큐 변화량을 이용한 적응식 AQM 성능 향상에 관한 연구

김종화, 이기영

인천대학교, 정보통신공학과

A Study on Performance Improvement of Adaptive AQM Using the Variation of Queue Length

Jong Hwa Kim^{O}, Ki Young Lee

Department of Information and Telecommunication Engineering,

University of Incheon

E-mail : uickjh@empal.com

Abstract

Random Early Detection (RED), one of the most well-known *Active Queue Management* (AQM), has been designed to substitute Tail Drop and is nowadays widely implemented in commercially available routers. RED algorithm provides high throughput and low delay as well as a solution of global synchronization. However RED is sensitive to parameters setting, so the performance of RED, significantly depends on the fixed parameters. To solve this problem, the Adaptive RED (ARED) algorithm is suggested by S. Floyd. But, ARED also uses fixed parameters like target-queue length; it is hard to respond to bursty traffic actively. In this paper, we proposed AQM algorithm based on the variation of current queue length in order to improve adaptability about burst traffic. We measured performance of proposed algorithm through a throughput, marking-drop rate and bias phenomenon. In experimentation, we raised a packet throughput as reduced packet drop rate, and we confirmed to reduce a bias phenomenon about bursty traffic.

I. 서론

FIFO (First-In-First-Out) 큐잉 메커니즘의 단순성 때문에, drop-tail 알고리즘은 현재 가장 널리 쓰이고 있는 큐잉 메커니즘이다. Drop-tail 큐가 오버플로우(overflow) 가 되었을 때, 새롭게 유입되는 어떠한 응용 서비스가 제공되는가에 상관없이 무조건 패킷을 폐기하는 알고리즘이다. 버스트한 트래픽에 적응하기 위하여, drop-tail 의 라우터는 커다란 FIFO Queue 를 갖게 된다. 혼잡이 지속될 때, drop-tail 라우터는 병목현상 (bottleneck) 으로 인하여 높은 지연을 겪게 된다.

Active Queue Management (AQM) 은 최선형 네트워크에서 쓰이고 있는 drop-tail 의 단점을 보완하기 위해 설계된 큐잉 알고리즘들의 부류를 말한다. 현재 대부분의 AQM 알고리즘들은 라우터에서 큐잉 지연 (queuing delay) 없이 높은 처리율을 제공하는 것을 목표로 한다[4]. AQM 의 성능 향상을 위한 많은 연구들이 진행되고 있다. S.Floyd 가 제안한 Random Early Detection (RED) 는 가장 널리 알려진 AQM 메커니즘이다. RED 는 큐의 크기가 미리 정해 놓은 임계치를 넘게 되면 이후에 들어오는 패킷을 확률적으로 폐기함으로써, 큐의 크기가 차오르는 것을 미리 제어하여 TCP 트래픽의 전역 동기화를 막음으로써 링크의 사용률을 높이는 방식이다[1]. 그러나 RED 는 고정된 파라미터를 이용하기 때문에 네트워크 상황에 능동적으로 대처할 수 없어, 네트워크의 성능이 파라미터 설정에 좌우되는 단점이 있다[3]. 이러한 문제점을 해결하기 위한 RED 의 변종 알고리즘들이 있다 [4]. FRED (Fair RED) 는 흐름 당 큐 길이와 활성중인 흐름을 선형적인 폐기함수를 이용하여 폐기확률을 계산하여 공평하게 링크 대역폭을 사용할 수 있도록 하였다. SRED (Stabilized RED)는 활성 흐름을 단계적인 드롭함수를 이용하여 큐를 안정시킬 수 있는 방법을 제시하였다. WRED (Weighted RED)는 트랙픽을 클래스에 따라 나누어 각각 다른 패킷 폐기 확률을 적용하여 가중치를 부여하여 우선순위가 낮은 패킷부터 폐기를 하는 방식을 취한다. ARED (Adaptive RED) 는 라우터가 처리하는 트래픽의 부하에 따른 큐 길이의 변화에 따라 패킷 폐기 확률을 변화시켜 링크의 효율성을 높인다.[4][6] 그 중 ARED 는 RED 의 주요 장점을 훼손시키지 않으면서 트래픽 부하에 따라서 RED 의 파라미터를 조정하여 평균 큐 길이의 안정성 제공하여 높은 처리율을 얻게 하는 적응식 AQM 이라 할 수 있다[2]. 그러나 RED 와 ARED 는 각 응용 서비스들의 갖는 지연이나 처리율과 같은 성질에 상관없이 높은 처리율을 위하여 미리 세팅된 파라미

터에 의하여 유입되는 패킷을 처리하게 된다. 이는 여러 RED 의 변종 알고리즘들이 계속 연구 되고 있지만 고정된 파라미터를 이용하기 때문에 RED 의 근본적인 문제점인 파라미터 민감성에 대한 부분은 해결하지 못하고 있다 [4][6].

