종단간 순방향 역방향 전송 지연 측정을 이용한 TCP Vegas 성능 향상

신영숙*, 김은기
국립 한밭대학교 정보통신전문대학원
e-mail : seabeauty80@hanmail.net*, egkim@hanbat.ac.kr

Performance Improvement of TCP Vegas by measuring of End-to-End Forward/Backward delay variation

Young-Suk Shin*, Eun-Gi Kim
Graduate School of Information & Communications, Hanbat National University

요 약

TCP 구현의 하나인 Vegas는 패킷의 유실을 망의 혼잡으로 인지하는 Reno와 달리 RTT(Round Trip Time) 측정값을 바탕으로 혼잡을 인지하며 윈도우 크기 등 혼잡 제어를 위한 주요 인자를 결정한다. TCP Vegas는 TCP Reno보다 더 효율적인 네트워크 대역폭과 처리율을 가진다. 그러나 Vegas의 혼잡 회피 방안이 TCP 패킷 경로의 비대칭적 특성을 제대로 반영하지 못하며, 이것은 양방향(순방향,역방향) 패킷 전송 상태를 반영하는 RTT 측정값을 순방향 경로의 상태 해석에 이용하기 때문이다. RTT는 패킷의 왕복 시간만을 측정하기 때문에 패킷의 송수신시 순방향과 역방향에서 어느 정도의 혼잡이 발생하였는지 알 수 없다. 본 논문에서는 리눅스 커널의 TCP 소스에서 RTT 측정값으로 혼잡도를 측정하는 기존의 Vegas 혼잡 제어 알고리즘을 수정하여 순방향 경로의 혼잡과 역방향 경로의 혼잡을 구별할 수 있는 새로운 Vegas 혼잡 제어 알고리즘을 설계하고 구현하여 그 성능을 분석하였다.

1. 서론

TCP (Transmission Control Protocol)의 혼잡 제어 메커니즘은 1988년의 TCP Tahoe이래 1990년 TCP Reno, 1995년 TCP Vegas에 이르기까지 다양한 버전으로 구현되었다. TCP의 혼잡 제어 메커니즘의 기본적인 알고리즘은 저속 출발(Slow Start), 혼잡 회피(Congestion Avoidance), 빠른 재전송(Fast Retransmit), 그리고 빠른 복구(Fast Recovery)로 구성되어 동작된다[1][2][3].

TCP 구현의 하나인 Vegas는 패킷의 유실을 통해 망의 혼잡을 인지하는 Tahoe, Reno 등의 TCP 구현들과는 달리 RTT(Round Trip Time)측정 값의 변화를 분석하여 망의 혼잡을 인지한다[4].

Vegas의 RTT 기반 혼잡 인지 방법은 혼잡의 초기 상황을 빠르게 인지할 수 있어, 혼잡 발생 이후에 망혼잡을 인지하고 대처하는 기존 구현에 비해 패킷 유실이 적고 따라서 재전송하는 데이터 양이 적다. 또한 혼잡이 발생할 때까지 꾸준히 혼잡윈도우(CWND) 크기를 증가시키는 기존 구현과 달리, 네트워크 내에서 버퍼링 되는 패킷의 수를 특정 범위 이내로 제한하여 혼잡윈도우 크기의 과대화를 막아, 높은 처리율을 유지하면서도 혼잡 상황을 억제하는 특징이 있다.

그러나 Vegas의 기본 아이디어라고 할 수 있는 RTT 측정값 해석은 다음과 같은 문제를 갖는다. 즉, RTT 측정값은 양방향 경로의 속성을 나타내는 값으로 이를 이용하는 Vegas는 순방향 경로(응용계층의 데이터가 전달되는 경로)에서 혼잡이 발생했는지 역방향 경로(순방향 데이터에 대한 확인응답(ACK)이 전달되는 경로)에서 혼잡이 발생했는지 알 수 없다.

