을 설정하지 않은 순수한 TFRC일 때와 최소전송률을 350kbps로 설정한 상태에서 실험해 보았다.

Fig. 12와 Fig. 13은 각각 TCP와 TFRC, TCP와 TFRC-Modify의 시뮬레이션에 따른 결과 그래프이다. Fig. 12를 보면 에러를 추가하지 않았던 Fig. 10의 TCP와 TFRC의 전송률 그래프와 비교해 보았을 때 전송률의 변동폭이 급격히 증가함을 보인다. 이는 무선구간에 에러를 추가한 경우 이를 혼잡손실과 구별하지 못하고 전송률을 감소시킨 결과로 알 수 있다. 무선구간에 에러발생률을 1% 추가하고 최소전송률을 설정한 시뮬레이션의 결과인 Fig. 13을 보면 이 역시 무선구간의 에러를 혼잡손실과 구별하지 못하고 전송률을 감소시켜 그 변동 폭이 Fig. 9의 경우보다 크게 나타나지만 최소전송률을 보장해 준 결과, Fig. 12와 비교해 보았을 때 TFRC-Modify 트래픽의 전송률이 TFRC보다 높은 수치에서 유지됨을 보인다. 이는 Fig. 9의 결과보다 약 13% 향상된 전송률을 보인다.

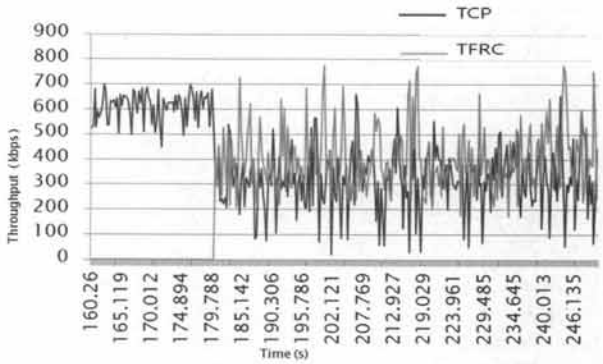


Fig. 12. Comparing TCP with TFRC when there is error(1%) in the wireless section

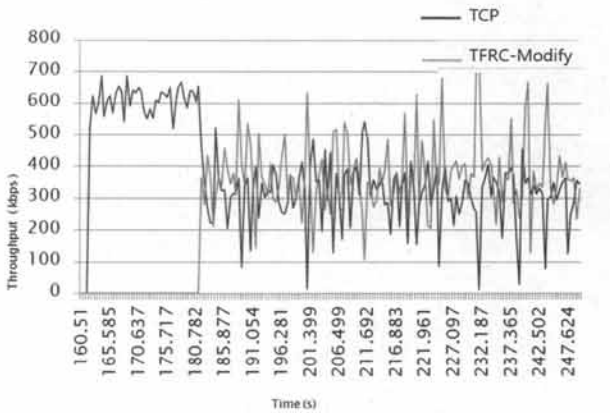


Fig. 13. Comparing TCP with TFRC-Modify that minimum rate is 350kbps when there is error(1%) in the wireless section

Fig. 14와 Fig. 15는 최소전송률을 보장하는 수정한 TFRC기법의 평균 RTT을 나타내는 결과이다. 실험환경은 (Fig. 6)의 시뮬레이션 환경에서 TFRC-Modify의 최소전송률을 350kbps으로 설정했을 경우이다.

Fig. 14는 실험에 따른 TCP와 TFRC-Modify의 평균 RTT을 구한 것이다. 10초의 간격으로 전송받은 패킷들에 대한 평균적인 RTT이다. TCP의 RTT가 TFRC-Modify에 의한 패킷이 전송되기 시작하는 180초 이후부터 급격히 상승한다. 이후에 시간이 지남에 따라 감소하고 TFRC-Modify의 RTT과 비슷하게 유지된다. 같은 환경에서 수정하지 않은 TFRC와 수정한 TFRC의 평균 RTT를 측정했다.

Fig. 15의 그래프는 같은 환경에서 TCP 트래픽과 경쟁하는 순수한 TFRC와 수정한 TFRC의 10초간격의 평균 RTT

시간이다. 수정한 TFRC의 평균 RTT는 시간대에 따라 큰 변동 없이 유지되고 있는 반면 수정 전의 TFRC는 시간대별로 RTT가 크게 달라지고 있다.

