논문번호 06-01-01

무선망에서의 TCP 성능 향상 방안에 관한 연구

A Study on TCP Performance Enhancements in Wireless Networks

朴 濤 用*, 金 榮 範*
Do-Yong Park*, Young-Beom Kim*

Abstract

The TCP protocol can provide some reliability using sliding window mechanism for data transmission, flow control, and congestion control. However, TCP has some limitations in that it has basically been designed solely for wired communication environments. If traditional TCP protocol is used also in wireless networks, the end-to-end data transmission performance degrades dramatically due to frequent packet losses caused by transmission errors and hand-offs. While there have been some research efforts on TCP enhancements considering the mobility of wireless communication devices, in this paper we propose a new method to improve the TCP performance by combining the Snoop and the Freeze-TCP methods. In the proposed scheme, the TCP end-to-end semantics is maintained and no changes of existing protocols in sending systems or in routers are required. It has the advantage of simple implementation because TCP code changes are limited to mobile devices for applying the Freeze-TCP and it requires only to add Snoop modules in base stations. Accordingly, the proposed scheme can operate well in the existing networks. Finally, in this study, we compared the performance of the proposed scheme with traditional TCP, other approaches through simulations using ns-2.

요 약

TCP는 데이터 전송, 흐름 제어 및 혼잡 제어를 위해 sliding window mechanism을 사용하여 신뢰성을 보장하지만 기본적으로 유선 통신망 환경만을 고려하여 설계된 프로토콜이라는 한계점을 갖고 있다. 무선 링크 상에서는 주로 전송 오류와 핸드오프에 의해 빈번한 패킷 유실이 발생함으로써 기존의 TCP 프로토콜을 무선망에 그대로 적용하는 경우 종단간 전송 성능은 급격히 떨어지게 된다. 무선통신기기 이동성을 고려한 TCP 최적화 방안에 관하여 여러 연구가 이루어져 왔으나 본 논문에서는 Snoop와 Freeze-TCP를 혼합적으로 적용하여 TCP 성능을 향상시키는 방안을 제안한다. 제안된 방안의 경우 end-to-end semantics가 그대로 유지되며 송신측이나 중간 라우터의 기존 프로토콜의 수정을 요구하지 않는다. 이 방안을 적용하는 경우 TCP 코드의 변경은 Freeze-TCP의 적용을 위해 이동통신기기에 국한되며 기지국에 Snoop 모듈을 추가하는 것만으로 충분하여 구현이 용이하다는 장점을 가지고 있다. 따라서 제안된 방안은 현재 구축되어 있는 망에서 충분히 상호 동작할 수 있다. 본 논문에서는 NS-2를 이용한 시뮬레이션을 통하여 일반 TCP, 기존 방식, 그리고 제안된 방안간의 성능을 비교 분석하였다.

Ⅰ. 서론

현재 mobile computing과 무선 네트워크는 일상적인 음성 및 데이터 통신에 있어서 매우 중요한 위치를 차지하고 있다. 그러나 무선 링크를 통한 통신은 제한된 대역폭, 높은 지연시간, 산발적으로 일어나는 높은 비트 에러율, 일시적인 연결 중단 등의 특성들이 있으며 네트워크 프로토콜들은 이런 특성들을 다룰 수 있도록 설계되어야 한다. 또한 프로토콜들과 애플리케이

* 건국대 전자공학과
(Dept. of Electronics Engineering, Konkuk University)
接受日:2006年 2月 10日, 修正完了日: 2006年 7月 14日

션들은 사용자의 이동성과 cellular wireless networks 안에서 한 cell에서 다른 cell로의 사용자의 이동에 따른 핸드오프를 다룰 수 있어야 한다. 특히 핸드오프가 일어날 때에는 패킷 유실과 패킷 전송지연의 변화가 일어난다[1].

TCP와 같은 신뢰성 있는 전송 프로토콜은 고정된 호스트들과 전통적인 네크워크인 유선 링크 상에서 만들어지고 그에 맞게 조정되어 왔다. TCP는 종단간의 지연과 혼잡(congestion)에 의한 패킷유실에 적용되어 위와 같은 상황에서 아주 잘 동작하였다. 유선망 상에서의 낮은 비트 에러율로 인해서 유선망에서 일어나는 모든 패킷 유실은 혼잡에 의한 것으로 간주하도록 하였다. 무선 환경에서의 높은 비트 에러율, 간헐적으로 일어나는 연결의 끊김과, 핸드오프는 TCP에서 재전송하기 전에 전송 원도우 크기를 떨어뜨리고, 혼잡 제어(congestion control) 또는 혼잡 회피 메카니즘(congestion avoidance mechanism)을 초기화 시키고, 재전송 타이머를 reset시킨다. 무선망상에서의 이런 조치들은 불필요한 링크 대역폭 이용률의 감소를 가져오게 된다.

이런 무선망상에서의 TCP성능 향상을 위해 많은 연구가 진행되어 왔다. 그러한 예로 Fast retransmission, I-TCP(Indirect TCP), M-TCP, Snoop, Freeze-TCP등의 알고리즘들이 나왔다. 이들 알고리즘 모두 무선망상에서 TCP의 성능을 향상시키기는 하였으나 또한 단점을 가지고 있다. Fast retransmit은 높은 비트 에러율, 긴 연결중단(Long Disconnections), 잦은 연결중단(Frequent Disconnections)에 대한 대응이 안되며, I-TCP와 M-TCP는 end-to-end TCP semantics를 만족 시키지 못한다. 또한 Snoop은 Fast retransmit에 비해 높은 비트 에러율에는 대처가 가능하나 역시 긴 연결중단과 잦은 연결중단에는 대처할 방법이 없다. Freeze-TCP는 사전에 감지 가능한 연결중단에 대한 대처만 가능하다.

본 논문에서는 이러한 단점들을 극복하는 방안으로 Snoop알고리즘과 Freeze-TCP를 병행 하는 방법을 제시한다. 위의 두 알고리즘을 end-to-end TCP semantics를 유지하며 높은 비트 에러율에 의한 패킷 유실로 유발되었을 때 혼잡 회피 메카니즘 수행을 회피 할 수 있고 핸드오프에 의한 일시적인 연결중단에 대응할 수 있음을 보인다.

