Spurious RTO를 위한 효율적인 혼잡윈도우 회복 기법

이재형, 안수빈, 추현승
성균관대학교 정보통신공학부
e-mail: jhyunglee@skku.edu

BCR: Balanced Congestion Control for Spurious RTO

Jaehyung Lee, Soobeen Ahn, and Hyunseung Choo
School of Information and Communication Engineering
Sungkyunkwan University

요 약

인터넷상에서 TCP 프로토콜은 사실상 표준으로써 널리 이용되고 있다. 그러나 네트워크 환경이 발전해 감에 따라서 무선 네트워크, CDMA, 3G 네크워크등 다양한 환경이 공존하게 되었고, 이러한 환경에서 TCP의 성능저하가 이슈가 되어 많은 연구가 진행되었으며, 많은 알고리즘들이 발표되었다. 본 논문에서는 무선환경에서 다양한 TCP 성능저하 원인 중에서 spurious RTO에 관한 문제를 해결하고자 한다. 이 전의 연구인 E-RTO는 delay spike문제를 효과적으로 다루고 있다 하지만 delay spike이후 혼잡윈도우를 회복하는 과정을 효과적으로 다루고 있지 못하다. 우리는 BCR를 제안하여 이러한 문제를 해결하고자 한다. BCR의 성능평가는 NS2를 이용하여 이루어 졌으며 throughput, goodput에서 좋은 성능을 보여주고 있다.

1. 서론

인터넷상에서 TCP 프로토콜은 응용계층에 신뢰성 있는 전송을 제공하기 위한 프로토콜로서 사실상 표준 프로토콜로 사용된다. 유선 네트워크로부터 무선 네트워크로 발전한 현재의 네트워크 환경에서 TCP의 성능을 개선하기 위한 많은 시도가 있었고 무선 LAN과 CDMA 같은 무선 네트워크에서도 성능을 높이기 위한 알고리즘들을 제안하여 왔다. 무선 네트워크 환경에서 이러한 시도의 요인 중 하나는 높은 패킷 손실률, link-layer 재전송과 핸드오프에 의한 delay spike 발생이다.

유선 네트워크에서 TCP는 패킷 손실을 네트워크 혼잡으로 판단한다. 패킷 손실을 다루는 대표적인 한 가지 방법으로써 만약 수신자로부터 세 개의 duplicate ACK 메시지를 받는다면 TCP 송신자는 timeout까지 기다리지 않고 잃어버린 패킷을 전송한다. 이는 fast retransmit라 널리 알려져 있다. 다른 대표적인 방법으로 RTO (Retransmission Timeout)가 발생했을 때 TCP는 cwnd(congestion window)를 1로 줄이고 slow strat threshold(ssthersh)의 절반으로 줄인다. 그리고 ACK 메시지를 받지 못한 segment를 전송한다. 그 후 TCP는 전송한 segment의 ACK 메시지를 기다린다. 만약 TCP가 새로운 ACK 메시지를 받았다면 cwnd의 크기를 늘리고 이전보다 많은 양의 패킷을 전송한다.[1]

무선 네트워크에서 무선 링크의 높은 손실률과 link-layer 재전송, 그리고 핸드오프에 의한 long-term delay와 같은 다른 종류의 이유로 인한 RTO에 대한 연구를 진행한다. delay spike발생으로 인한 spurious RTO의 발생은 패킷이 수신쪽에 잘 도착했음에도 불구하고, 재전송 타임아웃이 발생하게 된다. E-RTO와 F-RTO는 Spurious RTO 문제를 다루기 위한 메커니즘을 제공하고 있다. 그러나, TCP에서 RTO가 발생한 다는 것은 네트워크의 상황이 이전보다 좋지 않음을 의미한다. 그럼에도 불구하고 E-RTO에서는 cwnd와 ssthersh를 그대로 회복하고 그럼으로써 다른 혼잡을 야기할 수 있다. 이러한 문제의 해결방안으로써 우리는 BCR 알고리즘을 제안한다. 이는 ABE기법을 사용함으로써 가용한 대역폭을 측정하여 합리적인 전송량을 유지할 수 있다. 그럼으로 인하여 무선 네트워크에서 TCP의 성능저하를 줄이는데 공헌한다.

