Efficient Adaptive RED 라우터 버퍼 관리 알고리즘 디자인과 구현

이종현[O] 임혜영 허의남 황준 김영찬
{saksi, huh, hjun}@swu.ac.kr, {naked97, yckim1}@sslab.cse.cau.ac.kr

Design and Implementation of Efficient Adaptive RED Router Buffer Management Algorithm

Jonghyun Lee[O], Youngchan Kim
Dept. of Computer Science and Engineering, ChungAng University, Seoul, Korea
Hyeyoung Lim, Euinam Huh, Jun Hwang
Dept. of Computer Science, Seoul Women's University, Seoul, Korea

요 약

RED(Random Early Detection) 라우터를 성공적으로 배치하기 위해서는 RED 알고리즘을 구성하는 각 파라미터를 적절히 조절할 수 있어야 한다. 특히 다수의 TCP 연결이 하나의 라우터를 공유하는 네트워크 병목구간에서는 그 중요성이 한층 강조된다. 그러나 RED가 TD 라우터와 같은 네트워크 퍼포먼스를 유지하면서 TCP 커넥션 간 페어니스(fairness)를 향상시키기 위해서는, 네트워크 상황에 따라 다수의 컨트롤 파라미터 값을 적절하게 설정해줘야만 한다. 문제는 다양한 네트워크 환경에서 효과적으로 RED 알고리즘이 동작하기 위해 파라미터를 설정하는 것이 매우 어렵다는 것이다. 본 논문에서는 네트워크 상황에 따라 동적으로 RED 파라미터를 조절하면서 빠르게 안정적인 상태로 적응하는 진보된 RED 알고리즘인 ea-RED(Efficient Adaptive RED)를 디자인하고 구현하여 알고리즘의 효율성을 확인한다.

1. 서 론

네트워크 혼잡을 해결하거나 피하기 위해서 다양한 연구들이 과거로부터 지속되어 왔다. 이 중 하나가 인터넷 통신의 90% 이상을 차지하는 통신 프로토콜인 TCP(Transmission Control Protocol)의 행동 특성을 밝혀내고 효율적인 퍼포먼스를 수행할 수 있도록 알고리즘을 제안하는 것이고, 다른 하나는 다수의 TCP 커넥션이 경쟁하는 병목구간에서 인터넷 라우터 버퍼를 능동적으로 관리하는 다양한 AQM(Active Queue Management) 알고리즘을 연구하여 네트워크 퍼포먼스와 페어니스(Fairness)를 향상시키는 것이다. RED(Random Early Detection)[1]는 AQM 알고리즘 중의 하나이다. 전통적인 TD(Tail-Drop) 라우터는 버퍼가 꽉 찬 이후 도착하는 패킷들을 단순히 드롭(Drop)시키는데 비해 RED 라우터는 초기 설정된 네트워크 파라미터 값에 따라서 라우터 버퍼가 꽉 차기 전 주기적으로 특정 확률에 따라 패킷들을 드롭시킨다. RED 알고리즘의 초기 목적은 특정 TCP 커넥션의 패킷들만이 드롭되는 것을 방지하기 위함이었으나, 현재는 다양하게 TCP 커넥션의 패킷들을 드롭시킴으로써 기존의 TD 라우터보다 TCP 커넥션 간 페어니스를 향상시킬 수 있는 방법으로 사용되고 있다. 그런데 RED 라우터가 TD 라우터와 같은 네트워크 퍼포먼스를 유지하면서 TCP 커넥션 간 페어니스를 향상시키기 위해서는 link bandwidth, active 커넥션 수, congestion level 등에 대한 네트워크 상태를 고려하여 파라미터에 적절한 값을 설정해야만 한다. 문제는 다이내믹하게 변하는 네트워크 상황에 적합한 파라미터 값을 초기에 설정해 주는 것이 매우 어렵다는 점이다[2,3]. [2,3] 논문에서 저자들은 이러한 문제 때문에 인터넷에 RED Router를 실제 배치하기에 많은 어려움이 있다고 결론 내렸다. 이 후, RED 라우터에 적합한 파라미터 값 설정 문제를 해결하기 위해서 큐 길이, active 커넥션 수 등에 따라 다이내믹하게 파라미터 값이 변화하는 몇몇 알고리즘들이 제안되었다[4-6]. [4-6] 논문에서 저자는 패킷 드롭 확률인 p의 다이내믹한 컨트롤에 포커스를 맞추었으나 p값의 빠른 변화는 TCP 커넥션들의 퍼포먼스를 저하시켰다. 왜냐하면 TCP Connection의 처리량이 Packet Loss Ratio에 의해 직접적으로 영향을 받기 때문이다[7].

