TFRC 혼잡제어 프로토콜 공정성 개선 방안

조 경 연, 김 영 복, 장 주 욱
서강대학교 전자공학과
{kycho,virtuoso,jjang}@eeca1.sogang.ac.kr

Improving the Fairness of TFRC Congestion Control Protocol

Kyung-yon Cho,Young-bok Kim, Ju-Wook Jang
Dept. of Electronic Engineering, Sogang Univ.

요 약

TFRC 혼잡제어는 멀티미디어, 인터넷 전화등 실시간성이 강조된 서비스에 적합한 프로토콜이다. 그러나 이러한 TCP-friendly 혼잡제어 알고리즘은 네트워크 혼잡이 심해져 손실률이 커짐에 따라 TCP와의 공정성(fairness)이 떨어지는 문제점을 가지고 있다. 이러한 문제를 해결하기 위해서는 전송률을 결정하는 제어식을 높은 손실률일 때 보정을 해주어야한다. 본 논문에서는 TFRC 혼잡제어 알고리즘의 전송률 결정에 있어서 중요한 변수인 재전송 타임아웃 값(RTO)의 재 정의를 통하여 전송률을 보정함으로서 공정성 하락을 방지함을 보이고, 시뮬레이션을 통하여 이를 입증한다.

1. 서 론

현재 인터넷 프로토콜로 가장 많이 쓰이는 TCP는 급격한 전송률의 변화로 인해 시간 데이터 교환에는 부적합하여 UDP 프로토콜을 많이 사용하였다. 하지만 UDP는 혼잡제어 기능이 없기 때문에 TCP와의 공정성의 문제가 발생할 수 있다. 이에 실시간 전송에 적합하며 혼잡제어 기능을 가진 TFRC(TCP Friendly rate control)이 제안되었다.[1] 하지만 TFRC는 전송률의 변화가 평활(smoothness)적이기 때문에 낮은 손실률의 경우 공정성(fairness)을 유지하지만 높은 손실률의 경우 공격적으로 사용 가능한 대역폭을 찾는 TCP와의 공정성이 떨어지는 문제가 [2]에 제기되었다.

본 논문에서는 혼잡한 네트워크 상황에서 대역폭의 공정성 문제를 인식, TFRC 혼잡제어 알고리즘의 전송률을 결정하는 제어식의 개선을 통하여 공정성의 급격한 하락을 방지함을 보이고 이를 시뮬레이션을 통하여 입증한다.

2. TFRC (TCP-Friendly Rate Control) Protocol 개요

방정식 기반의 혼잡제어의 목적은 이용 가능한 대역폭을 발견, 이용하는 것이 아니라 상대적으로 안정적인 상태를 유지하면서 혼잡상태에 적응하는 것이다. 이러한 목적을 위해 현재 다수를 차지하는 TCP와 상응하는 모델링 방정식을 혼잡제어알고리즘으로 이용한다[3]

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)} \qquad (2\text{-}1)$$

(T : 전송률, s : 패킷 크기, R : round trip time, t_{RTO} : 재전송 타임아웃, p : 손실률)

TCP와 공정성을 유지하면서 안정적 전송을 위해서는 공격적으로 이용 가능한 대역폭을 찾는 대신 손실사건률(loss event rate)에 맞춰 전송률을 서서히 증가시키고 하나의 손실사건에 대해 전송률을 반으로 줄이는 대신 몇 개의 연속적인 손실사건이 발생한 경우 전송률을 반으로 줄인다. 수신단은 해당 간격에 패킷을 하나라도 받으면 RTT마다 적어도 한번의 피드백을 전송하여야하며, 송신단은 몇 번의 RTT 이후에도 피드백을 받지 않으면 전송률을 감소시켜 궁극적으로는 전송을 멈춘다.

2.1 송신단 기능

송신단에서 전송되어지는 패킷에는 시퀀스넘버(sequence number)와 보낸 시각이 찍혀있는 시간도장(time stamp)이 담겨있다. 이 시퀀스 넘버는 연속적으로 증가한다. 시퀀스 넘버 i 패킷의 시간도장을 t_i^s라하자. 시퀀스넘버가 1 이상인 모든 패킷에 대해 현재 RTT정보를 담는다. RTT_i를 i번째 패킷의 RTT로 가정하기로 하자. 현재 시간을 t라하고, 수신단에서 패킷 i를 받은 시각을 t_i^r 라 하면 t_i^d는 다음과 같이 정의하자.

$$t_i^d = t - t_i^r \qquad (2\text{-}2)$$

수신단은 피드백(feedback) 정보에 t_i^s 와 t_i^d를 담는다. t'를 송신단에 도착한 시각이라 가정하면 송신단은 RTT_s를 계산한다.

$$RTT_s = t' - t_i^s - t_i^d \qquad (2\text{-}3)$$

송신단은 다음 식으로 RTT를 갱신한다.[1]

$$RTT = \alpha * RTT + (1-\alpha) * RTT_s \quad (2\text{-}4)$$

α가 작으면 RTT가 패킷지연에 빠르게 응답하고 반면 크면 점진적으로 변화할 것이다. 송신단은 또한 TCP와 같이 RTT_s를 이용하여 T_0를 갱신한다. 매번 피드백이 돌아올 때마다 p,RTT, T_0를 이용하여 최근 전송율 T_n를 계산하여 현재 전송율 T가 T_n보다 작으면 다음 식으로 전송률을 높인다.[1]

$$T = \min(T + 1/RTT, T_n) \quad (2\text{-}5)$$

만약 T가 T_n보다 작으면 T_n으로 전송률을 감소시킨다.