본 논문에서는 RTT 측정값으로 혼잡도를 측정하는 기존 Vegas의 단점을 보완하여 Vegas가 순방향 경로의 혼잡과 역방향 경로의 혼잡을 구별하여 효율적으로 혼잡 제어를 할 수 있는 새로운 혼잡 제어 알고리즘을 설계하고 구현하여, 그 성능을 분석하였다.

2. 관련 연구

2.1 TCP Vegas 알고리즘

① 해당 TCP 연결에 대하여 BaseRTT를 구한다.

- 해당 TCP 연결이 혼잡을 겪지 않을 때의 RTT값을 BaseRTT라고 정의한다. Vegas구현에서는 해당 연결의 모든 RTT 측정값들 중 최소값을 BaseRTT로 지정한다.
- 이 TCP 연결의 현재 윈도우 크기가 W라면 이 연결에 대하여 기대할 수 있는 전송률, Expected는 다음과 같다.

$$Expected = W / BaseRTT$$

② 현재의 실제 전송률, Actual을 구한다

- 실제 전송률은 W를 현재 RTT 측정값(가장 최근의 RTT 측정값) 으로 나누어 구한다

$$Actual = W / RTT$$

- BaseRTT <= RTT이므로 Actual <= Expected가 항상 성립한다.
- 실제 전송률 계산은 매 RTT 마다 한번 수행한다.

③ 실제 전송률과 기대 전송률을 비교하고, 이를 바탕으로 윈도우 크기 W를 적절하게 변경한다.

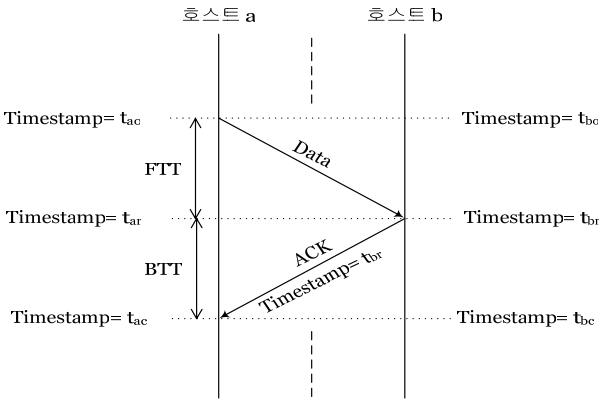
$$Diff = (Expected - Actual) \times BaseRTT$$

- α , β 라는 두 가지 임계값을 정의한다. α는 너무 적은 데이터를, β는 너무 많은 데이터를 네트워크 내에서 가지고 있음을 판단하는 기준 값이다. Vegas는 TCP 연결이 네트워크 내에서 가지는 여분의 데이터, 즉 망 경로의 버퍼 용량을 차지하는 데이터 양을 α 와 β값 사이로 제한하고자 한다.
- 이를 위해 Vegas는 Diff < α 일 경우 W를 선형 증가시키고, Diff > β 일 경우 W를 선형 감소 시킨다. Vegas의 선형 증가/선형 감소 방법은 기존 구현의 선형 증가/지수형 감소에 비해 TCP 연결의 W변화가 급격하지 않아 연결의 전송 상태가 안정적이며, TCP 트래픽의 Bursty 속성을 완화시켜 네트워크 역시 안정적인 동작을 할 수 있도록 한다[5][6].

2.2 순방향/역방향 전송 시간 측정 알고리즘

기존 RTT 측정 방법은 호스트간 타임스탬프 카운터의 시간차로 인하여 망에 혼잡이 발생했을 때 혼잡 경로 파악이 불가능 하였다. 이러한 문제의 해결을 위해 다음과 같은 알고리즘을 [7]에서 제안하였다.

이 알고리즘은 다수의 RTT를 측정하여, 가장 작은 RTT의 경우 왕복 전송 경로에 가장 혼잡이 없고, 이때의 순방향과 역방향 전송 시간이 동일하다는 기본 가정으로부터 시작한다. 이러한 가정에서, 가장 작은 RTT값을 갖는 경우로부터 계산된 순방향/역방향 전송 시간을 사용하여 호스트간 타임스탬프 카운터의 시간 차를 계산한다. 계산된 시간차를 이용하여 패킷을 받을 때 마다 순방향과 역방향 전송 시간을 계산한다.



