실시간 비디오 스트림의 공정성 개선를 위한 TCP 친화적 하이브리드 혼잡제어기법

A TCP-Friendly Congestion Control Scheme using Hybrid Approach for Enhancing Fairness of Real-Time Video Stream

김현태, 지석근, 나인호
군산대학교 전자정보공학부

Hyun-Tae Kim, Suk-kun Ji, In-ho Ra
School of Electronics and Information Engineering Kunsan National University
E-mail : camelk@kunsan.ac.kr

요 약

최근 인터넷의 발전으로 디지털 오디오 및 비디오와 같은 멀티미디어 스트림에 대한 요구가 증가하고 있다. 이러한 멀티미디어 스트리밍을 UDP로 전송할 경우 TCP와 같은 혼잡제어를 수행하지 않기 때문에 동일한 전송경로의 TCP 트래픽 궁핍을 일으켜 혼잡붕괴 및 막대한 전송지연을 초래한다. 이러한 문제점으로 인하여 실시간 멀티미디어 스트림의 전송지연과 혼잡제어를 위한 새로운 전송기법과 프로토콜에 대한 다각적인 연구가 수행되고 있다. TCP 친화적 혼잡제어 기법은 크게 일반적인 혼잡윈도우 관리기능을 이용하는 윈도우 기반 혼잡제어와 TCP 모델링 방정식 등을 이용하여 전송율을 직접 조절하는 율 기반 혼잡제어로 나눌 수 있다. 본 논문은 윈도우 기반과 율 기반을 복합적으로 다룬 하이브리드형 TCP-friendly 혼잡제어 기법에서 Square-root 혼잡회피 알고리즘을 제안하였으며, NS를 사용하여 제안한 TEAR의 성능을 실험하였다. 실험을 통하여 제안한 기법의 적용으로 TEAR의 안정성을 개선함을 알 수 있었다.

1. 서론

본 논문은 비디오와 같은 대용량의 스트리밍 데이터를 실시간에 IP 프로토콜을 이용하여 전송하고자 할 때 TCP나 UDP와 같이 기존의 트랜스포트 프로토콜 특성에 의해 발생하는 트래픽 혼잡을 제어하기 위한 기법에 대한 연구이다. 멀티미디어 스트림을 전송하기 위해 UDP을 이용할 경우, UDP는 TCP와는 달리 혼잡제어를 수행하지 않기 때문에 대역폭을 독점한다. 따라서 동일한 전송경로에서의 TCP 트래픽 궁핍을 일으킨다. 이러한 현상을 혼잡붕괴(congestion collapse)라 하며, 혼잡제어를 수행하는 TCP에 비해 상대적으로 UDP에 할당되는 채널 대역폭이 증가하기 때문에 발생하는 현상이다[1]. 이와 유사하게 실시간 멀티미디어 응용에 TCP 혼잡제어를 적용하면 멀티미디어 데이터의 서비스 품질이 크게 저하되는 문제점이 발생하는데, 이것은 송신측의 데이터 전송율이 급격히 변화될 경우에 전송 지연시간이 급속히 증가 또는 감소됨으로서 발생되는 현상이다. 이러한 문제점을 해결하기 위해 실시간 멀티미디어 데이터의 전송에 적합한 TCP 친화적 혼잡제어기법에 대한 다각적인 연구가 진행되고 있다.

본 논문에서는 인터넷과 같은 공유된 네트워크 환경에서도 안정적인 동작이 가능하도록 지연기반 square-root 혼잡회피기법을 적용한 하이브리드형 TCP 친화적 혼잡제어기법을 제안하였다. 본 논문의 구성은 다음과 같다. 먼저2장에서는 TCP 친화적 혼잡제어기법의 특징과 방법에 대해 기술하고, 3에서는 TEAR의 소개와 더불어 제안한 기법을 적용한 하이브리드형 TCP 친화적 혼잡제어기법에 대해 기술한다. 4에서는 성능 평가 시뮬레이션 방법과 결과에 대해 설명한다. 마지막으로 5장에서는 결론 대해 기술한다.

