Snoop 프로토콜에서 혼잡 제어 지연을 통한 이동망상에서의 TCP 성능향상 기법

김 용[†] · 성 호 철[†] · 현 호 재[††] · 한 선 영[†††]

요 약

무선망에서는 유선망에 비해 그 특성상 비교적 많은 패킷을 손실된다. TCP 프로토콜은 흐름제어나 에러정정, 혼잡 제어 등의 기능을 통해 보다 효율적이고 안정적인 통신을 지원하고 있다. 하지만 표준 TCP 프로토콜은 유선망의 특성을 고려하여 개발되었기 때문에 무선망에서 혼잡한 상황에서 패킷이 도달하지 못한 경우와 실제로 패킷이 손실되어 전달되지 못하는 경우를 구분하지 못한다. 최근까지 제시된 여러 이동망 TCP에 대한 논문은 무선망에서 패킷이 손실된 경우 혼잡 제어를 일어나지 못하게 하는 방법을 제시하고 있다. 본 논문에서는 TCP Persist Timer를 이용하여 혼잡제어를 회피하는 방법을 기존에 제시된 Snoop 프로토콜에 적용하여 자체적인 이동망상에서의 TCP 성능향상에 더하여 연속적인 에러에 대한 성능 향상을 제공하고 있다. 개선된 Snoop 프로토콜은 WZACK(Window Size Zero ACKnowledge Packet)을 이용하여 혼잡제어를 정지시킴으로써 비효율적인 혼잡제어를 막도록한다.

Improving the performance of TCP over networks of mobile with delaying congestion control in Snoop

Yong Kim[†] · Ho-Cheol Sung[†] · Ho-Jae Hyun[††] · Sun-Young Han[†††]

ABSTRACT

There are more packet loses in wireless networks than in wired networks. The TCP protocol supports effective and stable network services with flow control and error control, congestion control. Since the standard TCP protocol is designed on characteristics of the wired network, it doesnt demarcate congestion from general packet loss over the wireless network. In these days, many papers describe about mobile TCP which have suggested how to avoid congestion control when packet loss over the wireless network. This paper suggests avoiding congestion control by using enhanced Snoop protocol. To enhance the original Snoop protocol, TCP persist timer is added. An enhanced Snoop protocol blocks inefficient congestion control by using WZACK(Window Size Zero ACKnowledge Packet).

키워드 : 이동망(Mobile), TCP, 무선망(Wireless), 스눕(Snoop), 혼잡(Congestion)

1. 서 론

데이터 단말의 증설과 이동이 빈번해지면서 이에 따른 케이블 공사가 커다란 부담으로 작용하기 마련이다. 그래서 케이블을 미리 설치해 놓고 증설되는 단말은 여기에 연결하여 사용함으로써, 케이블공사의 부담을 줄이겠다는 발상에서 70년대 말 탄생된 것이 LAN(Local Area Network)이라 할 수 있다. 즉, LAN의 개발목적 중 하나는 정보 기기를 연결하는 케이블의 간소화에서 찾을 수 있다.

그러나, 최근 LAN의 구성이 커지고 노트북 PC와 같이 휴대형 데이터 단말이 빠른 속도로 보급되면서, LAN 이용자들 사이에서는 또 다른 새로운 요구가 일고 있다. 그것은 무선 LAN이 갖는 장점과 그 맥을 같이 한다고 볼 수 있다. LAN의 케이블을 무선으로 대체함으로써 단말을 케이블로부터 해방시키기 위한 무선 LAN의 실용화가 추진되고 있다. 무선 LAN에서도 유선 LAN에 적합한 TCP/IP를 이용함으로써 무선 LAN의 특성으로 인한 문제가 제기되었다. 그래서 Mobile IP나 Mobile TCP 같은 이름의 무선 LAN상에 적합한 형태로 변형된 프로토콜이 제시되기 시작했다.