(그림 1) 종단간 순방향/역방향 전송 지연 측정 파라메타

(그림 1)은 호스트 a와 b의 패킷 교환과 종단간 순방향/역방향 전송 지연 시간 측정을 위한 요소들을 자세하게 보여주고 있으며, 알고리즘을 설명하기 위하여 필요한 변수들을 다음과 같이 정의한다.

- t_{hi} : 호스트 h의 TSC 값, i=o(originate), r(receive), c(current).
- TD_{ab} : 호스트 a와 b에 세트 된 TSC의 차이.
- RTT_{cp} : 현재 패킷의 RTT 값.
- RTT_{min} : 지금까지의 RTT 측정값들 중 최소값.
- FTT_{cp}: 현재 패킷의 순방향 전송 시간(Forward Transmission Time).
- FTT_{ref} : RTT_{min} 일 때의 FTT_{cp} 값 ($FTT_{ref} = RTT_{min}/2$).
- BTT_{cp}: 현재 패킷의 역방향 전송 시간 (Backward Transmission Time).
- BTT_{ref} : RTT_{min} 일 때의 BTT 값($BTT_{ref} = RTT_{min}/2$).

(그림 2)는 본 논문에서 제안한 알고리즘을 의사코드(pseudo code)로 나타낸 것이다. 알고리즘의 동작을 간단히 설명하면 다음과 같다. 지금까지 전송된 모든 패킷의 RTT 값들 중에서 가장 작은 값을 찾아, 이 값을 기준으로 FTT_{ref}, BTT_{ref}, TD_{ab} 값을 계산한다. 그리고, 8~9 줄에서는 현재 패킷의 순방향 전송 시간과 역방향 전송 시간을 계산한다.

```
01: RTT_min = big number;
02: until (end of packet) {
03:     if (RTT_min > RTT_cp) {
04:         RTT_min = RTT_cp;
05:         FTT_ref = BTT_ref = RTT_min / 2;
06:         TD_ab = t_ac - t_br - BTT_ref;
07:     }
08:     FTT_cp = t_br - t_ao + TD_ab;
09:     BTT_cp = t_ac - t_br - TD_ab;
10: }
```

(그림 2) 제안된 순방향/역방향 측정 알고리즘

3. 새로운 TCP Vegas 혼잡 제어 알고리즘의 설계

순방향 경로에서 혼잡이 발생할 경우 순방향 데이터 전송 량을 줄이는 것은 혼잡을 제어하기 위한 적당한 방법이다. 그러나 역방향 경로에서 혼잡이 발생할 경우 순방향 데이터 전송 량을 줄이는 것은 옳지 않을 수 있다.

역방향 경로의 혼잡 원인은 일반적인 경우 역방향 데이터의 과다에 따른 것이므로, 역방향 경로의 데이터를 감소시켜서 해결할 수 있다. 물론 순방향 경로에서 전송되는 데이터를 감소시켜 상대적으로 역방향 경로상의 ACK 양을 줄임으로서 역방향의 혼잡을 어느 정도 해소할 수는 있을 것이다. 그러나 순방향 데이터의 감소량에 비해 역방향 데이터의 감소가 미미하여 역방향 경로의 혼잡은 해결하지 못한 채 순방향 경로의 전송률만 떨어뜨릴 수 있다.

이러한 결과를 반영하여 다음과 같이 기존의 Vegas 혼잡제어 알고리즘을 수정한다.