2. TCP 친화적 혼잡제어기법

2.1 윈도우 기반 혼잡제어기법

윈도우 기반 혼잡제어기법은 혼잡윈도우 관리를 통해 전송율을 조절하는 기법이다. 송신측은 혼잡윈

도우의 최대크기로 패킷을 전송할 수 있으며, 수신측은 수신한 패킷마다 Ack를 보냄으로써 Ack가 송신측의 전송율을 조절하게 한다. 이러한 윈도우 기반 혼잡제어기법에는 AIMD, GAIMD, Binomial 등이 있다.

2.1.1 AIMD 혼잡제어기법

송신측은 가용대역폭을 이용하기 위해 혼잡윈도우 크기를 선형적으로 증가시키고, 혼잡 지시자인 패킷 손실이 발생할 때는 혼잡윈도우 크기를 절반으로 감소시킨다. AIMD 알고리즘은 식 (1)과 같다.

$$I: w_{t+R} \leftarrow w_t + \alpha \quad ; \quad \alpha > 0$$
$$D: w_{t+\delta t} \leftarrow (1-\beta)w \; ; \quad 0 < \beta < 1 \qquad (1)$$

(단, I : 혼잡 윈도우 크기의 증가,D: 혼잡 윈도우 크기의 감소[2].)

2.1.2 GAIMD 혼잡제어기법

GAIMD는 AIMD 기법을 확장하여 일반화한 것이다. GAIMD가 TCP 친화적이기 위해서는 $\alpha(0.31)$와 $\beta(7/8)$가 식 (2)와 같은 관계를 가져야 한다[3].

$$\alpha = \frac{4(1-\beta^2)}{3} \qquad (2)$$

2.1.3 Binomial 혼잡제어기법

Binomia 혼잡제어기법은 선형 혼잡제어를 수행하는 AIMD와 달리 비선형 혼잡제어를 수행하여 가용대역폭을 보다 빨리 사용할 수 있는 기법이다. Binomial 알고리즘은 식 (3)과 같다.

$$I: w_{t+R} \leftarrow w_t + \alpha / w_t^k \; ; \quad \alpha > 0$$
$$D: w_{t+\delta t} \leftarrow w - \beta w_t^l \quad ; \quad 0 < \beta < 1 \qquad (3)$$

여기서 Binomial 기법이 TCP 친화적이 위해서는 적절한 α, β 값과 $k+l=1$, $l<=1$의 조건을 만족하면 된다[4].

2.2 율 기반 혼잡제어기법

율 기반 혼잡제어기법은 혼잡을 지시하는 네트워크 피드백 기법을 이용하여 동적으로 전송율을 조절하는 기법으로 TCP 모델링 방정식을 이용하는 TFRC(TCP Friendly Rate Control) 등이 있다.

2.2.1 TCP 모델 방정식

TCP 처리율은 왕복시간 t_{rtt}, 재전송타임아웃t_{rto}, 패킷크기 s, 패킷손실율 p를 근거로 하여 식 (4)과 같이 구할 수 있다.

$$T = \frac{s}{t_{rtt}\sqrt{\frac{2p}{3}} + t_{rto} min\,(1.3\sqrt{\frac{3p}{8}})\,p(1+32p^2)} \qquad (4)$$

2.2.2 TFRC

TFRC는 TCP 모델링 방정식을 이용하여 전송율을 조절하는 방법이다. 율 기반 혼잡제어를 수행하기 위해 식 (4)를 사용하기 위해서는 패킷 손실율을 정확히 측정하는 것이 매우 중요하다. 이에 따라 TFRC는 연속적인 패킷손실 간의 패킷 수를 측정하는 평균손실간격(Average Loss Interval)과 EWMA기법을 이용하여 패킷 손실율을 측정한다. TFRC 수신측은 매 RTT마다 TCP 방정식에 필요한 파라미터를 업데이트하여 송신측에 보내고 송신측은 업데이트된 정보를 이용하여 계산한 전송율로 패킷을 전송한다. TFRC는 지연기반 혼잡회피(delay-based congestion avoidance)를 사용하여 프로토콜의 성능을 향상시키고 있으며, 대역폭을 놓고 서로 경쟁하는 트래픽에 대해 충분한 반응을 보임으로써 상대적으로 안정된 전송율을 유지할 수 있다[5][6].