Mobile IP는 기본적으로 IP 레이어에서 무선 LAN 상의 핸드오프시 라우팅 경로를 찾기 위한 방향으로 기술이 발전되어 왔으며 Mobile TCP는 IP 레이어의 상위 레이어인 TCP 레이어에서 보다 안정적인 속도를 목표로 발전되어왔

※ 본 논문은 1999년도 건국대학교 학술진흥연구비지원에 의한 결과임.
† 준 회 원 : 건국대학교 대학원 컴퓨터정보통신공학과
†† 정 회 원 : 건국대학교 대학원 컴퓨터정보통신공학과
††† 정 회 원 : 건국대학교 컴퓨터공학과 교수
논문접수 : 2000년 10월 30일, 심사완료 : 2001년 5월 25일

다. 그래서 Mobile TCP에서 개발된 여러가지 기법들이 발표가 되어있으며 이러한 프로토콜들은 상호 장단점을 가지고 있다.

본 논문은 이동망에서의 성능향상 프로토콜 중 The Berkeley Snoop Protocol (이하 Snoop)이라는 프로토콜을 이용하여 연속적인 에러에 대한 모듈을 추가하는 것으로 Mobile 상의 TCP 성능향상 방법을 제시하고자 한다.

2. 관련 연구

2..1 TCP 프로토콜 혼잡 제어

혼잡(Congestion)은 하나 또는 그 이상의 스위칭 점 (즉 라우터)에서 데이터그램의 과부하로 인해 심각한 지연이 발생된 상태이다. TCP는 혼잡으로 인해 지연이 늘어나면 자동적으로 전송 속도를 줄여서 혼잡을 피하게 한다. 혼잡을 피하기 위해서 TCP는 다음과 같은 두 가지 기술을 사용한다.

㉠ Slow-Start
㉡ Multiplicative decrease

TCP는 연결이 이루어지면 수신자의 윈도우 크기를 기억하게 된다. 그리고 혼잡을 제어하기 위해 TCP는 혼잡윈도우(Congestion Window)를 유지한다. 윈도우 크기는 다음과 같이 정의된다.

$$\textbf{\textit{Allowed\_window}} = \min(\textbf{\textit{receiver\_advertisement}}, \textbf{\textit{congestion\_window}})$$

혼잡이 없는 안정된 전송 상태에서 혼잡 윈도우는 수신자의 윈도우 크기와 같다. 혼잡 윈도우를 줄이게 되면 TCP가 전송하는 트래픽은 감소한다. TCP는 세그먼트 손실이 발생하면 혼잡상황으로 취급하고 혼잡 윈도우를 반으로 줄인다. 따라서 TCP는 매 손실마다 혼잡 윈도우를 반으로 줄이기 때문에 손실이 조금만 지속되면 지수적으로 윈도우를 감소시키게 된다. 계속적인 손실이 발생하면 TCP는 하나의 데이터그램으로 전송을 제한하고 재전송하기 전에 만료되는 시간을 두 배로 늘린다.

혼잡한 상황이 끝나면 TCP는 Slow-Start 기술을 이용한 회복을 시도한다. Slow-Start는 새롭게 연결이 시작되거나 혼잡한 상황이 지나면 트래픽이 증가될 때마다 단일 세그먼트에서 혼잡 윈도우를 시작한다. 그리고 ACK이 돌아오면 한 세그먼트 단위로 혼잡 윈도우를 증가시킨다. Slow-Start의 사용으로 혼잡이 끝난 후 동일한 혼잡 상황이 발생되는 것을 막게 된다. 그러나 Slow-Start 라고 해서 시작이 느린 것은 아니다. TCP는 혼잡 윈도우를 이전의 2배로 반복해서 늘려나간다. 그러나 같은 혼잡 상황을 회피하기 위해, TCP는 하나의 추가적 제약을 가지고 있다. 일단 혼잡 윈도우가 혼잡 전 크기의 반이 되면 TCP는 혼잡 회피 단계에 들어가고 혼잡 윈도우를 1씩 증가시킨다. 이러한 방법으로 TCP는 흐름제어를 하게 되어 초기의 TCP에 비해 두 배에서 열 배까지의 성능 향상을 가져왔다[1].

