유무선 망에서의 TCP Westwood 의 성능 향상 기법

이재형 조인휘 ○

한양대학교 전자컴퓨터통신공학과

njaylee@hanyang.ac.kr, iwjoe@hanyang.ac.kr

An Enhancement TCP-Westwood over Wire/Wireless Networks

Lee, Jaehyung Joe, Inwhee○

Department of Electronics Computer Engineering, Hanyang University

요 약

TCP는 유선망에 최적화되도록 변화 된 결과, 무선망에서는 무선망의 단점인 비연속적인 연결 링크로 인해 전송손실이 발생하게 된다. 그러나 어플리케이션 프로그램에서는 이러한 문제로 인해 발생되는 오류를 네트워크의 혼잡으로 인한 손실로 오인하게 된다. 그 결과 어플리케이션 프로그램에서는 혼잡 제어 메커니즘이 수행되어 전송율을 줄이므로 네트워크 성능이 저하되는 문제점을 초래한다. 이러한 이유로 최근 유무선 혼합망에서 TCP의 성능을 향상시키기 위한 연구가 활발히 이루어지고 있다. 본 논문에서는 기존에 제안된 개선 모델들 중 상대적으로 많이 사용되고 있는 TCP Westwood에 대해서 성능을 보완하는 기법을 제안하고자 한다.

1. 서 론

무선 인터넷이 보급됨에 따라 기존에 널리 사용되던 인터넷의 서비스(웹 브라우저, email 등)를 그대로 무선 인터넷에도 적용하려 하고 있다. 이러한 서비스들은 Transmission Control Protocol (TCP)를 기반으로 서비스를 하게 되지만 이 프로토콜은 에러가 거의 없는 안정한 링크(link)를 대상으로 설계되었기 때문에 무선 링크처럼 지연 시간이 길고 패킷 유실이 많은 환경에서는 적절하게 동작하지 못하는 단점이 있다. 이는 TCP가 패킷의 유실이 있을 때마다 그 원인이 네트워크의 혼잡(congestion)이라 판단하고 fast retransmit와 fast recovery같은 종단 혼잡 제어(end-to-end congestion control) 알고리즘을 사용하기 때문이다. 그 결과, 링크의 활용도(link utilization)가 낮아질 수 있으며, TCP의 성능 자체도 크게 감소한다[1].

이와 같은 TCP의 문제점을 개선하기 위해 많은 연구들이 진행되어 왔다. 대표적으로 J-TCP, I-TCP TCP Jersey 연구로는 등이 있는데, 이들은 대부분 매우 제한적인 환경에서만 적용 가능한 경우이며, 기존 네트워크 프로토콜의 많은 수정이 요구된다는 단점이 있다. 반면 TCP-Westwood의 경우에는 무선과 유선환경에 모두 적용 가능하며 프로토콜의 수정이 많지 않아 사용하는 데 이점이 있다.[1]

이 논문에서는 앞에서 언급한 TCP-Westwood를 이용하여 무선네트워크 환경에서의 성능을 향상시키기 위한 방법을 제시하고자 한다. TCP-Westwood는 앞서 언급한 TCP들 중 가장 일반적으로 적용가능하며 sender-base를 기본으로 하고 있기 때문에 서버에서만 TCP-Westwood를 사용함으로써 보다 나은 서비스를 제공할 수 있다. 이에 대해서 본문에서는 TCP-Westwood의 동작방식을 설명하고, 성능 향상을 위한 방법을 도출하고, 시뮬레이션을 통해서 제안모델의 성능에 관해 분석하고자 한다.

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음(IITA-2008(C1090-0801-0047))

2. 관련 연구

2.1 TCP

TCP는 현재 인터넷의 가장 대표적인 전송 프로토콜로 이용되고 있다. 그러나 TCP를 무선 환경에서 사용할 경우 몇 가지 문제점이 발생한다. TCP는 모든 패킷 손실을 혼잡으로 인한 손실로 인지하며, 혼잡을 인한 손실이 아닐 경우에도 불필요한 혼잡제어를 실행하여, 전송률 저하시키는 문제를 발생시킨다[2].