본 논문의 구성은 다음과 같다. 다음 장에서는 Mobile 환경에서의 TCP 혼잡회피 메카니즘의 문제점을 알아보고, 3장에서는 mobile TCP에 관한 기존 연구들에 대한 간략한 소개를 하며 여러 방식들간의 장단점 비교와 함께 본 논문에서 제안한 방식에 대한 기술을 한다. 4장에서는 본 논문에서 제안한 방식의 성능을 검증하기 위한 시뮬레이션 환경 설정 및 구현을 보이고 5장에서는 시뮬레이션 결과 및 검토를, 마지막으로 6장에서 결론을 맺는다.

II. Mobile 환경에서의 TCP congestion control mechanism의 문제점

무선망은 기존의 유선망에 비해 느린 속도와 잦은 에러를 발생시킨다. 이러한 전송매체의 차이는 기존의 유선망에서 사용되던 TCP를 무선망에 적용함 에 있어서 특성에 맞지 않는 부분들에 의해 성능저하가 일어나는 것은 어떻게 보면 필연적일 수밖에 없다. 유선망에서는 전송매체의 안정화로 패킷의 유실이 적다. 그러나 유선망에서도 전송 중 에러가 발생해서 패킷이 버려지는 경우가 존재한다. 망이라고 하는 것은 전송대역폭을 가지고 있고 그 대역폭 이상의 패킷 전달 요구가 혼잡하게 되면 그 패킷 전송을 감당하지 못하는 네트워크에 연결된 라우터는 전달되지 못하는 패킷을 버리게 되고 패킷을 전달하고자 했던 송신측 호스트는 Time-out이 걸리거나 그 패킷을 받아야 하는 Receive 호스트로부터 재전송 요구를 받게 된다. 이 때 송신측 호스트에서는 혼잡제어(Congestion Control)를 발생한다. 혼잡제어가 발생하면 혼잡윈도우(cwnd)가 절반씩 줄어들어 늦어진 속도만큼 전송 속도를 떨어뜨린다.

무선망에서의 TCP가 문제가 되는 점이 바로 이 패킷 유실 때의 혼잡제어다. 무선망의 특성상 패킷 유실이 잦음 에도 불구하고 TCP는 무선망에서의 패킷 유실과 혼잡에 의한 패킷 유실을 구분 할 방법이 없다. 따라서 전송 대역폭 자체가 줄어든 것이 아니라 잠시 무선망의 특성에 의해 발생한 패킷 유실을 혼잡에 의한 패킷 유실로 보고 혼잡제어를 하게 됨으로써 매번 패킷 유실이 일어날 때마다 송신측은 보내는 속도를 낮추게 된다. 이러한 TCP의 특성은 에러가 잦은 무선망에서 TCP 성능을 크게 저하시키게 된다.

무선망에서 나타날 수 있는 또 다른 문제점은 잦은 연결의 끊어짐이다. 연결이 끊어지는 이유로는 signal fading, 링크에러, 또는 이동호스트의 움직임으로 인한 핸드오프등을 들 수 있다. TCP를 이용하여 데이터를 전송 중에 핸드오프가 일어난다면 그 과정동안 전송된 패킷들은 모두 유실되게 된다. 이렇게 잃어버린 패킷들 역시 기존의 TCP에서는 혼잡으로 간주하고 TCP 윈도우 크기를 줄이고 혼잡회피 단계에 들어가게 된다. 핸드오프에 의한 일시적인 연결 중단 상황도 대역폭을 줄일 필요는 없으므로 혼잡회피 알고리즘은 대역폭의 이용률을 떨어뜨리게 되는 것이다.

이러한 문제점을 해결하기위해 여러 가지 이동 TCP 프로토콜이 제안되었다. 이동 TCP는 혼잡과 유실을 구분하여 서로 다른 방식으로 처리 할 수 있도록 하는 것이 일반적인 방식이다. 그러한 방식들에는 MTCP[2], I-TCP[3], Fast-Retransmit[4], Snoop[5], M-TCP[6], Freeze-TCP[7] 등이 있다.
다음 장에서는 위에 열거한 각각의 방법들에 대해 동작 원리를 설명하고 장단점을 비교해 보기로 한다.

Ⅲ. 관련 연구 동향

3.1 I-TCP

Split Connection Approach 방식은 무선망과 유선망 연결 부분을 분리하여 한 TCP연결을 두개의 TCP 연결로 바꾼다. 이 방식을 이용한 프로토콜에는 Indirect TCP (I-TCP)[3]가 있다. I-TCP는 BS(또는 MSR: Mobile Support Router)에 I-TCP Agent를 두고 송신측에서 연결을 요청하면 Agent가 연결을 맺고 Agent는 다시 MH(mobile host) 와 연결을 맺어 데이터 전송을 하면 Agent는 중간에서 다시 다른 연결로 포워딩 해주는 방식을 사용했다. 이러한 방법은 서로 다른 연결을 맺기 때문에 무선망 연결은 기존의 TCP 대신 무선망에 적합하도록 변경된 TCP를 사용하여 연결할 수 있기 때문에 성능향상에 많은 영향을 끼칠 수 있다.

이 방법은, 유선망(Ethernet, long-haul links, ATM 등)은 신뢰성이 높고 전송 대역폭이 큰 반면, 무선망은 작은 대역폭을 가지며 높은 비트 에러율을 유발시키는 잡음과 페이딩에 의한 신호의 유실을 입기 쉬운 특성에 따라, 유선과 무선에서의 흐름제어와 혼잡제어를 기능적으로 분리하고 있다. 무선 링크를 위한 분리된 전송 프로토콜은 전속 가능 대역폭, 이동, 연결중단과 같은 이벤트의 지원을 가능하게 하여 상위 계층에 알려줄 수 있게 한다.

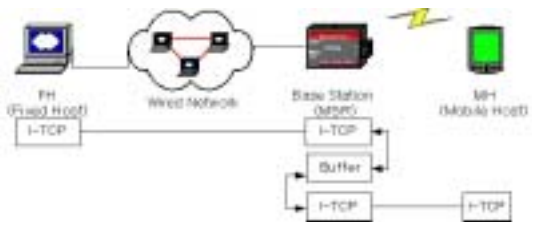

그림 1. I-TCP의 구조.
Fig. 1. The structure of I-TCP.