2.2. Mobile TCP

무선망은 기존의 유선망에 비해 느린 속도와 잦은 에러를 발생시킨다. 이러한 전송매체의 차이는 기존의 유선망에서 사용되던 TCP를 무선망에 적용함에 있어서 특성에 맞지 않는 부분이 생긴다. 이로 인해 성능저하가 일어나는 것은 필연적이라 할 수 있다.

TCP의 문제점은 무선망의 패킷 손실 발생시 동작하는 혼잡 제어이다. 무선망의 특성상 패킷 손실이 잦음에도 불구하고 TCP는 무선망에서의 패킷 손실과 혼잡에 의한 패킷 손실을 구분하지 않고 유선망에서와 같이 혼잡상황으로 인식한다. 무선망의 특성상 자주 일어나는 패킷 손실이 일어날 때마다 송신측은 보내는 속도를 낮추게 된다.

이러한 TCP의 특성은 에러가 잦은 무선망에서 TCP 성능을 크게 저하시키므로 이런 문제점을 해결하기위해 여러가지 이동망 TCP(Mobile TCP)가 제안되었다. 이동망 TCP는 혼잡과 손실을 구분하여 다른 방식으로 처리 하는 방식이 많이 제시되어 있다[2]. 그러한 방식에는 다음과 같은 것들이 있다[3].

㉠ The Split Connection Approach
㉡ The Fast-Retransmit Approach
㉢ Link-level Retransmissions

The Split Connection Approach 방식은 무선망과 유선망 연결 부분을 분리하여 한 TCP연결을 두개의 TCP연결로 바꾼다. 이 방식의 Indirect TCP(I-TCP) I-TCP는 기지국에 I-TCP 대리인을 둔다. 그리고 송신측 측에서 연결을 요청하면 대리인이 연결을 맺고 대리인은 다시 이동 노드와 연결을 맺는다. 그렇게 연결을 유지하고 데이터 전송을 하면 대리인은 중간에서 다시 다른 연결로 포워딩 해준다. 그래서 대리인과 이동 노드간에 기존의 TCP 대신 무선망에 적합하도록 변경된 TCP를 사용하여 많은 성능향상을 할 수 있다[4].

그러나 이 방법은 여러가지 단점이 있다. 첫 번째로, 기본적인 TCP semantic을 무시하게 된다. 일반적인 TCP는 종단간(end-to-end)연결을 이루게 되지만 I-TCP는 중간에 I-TCP가 연결을 맺기 때문에 종단간연결을 만족하지 못한다. 두 번째로 응용프로그램의 새로운 링크가 필요하다. 기존의 TCP를 대체하는 I-TCP를 사용하기 때문에 TCP를 사용하는 모든 어플리케이션을 TCP 소켓시스템 콜대신 I-TCP 소켓시스템 콜을 이용하여 재링크 시켜야 한다. 마지막으로 TCP상에서의 부담을 줄이기 위해 I-TCP를 사용했기 때문에 4번의 패킷 복사가 일어나게 된다. 송신측 쪽

에서 한번, 이동 노드쪽에서 외에 기지국에서 2번의 복사가 일어난다[5].

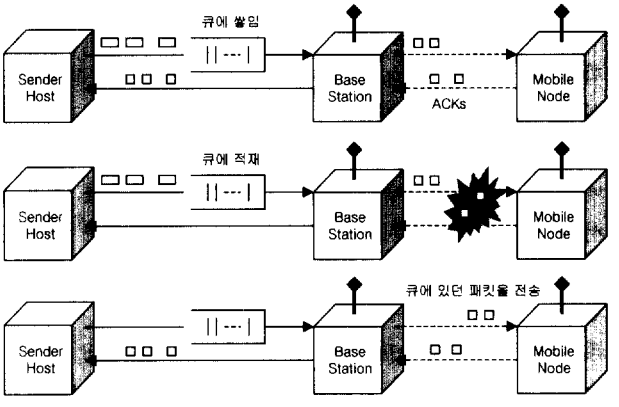
The Fast-Retransmit Approach 방식은 핸드오프(Handoff) 시 속도 향상을 위한 방법으로 송신측쪽이 핸드오프에 의한 데이터그램 손실을 혼잡에 기인한다고 간주하고 윈도우 사이즈를 줄이고 패킷을 재전송한다. 일반적으로 핸드오프되는 동안 자주 패킷이 손실된다. 이 때 이동 노드 쪽에서는 중복된 ACK을 보내게 되면 송신측 측에서는 바로 패킷 손실로 인식하고 재전송할 패킷을 보내주게 된다. 하지만 이 방법은 중복된 ACK과 손실된 모든 패킷에 대한 송신측 쪽에서 반복되는 재전송 때문에 혼잡한 상황을 가중시키는 부담을 안고 있다는 것이 문제이다[6].