2.2 TCP-Westwood

TCP-Westwood는 TCP의 고유한 end-to-end 특성을 이용한 접근 방법으로 유무선 통합 링크에서 ACK를 이용한 샘플링 기법을 사용하여 현재의 가용 대역폭을 측정한다. 유선 링크에 있는 송신단들이 병목구간을 거쳐 무선 링크에 있는 수신단으로 패킷을 전송할 경우에 송신단은 사용할 수 있는 최대 가용 대역폭(Bandwidth estimated, BWE)을 계산하여 혼잡 윈도우 크기(Congestion window, cwnd) 및 슬로우 스타트 임계값(Slow Start threshold, ssthresh)을 설정한다. TCP-Westwood는 계산된 가용 대역폭을 이용하여 효과적인 데이터 전송을 한다는 장점을 갖는다.

TCP-Westwood(TCPW)는 무선뿐 아니라 유선의 네트워크에서도 TCP Reno 의 성능을 향상시키는 sender 측 TCP congestion window algorithm의 수정이다. 일반적 아이디어는 대역폭을 사용하는 것은 BWE의 측정을 congestion episode 이후 congestion window(cwin)와 slow start threshold(ssthresh)를 설정하기 위해 사용한다는 것이다. TCP-Reno에 관한 TCP-Westwood 의 중요한 특색이 있는 특징은 TCP reno는 세 개의 중복된 ACK을 받거나 slow start threshold와 time congestion에 사용되는 유효 대역폭과 일치하는 congestion window의 선택을 시도하고 난 후에 time out이 될 때congestion window를 반으로 나눈다는 것이다. source는 returning ACKs의 비율을 측정하고, 평균을 내는 것에 의해 TCP connection 을 따라 이용할 수 있는 대역폭의 end to end의 예상치를 수행하게 된다. sender가 패킷 손실을 알아차리든지(즉 타임 아웃은 발생하거나 또는 3 duplicate ACK을 받았을 때) sender는 대역폭 예상치를 혼잡 윈도우 크기(cwin)와 슬로우 스타트 임계치(ssthresh)를 적절히 설정하기 위해 사용하게 되고, 이 방법은 TCP-Westwood가 cwin과 ssthresh의 과도한 축소를 피하게 한다. 따라서 이는 더욱 빠른 faster recovery를 보장한다[3].

2.3 Congestion Control algorithm

TCP-Westwood의 동작 방식을 간단히 설명하면, 두 가지로 나눠서 볼 수 있다. 하나는 중복된 ACK를 받는 경우이고 나머지 하나는 timeout이 발생하는 경우이다.

1) 정상적으로 ACK를 받는 경우

Congestion window의 크기를 TCP Reno방식에 따라 증가시킨다. End-to-end에 걸리는 가용 대역폭을 측정한다.

2) 3 duplicated ACK를 받는 경우

슬로우 스타트 임계값과 혼잡 윈도우 크기를 다음과 같은 방식으로 수정한다.

$$ssthresh = \max(2, (BWE \times RTT_{\min}) / seg\_size);$$
$$cwnd = ssthresh;$$

3) timeout 이 만료되는 경우

슬로우 스타트 임계값과 혼잡 윈도우 크기를 다음과 같이 수정한다.

$$ssthresh = \max(2, (BWE \times RTT_{\min}) / seg\_size);$$
$$cwnd = 1;$$

3. TCP Enhancement

위에 식들을 사용하여 혼잡윈도우의 크기는 다음 식(1)과 같이 나타낼 수 있다.

$$ssthresh = (BWE \times RTT_{\min}) / seg\_size \quad (1)$$

위 식(1)에서 구하데 되는 혼잡 윈도우 크기는 측정한 가용 대역폭을 최소 시간으로 곱하고 이를 다시 세그먼트 크기로 나누는 것을 보여주는 것이다. 여기서 RTT_{min} 의 최소값을 곱하는 이유는 병목현상이 없을 때의 RTT 값으로 혼잡 윈도우 크기를 설정하기 위해서이다. 하지만 이 변수에 설정된 값은 병목현상이 점차 감소하여 최종에는 모두 소진되는 환경을 고려하여 설정되었다. 여기에서 병목현상이 모두 소진되는 시점을 파악하기 보다는 병목현상이 발생하여 대기중인 패킷을 저장하는 Buffer에 저장된 패킷을 점차적으로 전송하여 모두 정상적으로 전송되어 Buffer가 모두 비워졌을 때 다른 패킷이 도착하여 buffer에 저장되기까지의 시간을 고려하여 ssthresh를 가정을 할 수 있게 된다.

