타임 스템프를 이용한 양방향 분리 TCP 혼잡 제어

조현주° 김정애 유기영
경북대학교 컴퓨터공학과 정보보호연구실
{i_hjcho°, sarah}@infosec.knu.ac.kr yook@knu.ac.kr

TCP Congestion Control using Timestamp

Hyun-Ju Cho° Jung-Ae Kim Kee-Young Yoo
Dept. of Computer Engineering, Kyungpook National University.

요 약

TCP Vegas나 FAST TCP는 RTT(Round Trip Time) 동안 실제 처리량(actual throughput)이 기대 처리량(expected throughput)보다 특정 임계값보다 더 작으면 혼잡 상황으로 판단하고 송신측에서 내보내는 데이터 양을 감소시킴으로써 혼잡을 제어한다. 그러나 RTT 기반의 처리량(Throughput) 측정은 송수신 경로가 다를 경우 양방향 경로의 혼잡 상황을 구분하지 않는 문제점을 가지고 있다. 따라서 본 논문에서는 TCP Timestamp를 이용하여 양방향 혼잡 상황을 구분하여 혼잡을 제어하는 메커니즘을 제안한다. 그리고 제안한 방식에 대한 성능 분석을 위해 NS-2의 TCP Vegas를 수정하여 시뮬레이션한 결과를 제시한다.

1. 서 론

인터넷이 보급된 이후 인터넷 사용자들이 지속적으로 증가되어 왔으며, 사용자들이 주고받는 데이터 형태도 단순 텍스트뿐만 아니라 용량이 큰 멀티미디어 데이터 등 다양해졌다. 이렇게 인터넷을 통해 전송되는 데이터의 양이 증가함에 따라 망의 혼잡을 제어하는 메커니즘이 더욱 중요하게 되었다.

망의 혼잡 상황을 제어하기 위한 프로토콜로 TCP Tahoe[1], TCP Reno[1], Fast Reno[2], TCP Vegas[3], FAST TCP[4] 등이 연구되어 왔다. 또한 TCP Reno에서 혼잡 상태를 잘못 감지한 것을 TCP Timestamp를 이용하여 감지하는 Eifel Detection 알고리즘[5]과 잘못된 감지로 인하여 혼잡제어 상태에 들어간 소스 측을 혼잡 제어 직전 상태로 복원시키는 Eifel Response 알고리즘[6]도 연구되었다.

TCP Vegas는 직접적인 loss를 감지해야 혼잡 상황을 탐지하는 TCP Tahoe나 TCP Reno와 달리, RTT(Round Trip Time) 동안 실제 처리량(actual throughput)이 기대 처리량(expected throughput)보다 특정 임계값보다 더 낮으면 혼잡 상황으로 판단하고 송신측의 데이터 송신양을 선형적으로 감소시킴으로써 혼잡을 제어한다. 하지만 TCP Vegas에서의 혼잡 제어방식은 RTT를 기반으로 처리량을 계산하기 때문에 송신 방향의 혼잡 상황과 수신 방향의 혼잡 상황을 구별하지 못한다. 따라서 양방향 경로 모두가 혼잡 상황이 아닌 경우에도 양방향 모두를 혼잡 상황으로 판단하게끔 한다. 예를 들어, 송신 방향의 경로가 혼잡하지 않은데 수신 방향의 경로가 혼잡하여 응답이 지연될 경우 송신측은 망이 혼잡한 것으로 판단하고 혼잡 제어용 윈도우 크기(CWND)를 줄인다. 따라서 망의 전체 처리량(Throughput)이 떨어지게 된다.

본 논문에서는 TCP Timestamp를 이용하여 송신 방향의 전송시간을 측정한 다음, RTT를 기반으로 한 기대 처리율과 실제처리율의 차와 전송시간을 기반으로 한 기대 처리율과 실제처리율의 차를 비교하여 혼잡한 경로를 구분함으로써 송신측이 혼잡을 제어할 수 있도록 하는 메커니즘을 제안한다. 이 메커니즘을 사용해 수신 방향의 경로가 혼잡한 경우 타임아웃 값을 늘려서 패킷 재전송을 시작하는 시간을 늦춤으로써 잘못된 재전송으로 인해 망의 트래픽이 낮아지는 것을 막았다. 이렇게 함으로써 전체 망의 처리량이 높아질 수 있도록 한 것이다. 제안하는 메커니즘은 TCP Vegas나 FAST TCP에 적용할 수 있는데 본 논문에서는 TCP Vegas에 적용하여 시뮬레이션 한 결과만을 보여준다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 연구에 대하여 살펴보고, 3장에서는 Timestamp를 이용한 혼잡 제어 방식을 논하고, 시뮬레이션 결과를 살펴본 다음, 4장에서 결론 및 향후 연구 과제를 제시한다.

2. 기존 연구

혼잡을 제어하는 메커니즘들은 크게 소스 기반 메커니즘과 링크 기반 메커니즘으로 나뉘어지는데, 여기에서는 소스기반 알고리즘인 TCP Vegas와 FAST TCP에 관해서만 살펴본다.