그러나 이 방법은 다음과 같은 단점이 지적되고 있다. 우선 첫 번째로, 기본적인 TCP semantics을 무시하게 된다. 일반적인 TCP는 end-to-end 연결을 이루게 되지만 I-TCP는 중간에 I-TCP가 연결을 맺기 때문에 end-to-end semantics을 만족하지 못한다. 두 번째로 응용프로그램의 새로운 링크가 필요하다. 기존의 TCP를 대체하는 I-TCP를 사용하기 때문에 TCP를 사용하는 모든 애플리케이션을 TCP Socket System Call대신 I-TCP Socket System Call을 이용하여 재링크 시켜야 한다. 마지막으로 TCP상에서의 부담을 줄이기 위해 I-TCP를 사용했기 때문에 4번의 패킷 복사가 일어나게 된다. 송신측에서 한번, MH쪽에서 외에 BS에서 2번의 복사가 일어난다.

3.2 Fast-Retransmit

Fast-Retransmit 방식[4]은 1~2개 정도의 데이터 세그먼트만이 유실되는 짧은 시간의 연결 단절로 인한 전송상의 문제를 해결하기 위한 목적으로 제안되었다. 이 방법에서는 TCP연결을 분할하지 않으며 다음과 같은 경우를 고려하고 있다. 즉, 핸드오프동안 MH는 패킷 수신이 안되므로 송신측은 기존의 TCP를 그대로 사용하는 경우 전송 타이머의 타임아웃 이후에 혼잡상태가 발생한 것으로 판단하고 혼잡제어 모드로 들어가게 되고 윈도우 크기를 줄이고 재전송을 하게 된다. 타이머의 타임아웃 시간이 긴 경우 MH는 핸드오프를 완료한 후에도 송신측 전송 타이머가 타임아웃될 때까지 기다리고 나서야 비로소 송신측으로부터 패킷을 다시 받을 수 있게 된다. 따라서 이 방식에서는 MH에서 핸드오프를 마친 즉시 최종 송신한 ACK를 세 번 재전송하도록 함으로써 송신측에서 혼잡 윈도우 크기를 반감시키고 1개 세그먼트를 즉각적으로 재전송하도록 하고 있다. 하지만 이 방법은 중복된 ACK와 유실된 모든 패킷에 대한 송신측 쪽에서 반복되는 재전송 때문에 혼잡한 상황을 가중시킬 수 있다는 점이 문제이다.

3.3 Snoop

이번 절은 유선망에서는 현재 존재하는 TCP구현을 변경하지 않고, BS와 MH에 대한 제어를 변경하여 무선링크에서의 TCP성능을 향상시키려는 Snoop module[5]을 소개한다. 먼저 송신측에서 MH로 전송이 일어날 때, 송신측은 무선망의 존재를 모르는 채, 무선망상의 패킷 손실로 인한 혼잡제어를 막기 위해서 BS의 Network-Layer 소프트웨어를 변형하여 TCP 윈도우의 최대 크기 정도의 버퍼를 갖는 Snoop이라는 Module을 추가하여, 송신측에서 오는 패킷을 모니터하고, MH로부터 응답(ACK)을 받지 못한 패킷을 버퍼링하여 중복된 응답(Duplicated ACK: DUPACK)이나

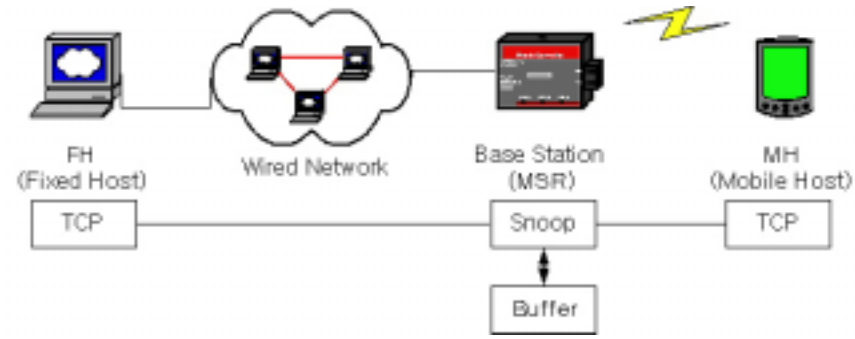

그림 2. Snoop의 구조.
Fig. 2. The structure of Snoop.

timeout이 발생하였을 때 BS에서 잃어버린 패킷을 재전송하는 지역 재전송(local retransmission)을 수행할 수 있게 한다. 따라서 TCP상의 재전송 이 아니라 BS와 MH간의 부분적 지역 재전송만이 일어난다.

이 때의 알고리즘은 설명해보면 다음과 같다 (그림 2). 먼저 송신측로부터 BS가 Packet을 받을 때, 정상적인 순서의 TCP 패킷이 도달하면 패킷을 Snoop 캐시에 저장하고 MH에 전송한다. 이전에 저장된 순서에 어긋난 패킷이 도달하면 이는 송신측 timeout이나 TCP의 빠른 재 전송(Fast Retransmission)에 의한 것이므로 저장된 패킷을 재 전송하게 한다. 아직 저장이 되지 않은 상태의 순서가 어긋난 패킷이 도달하면 이는 패킷이 비정상 적으로 도달한 혼잡이 일어난 경우이므로 이 패킷들은 혼잡을 경험했다는 기록을 해둔다. 이 정보는 BS에서 MH로부터 ACK를 받을 때 이용된다. BS가 MH로부터 ACK를 받을 때 비정상적인 ACK를 받았을 때 이 ACK은 무시한다. 이전에 받은 ACK와 같은 Duplicated ACK를 받으면 MH가 다음 패킷을 받지 못했으므로 BS은 다음의 Action을 취한다. 만약 원하는 패킷이 Snoop 버퍼에 없거나 혼잡 Loss를 겪었다면 DUPACK를 송신측으로 보내 혼잡제어 메커니즘을 일으킨 후 패킷을 재 전송하게 한다. 그렇지 않다면, 즉 원하는 패킷이 Snoop 버퍼 안에 존재할 때, 기대하지 않았던 DUPACK을 받으면 패킷을 재 전송한다. 만약 패킷이 Snoop 버퍼 안에 존재하고, 예상했던 DUPACK를 받으면, DUPACK를 무시한다. 새로운 ACK의 경우 MH가 받은 Sequence의 증가를 가리키므로 현재 상태정보(State Information)을 이용해 무선망에 의한 패킷 유실인지 검사하고 무선망에 의한 패킷 유실이면 재전송하고 Retransmission Timer를 Set하고 상태를 수정해야 한다. 그렇지 않다면 ACK를 송신측으로 보내고 캐시를 비운다.