본 논문은 다음과 같이 구성한다. 2장에서는 RTO 발생에 의한 문제를 다룬 기존 알고리즘들을 소개하고 3장에서는 이러한 문제를 해결할 수 있는 알고리즘을 제안한다. 4장에서는 ns-2를 이용하여 기존 알고리즘들과의 비교를 통해 제안 알고리즘의 성능을 분석한다. 마지막으로 5장에서는 본 논문의 결론을 제시한다.

2. 관련연구

TCP에서 재전송은 두 가지 경우로 나눈다. 첫 번째 경우는 fast retransmit에 의한 재전송이고 다른 한 가지 경우는 RTO 발생에 의한 재전송이다. fast retransmit 알고리즘이 수행되지 않을 때 손실된 segment를 회복하기 위해 RTO에 의해 발생한 재전송이 이루어진다. 게다가

segment의 delay spike는 RTO 발생의 또 다른 원인으로 알려져 있다. 네트워크에서 특별한 segment가 손실 없이 갑작스런 지연에 의해 방해 받는 다면 전통적인 TCP 송신자는 불필요하게 unacknowledged segment들의 전부를 재전송함으로써 slow start단계를 수행한다. 이러한 spurious RTO 발생을 해결하기 위한 두 가지 선행 알고리즘이 존재한다. F-RTO[2]와 E-RTO[3] 알고리즘이 바로 그것이다.

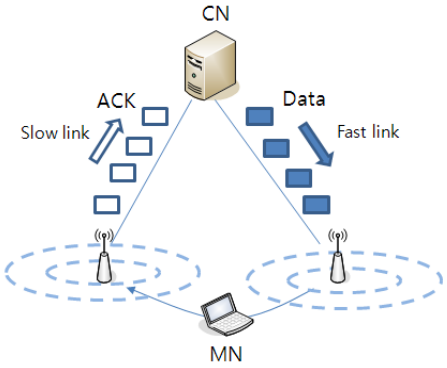
E-RTO는 F-RTO를 개선한 기법으로 delayspike에 의한 RTO발생에서 불필요한 재전송을 막기 위한 스킴으로 송신측기반의 RTO 알고리즘이다. 또한 SACK을 사용하여 RTO이후 패킷의 손실에 대한 처리를 하고 있다. E-RTO는 E-RTO Start, E-RTO Hold, 그리고 E-RTO Stop의 세 단계로 이루어져 있다. RTO가 발생 하면 먼저 RTO가 연속적으로 몇 번 발생했는지부터 체크한다. 만약 세 번 이하로 발생했다면 E-RTO의 단계는 Start단계가 되고 현재의 congestion widow값과 Slow start threshold값을 저장해둔다. 그리고 재전송 세크먼트가 아닌 다음 보낼 새로운 세그먼트를 전송하게 된다. 이렇게 RTO가 3번이상 연속적으로 발생하게 되면 기존의 TCP의 RTO 알고리즘에 따라 Slow Start단계를 수행하게된다. 이유는 계속 새로운 세그먼트를 전송하게 되면 RTO가 발생해도 한 윈도우 크기만큼 새로운 세그먼트를 전송하는 것을 막기 위해서이다. 이렇게 새로운 세크먼트를 전송하고 ACK가 돌아오기를 기다린다. 첫 번째 도착한 새로운 ACK의 시퀀스 넘버를 확인하게 된다. 만약 시퀀스 넘버를 확인하여 RTO 이후 전송한 새로운 세그먼트의 시퀀스 넘버와 같다면 지금까지 보낸 세그먼틀 수신측에 모두 잘 도착했다고 판단하게 되어 저장해 두었던 congestion window 크기와 Slow start threshold의 크기를 그대로 복원해주고 E-RTO 상태를 Stop으로 두고 알고리즘을 끝내게 된다. 도착한 첫 번째 ACK의 시퀀스 넘버가 새로 전송한 세그먼트의 시퀀스 넘버와 같지 않다면 두 번째 ACK를 기다리고 두 번째 ACK도 Dupack이 아니라면 같은 방법으로 저장해 두었던 congestion window 크기와 Slow start threshold의 크기를 그대로 복원해주고 E-RTO 상태를 Stop으로 두고 알고리즘을 끝내게 된다.