2.2 수신단 기능

수신단은 송신단이 RTT를 계산하기 위한 피드백정보를 보낸다. 또한 p(loss event rate)를 계산하고 송신단에 보낸다. 손실사건의 계산에 대한 원칙으로, 손실률 계산치는 하나의 패킷손실을 측정하는 것이 아니라 손실사건률을 측정하는 것이다. 손실사건률은 한 RTT내 몇 개의 패킷 손실로 구성된다. 또한 손실률 계산치는 새로운 손실사건에만 반영되어야한다. 이전의 계산평균보다 큰 새로운 손실간격(loss interval) 혹은 지난 손실사건보다 충분히 큰 간격에서 손실률 계산치는 감소하여야한다.[1]

3. TFRC의 전송시 문제점및 개선방안

높은 손실률을 보이는 네트워크상황에서 TFRC 혼잡제어 프로토콜은 TCP보다 공정성이 크게 떨어진다.[2]따라서 높은 손실률(대략 0.1보다 큰 경우)일 경우 TCP와의 공정성을 형성하기 위해서 전송률의 보정이 필요하다. 그러므로 전송률 제어식에서 손실률이 증가함에 따라 가장 큰 영향을 갖는 t_{RTO}값을 변경함으로써 전송률을 보정한다. TCP에서 t_{RTO}의 값은 다음과 같이 정의된다.[1]

$$t_{RTO} = SRTT + 4 \times RTT_{var} \quad (3\text{-}1)$$

(SRTT : 예상되는 RTT 값, RTTvar : RTT의 분산)

전송률 계산 시 수신단의 feedback에 의해 알려진 손실률의 정도에 따라 t_{RTO}를 감소시킴으로써 높은 손실률이 발생하는 상황에서도 전송률과 공정성이 급격히 떨어지는 상황을 방지한다. 전송단은 수신단의 피드백정보에서 손실률(p)을 모니터링하여 10%이상의 손실률이 발생했을 경우, t_{RTO} (retransmit timeout)을 손실률에 비례해서 줄여나간다. 전송률 계산 시 재정의된 t_{RTO} (retransmit timeout)값의 대입으로 계산한다.

$$t_{RTO} = \frac{t_{RTO}}{(a + p)} \quad (p>0.1\text{일때}) \quad (3\text{-}2)$$

(p > 0.1 일 때)
a:상수 , p:손실률

위 식에 의하면 t_{RTO}는 손실률에 따라 변화함을 알 수 있다. 손실률이 커짐에 따라 기존의 t_{RTO}에서 더 많은 비율로 감소되어진다. 상수a는 실험에 의하여 1.4에서 가장 안정적인 결과를 보인다.

t_{RTO}의 값을 손실률이 증가함에 따라 적당하게 감소시켜 2-1식의 분모항의 전체의 값이 갑작스럽게 변하는 것을 억제하여 전송속도의 변화량을 줄임으로써 TCP와의 대역폭에서 공정성을 유지시킨다. 즉, 2-1의 전송 식에 3-1식의 t_{RTO}값을 넣지 않고 제안된 3-2의 t_{RTO}의 값을 넣음으로써 공정성을 개선하였다.

4. 시뮬레이션

제안된 방법의 성능을 확인하기 위해 NS(Network Simulator) 시뮬레이터를 사용하여 시뮬레이션 하였다.[4] 시뮬레이션을 위한 망의 배치는 TFRC를 이용하여 전송하는 노드와 TCP를 이용하여 전송하는 노드가 하나의 병목 링크를 공유할 수 있게 배치를 하였고 TFRC와 TCP의 노드 수를 같게 하고 각 노드 수를 1개부터 70개까지 증가시켜가며 병목링크에서의 손실률의 변화에 따른 공정성을 확인할 수 있게 하였다. 병목링크의 전송속도는 1Mbps이고 지연은 20ms로 설정하였고 드롭테일 큐(Droptail queue)를 사용하였다.

그림 1은 기존 TFRC와 TCP가 같은 병목지점을 공유할 때 손실률변화에 따른 전송률을 보인다. 각 포인트는 병목링크를 지나는 각 플로우의 일반화된 전체 평균 전송률이다. 손실률이 증가함에 따라서 병목링크에서 TFRC의 전체적인 평균전송률은 급격히 감소하게 된다. 반면 TCP의 전송률은 TFRC에 반해 상승하는 것을 볼 수 있다. 다시 말해 두 프로토콜간의 전송대역폭의 공정성은 크게 떨어지는 것을 볼 수 있다.

