NS-2를 이용한 Efficient Adaptive RED 라우터 버퍼 관리 알고리즘 성능평가

임혜영○ 이종현 허의남 황준
{saksi, huh, hjun}@swu.ac.kr, naked97@sslab.cse.cau.ac.kr

Evaluation of Efficient Adaptive RED Router Buffer Management Algorithm Using NS-2

Hyeyoung Lim○, Euinam Huh, Jun Hwang
Dept. of Computer Science, Seoul Women's University, Seoul, Korea
Jonghyun Lee
Dept. of Computer Science and Engineering, ChungAng University, Seoul, Korea

요 약

RED 파라미터를 가장 적합하게 설정할지라도 RED 라우터의 처리량(Throughput)이 TD 라우터 보다 획기적으로 향상되기는 매우 어려운 일이다. 하지만 RED 라우터를 이용하여 TCP 커넥션들간의 페어니스(Fairness)와 페어니스 향상에 따른 전체 네트워크 이용률(Utilization)을 높일 수는 있다. 본 논문에서는 네트워크 상황에 따라 동적으로 RED 파라미터를 조절하면서 빠르게 안정적인 상태로 적응하는 진보된 RED 알고리즘인 ea-RED(Efficient Adaptive RED)의 성능을 네트워크 시뮬레이션 툴인 NS-2를 이용하여 TD, RED 알고리즘과 다양하게 실험하고 비교 분석한다. 이 실험을 통해 병목구간에서 혼잡이 심할 경우, ea-RED 라우터가 RED나 TD 라우터에 비해 큐를 효율적으로 사용하면서도 보다 높은 페어니스 인덱스 값을 갖는다는 점을 확인할 수 있다.

1. 서 론

네트워크 혼잡상태가 심각하지 않고 RED[1] 파라미터에 적당한 값을 설정할 경우, RED 라우터는 모든 TCP 커넥션에 대해서 TD 라우터 보다 공평하다. 그러나 다이내믹한 네트워크 상황에 적합한 RED 파라미터 값을 설정하기란 쉽지 않다[2, 3]. 특히 최대 드롭 확률을 설정하기 위해 사용되는 파라미터 값 max_{th}, 드롭 확률을 계산하는데 사용하는 최소 threshold 값 min_{th}, 드롭 확률을 계산하는데 사용하는 최대 threshold 값 max_p의 경우는 더욱 그렇다. 이런 문제점을 해결하기 위해, 패킷 드롭 확률인 p의 다이내믹한 컨트롤에 포커스를 맞추는 연구가 진행되었다[4]. 하지만, p값의 빠른 변화는 TCP 커넥션들의 퍼포먼스를 저하시켰다[5]. 한편 현재 큐 길이를 완만하게 반영하기 위해 사용되는 평균 큐 길이 파라미터 avg 값이 max_{th}값과 min_{th}값 사이에 있을 때 가장 좋은 성능을 보인다는 사실에 기초한 dt-RED[7] 알고리즘이 제안되었다. dt-RED는 max_{th}를 다이내믹하게 조절함으로써 이전 알고리즘들에 비해 많은 개선을 이루었으나, TCP 커넥션의 수가 100개를 초과할 경우, TD 라우터보다 훨씬 낮은 페어니스 인덱스(Fairness Index)[6] 결과 값을 나타내게 된다[7]. 또한 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.

2. ea-RED

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 변화시켜서 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) 조절할 수 있다.

3. ea-RED 시뮬레이션

이 절에서는 ea-RED의 성능을 TCP 커넥션들 간 페어니스 측면에서 평가하기 위해 몇 가지 시뮬레이션 결과를 보일 것이다. 시뮬레이션 툴로는 NS-2를 사용했다.

3-1. 시뮬레이션 환경

우리는 N개의 송신 호스트와 1개의 수신 호스트 그리고 1개의 라우터로 구성된 병목 구간 토폴로지를 구성했다. N개의 송신 호스트는 라우터와 100Mbps 대역폭과 10ms 지연시간을 갖는 양방향 링크로 연결하였다. 라우터와 수신 호스트는 10ms 지연시간을 갖고 0.01Mbps, 0.01Mbps, 0.5Mbps, 1Mbps 대역폭을 갖는 양방향 링크로 연결되었다. N개의 송신 호스트와 1개의 수신 호스트는 모두 TCP Reno 연결방식을 사용했다. TCP Reno의 세부 설정은 다음과 같다.