Snoop Module과 Routing Protocol간의 상호 작용에 대해 알아보면 핸드오프가 일어날 때 새로운 BS는 Snoop Module이 버퍼링하고 있는 패킷을 저장하고 있어야 성능을 기대할 수 있다. 따라서 Primary BS로 전달되는 패킷을 Multicast Group에 참여하는 모든 BS들도 유지하고 있어야 한다. 하지만 MS로부터 오는 패킷과 ACK은 단지 Primary BS만을 통해 전달되므로 다른 Multicast Group의 BS는 FIFO Cache를 유지하다가 버퍼가 부족하게 되면 버퍼를 비워야 한다. 핸드오프가 발생해서 새로운 BS가 ACK를 받으면 다음에 전달 될 패킷을 바로 보낼 수 있게 된다. 이러한 방식의 장점은 새로운 BS와 이 전의 BS간의 상태정보의 전달이 없어 핸드오프에 걸리는 시간을 최소화 할 수 있다는 것이다. 하지만 이 경우 BS에서 필요 이상의 많은 버퍼를 유지해야 하고, 각 BS사이에 항상 동일한 상태를 유지해야 할 필요가 생긴다. 또 Non-Overlapped 핸드오프에 대한 해결책이 없다.

3.4 M-TCP

M-TCP[6]는 SH(Supervisor Host)에서 TCP 연결을 분리하여 흐름 제어를 하는 것으로 무선망의 상황이 낮은 비트 에러율을 가지며 핸드오프와 같은 잦은 연결 중단에 초점을 맞추고 있다. 그림 3에서와 같이 M-TCP의 구조를 살펴보면 3-레벨 계층구조 로서 MSS기능을 줄이고 MH가 동일한 SH도메인 내 새로운 MSS로 이동시 상태정보 교환이 수행되지 않아도 되도록 하였다. 종단간의 ACK를 유지하기 위해 SH는 MH로부터 ACK 수신 후 송신단으로 ACK를 보낸다. 핸드오프후에 TCP 윈도우의 크기를 그대로 유지하게 위해 TCP persist모드를 이용한다. SH는 MH로부터의 마지막 ACK를 보류하고 있다가 핸드오프가 발생할 때에 SH는 수신 윈도우의 크기를 0으로 설정하여 보류된 ACK를 전송하고 이것을 수신한 송신단이 persist모드로 들어가게 하는 것이다. 핸드오프완료후

에는 전송된 ACK에 의해 persist모드를 해제한다.

이 방법의 단점은 기본적으로 낮은 비트 에러율을 가정하여 설계되었기 때문에 FEC와 신뢰성있는 link-layer 프로토콜로써 낮은 비트 에러율을 가질 수 있도록 하는 설계가 필요하며, MH는 핸드오프나 물리적 간섭 때문에 긴 연결중단에 접할 수 있는데 이때에는 데이터의 유실로 Poor TCP throughput을 야기 시킨다.

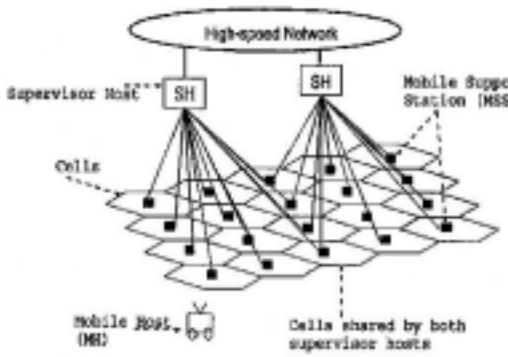

그림 3. M-TCP의 구조.
Fig. 3. The structure of M-TCP.

3.5 Freeze-TCP

Freeze-TCP[7]의 핵심 아이디어는 임박한 연결중단에 대한 부담을 MH에 지우는 것이다. MH는 무선 안테나로부터 신호의 세기를 모니터하여 임박한 핸드오프를 감지할 수 있다. 또한 어떤 경우에는, 어떤 임의적인 연결중단을 예측할 수도 있다(예로 신호의 세기에 페이딩이 생길 경우). 그러한 경우에 MH는 zero window size를 송신측에 보내어 ZWP모드로 들어가게 하여 송신측의 혼잡 윈도우를 떨어뜨리게 되는 것을 방지할 수 있다. 이 방법을 구현하기 위해서는 송신측이나 중간노드들(예로 BS)의 코드변경은 필요 하지 않으며 단지 MH의 코드만 변경하면 된다.

수신측에서 임박한 연결중단을 감지한다면, 수신측에서는 몇 개의(적어도 한개 이상의) ACK를 보내는 것을 시도한다. 이 때 보내지는 ACK에는 윈도우 크기가 0으로 설정된다(0의 윈도우 수신 크기를 보내는 ACK를 ZWA- Zero Window Advertisement라 한다). 여기서 연결이 중단되기 얼마 전에 ZWA신호를 보내야 하는가라는 문제를 발생한다. 이 기간은 연결중단 전의 warning period이라 하는 기간동안 하는 것으로 제안한다. 이상적으로 warning period는 최소한 하나의 ZWA 패킷이 송신 측에 확실히 도착할 수 있을 만큼 충분히 길어야 한다. 만일 warning period가 너무 길 경우 송신측은 연결중단에 앞서 너무 빨리 idle모드로 들어가게 되고, warning period가 너무 짧다면 수신측에서 ZWA를 송신측으로 보낼 시간이 짧아서 연결 중단 동안 송신측은 혼잡 윈도우 크기를 줄이게 될 것이다. 실험에 의해 검증된 합리적인 warning period는 RTT(Round Trip Time)로 알려져 있다. RTT보다 길거나 짧은 warning period는 대부분의 경우에 평균 성능보다 낮은 성능을 보인다.