3. 하이브리드 TCP 친화적 잡제어기법

TEAR(TCP emulation at receivers)는 윈도우 기반과 율 기반을 결합한 하이브리드 TCP -friendly 혼잡 제어 기법 중 대표적인 프로토콜[6][7]이며, 본 논문에서는 이것을 기반으로 TEAR 성능 개선을 위한 Square-root 혼잡회피 알고리즘을 제안하였다.

3.1 TEAR

TEAR 수신측은 혼잡윈도우 크기를 관리하고 이를 토대로 처리율을 계산한다. 또한, 계산된 처리율을 피드백을 통해 송신측에 전송하고 송신측은 이를 토대로 전송율을 결정한다.

3.1.1 TEAR 상태 전이

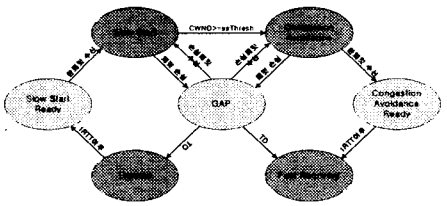

그림 1. TEAR 상태전이

3.1.2 혼잡윈도우 증가 알고리즘

Slow-start, Congestion-avoidance 상태에서 하나의 패킷을 수신하는 경우 혼잡윈도우 크기의 증가는 그림 2와 같은 동작에 의해 수행된다.

```
Void TEARSinkAgent::IncreaseWindow()
{
        switch(state){
                case SLOW_START:
                        cwnd=cwnd+1;
                        if(ssThresh<=cwnd)
                        cwnd=ssThresh;
                case CONGESTION_AVOIDANCE:
                        cwnd+=1/lastcwnd;
        }
}
```

그림 2. 혼잡윈도우 증가 알고리즘

여기서 lastcwnd는 매 round가 시작할 때마다 다음 round의 혼잡윈도우 증가를 계산하기 위해 현재

혼잡윈도우로 업데이트 된다.

3.1.3 혼잡 윈도우 감소 알고리즘

패킷 손실이 발생하면 Slow-start, Congestion -avoidance 상태는 Gap상태로 전이된다. Gap 상태에서는 혼잡윈도우크기가 변하지 않는다. Gap 상태는 수신측이 패킷 손실이 타임아웃에 의한 것인지, 세 개의 중복 Ack에 인한 것인지를 결정하는 중계상태이다. 그림 3은 혼잡윈도우크기 감소 알고리즘을 나타내고 있다.

```
Void TEARSinkAgent::DecreaseWindows()
{
        switch(state){
                case FastRecovery:
                        cwnd=cwnd/2;
                        ssThresh=cwnd;

                case Timeout:
                        cwnd=1;
                        ssThresh=cwnd;
        }
}
```

그림 3. 혼잡윈도우 감소 알고리즘

3.2. TEAR 성능 개선을 위한 지연 기반 Square-root 혼잡회피 알고리즘

TEAR 수신측은 TCP 톱니형태의 전송율 변화를 피하기 위해 매 RTT 대신 epoch단위로 처리율을 계산한다. Epoch는 연속적인 율 감소 사건 간의 시간으로 정의된다.

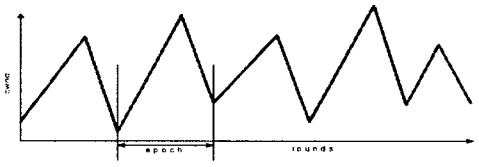

그림 4. 구간

그림 4는 epoch를 나타내고 있다. 노이즈에 의한 불필요한 율 변화를 방지하기 위해 지난 w개 epoch의 처리율에 대한 가중치 평균을 사용하여 평활화한 처리율을 계산하고, 계산된 처리율은 피드백을 통해 송신측에 보낸다. 송신측은 이를 토대로 전송율을 결정하기 앞서 인터넷과 같은 다양한 네트워크 상태에 적응하도록 지연기반 Square-root 혼잡회피기법을 도입하여 패킷간 전송간격 $t_{inter-packet}$를 식(5)와 같이 조절하였다.

$$t_{inter-packet} = \frac{S \times RTT_{current}}{T_{estimated}\sqrt{SRTT}} \qquad (5)$$

(단, $T_{estimated}$ 수신측 처리율, s:패킷크기)

제안한 기법은 현재의 네트워크 지연이 과거의 지연보다 크면 현재 네트워크 상태가 혼잡할 가능성이 크므로 이를 피하기 위해 전송간격을 길게 하였고, 그 반대의 경우 전송간격을 짧게하여 안정된 TEAR 성능을 유지하도록 하였다. 그림 5는 전체적인 동작 메카니즘을 나타낸 것이다.