- Expect = W / BaseRTT ---------- 식1
- Actual = W / RTT ---------- 식2
- RTT = FTT_Delay + BTT_Delay + BaseRTT ------ 식3
- FTT_Delay + BTT_Delay = RTT - BaseRTT
- FTT_Delay = FTT – BaseFTT ---------- 식4
- M_Actual = W / (BaseRTT + FTT_Delay) ----- 식5
- Diff = (Except – M_Actual) × BaseRTT ----- 식6

• FTT : 순방향 전송 시간
• BTT : 역방향 전송시간
• FTT_Delay : 현재 측정된 순방향 경로의 지연값
• BTT_Delay : 현재 측정된 역방향 경로의 지연값
• BaseRTT : 해당 TCP 연결이 혼잡을 겪지 않을 때의 RTT값

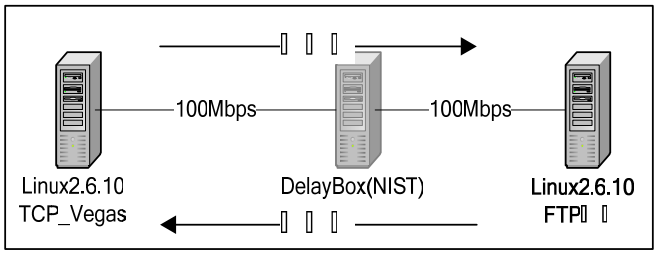
식1의 Expect의 경우 순방향/역방향 모두에 대해 혼잡을 배제하였으므로 그대로 사용해도 된다. 그러나, 식2의 기존 Actual의 계산 과정에 이용되는 RTT 측정값은 양방향의 상태를 나타내므로, 식5처럼 RTT 측정값에서 역방향 경로의 지연을 제거함으로써 역방향 경로의 혼잡으로 순방향 경로의 데이터 전송을 줄이는 원인을 제거할 수 있다. 식3과 같이 혼잡이 가장 없을 때의 RTT값에 순방향/역방향 경로의 지연값을 더해 계산된 값이 현재 측정된 RTT이다. 식4와 식5에 서 계산된 값을 식6에 적용시켜 Diff값을 계산한다.

4. 새로운 TCP Vegas 혼잡 제어 알고리즘 성능 분석

4.1 테스트 환경

새롭게 설계된 TCP Vegas의 성능측정과 분석을 위해 리눅스 2.6.10 커널을 수정 하였다.

다양한 환경에서의 성능 분석을 위해 종단간 시스템 사이에 임의적인 전송지연을 발생하기 위한 딜레이 박스(NIST)[8]를 사용하였다. 실험 환경은 (그림 3)과 같다.



(그림 3) 실험 환경 구성

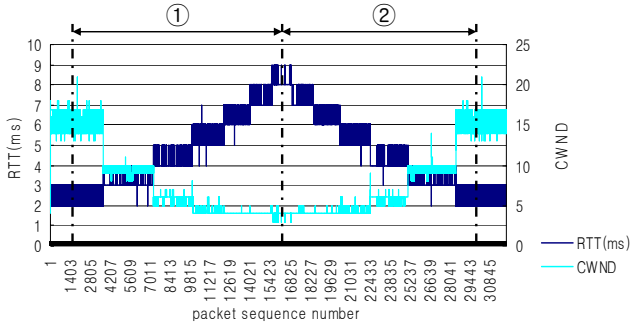
실제 네트워크와 비슷하게 트래픽을 발생시키기 위해 기존의 Vegas와 수정된 Vegas에 <표 1>과 같이 순방향/역방향 경로에 각각 1ms씩 증가 또는 감소하도록 임의적인 지연을 주고 각 지연 시간은 10초 동안 지속 되도록 하여 총 130초 동안 파일 전송을 테스트 하였다.