ZWP는 지수 함수적으로 back-off 되기 때문에 핸드오프가 완료된 후 다시 데이터를 전송할 수 있음에도 불구하고 상당한 기간의 idle 시간을 유발할 가능성이 있다. 예를 들면, 연결중단 시간이 길었던 상황에서 재 연결이 되고 재 연결 직전에 송신측으로부터 보내진 ZWP를 잃었다면, 송신측은 다음 probe 패킷을 보내기 전에 더욱 긴 시간의 back-off를 하게 될 것이다. 이러한 긴 idle기간을 회피하기 위해, 수신측에서는 재 연결이 되자마자 연결이 끊어지기 전에 받았던 마지막 패킷에 대한 세 개의 중복된 ACK를 보내게 해서 송신측에서 즉각 데이터를 보낼 수 있게 한다. 아래 그림4는 일반적인 TCP와 Freeze-TCP에서 연결이 중단되었다가 다시 연결되었을 때의 성능비교를 도식적으로 나타낸 것이다.

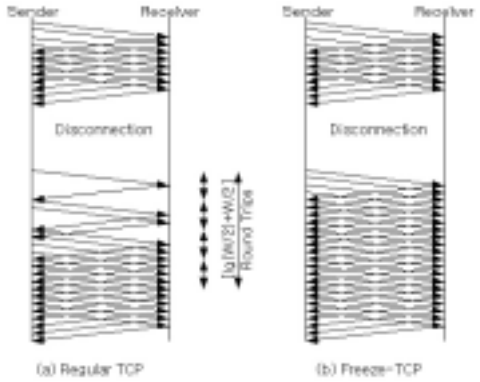

그림 4. 연결중단후의 Regular TCP와 Freeze-TCP의 성능비교.
Fig. 4. The performance comparison of Regular TCP and Freeze-TCP after disconnection.

그림 4(a)에서는 재 연결후 slow-start단계로 들어가지만 그림 4(b)에서는 재 연결 후에도 혼잡 윈도우를 연결중단 이전과 같이 유지하여 대역폭을 유지하고 있음을 보여준다.

3.6 기존 방안들의 성능 비교

위와 같이 TCP의 성능을 향상시키는 방법들에 중

표 1. I-TCP, M-TCP, Snoop, Freeze-TCP의 비교.
Table 1. A comparison of I-TCP, M-TCP, Snoop, Freeze-TCP.

	I-TCP	M-TCP	Snoop	Freeze-TCP
중간노드에 TCP모드가 필요한가?	O	O	△	X
암호화된 트래픽을 다룰 수 있는가?	X	X	X	O
End-to-end semantics가 유지되는가?	X	X	O	O
긴 연결중단을 다룰 수 있는가?	△	O	X	O
잦은 연결중단을 다룰 수 있는가?	△	△	X	O
높은 BER을 다룰 수 있는가?	O	O	O	X

요하게 고려되어야 할 것 중의 하나가 현재의 기간시설들간의 상호 운용성(inter-operability)이다. 이상적으로는 이 목표를 이루기 위해서는 중간의 라우터나 송신측은, 다른 단체나 기관에 속해 있기 때문에 어떤 수정도 허용되지 않는다.

두 번째로 고려되어야 할 중요한 점은 암호화된 트래픽의 경우이다. 네트워크 보안은 갈수록 중요한 부분이 되어가고 있으며 암호화는 이에 따라 광범위하게 적용되어가고 있다. 예로써, IPSEC는 IP의 차세대 프로토콜인 IPv6의 한 부분으로 완전히 자리잡고 있다. 이런 경우, 모든 IP payload는 암호화되며 중간 노드들(BS이나 router)은 payload에 TCP가 실려 있다는 것을 알수 없다. BS에 의존하여 중계하는 방법들(Snoop, I-TCP, M-TCP)은 트래픽이 암호화된 경우에는 동작하지 못할 것이다. 또한 데이터와 ACK는 다른 경로를 택할 수도 있다(예로 위성을 통한 경로). 이러한 경우 중계에 의한 방법들은 심각한 문제를 유발할 수도 있다.

True end-to-end semantics를 유지하고 있는지도 고려되어야 할 사항중의 하나이다. I-TCP는 true end-to-end semantics를 유지하지 못한다. M-TCP는 true end-to-end semantics를 유지하기는 하지만 BS에 부가적으로 TCP기능을 첨가, 수정해야 한다.

BS에 기능을 추가함으로써 성능을 향상 시킬 수 있고 위에 열거한 것들이 그리 중요하지 않다고 하더라고 또 하나 고려되어져야 할 사항은 Snoop, I-TCP, M-TCP등이 BS에 어느 정도의 데이터를 담아둘 버퍼를(지역 재 전송을 위해) 필요로 하며 각각의 연결에 데이터를 처리하여 보내기 위한 추가적인 프로세싱이 필요하다. 만일 수백, 수천의 이동호스트가 하나의 BS에 있다면, 각각의 연결에 연관된 트래픽의 프로세싱에 의해 과부하가 걸릴 것이다. 이동 호스트가 하나의 BS지역에서 다른 BS로 옮겨갈 때 연결에 관한 모든 상태가(재 전송을 위해 버퍼에 있던 모든 데이터를 포함해서) 새 BS로 옮겨져야 한다. 이것은 상당한 양의 오버헤드를 유발시킬 수 있으며, 몇몇 패킷을 잃어버릴 수도 있다. 그렇게 되면 송신측은 혼잡 회피단계로 들어가 혼잡 윈도우를 떨어뜨려 성능을 향상시키려던 원래의 목적과는 멀어지게 될 것이다.

아래 표1에서는 현재 나와 있는 방법들 중에서 I-TCP , M-TCP, Snoop, Freeze-TCP들의 특징들을 비교하여 나타내었다.