본 논문에서는 기존의 ARED 큐의 변화량을 동적으로 변화시킨 제안한 알고리즘을 이용하여 RED 의 장점인 링크의 효율성을 향상시키고, 버스트한 트래픽에 대하여 높은 처리율을 얻게 함으로써 응용 서비스에 맞도록 능동적으로 패킷을 폐기할 수 있도록 하는 알고리즘을 제안하였다.

본 논문의 구성은 다음과 같다. II 장에서는 본 연구에 기초가 되는 큐 관리 기법에 대하여 상세하였고, III 장에서는 제안한 알고리즘에 대하여 설명을 하고, IV 장에서는 제안한 알고리즘에 대한 시뮬레이션 결과를 분석하고, V 장에서는 본 논문의 결론을 기술하였다.

II. 관련연구

2.1 RED (Random Early Detection) 알고리즘

RED 알고리즘의 설계의 목적은 패킷 손실과 큐잉 지연을 최소화하여, 높은 링크 이용률을 얻고, 바이어스 현상을 제거하여, 전역 동기화 (global synchronization) 와 같은 현상을 피하기 위하여 디자인 되었다.[1]
RED 는 EWMA (Exponential Weighted Moving Average) 방식을 사용하여 평균 큐 길이 (*avg*) 를 계산한다. 이 때 평균 큐 길이는 그림 1 과 같이 구해진다.

$$avg \leftarrow (1- w_q)avg + w_q q$$

w_q : queue weight
q : current queue size

그림 1. RED 의 평균 큐 길이

위의 식에서 구해진 avg 값이 최소 임계치(min_{th}) 와 최대 임계치(max_{th}) 사이의 값이 될 때 다음의 확률로서 랜덤하게 패킷을 폐기하게 된다.[1]

$$p_b \leftarrow max_p (avg - min_{th}) / (max_{th} - min_{th})$$

p_b : packet-marking probability
max_p : maximum value for pb
avg : average queue size
min_{th} : minimum threshold for queue
max_{th} : maximum threshold for queue

그림 2. RED 의 패킷 폐기 확률

만약 avg 값이 최소 임계치 값보다 작으면 새로 들어오는 패킷을 폐기하지 않지만 만약 *avg* 값이 최대 임계치 값보다 크게 되면 새로 들어오는 모든 패킷을 폐기하게 된다.

그러나 RED 는 링크가 조금 혼잡되면 max_p 가 커지고, 평균 큐 사이즈는 최소 임계치에 가까워지게 된다. 링크가 더 혼잡되면 max_p 는 낮아지고 평균 큐 사이는 max_{th} 와 비슷하게 된다. RED 는 평균 큐 사이즈가 max_{th} 보다 커지게 되면 제대로 수행을 하지 못하게 된다. RED 의 주된 단점은 혼잡의 정도와 파라미터의 세팅에 따라서 평균 큐 사이즈가 변한다는 것이다. 즉, 고정된 파라미터에 따라 성능이 좌우되기 때문에 능동적이지 못하다. 뿐만 아니라 RED 의 처리율 역시 트래픽 로드와 RED 파라미터에 민감하다는 것이다. 이러한 RED 의 문제점을 보완하고자 나온 알고리즘이 Adaptive-RED 이다.

2.2 Adaptive RED 알고리즘

RED 의 성능은 트래픽 부하와 RED 파라미터 세팅에 좌우된다. 적절하지 않은 파라미터 세팅은 drop-tail 보다 그 성능이 오히려 훨씬 나빠질 수 있다 [3]. S. Floyd 등이 제안한 ARED 알고리즘은 Feng 등이 제안한 Self-configuring RED 알고리즘을 새롭게 해석하여 제안한 알고리즘이다 [2]. ARED 는 기존의 RED 알고리즘이 가지고 있는 파라미터에 대한 민감성을 유연하게 하고자 제안된 알고리즘이다. ARED(Adaptive-RED) 알고리즘은 기존 RED 알고리즘이 갖고 있었던 문제점인 파라미터에 대한 민감성을 완화하고자 제안된 능동적 큐 관리 알고리즘이다. ARED 는 RED 알고리즘을 최소한의 변경으로 주요 장점을 훼손시키지 않고 평균 큐 길이를 안정화 시키고, 다른 RED 파라미터를 자동적으로 세팅하여 다양한 망 상황에서 적응성을 우수하게 만들고자 한 알고리즘이다. 최소화한다는 것이다. ARED 는 패킷 로스 레이트와 큐잉 지연에 분산을 감소시킴으로써 RED 알고리즘이 갖고 있었던 파라미터에 민감한 문제를 완화시켰다. ARED 의 알고리즘은 그림 3 과 같다 [2].