본 논문에서는 다이내믹하게 변화하는 네트워크 상태에 따라 threshold 값을 변화시킴으로써 빠르게 네트워크 상황에 적응하는 ea-RED(Efficient Adaptive RED)를 제안한다.

2. TD and RED 라우터

역사적으로 인터넷 라우터는 TD(Tail-Drop) 방식을 사용해 왔다. TD 버퍼 관리(Management) 매커니즘은 FIFO 방식으로 Send 커넥션에서 전달된 Packet을 Rec-

eive 커넥션으로 전송하고, 버퍼가 패킷들로 꽉 찰 경우에는 버퍼에 여유 공간이 생기기 전까지 새롭게 도착하는 패킷들을 모두 드롭시키게 된다. TD 매커니즘의 문제점은 바로 이 부분에 있다. 네트워크 전송량이 갑작스럽게 증가할 때만 패킷 드롭이 발생하기 때문에, 결과적으로 fast retransmit 알고리즘이 타임아웃 만료(expirations) 회피를 위해 사용될 수 없으며, 글로벌 동기화(synchronizat-ion) 문제[8]를 야기시킨다. 결국 TD Router는 경쟁적인 TCP 커넥션들간에 높은 처리량(Throughput)과 페어니스를 제공할 수 없다.

RED 알고리즘은 TCP 커넥션들의 혼잡제어를 위해 디자인 되었다. RED 알고리즘은 라우터의 평균 큐 길이를 모니터링함으로써 혼잡이 시작됨을 발견하고, 특정 확률에 따라서 전체 TCP 커넥션들이 보내오는 패킷을 공평하게 드롭시키게 된다. 평균 큐 길이를 낮게 유지함으로써 라우터에 입력되는 패킷의 수가 폭발적으로 증가하여 한계 큐 길이를 넘어서게 되어 새롭게 도착하는 패킷을 모두 드롭시키는 상황을 미연에 방지하게 된다.

RED 알고리즘의 대표적인 파라미터는 avg, max_{th}, min_{th}, w_q, max_p 이다. avg 파라미터는 빠르게 변화하는 현재 큐 길이를 완만하게 반영하기 위해 사용되는 평균 큐 길이다. max_{th} 파라미터는 드롭 확률을 계산하는데 사용하는 최대 threshold 값이고, min_{th} 파라미터는 드롭 확률을 계산하는데 사용하는 최소 threshold 값이다. w_q 파라미터는 avg 계산을 위한 가중치 값이다. max_p 파라미터는 최대 드롭 확률을 설정하기 위해 사용되는 파라미터 값이다.

패킷 드롭 확률은 다음과 같이 세 가지 방식에 의해서 결정된다.

- If $avg < min_{th}$, 패킷 드롭 확률 0%
- If $min_{th} < avg < max_{th}$, 패킷 드롭 확률 p_{red}
  $p_{red}(avg) = ((avg - min_{th}) * max_p) / max_{th} - min_{th}$
- If $max_{th} < avg$, 패킷 드롭 확률 100%

RED 알고리즘은 fast retransmit algorithm을 사용하여 retransmission timeout을 방지하고, 글로벌 동기화 문제를 해결할 수 있다.

그러나 RED 알고리즘의 가장 큰 문제는 다양한 네트워크 환경에 따라 일관되게 파라미터 값을 사용할 수 없고, 올바른 파라미터 값 설정을 위해 많은 노력이 필요하다. 만약 RED 라우터에 적합한 파라미터 값이 설정되지 않는다면, TD 라우터 보다 낮은 성능을 발휘하기도 한다. 또한 TCP 커넥션의 수가 많지 않을 경우에는 페어니스 인덱스 (Fairness Index)[9]가 TD 알고리즘 보다 높게 측정되지만, TCP 커넥션의 수가 많아지고 전송량이 많아질수록 TD 알고리즘 보다 낮은 성능을 발휘한다.