2.1 TCP Vegas

TCP Vegas는 큐잉 지연을 기반으로 하는 TCP 변종으로, 식(1)과 같이 RTT(Round Trip Time)동안의 평균

처리량(Average throughput)을 토대로 예측 처리량(Expected throughput)을 계산해서, 실제 처리량(Actual throughput)이 예측 처리량보다 임계값 이상 낮으면 망이 혼잡한 것으로 판단하고 송신측에서 내보내는 데이터의 양을 1씩 감소시켜서 혼잡을 제어한다.

$$w_i(t+1) = W_i(t) + \frac{1}{T_i(t)} sign(a - x_i(t)q_i(t)) \quad \text{식(1)}$$

where $w_i(t)$: window size of flow i in the current update period t

$x_i(t)$: current throughput

$q_i(t)$: current queuing delay

$T_i(t)$: current RTT

a_i: protocol parameter

이렇게 선형적으로 혼잡을 제어하는 방식은 망의 혼잡과 정상 상태에 대한 적응 속도가 느리다는 단점을 가지고 있다.

2.2 FAST TCP

FAST TCP는 TCP Vegas의 고속 버전으로, 가장 큰 특징은 망이 혼잡 하게 되었을 때 혼잡 정도에 따라 혼잡 제어 윈도(CWND)의 크기를 식(2)와 같이 조절함으로써 망의 혼잡 상황에 대한 적응 속도가 TCP Vegas보다 빠르다는 장점을 가지고 있다.

$$w_i(t+1) = w_i(t) + \gamma(a_i - x_i(t)q_i(t)) \quad \text{식(2)}$$

where γ: constant between 0 to 1

그러나 TCP Vegas와 FAST TCP 둘 다 RTT 시간 동안의 실제 처리량이 예측 처리량보다 임계값 이상 낮으면 망이 혼잡한 것으로 판단한다. 송수신 양방향 중에 어느 방향이 혼잡 상황인지에 대한 구별을 하지 않는다. 하지만 데이터의 송수신 경로가 다를 경우 어느 한쪽 방향만 혼잡 상황이 발생할 수도 있다. 그런데 TCP Vegas와 FAST TCP에서는 RTT 기반으로 실제 처리량(Actual throughput)을 측정하므로 그러한 혼잡 상황에 대한 정확한 구분을 할 수가 없다. 예를 들어, 노드 A가 노드 B에게 데이터를 전송하려고 할 때, A에서 B로 데이터가 전달되는 경로와 응답이 수신되는 경로가 다를 수 있다. 이때 데이터 전달 경로가 혼잡하지 않지만 응답이 수신되는 경로가 혼잡한 경우 노드 A에서는 RTT 시간이 길어지게 되면 혼잡 제어 윈도우(CWND)의 크기를 줄여서 노드 B로 보내는 데이터 양을 줄이게 된다.

따라서 본 논문에서는 TCP의 Timestamp를 이용하여 송수신 양방향에 대한 혼잡 상황을 구분하여 혼잡을 제어함으로써 망의 전체 처리량(Throughput)을 높일 수 있는 메커니즘을 제안하고자한다.

3. 제안하는 혼잡 제어 메커니즘

본 절에서는 Timestamp를 이용하여 송수신 시간을 구별하여 측정을 함으로써 송신방향과 수신방향의 혼잡 상황을 구분하고자 한다. 이러한 구분을 기초로 송신방향이 혼잡할 때는 송신측 혼잡 제어 윈도우(CWMD) 크기를 조절하고 수신방향이 혼잡할 때는 TimeOut 시간을 조정할 수 있게 함으로써 혼잡을 제어하고 잘못된 재정송(Spurious Retransmission)을 막는다.

3.1 Timestamp를 이용한 송신 시간 측정

송수신측 간의 시간 동기화는 이루어져 있는 것으로 가정하고, 전송하는 패킷의 Timestamp에는 송신측의 송신시간(S.SendTime)과 응답하는 상대방 패킷의 수신시간(S.RecvTime)을 포함시켜 보낸다. 수신측도 자신이 전송하는 패킷의 송신시간(R.SendTime)과 응답하는 상대방 패킷의 수신시간(R.RecvTime)을 포함시켜 보낸다. 송신측은 도착하는 응답 패킷의 도착 시간(S.Arrived Time)을 측정한 다음, 식(3)를 이용하여 데이터 전송시간(ST)을 측정한다.

ST = R.RecvTime -S.SendTime 식(3)

그리고 평균 송신시간을 이용해, 송신 방향 현재 처리량(x_i)을 구한다.

3.2 제안하는 메커니즘의 혼잡 제어 윈도우 크기 조절

TCP Vegas에서 예측 처리량과 실제 처리량의 차가 특정임계값(beta)값 보다 큰 경우 혼잡상황으로 판단할 때, 제안하는 메커니즘은 다음과 같이 혼잡을 제어한다.