Every *interval* seconds:
if ($avg > target$ and $max_p \leq 0.5$)
  increase max_p :
  $max_p \leftarrow max_p + a$
elseif ($avg < target$ and $max_p \geq 0.01$)
  decrease max_p:
  $max_p \leftarrow max_p + \beta$

Varibles:
avg : average queue size
Fixed parameter:
$interval$: time; 0.5 seconds
$target$: target for avg:
$[min_{th} + 0.4(max_{th} - min_{th}), min_{th} + 0.6(max_{th} - min_{th})]$
a : incresement: $\min(0.01, max_p/4)$
β: decrease factor: 0.9

그림 3. Adaptive-RED 알고리즘

ARED 는 고정된 평균 큐 사이즈를 유지하려고 노력을 한다. 그러므로 적응성있는 드롭 레이트에 의해 예상할 수 있는 평균 큐잉 딜레이를 제공한다. 그러나 고정된 평균 큐 사이즈 트래픽이 대용량 실시간 트래픽이라든지 웹 트래픽과 같은 트래픽이 혼재되었을 경우에는 적절히 수용할 수가 없다. 특히 인터넷 트래픽의 많은 혼잡 원인인 버스트한 트래픽을 적절히 처리할 수 있는 방안이 연구되어야 한다.

III. 제안한 알고리즘

3.1 시간당 큐 변화량에 따른 고려사항

RED 와 ARED 는 새로운 패킷이 중간 라우터에 들어올 때마다 평균 큐 길이를 계산하게 된다. 현재 큐 길이에 대하여 얼마나 가중치를 두느냐에 따라 W_q 를 설정하여 새로운 큐 길이를 구하게 된다.

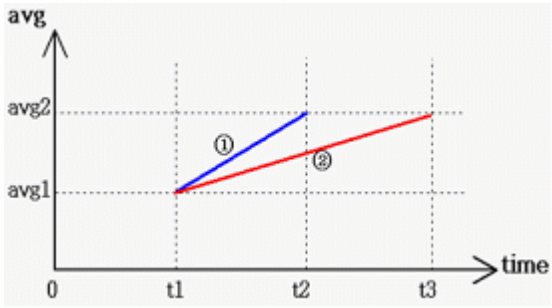

그림 4. 단위 시간당 큐 길이 변화

그림 4 를 보면 기존의 RED 는 ①과 ②를 같은 것으로 인식한다. 그 이유는 현재 시간 t1 에서 큐 길이가 avg1 일 때, t2-t1 의 시간이 지나서 avg2 가 되던지, t3-t1 의 시간이 지나서 avg2 가 되던지, 새로 계산된 avg 의 값은 같기 때문이다. 따라서, ①과 ②는 패킷 폐기 확률이 같게 된다. 이는 시간을 고려하지 않은 결과이다. 본 논문에서는 시간에 관한 부분까지 고려해서 패킷 폐기 확률을 계산 하였다. 시간당 큐 변화량이 양의 값을 가지면서 그 값이 크다면 버스티한 트래픽이라고 할 수 있다. 이 값이 양의 값이면 패킷 폐기 확률을 높이고, 음의 값이면 패킷의 폐기 확률을 낮추게 된다.

3.2 QVARED (Queue Variation Adaptive RED)

RED 와 ARED 는 현재의 큐 길이의 변화는 고려하지 않고 패킷을 폐기한다. 본 논문에서는 제안한 QVARED (Queue Variation Adaptive RED) 는 ARED 의 기본 구조 즉, max_p 를 구함에 있어서 target-range 를 정하여 자체 조율하는 알고리즘은 그대로 적용 하였고[16], 이전의 큐 길이와 현재의 큐 길이를 비교하여 만약에 큐가 감소 중에 있다면, 그 확률을 큐 길이가 증가하고 있을 때의 절반에 해당하는 확률로 패킷을 폐기할 수 있는 알고리즘을 제시하였다. 알고리즘의 드롭함수를 도식화하여 각각 그림 5 에 나타내었고, 자세한 알고리즘은 그림 6 에 나타내었다.