<표 1> 순방향/역방향 경로의 지연 시간

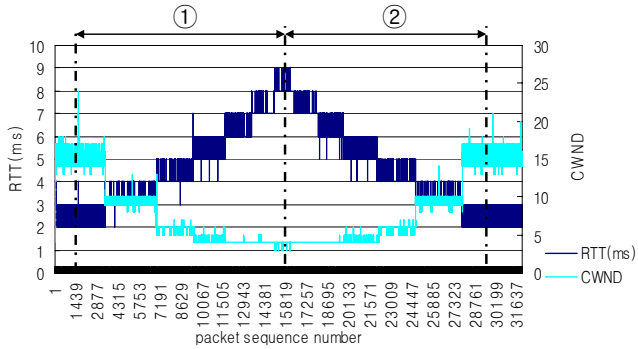
순방향지연	FTT (ms)	0	1	2	3	4	5	6	5	4	3	2	1	0
	BTT (ms)	0 (임의적인 지연을 주지 않음)												
역방향지연	FTT (ms)	0 (임의적인 지연을 주지 않음)												
	BTT (ms)	0	1	2	3	4	5	6	5	4	3	2	1	0

4.2 성능 분석

4.2.1 기존 TCP Vegas의 성능분석



(그림 4) 기존 TCP Vegas에서의 순방향 지연 시 RTT와 CWND의 관계

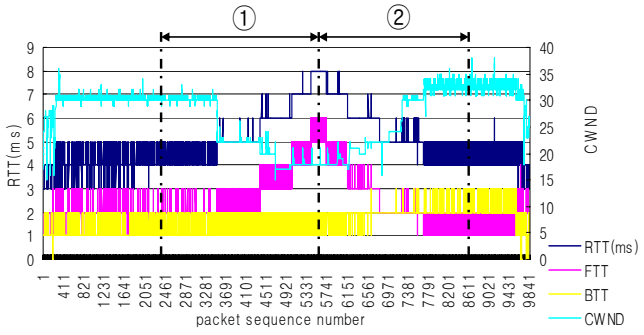


(그림 5) 기존 TCP Vegas 에서의 역방향 지연 시 RTT 와 CWND 의 관계

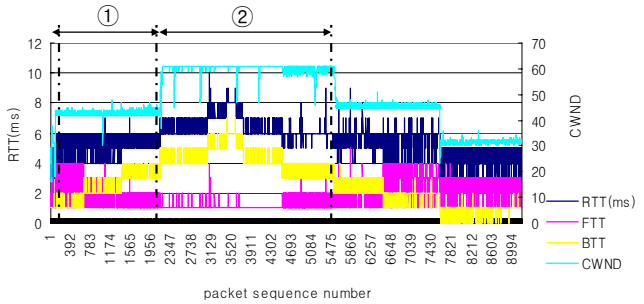
(그림 4)와 (그림 5)는 기존 TCP Vegas 에서 각각 순방향과 역방향에 전송지연을 주었을 때의 결과를 보여주고 있다. (그림 4)의 구간 ①과 같이 순방향 경로에만 지연을 1ms 씩 증가시킬 경우 CWND 를 서서히 감소시켜 데이터 전송률을 낮추고 있으며, 반대로 구간 ②와 같이 순방향 경로에만 지연을 1ms 씩 감소시킬 경우 CWND 를 서서히 증가시켜 데이터 전송률을 높이고 있다. 그러나 (그림 5)의 구간 ①과 같이 순방향 경로에는 어떠한 지연도 주지 않고 역방향 경로에만 지연을 1ms 씩 증가시켰는데도 불구하고 CWND 를 서서히 감소시켜 불필요하게 데이터 전송률을 낮추고 있다. 이는 결과에서 볼 수 있듯이 기존의 TCP Vegas 는 순방향과 역방향의 지연을 구분하지 않고 단순히 RTT 측정값만을 사용하여 혼잡 제어를 하기 때문이다. 또한 (그림 5)의 구간 ②는 역방향 지연이 감소함에 따라 CWND 를 서서히 증가시켜 데이터 전송률을 높이고 있다.

따라서 기존의 TCP Vegas 는 혼잡한 망에서 순방향지연과 역방향 지연에 관계없이 RTT 가 커지면 CWND 를 감소시켜 순방향의 데이터 전송률을 낮추게 되어 결국 불필요하게 데이터 전송률을 감소시키게 되는 단점이 있다.