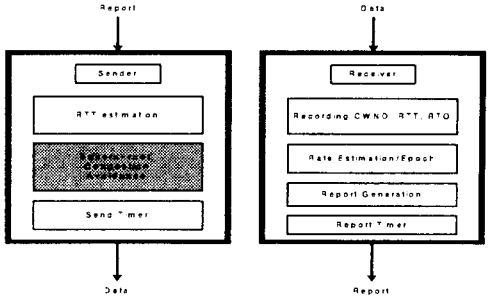

그림 5. 전체적인 동작 메카니즘

4. 시뮬레이션 결과

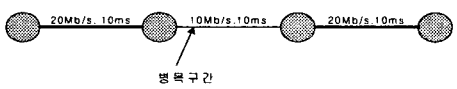

그림 6. 시뮬레이션 토폴로지

그림 6과 같은 시뮬레이션 토폴로지 환경에서 NS-2[8] 이용하여 제안된 기법의 성능을 테스트하였다. 성능평가를 위해 TCP와 제안된 TEAR의 총 전송율과 순간 전송율을 측정하였으며, 또한 동일한 조건(Feedback 1 RTT, DropTail, TCP-SACK)에서 기존의 TEAR와 제안된 TEAR의 성능(performance)을 비교분석하였다.

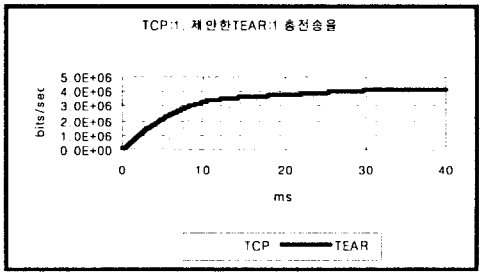

그림 7. 1개의 흐름을 갖는 TCP 및 TEAR의 전체 전송율 비교

그림 7부터 그림 10까지는 이미 알려진 TEAR의 실험환경과 동일한 조건에서 제안된 기법을 적용하였을 때 발생하는 성능평가 결과를 나타낸 것이다.