3. ea-RED

3-1. 제안동기

네트워크 상황에 따라 적합한 파라미터 값을 설정할 경우, RED 라우터는 모든 TCP 커넥션에 대해서 TD 라우터 보다 공평하다. 그러나 다이내믹한 네트워크 환경에서 적합한 RED 파라미터 값을 설정하기란 쉽지 않다. 특히 max_{th}, min_{th}, max_p 파라미터의 경우는 더욱 그렇다. 이런 문제점을 해결하기 위해, 패킷 드롭 확률인 p의 다이내믹한 컨트롤에 포커스를 맞추는 연구가 진행되었다. 하지만, p값의 빠른 변화는 TCP 커넥션들의 퍼포먼스를 저하시켰다.

한편 avg 값이 max_{th}값과 min_{th}값 사이에 있을 때 가장 좋은 성능을 낸다는 사실에 기초한 dt-RED[10] 알고리즘이 제안되었다. dt-RED는 max_{th}를 다이내믹하게 조절함으로써 이전 알고리즘들에 비해 많은 개선을 이루었으나, max_{th} 값의 변화가 observation interval에 의존하기 때문에 현재 큐의 변화에 가장 적합한 max_{th} 값과 min_{th} 값을 신속하게 찾아내는데 어려움이 있다. 이외에도 알고리즘 특성상 min_{th} 값의 변화가 거의 없고, max_{th} 와 min_{th} 값의 최고, 최저 값 설정이 효율적이지 못하다.

3-2. 알고리즘

ea-RED 알고리즘에서는 두 개의 threshold 값인 max_{th}와 min_{th} 값을 네트워크 상황에 따라 적절하게 설정하기 위해서 avg 파라미터와 현재 큐의 길이를 나타내는 $curq$ 파라미터를 모니터한다. avg 파라미터 값의 변화에 따라 max_{th}와 min_{th} 값을 다이내믹하게 변화시켜줌으로써, avg 파라미터 값이 지속적으로 max_{th}와 min_{th} 값 사이에 위치할 수 있도록 한다. 그리고 $curq$ 길이가 물리적인 버퍼 사이즈를 초과할 경우에도 max_{th}와 min_{th} 값을 다이내믹하게 변화시면서 avg 파라미터 값의 변화만으로 반영할 수 없었던 네트워크 상황을 정확히 반영하고 빠르게 적응할 수 있도록 한다. 한편 observation interval을 설정하지 않음으로써, 효율적으로 네트워크 상황에 적응할 수 있고 이미 지나간 상황을 뒤늦게 반영하는 부작용을 제거한다. 우리는 또한 p 파라미터 값을 직접적으로 조절하지 않고, threshold 값들을 조절함으로써 간접적으로 p 값을 좀 더 완만하게(smoothly) 조절한다.

세부 threshold 컨트롤 알고리즘은 다음과 같다.

```
if ((avg >= minth && curq > 1) && (avg >= maxth)){
    if(maxth < maxlim) maxth += maxth * αmax;
    else maxth = maxlim;
    if(minth < maxth) minth += minth * βmax;
    else minth -= maxth * γmin;
}else{
    if(maxth > minth) maxth -= maxth * αmin;
    else maxth = minth * γmax;
    if(minth < minlim) minth -= minth * βmin;
    else minth = minlim;}
```

```
if(curq >= qlim){
    if(maxth < maxlim) maxth += maxth * αmax;
    else maxth = maxlim;
    if(minth < maxth) minth += minth * βmax;
    else minth = maxth * γmin;}
```

max_{lim}과 min_{lim} 파라미터 값은 각각 max_{th}와 min_{th} 파라미터 값의 한계값을 나타낸다. α_{max}와 β_{max} 파라미터 값은 각각 max_{th}와 min_{th} 파라미터 값의 증가비율을 나타낸다. α_{min}와 β_{min} 파라미터 값은 각각 max_{th}와 min_{th} 파라미터 값의 감소비율을 나타낸다. γ_{max} 파라미터 값은 max_{th} 값이 감소하여 min_{th}값 이하로 내려갈 경우 min_{th}의 몇 배 크기로 max_{th}값을 설정할 것인가를 나타낸다. γ_{min}파라미터 값은 min_{th} 값이 증가하여 max_{th} 값 이상으로 올라갈 경우 min_{th} 값을 max_{th}의 일정한 비율로 감소시키기 위해 설정한다.

3-3. 알고리즘 구현

동일한 트래픽 환경에서 측정한 RED와 ea-RED 파라미터 값들의 변화는 그림 1에서 확인할 수 있는 바와 같다.