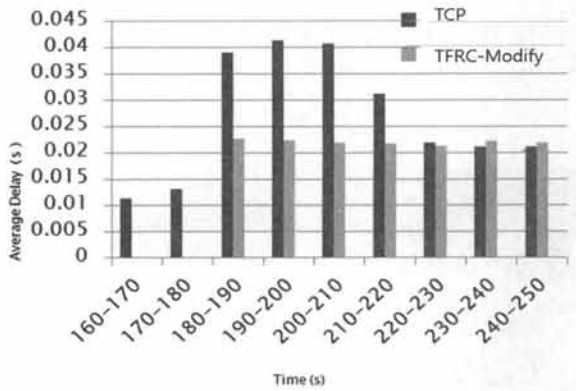


Fig. 14. Average RTT of TCP and TFRC-Modify

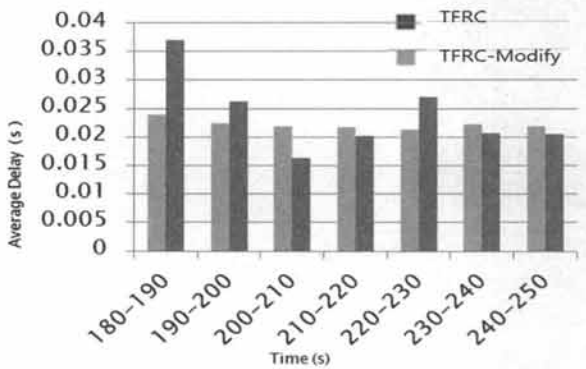


Fig. 15. Average RTT of TFRC and TFRC-Modify

이를 통해 TFRC-Modify기법을 이용한 영상 스트리밍이 더 안정적으로 data를 수신할 수 있다고 볼 수 있다.

최소전송률을 설정해준 Fig. 13의 시뮬레이션 결과에서 최소전송률 보다 제한한 결과의 전송률이 낮게 나오는 경우가 나타난다. 이는 피드백 타이머가 타임아웃 되었을 때 전송률을 절반으로 줄이는 과정에서 혼잡상황이 매우 높음에 따라 제한한 전송률보다 낮게 나오는 상황이라고 판단한다. 이렇게 혼잡상황이 높은 경우는 응용계층에서 보았을 때 성능이 낮게 나올 수 있다고 예상되었다. 이에 대해 추가적으로 제안하는 RTT에 따라 전송률을 제어하는 방법으로 혼잡손실로 판단하는 경우에만 최소전송률을 보장해 준다면 높은 혼잡상황에서 전송률이 감소하는 폭을 줄여줄 수 있을 것이라고 판단하고 실험 하였다. Fig. 16은 이에 대한 결과를 보여주는 그래프이다. TFRC-Modify에 의해서 처리되는 전송률이 평균적으로 TCP에 의해 처리되는 전송률 보다 높게 유지되었다. 또 RTT에 따른 제한을 추가하지 않은 Fig. 12의 TFRC-Modify와 비교하였을 경우 그래프 상의 전송률은 제한하는 전송률보다 떨어지는 경향이 커진 것처럼 볼 수 있다. 이는 손실의 원인을 구분하지 않고 최소 전송률을

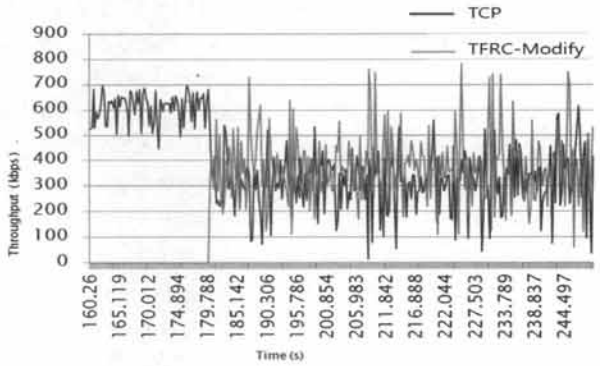


Fig. 16. Comparing TCP with TFRC-Modify that minimum rate is 350kbps when there is error(1%) in the wireless section (Rate control over RTT)

제한했던 방법을 혼잡이 크다고 판단한 상태에서만 최소전송률을 제한하도록 변경한 결과이다.