Link-level Retransmission은 무선망의 데이터 링크 계층에서 자체적으로 재전송하는 프로토콜을 이용하는 것으로 TCP 는 종단간의 연결을 보장 받을 수 있게 된다. 하지만 그에 따른 부하와 함께 설계에 관해 상위계층에 대한 충분한 정보가 필요하다. 그러한 문제로 인해 아직 이 방법에 대한 완전한 연구가 되어 있지 않지만 관련된 작업들이 수행중이다[7].

2.3 Snoop

Snoop은 유선망에서는 사용되는 TCP구현을 변경하지 않고 사용하면서 무선망의 TCP 성능을 향상시킨다. Snoop은 성능향상을 위해 Snoop 모듈과 라우팅 프로토콜을 소개하고 있다. Snoop 모듈은 송신측에서 이동 노드로 전송이 일어날 때, 송신측은 무선망의 존재를 모르는 채, 무선망상의 패킷 손실로 인한 혼잡 제어를 막기 위해서 기지국의 네트워크 계층 소프트웨어를 변형하여 TCP 윈도우의 최대 크기 정도의 캐시를 갖는 Snoop이라는 모듈을 추가하여, 송신측에서 오는 패킷을 모니터하고, 이동 노드로부터 ACK을 받지 못한 패킷을 캐싱하여 중복된 ACK이나 타임아웃이 발생하였을 때 기지국에서 잃어버린 패킷을 재전송하는 국부적 재전송(Local Retransmission)을 수행할 수 있게 한다. 그래서 TCP상의 재전송이 아니라 기지국과 이동 노드간의 부분적 국부적 재전송만이 일어난다.

(그림 1)은 Snoop 알고리즘을 나타낸 것이다. 먼저 기지국이 송신측으로부터 패킷을 받을때, 정상적인 순서의 TCP 패킷이 도달하면 패킷을 Snoop 캐시에 저장하고 이동 노드에 전송한다. 이전에 저장된 순서에 어긋난 패킷이 도달하면 이는 송신측 타임 아웃이나 TCP의 빠른 재전송(Fast Retransmission)에 의한 것이므로 저장된 패킷을 재전송하게 한다. 아직 저장이 되지 않은 상태의 순서가 어긋난 패킷이 도달하면 이는 패킷이 비정상적으로 도달한 혼잡이 일어난 경우이므로 이 패킷들은 혼잡을 경험했다는 기록을 해둔다. 이 정보는 기지국에서 이동 노드로부터 ACK를 받을 때 이용된다. 기지국이 이동 노드로부터 비정상적인 ACK를 받았을 때 이 ACK은 무시한다. 이전에 받은 ACK와 같은 중복된 ACK를 받으면 이동 노드가 다음 패킷을 받지 못했으므로 기지국은 다음의 동작을 취한다. 만약 원하는 패킷이 Snoop 캐시에 없거나 혼잡 상태에 의해서 손실 되었다면 DUPACK를 송신측으로 보내 혼잡 제어 메커니즘을 일으킨 후 패킷을 재전송하게 한다. 그렇지 않다면, 즉 원하는 패킷이 Snoop 캐시 안에 존재할 때, 기대하지 않았던 DUPACK을 받으면 패킷을 재전송한다. 만약 패킷이 Snoop 캐시 안에 존재하고, 예상했던 DUPACK를 받으면, DUPACK를 무시한다. 새로운 ACK의 경우 이동 노드가 받은 순서(Sequence)의 증가를 가리키므로 현재 상태정보을 이용해 무선망에 의한 패킷 손실인지 검사하고 무선망에 의한 패킷 손실이면 재전송하고 재전송 타이머를 리셋하고 상태를 수정해야 한다. 그렇지 않다면 ACK을 송신측으로 보내고 캐시를 비운다.