이런 가정하에, RTT_{min}값을 약간 큰 값을 취하게 하면 ssthresh값이 커지면서 전송률이 증가할 수 있다.

3.1 New RTT parameter for slow start threshold

위의 식(1)에서 사용하는 RTT_{min}은 너무 전형적인 값이기 때문에 보다 적절한 값으로 수정을 하면 TCP-Westwood의 성능을 향상시킬 수 있다. 이에 따라 위의 식(1)의 RTT_{min}을 $RTT_{estimate}$으로 변경하여 다음과 같이 ssthresh를 수정할 수 있다.

$$ssthresh = (BWE \times RTT_{estimate}) / seg\_size \quad (2)$$

위의 식(2)에서는 $RTT_{estimate}$값을 이용하여서 ssthresh를 TCP-Westwood보다 큰 값을 가지게 함으로써 보다 빨리 혼잡 윈도우 크기를 증가 시킬 수 있다. 이렇게 함으로써 전송률은 증가하게 될 것이다.

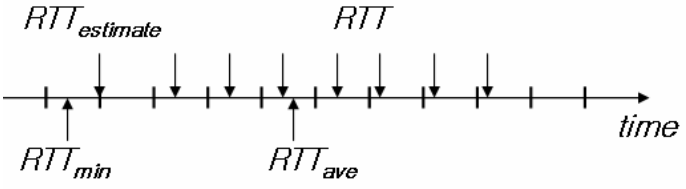

그림 1. RTT

위의 그림 1은 간단하게 RTT들을 값에 따라 보여주는 것이다. RTT_{min}과 RTT_{ave}의 간격은 네트워크의 상황에 따라 달라질 수 있다. RTT_{min}값의 경우는 최단 시간으로 보기 때문에 전송할 때 거쳐가는 router에서의 buffer에서의 대기시간이 0일 때로 본다. 반면 RTT값들은 대기시간이 조금씩 생길 수 있기 때문에

RTT_{ave}의 값은 그 때의 네트워크 상황을 반영하고 있다. 그러므로 $RTT_{estimate}$값은 buffer가 다 비워지는 데 걸리는 시간을 기다리지 말고 전송하는 데 걸리는 시간이 있기 때문에 RTT_{min}에 전송하는 데 걸리는 시간을 더해서 RTT_{min}값을 증가시킬 수가 있다.

먼저, congestion avoidance 구간에서의 RTT의 평균값을 구한다. 평균값을 구하는 이유는 RTT값들이 congestion을 대표하는 지수로 가정하고 이를 이용해서 $RTT_{estimate}$을 유추할 수 있다.

$$RTT_{ave} = \frac{\sum RTT}{number\ of\ RTTs} \quad (3)$$

$RTT_{estimate}$값은 RTT_{min}보다는 크고 RTT_{ave}값보다는 작아야한다. 하지만 가급적 RTT_{min}값에 근접한 설정치로 설정되어야 한다. 만약 RTT_{ave}에 근접한 설정값으로 ssthresh값을 설정하게 될 경우 congestion의 발생이 너무 빨리 진행되어 특정한 시간을 기준으로 할 경우 보았을 때 기존의 가정보다 더 많은 congestion이 발생할 수 있다. 다음 식(4)은 가지고 $RTT_{estimate}$값을 구하기 위한 것이다.

$$RTT_{estimate} = RTT_{\min} + \frac{RTT_{ave} - RTT_{\min}}{RTT_{\min}} \quad (4)$$

위의 식(4)은 RTT_{ave}값을 RTT_{min}값과의 비율로 증가된 $RTT_{estimate}$값을 설정하게 된다. 이러한 가정들을 통해 기존의 식(1)보다 향상된 ssthresh값을 유추 할 수 있다.

