ea-RED++ : 예측 알고리즘을 적용한 ea-RED 알고리즘

이종현°, 임혜영, 황준, 김영찬
중앙대, 서울여대
{naked97°, yckim}@ sslab.cse.cau.ac.kr, {saksi, hjun}@ swu.ac.kr

ea-RED++: Adding Prediction Algorithm for ea-RED Router Buffer Management Algorithm

Jonghyun Lee°, Youngchan Kim
Dept. of Computer Science and Engineering, ChungAng University, Seoul, Korea
Hyeyoung Lim, Jun Hwang
Dept. of Computer Science, Seoul Women' s University, Seoul, Korea

요 약

ea-RED(Efficient Adaptive RED)[1][2]는 다수의 TCP 커넥션이 경쟁하는 병목구간에서 인터넷 라우터 버퍼를 능동적으로 관리하는 다양한 AQM(Active Queue Management) 알고리즘 중의 하나로 RED 라우터 버퍼 관리 알고리즘의 성능을 개선한 라우터 버퍼 관리 알고리즘이다. 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 설정해주는 것이 매우 어렵다는 점이다[3]. ea-RED는 max threshold와 min threshold 값을 네트워크 상황에 따라 동적으로 조절함으로써 이런 문제를 해결했고, 기존 RED에 비해 라우터 버퍼는 50% 정도만 사용하면서도, 페어니스 인덱스(Fairness Index)[4]가 최대 41.42% 개선되었다.[1][2] 그러나 송신 TCP 커넥션의 수가 늘어날수록 성능향상에 대한 효과가 감소되었고, 드롭 패킷수가 TD나 RED 라우터 버퍼관리 알고리즘에 비해 많았기 때문에 라우터의 출력(output) 총 패킷 용량이 최대 약 2.3% 정도 TD나 RED 라우터 버퍼관리 알고리즘에 비해 적었다. 이 부분을 개선하기 위해 기존 ea-RED 알고리즘에 LR_Lines 예측 알고리즘을 적용한 ea-RED++ 알고리즘을 구현하였고, 실험 결과 페어니스 인덱스는 기존 ea-RED에 비해 최대 약 30% 정도 향상되었고, 총 output 패킷 용량의 손실률은 최대 50%정도 감소하여 기존 ea-RED에 비해 향상된 성능을 보여주었다.

1. 서 론

ea-RED(Efficient Adaptive RED)[1][2]는 다수의 TCP 커넥션이 경쟁하는 병목구간에서 인터넷 라우터 버퍼를 능동적으로 관리하는 다양한 AQM(Active Queue Management) 알고리즘 중의 하나로 RED 라우터 버퍼 관리 알고리즘의 성능을 개선한 알고리즘이다. 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 설정해주는 것이 매우 어렵다는 점이다[3]. ea-RED는 max threshold와 min threshold 값을 네트워크 상황에 따라 동적으로 조절함으로써 이런 문제를 해결했다. 기존 RED에 비해 라우터 버퍼는 50% 정도만 사용하면서도, 페어니스 인덱스(Fairne-ess Index)[4]가 최대 41.42% 개선되었다.[1][2] 그러나 송신 TCP 커넥션의 수가 늘어날수록 성능향상에 대한 효과가 감소되었고, 드롭 패킷수가 TD나 RED 라우터 버퍼관리 알고리즘에 비해 많았기 때문에 라우터의 출력(output) 총패킷 용량이 최대 약 2.3% 정도 TD나 RED 라우터 버퍼관리 알고리즘에 비해 적었다. 이 부분을 개선하기 위해 기존 ea-RED 알고리즘에 LR_Lines 예측 알고리즘을 적용한 ea-RED++ 알고리즘을 구현하였다.

2. LR_lines (Linear Regression Lines)

LR_lines는 통계적인 툴로서 과거 값으로부터 미래의 값을 예측하기 위해 사용된다. security price의 경우에서 일반적으로 값이 지나치게 확장되었을 때 값을 결정하기 위해 사용된다. LR_lines는 LR_trend_line이라고도 하는데, price와 예측선 사이에 거리를 최소화 하기 위해 price를 직선으로 최소 제곱법을 사용하여 그린 것이다. 만약 '내일 특정 가격이 어떻게 변할까?' 라고 추측할 때 오늘 가격에 상당히 근접할 것으로 생각하는 것이 논리적

이다. 특히 가격이 상승추세라면 오늘 가격은 상당히 상승에 편중되어 나타날 것이다. 이와 같이 LR_lines는 이러한 논리적인 가정의 통계학적 확증작업이라고 할 수 있다. LR_lines의 계산 방법은 다음과 같다.

$$y = a + bx$$

$$a = \frac{\sum y - b\sum x}{n}$$

$$b = \frac{n\sum(xy) - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

그림 2. LR_lines 계산 식

그림 2의 계산 식에서 y는 종속변수, x는 독립변수, a는 절편, b는 기울기, n은 총 기간을 의미한다.