(그림 1) Snoop 동작 구조

Snoop에 대한 초기 논문[8]에서는 이동 노드가 송신측으로 전송하는 것에 대해서는 나와있지 않다. 하지만 이후에 발표된 논문[6]에는 이에 대한 해결책을 제시하고 있다. 현재 TCP구현에서 쓰이지 않고 있는 SACK(Selective ACK) 옵션을 이용하여 NACK(Negative ACK)을 구현해서 이동 노드와 기지국간의 전송에서 일어나는 패킷 손실을 해결한다. 이동 노드에서 기지국으로 패킷을 전송할 때 단일 윈도우에서 일정한 수이상의 패킷이 기지국에 도달하거나 일정한 시간동안 새로운 패킷의 도달이 없다면 기지국의 Snoop이 NACK을 보낸다. NACK은 비트 벡터의 형태로 보내지므로 무선자원을 많이 차지하지 않는다. 이동 노드는 NACK을 이용하여 손실된 패킷을 선택적으로 다시 전송한다. 전송된 패킷들은 기지국의 버퍼에 저장되었다가 송신측으로 전달된다. 이 방법의 장점은 TCP의 종단간 연결이 유지되고, 고정된 네트웍의 TCP 구현이 바뀌질 필요가 없고 단지 새로 추가되는 이동 노드와 기지국의 코드가 변경되면 된다. 또한 기지국에서 Snooping을 하는데 트랜스포트 계층의 부하가 전혀 없다.

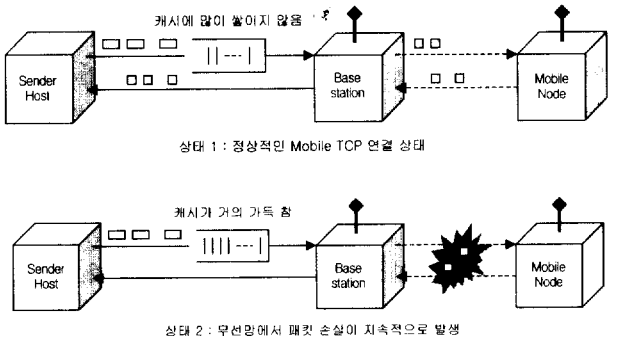
그리고 사용자의 이동에 의한 TCP성능 저하를 막기 위한 라우팅 프로토콜을 제시하고 있다. 이동 노드가 무선환경에서 셀(Cell)간을 이동하면 송신측과의 경로정보가 갱신되어야 하는데, Mobile IP에서는 이런 핸드오프가 일어날 때 패킷 손실이나 긴 시간지연이 발생한다. 이는 TCP 계층에서 심각한 성능저하를 가져온다. 그러한 문제를 극복하기 위해서 제안된 방법에는 멀티캐스트를 하기위한 기지국에서의 지능적 버퍼링을 사용한다. 기본적인 라우팅 방법은 Mobile IP와 비슷하고 차이점은 낮은 지연시간을 갖는 핸드오프를 지원하고 패킷 손실과 지연시간의 변화를 줄여준다는 점이다.