4. 실험 및 성능 평가

앞에서 제시한 방법으로 구한 슬로우 스타트 임계값을 가지는 TCP-Westwood를 NS-2 네트워크 시뮬레이터를 통해서 구현하고 이에 대한 성능을 평가하고자 한다.

4.1 실험 환경

다음 그림 2은 실험을 하기 위한 네트워크 모델을 나타내고 있다.

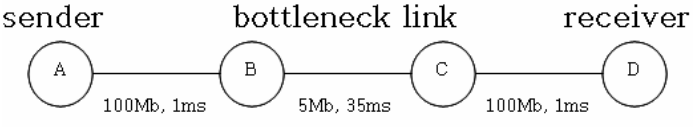

그림 2. 실험 환경 네트워크 구성도

그림 2에서는 병목구간에 5Mbps의 대역폭을 설정하고 양쪽 끝에 위치한 sender에서 receiver로 FTP를 이용하여 1400 바이트의 크기를 가지는 패킷을 연속적으로 40초 동안 보내는 실험으로 함으로써 네트워크에 병목현상을 유발해서 TCP의 성능을 평가하고자 한다. 이 때 병목구간에서 유선과 무선을 특성을 실험하기 위해 병목구간의 링크에서 링크 에러률을 입력 받아 이를 실험에 이용하도록 하였다. 이 경우 유선 망에서의 전송에는 전송에러가 발생하지 않고 가정하고, 무선에서는 손실율 0.1%를 가정하여 실험을 하였다.

4.2 실험 결과

TCL 스크립트를 수행하면서 혼잡 윈도우(cwnd)의 변화를 다음과 같이 그래프로 도시화 하였다.

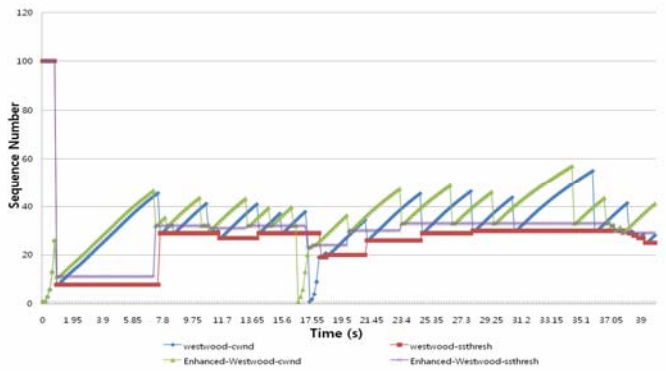

그림 3. 무선환경에서의 혼잡 윈도우 크기의 변화

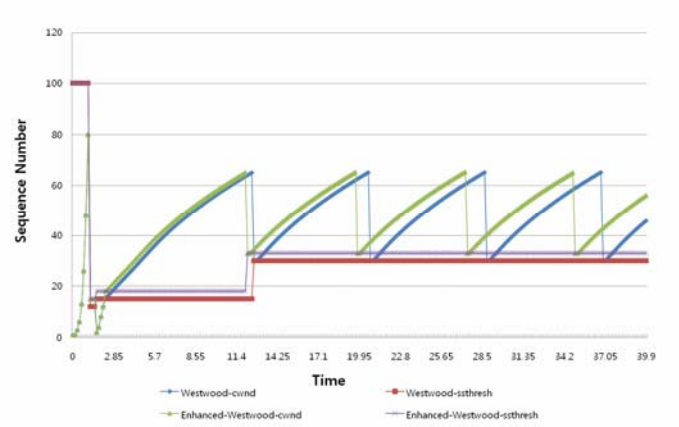

그림 4. 유선환경에서의 혼잡 윈도우 크기 변화

그림 3와 그림4의 결과에서 보여주는 바와 같이 $RTT_{estimate}$를 사용하여 슬로우 스타트 임계값을 설정한 TCP에서는 혼잡이 발생하여도 RTT_{min}값을 사용한 TCP보다 혼잡 윈도우 크기가 더 낮게 증가한다는 것을 알 수 있다.