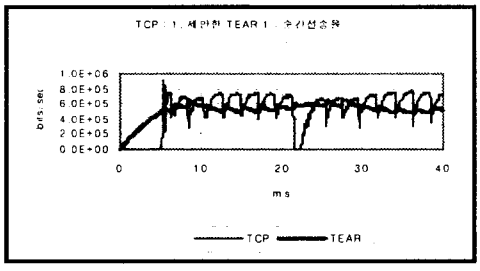

그림 8. 1개의 stream을 갖는 TCP 및 제안된 TEAR의 순간전송율 비교

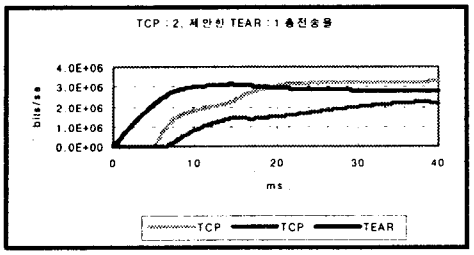

그림 9. 2개의 흐름을 갖는 TCP 및 1개의 stream을 갖는 제안된 TEAR의 총전송율 비교

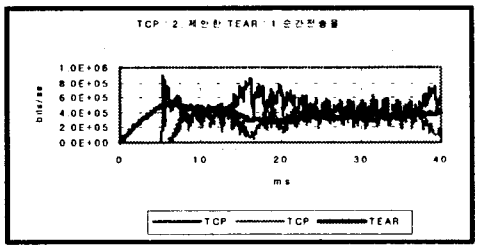

그림 10. 2개의 흐름을 갖는 TCP 및 1개의 stream을 갖는 제안된 TEAR의 순간전송율 비교

그림 7은 1개의 stream을 갖는 TCP와 1개의 흐름을 갖는 제안된 TEAR 간의 총전송율을 비교한 것이며, 그림 9는 TCP st흐름을 2개로 증가시켰을 때의 총전송율을 비교한 것이다. 그림 8은 각각 1개의 흐름을 갖는 TCP와 제안된 TEAR를 경쟁시켰을 때 순간전송율 변화를 비교하여 나타낸 것이고, 마찬가지로 그림 10은 2개의 흐름을 갖는 TCP와 1개의 스트림을 갖는 제안된 TEAR 간의 순간 전송율을 비교한 것이다.

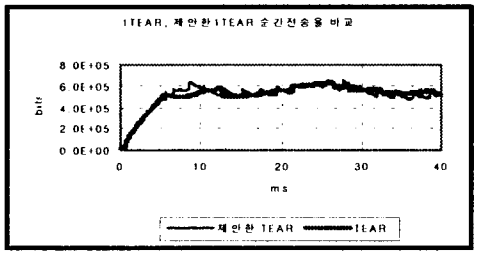

그림 11. 1개의 stream을 갖는 TEAR와 제안된 TEAR의 순간전송율 비교

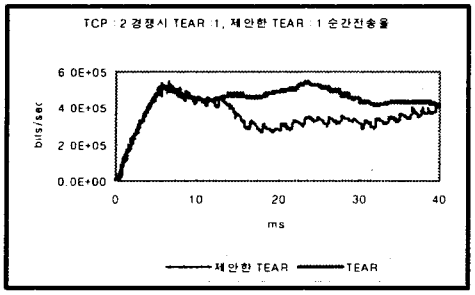

그림 12. 2개의 흐름을 갖는 TCP와 각각 1개의 스트림을 갖는 TEAR 및 제안된 TEAR의 순간전송율 비교

그림 11과 그림 12는 각각 TCP 및 기존 TEAR와 제안된 TEAR 간의 순간전송율을 비교한 결과이다. 여기서 알 수 있듯이 제안한 TEAR는 기존 TEAR에 지연기반 Square-root 혼잡회피기법을 적용하여 인터넷과 같이 공유된 네트워크 자원(shared network resources)을 경쟁하는 네트워크 환경하에서도 안정적으로 동작하여 거의 일정한 전송속도를 제공함으로서 멀티미디어와 연속적으로 끊기지 않고 일정한 속도를 제공해야하는 응용에 적절히 적용될 수 있음을 보였다.

5. 결론 및 향후과제

본 논문은 스트리밍 방식의 실시간 멀티미디어 전송을 위해 TCP 친화적인 혼잡제어기법을 제안 하였다. 전송계층(Transport Layer)에서 기존의 TCP와 경쟁할 경우라도 전송대역폭을 공정하게 나누어 사용할 수 있도록 TCP 동작을 적용하였고, 갑작스런 율 감소에 의해 멀티미디어 응용의 사용자 인지 품질이 저하되는 것을 피하기 위해 평활화한 전송을 수행할 수 있도록 epoch 단위로 처리율을 계산하였다. 또한 지연기반 Squre-root 혼잡회피기법을 적용하여 인터넷과 같은 공유된 네트워크 환경에서도 안정적으로 동작하도록 하여 기존에 제안된 TEAR보다 혼잡회피능력이 뛰어나도록 성능을 개선하였으며, 시뮬레이션을 통한 성능평가를 통해 이를 입증하였다.

6. 참고문헌

[1] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet", IEEE/ACM Transactions on Networking, Aug 1999.

[2] S. Jin, L. Guo, I. Matta, A. Bestavros, "A Spectrum of TCP-friendly Window-based Congestion Control Algorithms", July 2002.

[3] Y. Richard Yang and Simon S. Lam. "General AIMD congestion conrtol", In Proceedings of ICNP, November 2000.

[4] D. Bansal and H. Balakrishnam, "Binomial congestion control algorithms", In Proceedings of IEEE INFOCOM, April 2001.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications", Technical Report, ACIRI, Feb 2000.

[6] Network Simulator - ns-2
http:// www.isi.edu/nsnam/ns