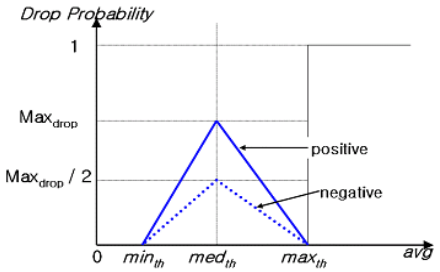

그림 5. QVARED 의 드롭함수

calculate q_t ← (avg - avg_prev) / (time - time_prev)
q_max ← max(q_t, q_tmax)
avg_prev ← avg , time_prev ← time
if $\min_{th} < avg < \max_{th}$
Increment *count*
calculate probability *pa*:
if avg < med_th
avg (max_p (avg - min_th) / (med_th - min_th)
else
avg (max_p (max_th - avg) / (max_th - med_th)
if q_t < 0 pb ← pb / 2
*if q_tmax > 0 pb ← pb + pb * q_t / q_max*
$pb \leftarrow max_p(avg - min_{th}) = (max_{th} - min_{th})$

그림 6. QVARED 알고리즘

IV. 시뮬레이션 결과 및 분석

본 논문에서 제안한 방법을 수행하기 위하여 네트워크 시뮬레이션 툴인 ns-2.27 (network simulator version 2.27) 을 사용하였고 [7], 네트워크 구성은 그림 7 과 같다.

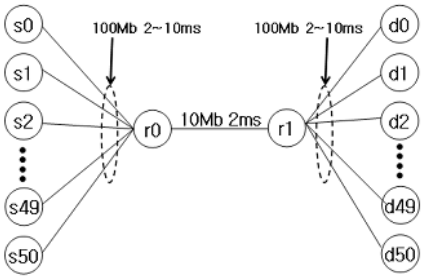

그림 7. 시뮬레이션 구성

TCP 노드의 수는 10 개부터 50 개까지 변화시켰고, 시뮬레이션 시간은 100 초 동안 수행하였다. 제안한 알고리즘의 성능을 비교하기 위하여 병목 링크에 RED, ARED 그리고 QVARED 를 각각 적용시켜 비교하였다. RED 에서는 두개의 임계치인 $\max_{th}$ 와 $\min_{th}$ 를 각각 80 과 20 packets 으로 잡았으며, $\max_p$ 는 0.02 로 설정하였다. ARED 는 ns-2 의 디폴트 파라미터를 그대

로 이용하였고[7], target-queue 만 50 packets 으로 하였다. 연구에서는 다양한 패킷 사이즈의 변화에 대해서는 고려하지 않았고, 평균 패킷 사이즈를 TCP 노드에서 1000Bytes 로 하였다.

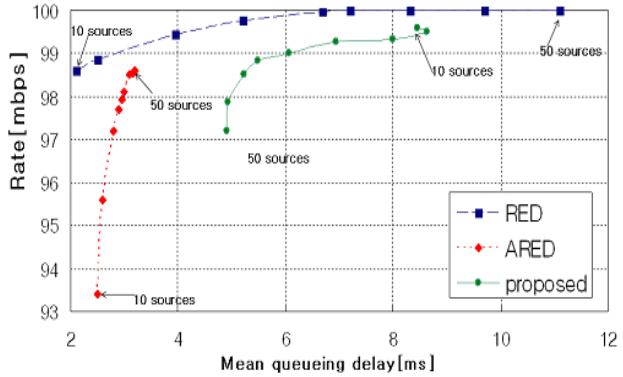

그림 8. TCP 소스 당 처리율

그림 8 은 TCP 총 소스 개수인 50 개를 5 개씩 끊어서, TCP 소스 5 개를 하나의 단위로 하여 하나의 단위에서 처리되는 초당 비트수를 평균 큐잉 지연 시간에 따라 다음과 같이 측정하였다.

$$Throughput = \frac{Packet\ time\ to\ have\ sent\ successfully}{Total\ simulation\ time}$$

처리율이 높다는 것은 그만큼 전송 능력이 우수하다는 것을 나타낸다. RED 는 우수한 링크 레이트와 처리율을 갖고 있음에도 불구하고, 큐잉 딜레이가 TCP 소스가 증가함에 따라 선형적으로 증가하는 것을 볼 수 있다. 본 논문에서 제안한 QVARED 는 RED 보다는 큐잉 딜레이가 훨씬 적게 나타남을 알 수가 있고, ARED 가 이득이 없었던 것에 비하여 효율성 면에서 좀 더 나은 것을 확인하였다. RED 와는 달리 QVARED 는 TCP 노드의 수가 증가함에 따라 큐잉 딜레이를 짧게 하기 위한 노력을 한다. 그 이유는 제안한 알고리즘이 큐의 변화량에 따라 큐잉 딜레이를 짧게 하여 버스트한 트래픽에 에 대비하여 링크의 안정성을 취하려 하기 때문이다.