[제안하는 혼잡 제어 메커니즘]

```
....
delta=int((expect-actual) * baseRTT + 0.5);
Send_delta=int((Send_expect-Send_actual)*baseST
            +0.5);
...
// congestion avoidance
if(delta>v_beta_) {
    if (Send_delta < delta)
        ++cwnd_;
    else
        --cwnd_;
    if(cwnd_<2) cwnd_ = 2;
        v_incr_ = 0;
}
```

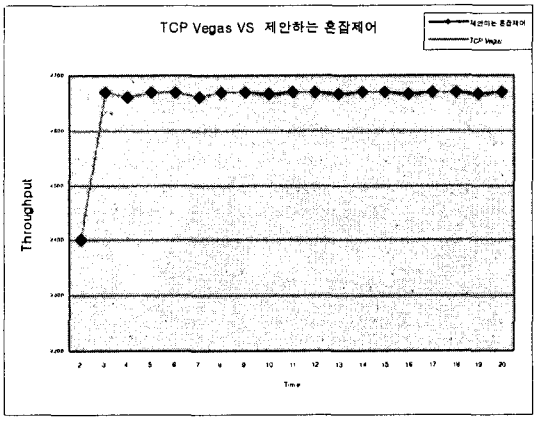
먼저, Timestamp를 이용하여 식(3)과 같이 송신 시간을

측정함으로써 송신 방향의 예측 처리량(Expected Through put)과 실제처리량(Actual throughput)의 차(Send_delta)를 구해서 RTT동안의 예측 처리량과 실제 처리량의 차(delta)보다 작으면 수신방향은 혼잡하지만 송신방향은 혼잡하지 않은 것으로 판단하여 송신 방향의 혼잡 제어 윈도우(CWND)의 크기를 Vegas와 같이 줄이지 않고 오히려 증가 시켜서 송신 방향의 처리율을 증가시켰다.

TCP Vegas에서는 RTT를 기반으로 Timeout 값을 조절하게 되는데, 이 Timeout 값이 TCP Reno에 비해서 짧다. 따라서 송신측에서 송신방향의 트래픽은 낮지만 수신 방향의 트래픽이 높아서 응답이 지연될 경우, 짧은 Timeout 값으로 인해 잘못된 재전송을 수행함으로써 망의 트래픽은 높이고 전체적인 처리율은 떨어뜨릴 수 있다. 따라서 제안하는 메커니즘에서는 Timeout값을 TCP Vegas보다 높임으로써 잘못된 재전송을 줄이고자 하였다.

3.3 시뮬레이션 결과

본 시뮬레이션은 NS-2를 사용하여 이루어졌으며, 6개의 노드로 구성된 망을 가정하고 소스측의 처리량(Throughput)을 측정하였다. 이때 소스에서 목적지까지의 송신 경로와 목적지에서 소스까지의 수신 경로가 서로 다른 것으로 가정하였다. 또한 양방향 간의 전송률을 비대칭적으로 설정하여 시뮬레이션을 수행하였다. 시뮬레이션의 수행결과가 그림1과 같이 나타났다. 그림1에서 보는 바와 같이, TCP Vegas에서의 처리량(Throughput)보다 아주 조금 나아졌음을 확인할 수 있다.



[그림 1] TCP Vegas와 제안하는 혼잡제어

[처리량 축소율 100:1]

4. 결론 및 향후 과제

본 논문에서는 Timestamp를 이용하여 송수신 방향의 혼잡을 구분해서 소스 측의 혼잡제어 윈도우의 크기를 조절할 수 있는 혼잡 제어 메커니즘을 제안하였다.

송수신 방향의 경로와 더불어 대역폭도 차이가 나는 경우 혹은 다양한 지연(Delay spikes)을 가지는 무선망(Ad-hoc Network)에서 제안하는 메커니즘을 사용하여 혼잡 제어 윈도우를 조절하면 성능이 더욱 향상 될 것으로 기대한다. 따라서 향후 보다 다양한 모델 설정하여 시뮬레이션 해봄으로써 제안하는 메커니즘의 성능을 살펴보고자 한다.

참고문헌

[1] K. Fall and S. Floyd. Simulation-based Comarisons of Tahoe, Reno, and SACK TCP,Computer Communication Review, 26(3):5-21, July 1996.

[2] S. Floyd,T. Henderson,"The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582

[3] Lawrence S. Brakmo and Larry L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet" IEEE Journal on selected areas in communications, Vol. 13. No 8. October 1995

[4] C. Jin, D. X. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, S. Singh., "FAST TCP: From Theory to Experiments", IEEE Network, 19(1): 4-11, January/February 2005.

[5] R. Luwig, and M. Meyer, "The Eifel Detection Algorithm for TCP", RFC3522, April, 2003.

[6] R. Luwig, and A. Gurtov, "The Eifel Response Algorithm for TCP", RFC 4015, February, 2005.