- advertised window size : 400
- ecn : true
- packet size = 1000 (byte)
- ssthresh : 65535
- max congestion window size : 70 (라우터 한계 큐 길이 50), 100(라우터 한계 큐 길이 75)

공통적으로 라우터의 한계 큐 길이는 50과 75로 설정했다. RED 라우터의 max_{th} 파라미터 값은 한계 큐 길이가 50일 때 45로 설정되고, 75일 때 70으로 설정했다. Ea-RED 라우터의 max_{th} 초기 파라미터 값은 항상 15이다. RED 라우터의 min_{th} 파라미터 값은 5로 설정하고, mean_packet_size는 500으로 설정했다. 라우터의 한계 큐 길이와 max_{th}, min_{th}의 기본 단위는 1,000 패킷으로 설정했다. ea-RED 라우터의 경우, $\alpha_{max} = \beta_{max} = \alpha_{min} = \beta_{min} = 0.05$ 로 설정하고, $min_{lim} = 5$, $max_{lim} = 45$로 RED 라우터의 max_{th} 파라미터 값과 동일하게 설정했다.

3-2. 시뮬레이션 결과 분석

TD, RED, ea-RED 라우터 간 최적의 성능 비교를 위해 우선, 동일한 환경에서 가장 최적화된 RED 라우터 maxth 파라미터 값을 찾아내야 했다. 이를 위해, 70초간 모든 송신 TCP 커넥션이 데이터를 전송하는 경우와 다이내믹하게 송신 TCP 커넥션의 수를 조정하는 경우로 나누어 실험을 실행했다. 실험 결과는 그림 1와 같다. 그림 1의 Static 그래프는 70초간 모든 송신 TCP 커넥션이 데이터 를 전송한 경우이고, Dynamic 그래프는 다이내믹하게 송신 TCP 커넥션의 수를 조정한 경우이다. 그림에서 살펴볼 수 있 는 바와 같이, 큰 차이는 없었지만 maxth 파라미터 값이 45일 때, 최적의 성능을 발휘했다. 이 결과를 근거로, 이후의 실험들에서 RED 라우터의 maxth 파라미터 구간의 라우터 버퍼 관리 방식을 TD, RED, ea-RED로 각각 설정하였다. N개의 송신 호스트는 라우터와 100Mbps 값은 모두 45로 설정된다.

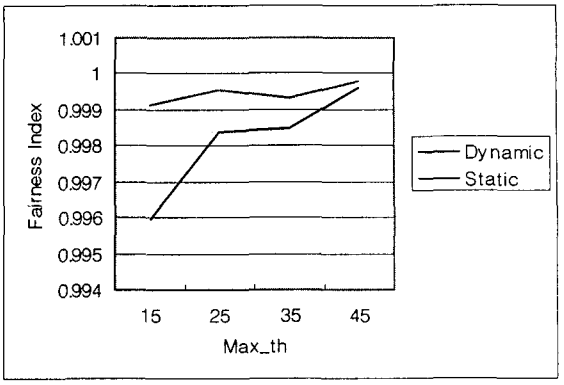

그림 1: max_{th} 설정 값 변화에 따른 RED 라우터의 페어니스 인덱스 변화

첫 번째 실험은 라우터와 1개의 수신 호스트 사이의 대역폭을 1Mbps, 0.5Mbps, 0.1Mbps, 0.01Mbps 로 변경하면서, 데이터를 전송하는 송신 TCP 커넥션의 수를 다이내믹하게 조정했다. 실험결과는 그림 2과 같다. 그림에서 알 수 있듯이, TD, RED, ea-RED 라우터간 큰 차이가 없는 비슷한 결과를 보여주고 있다.

두 번째 실험은 송신 TCP 수를 21, 50, 100, 150, 200으로 각각 설정한 상태에서 송신 TCP 커넥션의 수를 시간에 따라 변화시켰다. 실험결과는 그림 3와 같다. 실험결과 송신 TCP 커넥션의 수가 100개 이상일 경우, RED 라우터는 TD 라우터 보다 오히려 좋지 않은 성능을 보여줬다. 그에 비해 ea-RED의 경우, 송신 TCP 커넥션의 수가 100개 일 때, RED 보다 약 25% 이상 나은 성능을 보였고, 송신 TCP 커넥션의 수가 200개 일 때, TD 라우터 보다는 약 33.6% 나은 성능을 보였고, RED 라우터 보다는 약 21.25% 향상된 성능을 보여줬다.

세 번째 실험은 송신 TCP 수를 두 번째 실험과 동일하게 설정하면서, 70초간 모든 TCP 커넥션이 데이터를 보냈다. 실험결과는 그림 4와 같다. 대부분의 경우, ea-RED의 성능이 TD 라우터와 RED 라우터를 앞섰다. 송신 TCP 커넥션의 수가 100개 일 때, RED 라우터 보다 약 28.8% 성능이 앞섰고, 송신 TCP 커넥션의 수가 150개 일 때, TD 라우터 보다 약 13.3%, RED 라우터 보다 약 41.42% 성능이 앞섰다.