3.7 제안된 Mobile TCP 성능개선 방안

본 논문에서는 무선망을 고려한 TCP 성능개선 방안으로서 기존에 제안된 여러 방안 중에서 송신측과 MH간에 true end-to-end semantics가 유지되지 않는 I-TCP와 M-TCP의 적용은 배제하고 장단점면에서 상호 보완관계에 있는 Snoop와 Freeze-TCP을 동시에 적용하는 방안을 제시한다. Snoop 메카니즘의 경우 I-TCP나 M-TCP에서 처럼 중간 노드의 기능에 있어서 약간의 수정이 필요하지만 TCP 안의 알고리즘의 수정은 필요하지 않으며 단순히 BS에 Snoop 모듈을 추가하는 것으로 충분하다. 다른 한편으로 Snoop 메카니즘에서는 핸드오프의 경우와 같은 연결 중단에 따른 혼잡 윈도우 크기의 감소에 대한 대책이 없지만 이러한 문제는 MH의 TCP에 약간의 수정을 가하여 Freeze-TCP를 구현함으로써 해결할 수 있으며, M-TCP 혹은 Snoop에서 핸드오프를 위해 주변 BS들에까지 버퍼링을 하여 많은 메모리를 필요로 하게 하는 문제와 BS에 많은 부하가 걸리는 것을 경감시킬 수 있어 성능향상을 위한 비용 및 운용상의 부담을 줄일 수 있다. 따라서 제안된 메카니즘을 통하여 end-to-end semantics를 유지 하면서 핸드오프와 같은 사전에 감지가 가능한 연결중단에 관한 처리는 Freeze-TCP에서 담당하고 트래픽 전송 중에 일어나는 BER(Bit Error Rate)에 관한 것은 Snoop에서 처리하여 이동망에서 발생하는 할 수 있는 TCP의 성능저하를 최대한 줄일 수 있다. 이어지는 장에서는 제안된

방안을 각각BS와 MH에 적용하고 ns-2를 이용한 시뮬레이션을 통해 성능을 평가해 보기로 한다.

Ⅳ. 시뮬레이션 모델

본 논문에서는 ns-2[8]를 사용하여 제안된 방안의 성능평가를 수행하였다. 실험 환경은 Linux kernel 2.2.16 (Redhat 7.1)상에서 ns-2.1b7a 버전을 사용 하였다. 네트워크 구성을 위해 tcl script를 실행하여 나온 결과는 그림 5와 같다.

그림 5에서 node 0는 송신측 호스트로써 MH인 node 5로 데이터를 전송하게 된다. node 1은 유선망의 중간 라우터이며, node 3는 BS이며 또한 node 5의 HA(Home Agent)이고, node 4 역시 BS로서 FA (Foreign Agent)이다. Node 2는 유선망 안에서 혼잡을 유발하기 위해 추가된 FH(Fixed Host)이다. 각 node간의 링크 대역폭은 1Mbps이고 유선망에서의 node간의 propagation delay는 20ms 무선 링크상에서는 2ms로 하였다. 시뮬레이션 시작후 2초가 지난다음부터 데이터전송이 시작되며 4.8~5초에서 node2에서 node3으로의 1Mbps UDP 트래픽을 유발시켜 유선망에서의 혼잡이 유발된다. 16초와 41초 부근에서 무선망에서의 BER에 의해 패킷이 하나씩 유실되며, 27초에서 32초 사이에서 핸드오프가 일어난다.

그림 5는 시뮬레이션 진행을 나타내고 있는 화면으로 MH가 HA를 떠나 FA지역으로 들어가서 데이터를 전송받고 있는 모습이다. 길게 화살표 모양을 띄고 있는 것이 데이터 패킷이며 Mobile IP에 의해 송신측 호스트에서 HA를 거쳐서 FA로 보내져서 목적지인 MH에 패킷이 도달하고 있다. 또한 작은 사각형은 ACK 패킷으로 MH에서 FA를 거쳐 송신측 호스트에 직접 전송이 이뤄진다.

Ⅴ. Simulation 결과

본 장에서는 일반 TCP, Snoop, Freeze-TCP, 그리고 제안된 방안을 적용했을 경우에 대해 시뮬레이션을 수행하고 얻어진 결과에 대해 비교 분석을 하기로 한다. 그림 6은 일반 TCP를 적용하여 시뮬레이션 한 것에 대한 결과로 그래프에는 전송된 패킷의 양과 TCP window의 크기 변화를 나타내었다.

그림 6에서 보여지는 것과 같이 패킷이 유실된 5초, 16초, 41초 모든 부분에서 또한 hand off가 일어난 27초~32초 사이에서 TCP window 크기가 감소하였다가 다시 증가하는 것을 볼 수 있다. 시뮬레이션에서 수신측의 maximum window size는 43으로 설정하였다.

그림 7은 Snoop 모듈만을 적용하여 시뮬레이션한 결과이다. 그림 7에서 보여지는 것과 같이 무선상에서의 BER에 의해 패킷이 유실된 16초와 41초 부분에서 TCP window가 혼잡회피 단계로 들어가지 않고, Snoop 모듈이 적용되어 있는 HA와 FA에서 패킷 유실즉시 지역 재 전송을 함으로써 계속 maximum window size를 유지 하며 전송되어지고 있는 것을 볼 수 있다.

그림 8은 Freeze-TCP만을 MH에 적용하여 시뮬레

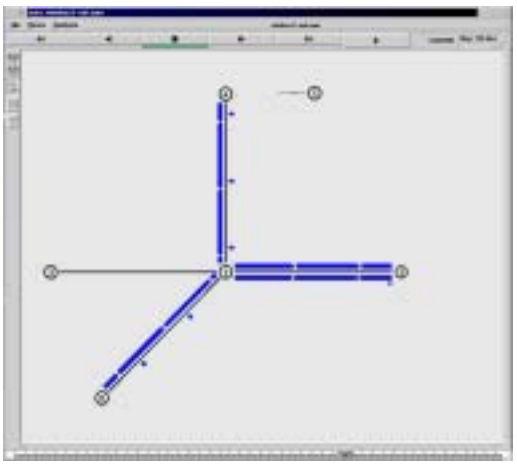

그림 5. ns-2 시뮬레이션 주 화면.
Fig. 5. The ns-2 simulation main window.

이션한 결과로서 핸드오프가 시작되기 전에 MH는 송신측 호스트로 ZWA패킷을 전송하여 송신측 호스트가 persist mode로 들어가게 하고 핸드오프 완료 직후에 MH가 핸드오프 시작 전에 받았던 마지막 패킷에 대한 ACK와 자신의 maximun widnow size를 실어 보냄으로써 송신측 호스트가 곧바로 최대 전송률로 데이터를 보낼 수 있도록 한다. 그림 8에서 보는것과 같이 핸드오프 완료 직후인 31초 부근에서도 window 크기는 그 전의 상태를 유지하고 있는 것을 볼 수 있다.

마지막으로 본 논문에서 제시하고자 했던 Snoop와 Freeze-TCP를 동시에 적용하여 TCP의 end-to-end semantics를 유지하면서 무선망에서 일어나는 BER과 핸드오프에 대한 성능개선을 보이고자 한다.