TCP Westwood는 선행 혼잡제어 방식의 무선 TCP 스킴으로 송신자가 단위시간당 받은 ACK 패킷에 대한 도착 비율을 계산한다. 본 스킴에서는 현재 네트워크의 가용 대역폭을 예측하여, 패킷 손실이 발생하면 예측된 대역폭 값을 기반으로 전송량을 조절한다. 송신자가 현재 네트워크의 상태를 기반으로 전송량을 조절하기 때문에 네트워크 상태에 따라서 대역폭의 이용 효율이 높아져 결과적으로 전송률의 향상을 가져온다.

3. 제안 알고리즘

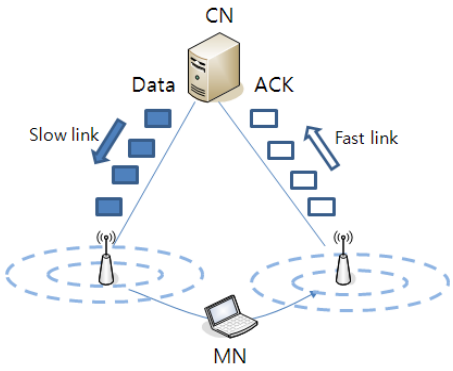
3.1 동기

E-RTO는 RTO 발생 원인을 판단하기 위해 RTO가 발생하면 현재의 cwnd값과 ssthresh값을 저장하고 새로운 패킷을 전송한다. 이후 TCP 송신자는 돌아오는 ACK를 확인하고 받은 ACK 가 RTO 이후 전송한 새로운 패킷과 같은 시퀀스 넘버라면 spurious RTO라 판단하여 RTO 발생 이전에 저장했던 cwnd와 ssthresh 값으로 복원한다. 하지만 RTO가 발생 했다는 것은 현재 네트워크가 이전의 네트워크 보다 혼잡함을 의미한다. 이러한 혼잡한 네트워크에서 E-RTO와 같이 RTO 발생 이전의 cwnd와 ssthresh 값으로 복원한다면 한 번에 많은 양의 패킷을 전송하여 네트워크가 더욱 혼잡해지는 경우가 발생할 수 있다.



(그림 1) 빠른 링크에서 느린 링크로 이동

또한 최근의 네트워크는 다수의 무선 네트워크가 공존하는 경우가 다수 존재한다. 예를 들어 3G Cellular 네트워크와 무선 LAN이 공존하는 경우 3G Cellular 네트워크는 높은 이동성을 제공하는 대신 낮은 대역폭을 제공하고 무선 LAN은 낮은 이동성을 제공하는 대신 높은 대역폭을 제공한다. 이와 같이 서로 다른 대역폭을 제공하는 네트워크의 사용자들이 이동함에 따라 발생하는 핸드오프로 인한 delayspike가 발생한다. 이 때 cwnd와 ssthreshold값을 delayspike 이전의 크기와 동일하게 복원하는 것은 새로운 네트워크 환경을 적절히 반영하지 못하는 결과를 초래한다.