각 이동 노드는 자신의 홈 위치에 관련한 홈 주소와 멀티캐스트를 위해 임시로 유지되는 IP 멀티캐스트 주소를 가진다. 이동 노드의 홈 에이전트(이하 HA)는 이 홈 주소로 전달되는 모든 패킷을 프록시 ARP를 이용하여 가로채서, 캡슐화하여 해당 멀티캐스트 그룹으로 전달한다. 멀티캐스트 그룹에는 이동 노드의 주변 기지국이 포함된다. 라우팅 시스템은 이동 노드의 위치를 결정하기 위해, 기지국이 주기적으로 경계 메시지를 브로드캐스트하고, 각 이동 노드는 위치와 이동을 파악하기 위해 최근에 받은 경계의 상태를 모두 기록하고 이 통계를 이용해서 자신의 주변 기지국을 확인한다. 또한 자신이 속해있는 셀과 앞으로 핸드오프를 할 가능성이 있는 셀도 결정한다. 이런 결정을 바탕으로 자신의 HA와 여러 기지국간의 라우팅을 설정한다. 라우팅이 설정되고나면 IP 멀티캐스트가 제공하는 동적 라우팅을 이용하여 HA에서 기지국으로 패킷 전송이 일어나고 이때 이동 노드를 포함하고 있는 셀의 모든 기지국이 IP 멀티캐스트 그룹에 포함된다. 멀티캐스트 그룹으로 전송된 패킷은 주(Primary) 기지국에 의해 이동 노드로 전달된다. 멀티캐스트 그룹의 나머지 기지국들은 이동 노드로 전송된 패킷 중 일부만 저장하고 있게 된다. 그리고 핸드오프가 일어나면 새로운 주 기지국은 이전 주 기지국로부터 패킷을 전달받을 필요가 없이 버퍼링하고 있던 패킷을 전송한다.

3. 설 계

3.1 연속적인 에러 감지 알고리즘

Snoop 프로토콜에서는 빠른 국부적 재전송(Local Fast Retransmission)을 위한 캐시를 유지하고 있다. (그림 2)는 ACK을 받지 못한 데이터그램이 캐시에 쌓여있는 모습을 보이고 있다. 정상적인 이동망 환경에서는 데이터그램이 캐시에 쌓임과 동시에 계속 빠져나가므로 캐시에는 항상 적당한 수준의 데이터그램만이 쌓여있다. Snoop은 국부적 재전송, 즉 기지국과 이동 노드간의 재전송을 지원하기 때문에 어느 수준의 에러 또한 뛰어난 수준으로 복구가 가능하다는 것을 이미 앞의 관련 연구에서 언급한 바 있다. 그러나 반복되는 에러상황에서 이동 노드로 재전송되는 패킷은 보내지지 않기 때문에 캐시에 계속 데이터그램이 쌓이기 시작한다. 이 특성을 이용하여 연속적인 에러(Burst-Error) 감지를 한다. 이러한 연속적인 에러는 통상의 핸드오프(Hand Off) 로 인한 연결 끊김외에도 통신 음영 지역에서 볼 수 있는 장시간의 연결 단절 현상에서 발생할 수 있다.



(그림 2) Snoop 에서 Burst Error 가 발생하는 경우

Snoop은 캐시에 데이터그램과 함께 <표 1>에서 보이는 관련 정보를 동시에 저장한다.

〈표 1〉 Snoop 내에서 캐싱되는 내용

struct mbuf *mb	캐싱될 데이터그램의 링크
struct iphdr iph	IP 헤더 중 몇 가지 필드
Int tcp_sum	TCP 체크섬
Tcp_seq seq	TCP 시퀀스 번호
Int size	캐싱된 TCP 데이터의 크기
Timev snd_time	전송되는데 걸린 시간
int num_rxmit	국부적 재전송 시도 횟수
int sender_rxmit	송신측 측에서 재전송되었는지의 여부

연속적인 에러를 감지하기 위하여 패킷이 들어올 때마다 패킷을 캐싱할 때 캐싱 버퍼의 크기를 알아본다. 이 경우 버퍼의 크기가 사전 정의된 수준 이상인 경우 재전송 하고자 하는 패킷의 국부적 재전송 횟수를 검사한다. 이때 연속적인 에러로 정의되어 있는 재전송 횟수를 넘어선 경우 연속적인 에러에 의한 재전송으로 간주하여 에러 처리 알고리즘을 수행한다. 그러나 실제로 연결이 끊어져 에러가 생기는 경우 무한적으로 송신측을 지연시키는 문제가 발생할 수 있으므로 재전송 횟수가 일정 횟수 이상이 되는 경우 WZACK을 더 이상 보내지 않도록 하여 연결이 끊긴 경우는 에러 처리 알고리즘을 중단시킨다.