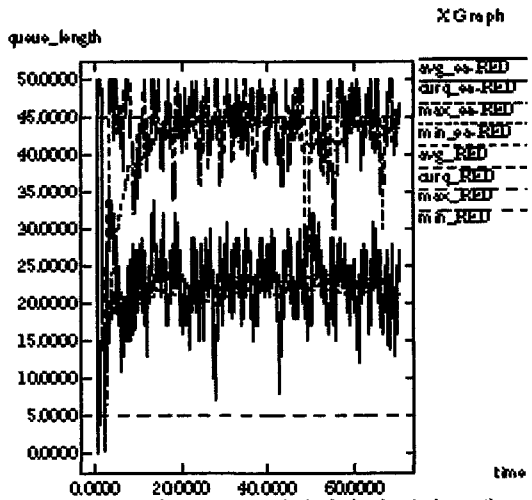

그림 1: RED와 ea-RED 파라미터 값 변화 그래프

입력 TCP 커넥션의 수는 21개이고, 70초간 실험을 진행하였다. $\alpha_{max}=\beta_{max}=\alpha_{min}=\beta_{min}=0.05$, RED의 max_{th} 파라미터 값은 45, min_{th} 파라미터 값은 5, ea-RED의 max_{th} 파라미터 값은 15, min_{th} 파라미터 값은 5로 설정했다. 실험결과 ea-RED 알고리즘을 사용한 라우터가 RED 알고리즘을 사용한 라우터에 비해서 전체 버퍼의 약50% 정도만을 사용했지만, 실험시간 동안 라우터에서 도착지로 송신한 총 패킷 수는 ea-RED가 근소하게(3패킷) 많았다.

4. 결론

본 논문에서는 RED 알고리즘의 개선된 형태인 ea-RED 알고리즘을 제안하였다. ea-RED는 RED의 threshold 파라미터 값을 네트워크 혼잡상황에 따라 다이내믹하게 변경할 수 있기 때문에, 초기에 RED의 max, min threshold 파라미터 값을 설정하는데 소요되는 오버헤드를 줄일 수 있고 또한 빠르게 네트워크 병목 상황에 적응할 수 있다. ea-RED는 dt-RED와 비교하여 max threshold 뿐만 아니라 min threshold도 네트워크 상황에 따라 보다 능동적으로 변화하기 때문에 *avg* 파라미터를 max threshold와 min threshold 사이에서 최대한 유지될 수 있도록 디자인하였다. 또한 *curq* 파라미터 값을 추가하여, dt-RED 보다 빠르게 다이내믹한 네트워크 환경에 적응할 수 있고, *avg* 파라미터가 완만하게 변화할 수 있도록 디자인하였다.

5. 참고문헌

[1] S. Floyd and V. Jacobson, " Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol. 1, pp. 397- 413, Aug. 1993.

[2] Martin May, Jean Bolot, Christophe Diot, and Bryan Lyles, " Reasons not to deploy RED," in Proceedings of IWQoS' 99, June 1999.

[3] Mikkel Christiansen, Kevin Jeffay, David Ott, F. Donelson Smith, " Tuning RED for web traffic," in Proceedings of ACM SIGCOMM 2000, August 2000.

[4] Haining Wang and Kang G. Shin, " Refined design of random early detection gateways," in Proceedings of Globecom' 99, pp. 769- 775, December 1999.

[5] Wu-chang Feng and Dilip D.Kandlur and Debanjan Saha and Kang G. Shin, " A self-configuring RED gateway," in Proceedings of IEEE INFOCOM' 99, March 1999.

[6] Wu-chang Feng and Dilip D.Kandlur and Debanjan Saha and Kang G. Shin, " BLUE: A new class of active queue management algorithms," Tech. Rep. CSETR-387-99, U. Michigan, March 1999.

[7] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, " Modeling TCP throughput: a simple model and its empirical validation," in Proceedings of ACM SIGCOMM' 98, pp. 303- 314, Aug. 1998.

[8] L. Zhang and D. D. Clark, " Oscillating behavior of network traffic: A case study simulation," Internetwoking: Research and Experience, vol. 1, pp. 101- 112, 1990.

[9] Raj Jain, " Throughput fairness index: An explanation," ATM Forum Contribution 99-0045, February 1999.

[10] Go Hasegawa, Kouichi Tokuda and Masayuki Murata, " Analysis and Improvement of fairness among many TCP connections sharing Tail- Drop and RED Routers" in Proceedings of INET 2002, November 2002