(그림 2) 느린 링크에서 빠른 링크로 이동

(그림 1)은 이동단말 노드가 높은 대역폭의 네트워크에서 낮은 대역폭의 네트워크로 이동하는 과정을 표현한 그림이다. (그림 1)에서와 같이 이동단말 노드가 이동성의 핸

드오버로 인핸 delayspike가 발생했을 경우 E-RTO 기법은 cwnd와 ssthreshold를 빠른 링크에서 사용하던 값으로 복원한다. 이는 느린 링크에서의 능력을 벗어난 전송을 유도하고 delayspike에 의한 RTO발생에 이어 또 다른 혼잡의 발생을 유도하고 이로 인한 TCP의 성능저하를 예상할 수 있다. 이와는 반대로 (그림 2)와 같이 이동단말 노드가 느린 링크에서 빠른 링크로 이동할 경우 E-RTO 기법은 delayspike 이후에 발생한 RTO에 대해 cwnd와 ssthresh 값을 느린 링크에서 사용하는 값으로 복원한다. 빠른 링크의 수용력이 충분히 있음에도 느린 링크의 cwnd와 ssthresh값을 사용함으로써 가용한 대역폭의 낭비를 가져온다. 또한 전송량을 떨어뜨리게 되며, 이는 TCP의 성능저하로 이어진다. 이러한 이유로 모바일 노드가 환경이 서로 다른 네트워크로 이동하는 경우 적절한 전송량 유지를 위해 그 상황에 맞는 cwnd와 ssthresh의 값을 측정하는 과정이 필요하다.

3.2 제안 알고리즘

Spurious RTO로 판단하면 이후의 네트워크 상황을 고려한 cwnd와 ssthreshold의 회복을 위해 가용한 대역폭을 측정한다. 측정된 값을 이용하여 혼잡윈도우 값을 적절한 값으로 회복시켜준다. 이를 위한 과정으로 Westwood의 ABE 기법을 도입하여 네트워크 상황에 적절한 cwnd와 ssthreshold값을 구한다.

ABE를 측정하기 위해 제안하는 알고리즘에서는 Westwood의 ABE 알고리즘을 사용하여 가능한 대역폭을 측정한다. 가용한 대역폭을 측정하기 위해서는 수신한 ACK로부터 다음 두 가지 정보를 수집하게 된다.

1) ACK를 수신하는 비율 : ACK를 수신하는 시간의 간격
2) 수신측에 보낸 데이터의 크기

위의 얻은 두 개의 정보를 바탕으로 다음과 같이 가용한 대역폭을 계산할 수 있다.

$$BW_{current}(k) = \frac{d_k}{t_k - t_{k-1}}$$

여기서 는 ACK를 수신하는 비율로서 t_k에 ACK를 수신한 것을 나타낸다. d_k는 수신측에 보낸 데이터의 크기를 말한다. 가능한 대역폭은 ACK를 수신할 때, 일정한 간격으로 측정되어 현재 가능한 대역폭을 계산하게 된다. 이렇게 계산된 현재의 가용한 대역폭과 과거의 측정된 대역폭의 계산을 통해서 대역폭을 다음과 같이 계산하게 된다.

$$BW_{Estimated}(k) = BW_{Estimatied}(k-1) \times \alpha + BW_{current}(k) \times (1-\alpha)$$

여기서 α 값은 0.2로 계산하였다. α의 값에 따라서 이전에 측정한 대역폭 값의 반영비율을 결정한다.

RTO가 발생하게 되면 기존의 TCP에서는 ssthresh 값을 반으로 줄이게 되고 cwnd값을 1로 줄임으로써 전송량을 줄인다. 하지만 이러한 메커니즘은 Delay spike에 의한 RTO의 발생의 경우 불필요한 전송량의 감소를 가져오게 되어 TCP의 성능저하의 원인이 된다. 그래서 BCR에서는 불필요한 전송량의 감소를 막고 적절한 전송량을 유지하기 위해서 다음 수식에 따르는 cwnd와 ssthresh값을 정정한 값으로 복원해준다.