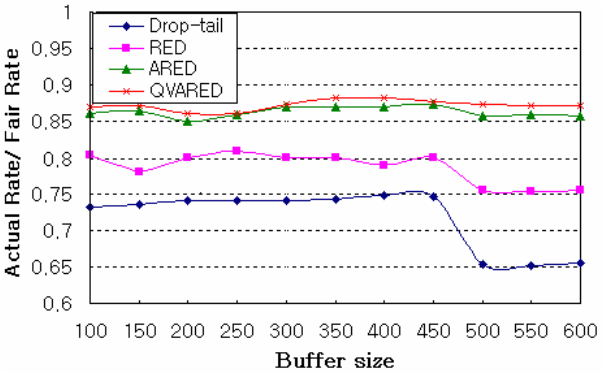

그림 9. 버스트한 트래픽에 대한 바이어스 현상

그림 9 는 drop-tail, RED, ARED, 그리고 제안한 QVARED 이 버스트한 트래픽에 얼마나 대하여 바이어스 현상에 대처할 수 있는가를 나타내고 있다. 이를 측정하기 위해 Actual rate(실제 마킹 비율) 에 대한 Fair rate(이상적인 비율) 의 비율로서 측정하였다. 이 비율이 커질수록 바이어스 현상을 줄이게 되어 버스트한 트래픽에 대해서도 안정적으로 패킷이 전송됨을 의미한다. 그림 9 에서 ARED 는 버퍼 크기가 늘어남에 따라 평균 0.863 정도로 버스트한 트래픽에 대하여 안정성을 보이며, QVARED 역시 그림에서 보는 바와 같이 제안한 방식은 버퍼 사이즈가 500bytes 이상이 될 경우에도 평균 0.871 정도로 ARED 방식과 유사하게 안정적으로 도달할 수 있음을 확인하였다.

V. 결론 및 향후 연구 과제

본 연구에서는 기존의 RED 알고리즘과 RED 의 변종 알고리즘인 ARED 에서 고려하지 않았던 시간당 큐의 변화량을 적용시켜 중간 라우터에서의 패킷 폐기를 보다 능동적으로 하는 방법을 제안하였다. 버스트한 트래픽은 큐의 오버플로우 (overflow) 현상을 일으켜, 어떤 흐름쪽으로 치우치게 되는 바이어스 현상을 야기한다. 실험을 통하여 편의 현상에 대하여 얼마나 안정적으로 대처할 수 있는가를 측정하였으며, QVARED 가 RED 나 ARED 에 비하여 바이어스 현상을 줄이게 되어 버스트한 트래픽에 대해서도 안정적으로 패킷이 전송될 수 있음을 보였다. 본 논문은 현재의 네트워크 상황에 중점을 두고 시뮬레이션을 하였다. 차후에는 실제 상황에 적용 가능 할 수 있도록 다양한 방식으로 접근할 것이다. 또한 여러 가지 RED 의 변종 알고리즘들이 갖는 장점들을 고찰하고 분석하여, 체증 제어 방식들의 성능 파라미터를 더욱 더 연구하여, 체증상황이 발생한 네트워크의 성능 개선 방안에 대하여 연구하여 능동적 큐 관리 알고리즘의 개선 방안에 대하여 포괄적으로 연구할 것이다.

참고 문헌

[1] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, August 1993.
[2] S. Floyd, R. Gummadi, and S. Shenker, “Adaptive RED : An Algorithm for Increasing the Robustness of RED’s Active Queue Management,” via- http://www.icir.o
via http://www.icir.org/floyd/papers/adaptiveRed.pdf
[3] S. Floyd, “RED: Discussions of setting parameters,” via http://www.aciri.org/floyd/REDparameters.txt, Nov. 1997.
[4] Seungwan Ryu and Sung Hee Kim, “Recent Advances in TCP Congestion Control”, Telecommunications Review Vol. 13 No.5 October 2003.
[5] V. Firoiu, and M. Borden, “A Study of Active Queue Management for Congestion Control,” 2000, IEEE INFOCOM
[6] Mahbub Hassan, and Raj Jain, High Performance TCP/IP Networking, Pearson Prentice Hall, 2003.12, pp.287-298.
[7] NS-2. http://www.isi.edu/nsnam/ns/