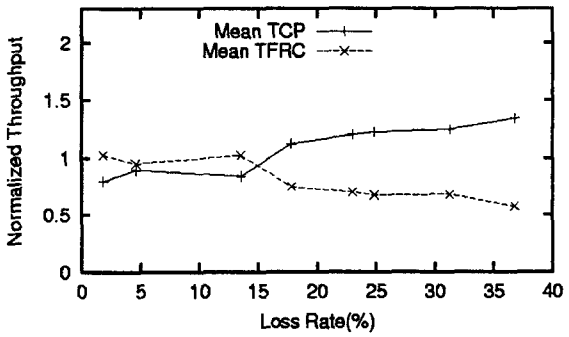

그림 1 기존 TFRC의 손실률에따른 throughput

그림 2는 본 논문에서 제안한 알고리즘을 TFRC에 대입했을 경우의 손실률변화에 따른 전송률을 보이고 있다. 그림에서처럼 손실률 변화에 따른 전송률은 기존의 두 프로토콜에서 보였던 급격한 변화를 보이지 않고 있고 전송대역폭의 공정성은 상당히 향상되어 안정된 모습을 볼 수 있다.

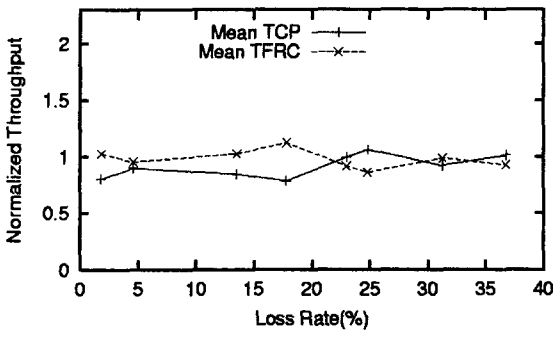

그림 2 개선된 손실률에따른 throughput

그림 3은 손실률에 따른 TFRC의 변동계수를 나타낸다. 각

포인트는 병목 링크에서 각 플로우의 변동계수이다. 변동계수에 대한 시뮬레이션을 위하여 32개의 TFRC와 TCP 플로우를 사용하였고 병목링크의 용량을 줄여 손실률을 증가시켜가며 각 플로우의 전송률에 대한 변동계수를 계산하였다. 그림 3에서 변동계수가 클수록 각 플로우 간의 전송률차이가 큰 것을 나타낸다. 따라서 기존 TFRC 플로우에 비하여 개선된 TFRC 플로우(그림3에서 TFRC_1.4)들 간의 전송률의 차이는 기존의 TFRC 플로우들에 비하여 낮아짐을 볼 수 있다. 다시 말해 제안 알고리즘이 적용한 개선된 TFRC의 경우 TFRC플로우들 간의 대역폭의 공정성 또한 향상된다.

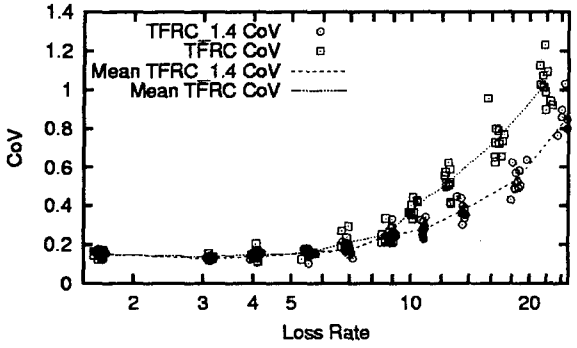

그림 3 TFRC의 변동계수

5. 결 론

본 논문에서는 혼잡한 네트워크 상황에서 TFRC의 문제점을 재시하고 그에 따른 성능저하를 방지하기 위하여 개선된 전송방법을 제안하였다. 혼잡한 네트워크 상황에서 TFRC는 기존의 TCP와의 전송대역폭 공정성이 상당히 떨어진다. 이런 공정성의 문제는 TFRC의 전송률계산에서 비롯되어지고 높은 손실률을 보이는 네트워크 상황에서 전송률은 재전송 타임아웃 변수 tRTO에의해 큰 영향을 받게된다. 이 tRTO를 적절한 값으로 재조정함으로써 전송대역폭의 공정성은 크게 향상된다.

참고 문헌

[1] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation -based congetion control for unicast applications. In Proceedings of ACM SIGCOMM 2000, August 2000.

[2] Y. Richard Yang, Min Sik Kim, Simon S. Transient Behaviors of TCP-friendly Congetion Control Protocols. In Proceedings of IEEE INFOCOM 2001.

[3] J. Padhye, V. Firoiu, D. Towsley, and J. Kurse: Modeling TCP throughput: A simple model and its empirical validation, In Proceedings of SIGCOMM'98, 1998.

[4] Steven Mccune and Sally Floyd, NS(network Simulator), http://www.isi.edu/nsnam/ns,1995.

[5] J. Padhye, J. KUrose, D. Towsley, and R. Koodli. A Model Based TCP-Friendly Rate Control Protocol. In Proceedings of NOSSDAV'99, 1999.