1) Slow Start threshold

$$ssthresh = \frac{BW_{Estimated} \times RTT_{base}}{Segment\ Size}$$

여기서 RTT_{base}는 설정된 연결 동안 측정된 round trip time값의 최소값을 의미한다.

2) Congetion window

If(ocwnd ↘ssthresh)
cwnd=ssthresh
Else if (ocwnd<ssthresh)
Cwnd=ocwnd

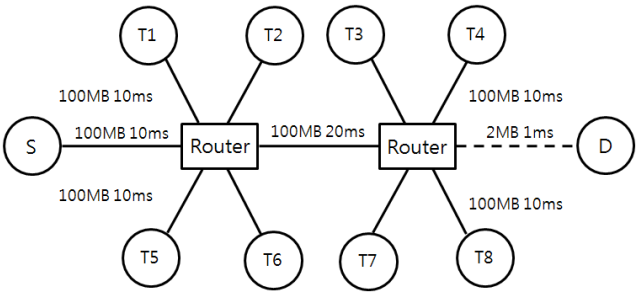
4. 성능평가

새로 제한한 스킴의 성능평가를 위하여 본 논문에서는 LBNL(Lawrence Berkly National Laboratory)의 NS-2(network Simulator)를 이용하여 실험하였다. 실험은 Throughput, goodput을 측정하여 비교하였다. 토플로지는 다음에 나오는 표에 나오는 파라미터들로 구성이 되어 있다.

<표 1> 시뮬레이션 매개변수

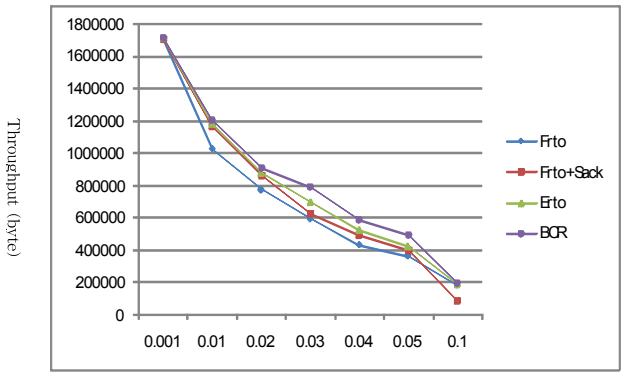
Bandwidth	Wired : 100MB / Wireles : 2MB
Packet Size	762byte
Propagation Delay	Wired : 10, 20ms / Wireless : 1ms
Queue Size	20-200 packets

양방향 백그라운드 트레픽이 있는 토플로지에서의 Throughput 과 goodput을 측정하였으며, 백그라운드 트래픽의 양을 조절하였다. 처음 실험은 delayspike를 기준으로 백그라운드 트래픽의 양이 많아지는 경우로 네트워크 환경이 delayspike 이후에 나빠지는 것을 의미하게 되고 두 번째 실험은 백그라운드 트래픽의 양이 줄어드는 경우로 네트워크 환경이 delayspike 이후에 좋아지는 것을 의미 하게 된다. 두 실험을 통해 평균값을 구하여 그래프로 나타내었다.