그림 9에서와 같이 16초와 41초 부분에서 일어났던 BER에의한 패킷유실로 TCP window 크기가 줄어드는 것은 BS에 들어있는 Snoop 모듈에서 지역 재전송

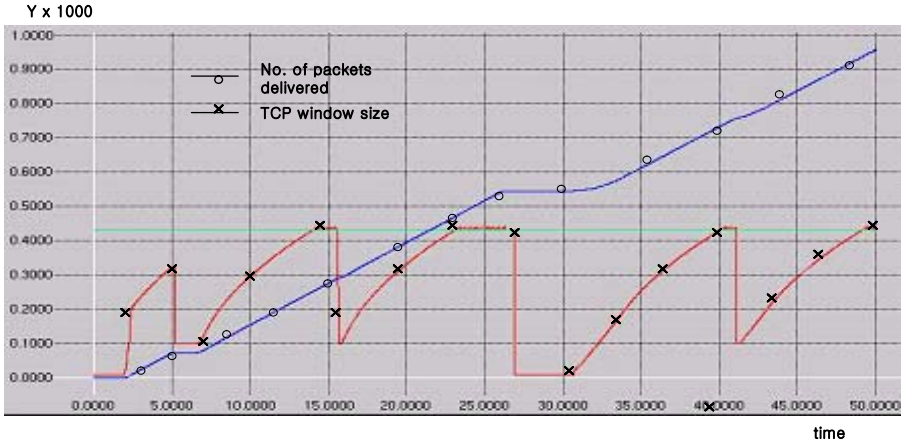

그림 6. 일반 TCP에 대한 시뮬레이션 결과.
Fig. 6. The simulation results for regular TCP.

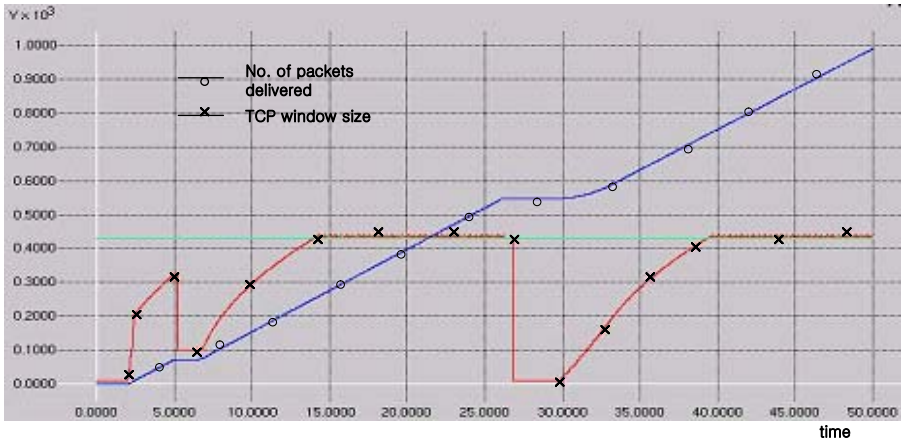

그림 7. Snoop에 대한 시뮬레이션 결과.
Fig. 7. The simulation results for Snoop.

으로 처리하여 주고 26초에서 31초 사이에 일어나는 핸드오프는 MH에 적용된 Freeze-TCP가 작동하여 송신측 호스트를 persist mode로 들어가게 함으로써 무선링크 상에서 할당받은 대역폭의 이용률을 떨어뜨리지 않고 사용하고 있음을 보여준다.

표 2는 각각의 시뮬레이션에서 50초 후에 MH에 도착한 데이터 패킷의 수를 비교한 것이다. BER에의한 패킷 유실의 경우 일반 TCP의 경우 재전송과 함께 TCP window 크기 감소가 이뤄지나 제안된 방안에서는 송신측의 재전송의 필요없이 Snoop 모듈에서 지역 재전송으로 처리하여 주고 핸드오프와 같은 짧은 연결 중단의 경우 MH에 적용된 Freeze-TCP가 작동하여 송신측에서 혼잡 윈도우 크기를 줄이지 않도록 함으로써 전체적으로 전송효율을 높게 유지할 수 있다.

무선망의 상황에 따라 BER이 틀려 질수 있으며 MH가 얼마나 많이 이동하느냐에 따라 핸드오프가 발생하는 횟수가 틀려지기 때문에 Snoop와 Freeze-TCP를 적용했을 때와 안 했을 때와의 차이를 정량적으로 나타낼 수는 없으나 표 2에서 보듯이 기존의 방식을 적용했을 때와 비교하여 제안된 방안을 적용한 경우 같은 시간에 많은 수의 데이터 패킷을 전달할 수 있다는 것을 알 수 있다.

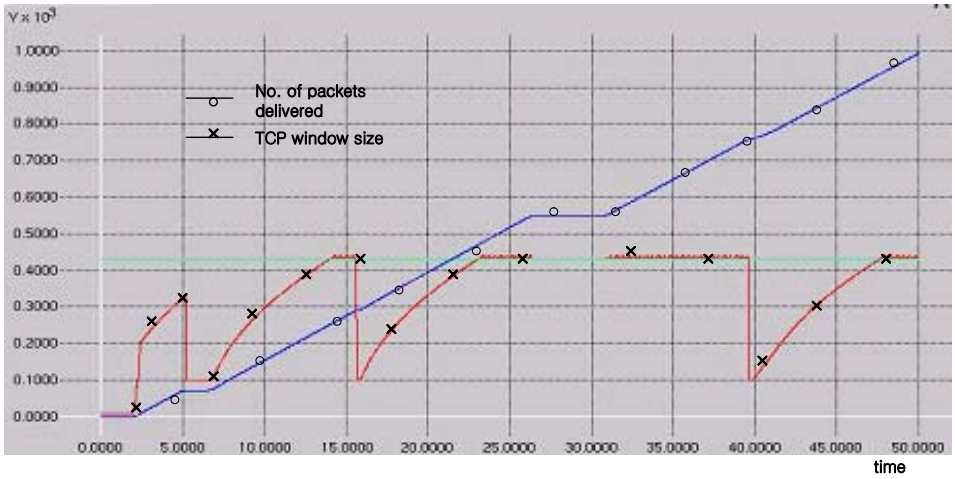

그림 8. Freeze-TCP만을 적용했을 경우에 대한 시뮬레이션 결과.
Fig. 8. The simulation results when Freeze-TCP only is applied.