3. ea-RED++

ea-RED++ 라우터 버퍼 관리 알고리즘은 기존 ea-RED[1][2] 버퍼관리 알고리즘과는 다르게 현재 싸이클의 평균 큐 길이 값을 max, min threshold값과 비교하는 것이 아니라, 현재 싸이클의 평균 큐 길이를 예측 알고리즘의 입력값으로 사용하여 다음 싸이클의 평균 큐 길이를 예측하고, 그 예측된 결과를 이용하여 max, min threshold 값과 비교하여 max threshold 값 보다 예측 평균 큐 길이가 길 경우, 미리 max threshold 값을 상향 조정하여, 다음 싸이클에서 평균 큐 길이가 max threshold 값을 침범하여 TD 버퍼관리 알고리즘과 동일한 결과를 야기하는 것을 예방하였다. min threshold 값보다 평균 큐 길이가 짧아진 경우는 min threshold값과 max threshold값을 하향 조정해주어 과도하게 max, min threshold값이 상향되는 결과를 방지하고, 가장 적합한 max, min threshold 값을 찾아내어 평균 큐 길이가 수렴할 수 있도록 유도했다.

4. 실험 및 결과 분석

실험 환경은 논문[1][2]에서와 동일한 환경에서 LR_lin-es 예측 알고리즘을 적용한 ea-RED++를 대상으로,70초 동안 input 노드 수를 21, 50, 100, 150, 200으로 하면서 70초가 될 때까지 모든 input 노드에서 패킷을 연속적으로 보내는 경우와 10초 단위로 활성 input 노드의 수를 변화시키는 경우의 두 가지 실험을 진행했다. 실험 후, 평균 큐 길이 변화, fairness index, 전체 input 노드로부터 전송된 총 input 패킷 전송량, ea-RED++ 라우터를 통해 전송된 총 output 패킷 전송량을 측정했다. ns 시뮬레이션 툴[5, 6]은 각 입력 노드가 보낼 수 있는 전체 데이터의 용량은 무한대로 지원하며, 총 input 패킷 전송량은 입력노드와 라우터간 link를 통해 전달된 총 데이터량을 뜻한다. 총 output 패킷 전송량은 라우터와 출력노드간 link를 통해 전달된 총 데이터량을 뜻한다. 결과로써 그림 1과 2는 input 노드 수가 10초마다 변화했을 때의 총 input 패킷 전송량과 input 노드 전체가 70초간 변화 없이 일정하게 패킷을 전송했을 경우 총 input 패킷 전송량이다. 실험 결과를 통해 ea-RED++가 근소하게 ea-RED보다 모두 높음을 확인할 수 있다. 이 실험 결과의 의미는 ea-RED++가 TD, RED, ea-RED에 비해서 input 노드에 대해 공평한 전송기회를 주어, input link 이용률을 높인다는 점이다.

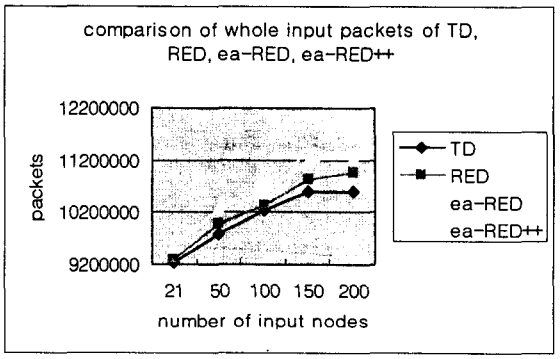

그림 1. input 노드 수가 10초마다 변화했을 때 총 input 패킷 전송량

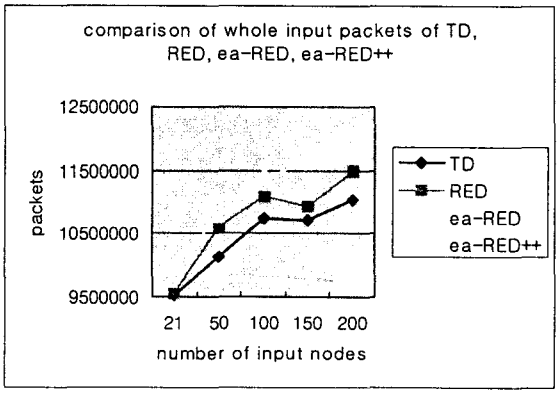

그림 2. input 노드 전체가 70초 동안 어떤 변화도 없이 일정하게 패킷을 전송한 경우 input 패킷 전송량

그림 3과 4는 input 노드 수가 10초마다 변화했을 때 총 output 패킷 전송량과 input 노드 전체가 70초 동안 변화 없이 일정하게 패킷을 전송한 경우 총 output 패킷 전송량에 대한 결과이다. 그림 4의 경우, ea-RED와 ea-RED++간에 성능이 개선된 점을 찾지 못했지만, 그림 3에서 보는 바와 같이 네트워크 환경이 빠르게 변화할 경우 ea-RED에 비해서 ea-RED++가 output 패킷 손실률에서 약 50% 정도 개선된 결과를 보여주었다. 무엇보다 input 노드의 수가 50일 경우를 제외하고는 TD, RED 라

우터 버퍼 관리 알고리즘과 거의 유사한 output 패킷 전송량 수치를 보인다는 점이다. 공평하게 input node에 기회를 주기 위해서 그 만큼 패킷 드롭량이 늘었음에도 불구하고, 총 output 패킷 전송량에 차이가 거의 없다는 것은 페어니스 인덱스 향상을 꾀하면서도 전체 네트워크 성능은 전혀 저하되지 않았다는 것을 의미한다.