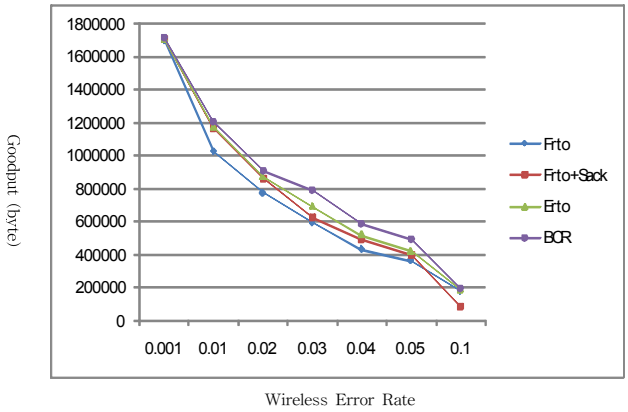


(그림 3) 토플로지

(그림 3)의 토플로지는 10노드와 2개의 라우터로 구성되어 있다 각 노드에서 라우터로 연결된 링크는 100MB의 대역폭과 10ms 전송지연의 유선링크로 연결되어 있다. 두 개의 라우터는 100MB의 대역폭과 20ms 전송지연의 유선링크로 연결되어 있으며 목적노드와 라우터는 2MB의 대역폭과 10ms 전송지연의 무선링크로 연결되어 있다. 백그라운 트래픽은 T1에서 T3, T2에서 T4로 FTP 트래픽이 존재하며, 반대방향으로 T7에서 T5으로, T8에서 T6로 FTP 트래픽이 존재한다. delayspike는 시뮬레이션이 시작한 후 10초 후에 발생하며 5초간 발생하는 상황에서 무선망의 에러율(0.1%, 1%, 2%, 3%, 4%, 5%, 10%)에 따라서 각 스킴들의 Throughput 과 goodput을 측정하였다. 에러율에 따른 각 각의 그래프는 다음과 같다.



(그림 4) Thoughput



(그림 5) Goodput

Delayspike의 영향을 최대화하기 위해서 시뮬레이션 시간을 짧게 설정하였다. 따라서 Thoughput에 Goodput 차이가 적다. 그러나 백그라운드 트래픽의 양이 변함에 따라 BCR은 적절한 혼잡 윈도우 값으로 회복하면서, 혼잡을 회피하거나 가능한 대역폭을 획득함으로써, Thoughput이나 goodput에서 좋은 성능을 보여주고 있다.

5. 결론

이 논문은 E-RTO의 congestion window 크기와 slow start threshold을 회복하는 단계에서 갑작스럽게 많은 데이터를 보내는 것을 완화하고자하는 것에서 시작되었다. 뿐만 아니라 현재의 네트워크 상황에서 흔히 존재하는 하이브리드 네트워크 상황에서 E-RTO의 delayspike 이후 congestion window 크기와 slow start threshold을 회복하는 메커니즘이 문제가 내재되어 있음을 설명하였다. 이러한 문제를 해결하고자 ABE를 사용하여 적절한 전송률을 유지할 수 있도록 하였다. NS-2을 통하여 실험결과 제안된 스킴은 다른 기법들보다 Thoughput에서 성능향상뿐만이 아니라 goodput에서 향상을 보였다. 이는 E-RTO의 경우 RTO발생이후 많은 패킷을 한 번에 전송하여 또 다른 혼잡을 유발하는 반면 제안된 스킴은 적절한 전송량을 측정하여 전송하여 혼잡을 유발하지 않고 안정적으로 패킷을 전송할 수 있는 것이 주요 원인이다.

ACKNOWLEDGMENT

본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-(C1090-0801-0046)). 교신저자 : 추현승

참고문헌

[1] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," RFC 2581, Apil 1999.

[2] P. Sarolahti, M. Kojo, and K. Raatikainen, "F-RTO: An Enhanced Recovery Algorithm for TCP Retransmission Timeouts," ACM SIGCOMM Computer Communication Review, vol. 33, no. 2, April 2003.

[3] M. J. Lee and O. C. Kwon, "E-RTO: An Enhanced TCP Retransmission Timeout Algorithm using ACK option" Proceedings of the IEEE WCNC, 2006.

[4]"UCB/LBNL/VINT Network Simulator," available on-line at http://www.isi.edu/nsnam/ns/.

[5] A. Gurtov and R. Ludwig, "Responding to Spurious Timeouts in TCP" IEEE International Conference on Computer Communication, vol. 3, pp. 2312-322, March 2003.

[6] N. C. Wang, Y. Y. Wang, S. C Chang, "A Fast Adaptive Congestion Control Scheme for Improving TCP Performance during Soft Vertical Handoff," Proceedings of the IEEE WCNC, 2007