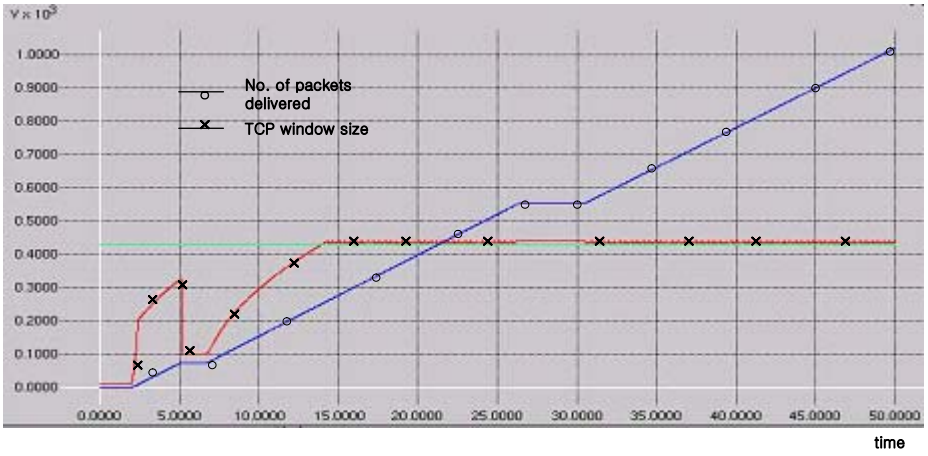

그림 9. 제안된 방안에 대한 시뮬레이션 결과.
Fig. 9. The simulation results for the proposed scheme.

표 2. 각 simulation에서 MH에 도착한 패킷의 수.
Table 2. The number of packets arriving at the MH for each simulation.

	Regular TCP	Snoop	Freeze-TCP	Snoop+Freeze-TCP
50초 후 MH에 도착한 데이터 패킷의 수	4785	4955	4960	5095

Ⅵ. 결론 및 추후 연구방향

본 논문은 현재 거론되고 있는 무선망 환경에서 보다 나은 TCP성능을 낼 수 있는 방법들 중에서 true end-to-end semantics를 유지하며, 송신측 호스트 또는 중간 노드에서의 최소한의 변경, BS에서의 추가적인 부하와 버퍼를 최소화 할 수 있는 방법으로 Snoop와 Freeze-TCP를 병행하는 방법을 제안하고, 시뮬레이션을 통해 그 성능을 보였다. Snoop모듈은 각 BS에 추가해야 하고 추가적인 버퍼가 필요하긴 하지만 I-TCP나 M-TCP와 같이 BS에서의 TCP코드의 변경 없이 Snoop모듈을 삽입하면 되므로 업데이트가 간편하며 핸드오프와 같이 사전 감지가 가능한 연결중단에 대해서는 MH의 TCP에 약간의 수정을 가하여 Freeze-TCP를 구현함으로써 M-TCP 혹은 Snoop에서 핸드오프를 위해 주변 BS들에까지 버퍼링을 하여 많은 메모리를 필요로 하게 하는 문제와 BS에 많은 부하가 걸리는 것을 경감시킬 수 있어 성능향상을 위한 비용 및 운용상의 부담을 줄일 수 있을 것이다.

본 논문에서 수행된 연구는 단지 FH에서 MH로의 TCP데이터 전송에 있어서의 성능개선에 관한 연구 결과 이며 반대방향인 MH에서 FH로의 TCP데이터 전송에 있어서의 문제점에서 대해서는 고려되지 않았다. MH로는 현재 Cellular phone 뿐만 아니라 노트북, PDA등 단순히 데이터를 수신하는 기능뿐만이 아닌 작업한 데이터를 전송하고자 하는 요구도 늘어날 것이고 그 데이터의 크기도 커질 것이라고 판단된다. 따라서 이 부분에 관한 연구도 이루어 져야 할 것이며 위성과 같이 송신측과 수신측 사이에 무선전송 부분이 존재할 때 어떻게 TCP의 성능을 유지할 수 있을 것인지에 관한 연구도 더 이루어 져야 할 것이다.

참고문헌

[1] A. K. Salkintzis, *"A Survey of Mobile Data Networks"*, in IEEE Communication Surveys, vol. 2, no. 3, third quarter, 1999.

[2] Ray Yavatkar and Namrata Bhagawat, *"Improving End-to-End Performance of TCP over Mobile Internetworks"*, in IEEE Worhhop on Mobile Computing Systems and Applications, Santa CNZ, CA, US, Dec. 1994.

[3] A. Bakre and B. R. Badrinath, *"I-TCP: Indirect TCP for Mobile Hosts"*, Proceedings of the 15th International Conference on Distributed Computing Systems, Vancouver, Canada, June 1995, pp. 136-143.

[4] H. Balakrishnan, S. Seshan, and Randy Katz, *"Improving Reliable Transport and HandoPerformance in Cellular Wireless Networks"*, Wireless Networks, Vol. 1, No. 4, December (1995), pp. 469-481.

[5] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and Randy Katz, *"A Comparison of Mechanisms for Improving TCP Performance over Wireless Links"*, ACM SIGCOMM'96, pp. 256-269, Palo Alto, CA, August (1996).

[6] Kevin Brown, Suresh Singh, *"M-TCP: TCP for Mobile Cellular Netwroks"*, ACM Computer Communications Review (CCR), vol. 21, no. 5,1997.

[7] Tom Goff, James Moronski, Vipul Gupta, *"Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments"*, in Proceedings of IEEE INFOCOM 2000, Israel.

[8] Network Simualtor 2(NS-2), http://www.isi.edu/nsnam/ns/

저 자 소 개

박도용 (정회원)

2000.3 ~ 2002.2 건국대 대학원 전자공학과
2002.4 ~ 현재 LG 정보통신
<주관심분야>
고속통신망 트래픽제어, 스위칭 시스템, 이동통신

김영범 (정회원)

79. 3~84. 2 서울대 전자공학과
84. 3~86. 2 서울대 대학원 전자공학과
90. 9~96. 7 미 메릴랜드 주립대
97. 9~현재 건국대 전자공학과 부교수
<주관심분야>
고속통신망 트래픽제어, 스위칭 시스템, 이동통신