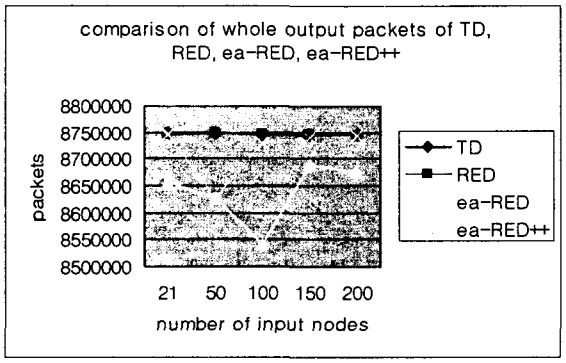

그림 3. input 노드 수가 10초마다 변화했을 때 총 output 패킷 전송량

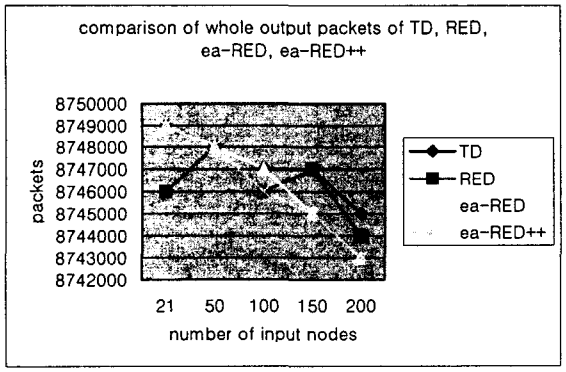

그림 4. input 노드 전체가 70초 동안 어떤 변화도 없이 일정하게 패킷을 전송한 경우 output 패킷 전송량

그림 5와 6은 input 노드 수가 10초마다 변화했을 때 페어니스 인덱스 변화와 input 노드 전체가 70초 동안 변화 없이 일정하게 패킷을 전송한 경우 페어니스 인덱스 변화에 대한 결과이다. 그림 5에서 보는 바와 같이 ea-RED++ 페어니스 인덱스가 ea-RED에 비해서 최대 약 30% 정도 향상되었음을 확인할 수 있다. 그림 6에서는 ea-RED++ 페어니스 인덱스가 ea-RED에 비해서 최대 약 15% 정도 향상되었음을 확인할 수 있다.

4. 결론

실험 결과를 통해 ea-RED++가 ea-RED에 비해서 페어니스 측면에서 최대 약 30% 정도 성능 향상이 있었고, 총 output 패킷 용량의 손실률에 대해서는 약 50% 이상 성능 개선이 있었음을 확인할 수 있었다.

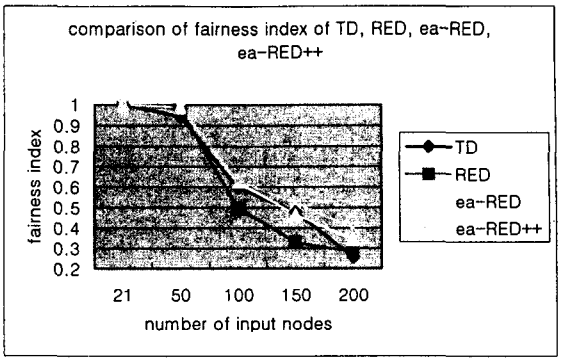

그림 5. input 노드 수가 10초마다 변화했을 때 페어니스 인덱스

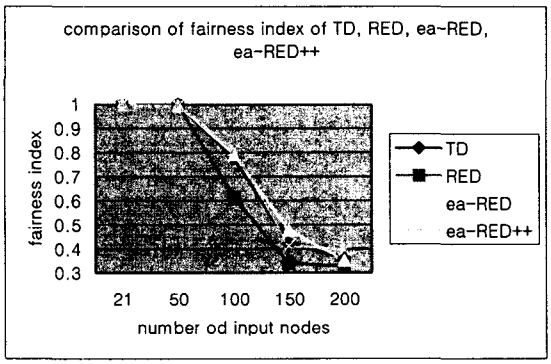

그림 6. input 노드 전체가 70초 동안 어떤 변화도 없이 일정하게 패킷을 전송한 경우 페어니스 인덱스

5. 참고문헌

[1] 임혜영, 이종현, 허의남, 황준, " NS-2를 이용한 Efficient Adaptive RED 라우터 버퍼 관리 알고리즘 성능 평가" 한국정보과학회 봄 학술발표논문집 Vol.30 No.1 2002. 4.

[2] 이종현, 임혜영, 허의남, 황준, 김영찬, " Efficient Adaptive RED 라우터 버퍼 관리 알고리즘 디자인과 구현" 한국정보과학회 봄 학술발표논문집 Vol.30 No.1 2002. 4.

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

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

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

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