4. 결론

ea-RED는 RED 라우터가 네트워크 상황에 맞는 파라미터 값 설정을 위해 소비하는 로드를 줄일 수 있다. 또한 실험을 통해, 기존 RED의 경우 송신 TCP 커넥션 연결수가 많아지면 TD 라우터 보다 좋지 못한 성능을 낸데 비해, ea-RED 라우터는 이 점을 개선하여 송신 TCP 커넥션의 수가 많아지더라도 네트워크 상황에 어느 방식의 라우터 보다 잘 적응하여 RED 라우터 보다는 최대 41.42%, TD 라우터 보다는 최대 33.6% 앞선 성능을 보였다. 또한 전체적으로 송신 TCP 커넥션들의 congestion

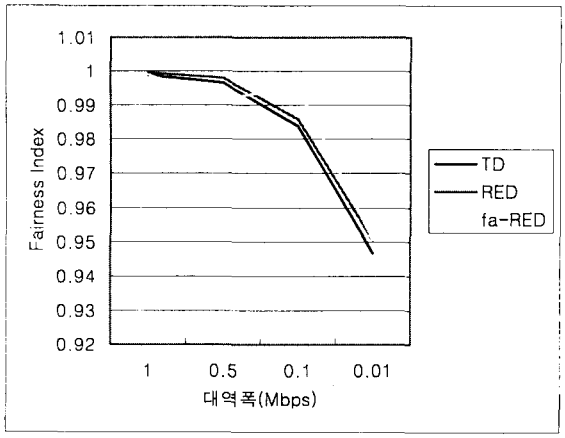

그림 2: 라우터와 수신 호스트 사이 대역폭 변화에 따른 페어니스 인덱스 변화

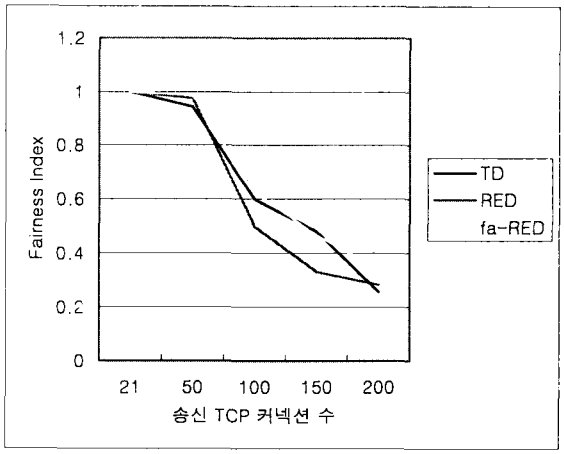

그림 3: 다이내믹하게 변하는 송신 환경에서 전체 송신 TCP 커넥션 수에 따른 페어니스 인덱스 변화

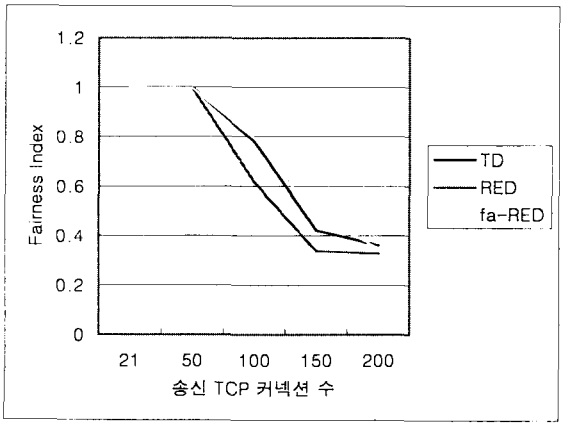

그림 4: 모든 송신 TCP 커넥션 데이터 전송시, 전체 송신 TCP 커넥션 수에 따른 페어니스 인덱스 변화

window의 크기는 하향 평준화 되었으나, 전체적인 네트워크 이용률은 RED와 TD 라우터에 비해서 늘었다. 예를 들어, 송신 TCP 커넥션의 수가 200개인 환경에서, 송신 TCP 커넥션의 수가 시간에 따라 다이내믹하게 조정되는 경우, 총 송신 packets 수는 다음과 같았다.

- TD : 10,600,000 packets
- RED : 10,950,000 packets
- ea-RED : 11,250,000 packets

이 결과를 통해 우리는 페어니스 인덱스만으로 알 수 없었던 전체 네트워크 퍼포먼스도 ea-RED 라우터가 RED나 TD 라우터에 비해 우수함을 알 수 있었다.

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] 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.

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

[7] 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

[8] NEST, ftp://ftp.cs.columbia.edu/nest/

[9] LBNL, LBNL Network Simulator-ns version 1, http://www-nrg.ee.lbl.gov/ns/

[10] UCB/LBNL/VINT Network Simulator - ns (version 2), http://www-Mash.CS.Berkeley.EDU/ns/

[11] Virtual InterNetwork Testbed(VINT), http://netweb.usc.edu/vint/

[12] 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.