4.2.2 수정된 TCP Vegas 의 성능분석



(그림 6) 수정된 TCP Vegas 에서의 순방향 지연 시 RTT 와 CWND 의 관계



(그림 7) 수정된 TCP Vegas 에서의 역방향 지연 시 RTT 와 CWND 의 관계

(그림 6)과 (그림 7)은 본 논문에서 제안한 TCP Vegas 혼잡 제어 알고리즘을 적용하여, 순방향과 역방향 경로에 각각 혼잡을 발생시켰을 때의 결과를 보여주고 있다.

(그림 6)의 구간 ①과 같이 순방향 경로에만 지연을 1ms씩 증가시킬 경우 기존의 TCP Vegas처럼 CWND를 서서히 감소시켜 데이터 전송률을 적절하게 낮추고 있으며, 반대로 구간 ②와 같이 순방향 경로에만 지연을 1ms씩 감소시킬 경우 CWND를 서서히 증가시켜 데이터 전송률을 높이고 있다. 또한 (그림 7)의 구간 ①과 같이 역방향 경로에만 지연을 1ms씩 증가시켰는데도 불구하고 역방향 경로에 비해 순방향 경로에 지연이 거의 없으므로 불필요하게 데이터 전송률을 낮추지 않고 있으며, 오히려 구간 ②와 같이 순방향 경로에 지연이 줄어드는 경우 데이터 전송률을 높이고 있다.

따라서 제안된 TCP Vegas는 효과적으로 순방향과 역방향의 혼잡 상태를 구분하고 있으며, 이에 따라 적절하게 데이터 전송률을 제어하는 것을 볼 수 있다.

5. 결론 및 향후 계획

TCP Vegas는 양방향 경로의 속성을 나타내는 RTT 측정값을 바탕으로 혼잡 제어를 한다. 따라서, 전송 경로에 혼잡이 발생했을 때 혼잡의 원인이 순방향 경로인지 아니면 역방향 경로인지 알 수가 없는 문제를 가지고 있다. 따라서, Vegas는 역방향 경로에서 혼잡이 발생했는데도 불구하고 순방향 데이터의 전송률을 낮추게 된다.

하지만 본 논문에서 제안한 새로운 TCP Vegas 혼잡제어 알고리즘은 비대칭적인 RTT 환경에서 순방향과 역방향 지연을 효과적으로 구분하고, 이를 이용하여 순방향 경로에 혼잡 발생시 혼잡을 회피하기 위해서 전송률을 낮추고, 역방향 경로에 혼잡 발생시 순방향의 전송률을 불필요하게 낮추지 않아 효과적인 데이터 전송률을 지속시킬 수 있었다.

향후 과제로는 보다 다양한 혼잡 요소를 적용시킨 환경에서 기존 TCP Vegas와 제안된 TCP Vegas의 성능 비교를 통해 제안된 TCP Vegas의 우수성을 검증해야 할 것이다.

참고문헌

[1] RFC 793, “Transmission Control Protocol”, http://www.ietf.org

[2] 안순신, 김은기, “정보통신 네트워크”, 이한출판사, pp.168-180, 1998.

[3] Behrouz A. Forouzan, “TCP/IP Protocol Suite”, McGraw-Hill, 2003.

[4] W. Richard Stevens, “TCP/IP Illustrated, Volume1 The Protocols”, Addison-Wesley, 1994.

[5] Lawrence S. Brakmo, Sean W. O’Malley, Larry L. Perterson, “New Techniques for Congestion Detection And Avoidance”, 1994. http://citeseer.ist.psu.edu/brakmo94tcp.html

[6] Jeonghoon Mo, Richard J.La, “Analysis and Comparison of TCP Vegas and TCP Reno”, 1998.

[7] 황순한, “Measurement of End-to-End Forward/Back-ward Delay variation”, 한밭대학교, 2005.

[8] “NIST(National Institute of Standards and Techonology) “, http://snad.ncsl.nist.gov/itg/nistnet/