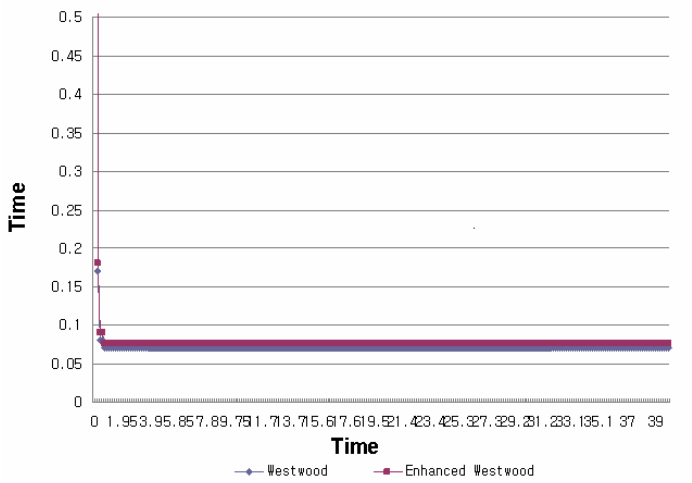

그림 5. RTTmin과 RTTestimate 의 비교

위의 그림 5는 기존의 TCP-Westwood와 제안한 TCP-Westwood 에서 슬로우 스타트 임계값을 결정하는 데 있어서 사용되는 RTT값을 비교한 것이다.

그림 5의 결과와 같이, 제안모델의 TCP-Westwood에서 $RTT_{estimate}$값이 더 크게 나오는 것을 알 수 있다. 이 값에 의해서 혼잡 윈도우 크기가 향상됨으로 아래 표와 같은 전송률에 대한 결과를 얻을 수 있다.

표 1. 전송률 비교

	Original TCP-Westwood	Enhanced TCP-Westwood
유선	4.48 Mbps	4.53 Mbps
무선	4.04 Mbps	4.25 Mbps

5. 결 론

본 논문에서는 TCP-Westwood에 대해서 전송률을 개선하는 방법에 대한 연구를 통해서 기존의 TCP-Westwood보다 성능적 개선을 해 보았다. NS-2 시뮬레이터에서의 실험을 통해서 얻은 결과를 보면, 기존의 TCP-Westwood보다 무선과 유선에서 각각 5%, 1.2%의 전송률이 증가함을 보았다. 기존의 TCP-Westwood 방식보다 그렇게 크게 성능을 증가시키지 못한 이유는 임계값 수정만으로는 궁극적으로 TCP에서 무선에서 발생하는 손실을 구별할 수 없기 때문에 TCP-Westwood와 같이 무선에서 발생하는 손실과 네트워크 혼잡 손실에 대해서 구별하는 방법이 없는 방식에서는 최상의 성능 향상을 하였다고 볼 수 있다. 다시 말해서 무선에서의 손실과 네트워크 혼잡 손실을 구별할 수 있는 방법이 TCP안에 들어 있어야만 더 좋은 성능을 가져올 수 있다. 더 나은 TCP 성능 향상을 무선 손실과 네트워크 혼잡 손실을 구별 할 수 있는 방법을 추가적으로 연구해야 할 것이다.

참고문헌

[1] 최진희, 진현욱, 유혁, “무선네트워크에서 TCP의 성능을 향상시키기 위한 기지국 버퍼 관리 기법”, 한국정보과학회 학술발표논문집, 제 28권, 제2호(III), p. 166 ~ 168, 2001.10

[2] Saverio Mascolo, Claudio Casetti, Mario Gerla, M.Y. Sanadidi and Ren Wang, “TCP Westwood: Bandwidth estimation for enhanced transport over wireless links”, Proceedings of the 7th annual international conference on Mobile computing and networking, p.287 ~ 297, July, 2001.

[3] Luigi A. Grieco and Saverio Mascolo, “Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control”, ACM SIGCOMM Computer Communication Review, Volume 34, Issue 2, p. 25~38, 2004