## 5. 결론 및 향후 과제

본 논문에서 네트워크에서 패킷의 손실에 따른 전송률 감소를 직접적으로 제한하는 TFRC-Modify기법을 제안했다. 이는 직접적으로 최소전송률을 제한하여 일정수치 이하로 내려가지 않게 전송률을 유지해 줌으로써 영상 스트리밍 서비스를 지원하는데 그 품질을 향상시킬 수 있다.

더불어 본 논문에서 제안 하고자 하는 직접적으로 전송률을 제한하는 방법이 기존에 연구되어온 손실의 원인을 구별하여 전송률을 계산하는 수정된 TFRC기법들과 비교했을 때 성능의 차이를 알 수 있는 연구가 앞으로 필요하다.

## 참 고 문 헌

[1] Kyumin Jeong, Jahon Koo, Kwangsue Chung "Loss Discrimination Mechanism for Improving the Performance of TFRCin Last-hop Wireless Networks", in Korean Institute of Information Scientists and Engineers(KIISE), Feb., 2010.

[2] Hyungho Lee, Chong-ho Choi, "A Loss Discrimination scheme for TFRCin Last Hop wireless Networks" in Wireless Communications and Networking Conference(WCNC), Mar., 2007.

[3] M. handly, S. Floyd, J.Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): protocol specification" RFC 3448, Jan., 2003.

[4] S. Floyd, M. Handley, J. Padhye, J. Widmer "TCP Friendly Rate Control (TFRC): Protocol Specification" RFC 5348, Sep., 2008.

[5] J. Garcia and Anna Brunstrom, "Checksum-based Loss Differentiation of Wireless and Congestion Losses" in Distributed Computing Systems Workshops, May, 2003.

[6] Hyun-Tae Kim, Suk-hun Ji, In-ho Ra "A TCP-Friendly

Congestion Control Scheme using Hybrid Approach for Enhancing Fairness of Real-Time Video Stream" in Proceedings of KFIS Spring Conference, June, 2004.

[7] S. S. Khanloo, M. Fathy, and M. Soryani, "Wireless TCP-Freindly Rate Control over the DCCP Transport Protocol" in Wireless and Mobile Communications, July, 2008.

[8] R. Chaudly, and L. Jacob, "ECN based TCP-Friendly Rate Control For Wireless Multimedia Streaming" in Computer Communications and Networks, Oct., 2003.

[9] S. Biaz, and N, Vaidya, "Discriminating Congestion Lossws from Wireless Losses Using Inter-arrival Times at the Receive" in IEEE Symposium ASSET, Mar., 1999.

[10] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Contro," RFC 4340, Mar., 2006.

[11] S. Cen, c. pu, and J. Walpole, "Flow and Congestion Control for Internet Streaming Application," in Multimedia Computing and Networking, Jan., 1998.

[12] Y. C. Lai, "DCCP: Transport Protocol with Congestion Control and Unreliability," in IEEE Internet Computing, Vol.28, No.4, Oct., 1998.

[13] S. Cen, P. C Cosman, and G. M. Voelker, "End-to-End Differentiation of Congestion and Wireless Losses," in IEEE/ACM Transactions on Networking, Vol.11, No.5, Oct., 2003.

[14] S. Bae, and S. Chong, "TCP—Friendly Wireless Multimedia Flow Control Using ECN Marking," in IEEE GLOBECOM, Vol.2, Nov., 2002.

[15] B. Song, K. Chung, and Y. Shin, "SRTP: TCP-ftriendly adaptation scheme," in International Workshop on Network and Operating System Support for Digital Audio and Video(NOSSDAV), Jul., 1998.

[16] Jams F. Kurose, Keith W.Ross "Computer Networking A Top-Down Approch" Addison Wesley, Mar., 1999.

[17] The Network Simulator NS-2, www.isi.edu/nsnam/ns.

이 강 섭

e-mail : lks0604@incheon.ac.kr

2007년~현 재 인천대학교 컴퓨터공학과 학부생

관심분야 : 무선인터넷 프로토콜, 라우팅 프로토콜

최 승 식

e-mail : sshoi@incheon.ac.kr

1998년 연세대학교 전자공학과(학사)

1990년 KAIST 전기 및 전자공학과(석사)

2002년 KAIST 전기 및 전자공학과(박사)

1990년~2004년 KT서비스 개발연구소 선임연구원

2004년~현 재 인천대학교 컴퓨터공학과 부교수

관심분야 : 무선 엑세스제어, 무선자원관리, 무선인터넷 프로토콜