본 논문에서는 캐싱 버퍼의 크기를 배포되고 있는 Snoop 소스에 정의된 32(0x20)개로 사용하고 재전송 횟수는 3(0x03) 회로 계산하여 연속적인 에러의 경계로 삼았다. 또한 연결 끊김에 대비한 재전송 횟수 제한은 16(0x10)회로 사용했다.

3.2 혼잡 제어 지연 알고리즘

이동망상에서의 TCP는 혼잡(Congestion)과 손실(Corruption)을 구분하지 못한다. 가장 단순한 형태에서의 발생될 수 있는 이동망 TCP는 TCP 자체 내에 혼잡과 손실을 구분할 수 있는 것을 구현하는 것이겠지만 실제적으로 이미 인터넷을 이용하고 있는 수많은 호스트 및 라우터 모두 기존의 TCP를 뒤엎고 새로운 TCP를 이식한다는 것은 비용 및 인력 문제 등 수많은 문제들이 발생할 수밖에 없다 그렇기 때문에 Snoop의 경우 기존의 TCP의 특성을 최대한 살린 채 이동망을 구성하는 기지국과 이동 노드에 대해서만 새로운 Snoop 계층을 포함하고 있다. 따라서 기존의 고정된 호스트는 수정없이 이동망상에서의 성능향상을 한다. 혼잡 제어 지연 알고리즘 또한 기존의 TCP 특성을 이용하여 고정된 호스트의 TCP는 변경하지 않은 채 Snoop 계층에서 연속적인 에러에 대한 모듈을 추가한다.

혼잡 제어 지연을 하기 위해 TCP 특성 중 Persist 타이머를 이용한다. TCP 헤더 중 윈도우 크기를 나타내는 윈도우 필드에 0을 셋팅해서 보내면 송신측에서는 수신측에서 버퍼가 가득찬 것으로 간주하고 데이터 전송을 중지한 채 Persist 타이머를 시작한다.

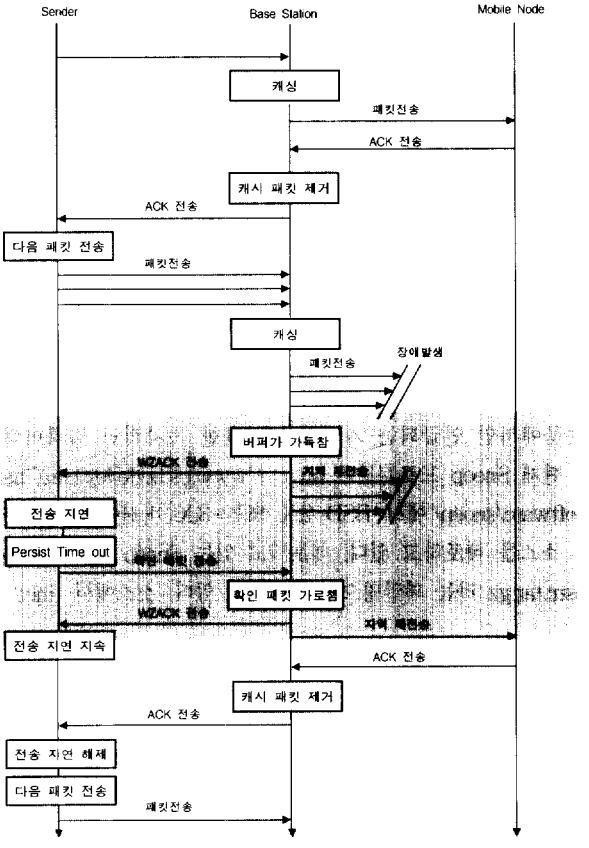
윈도우 필드가 0으로 셋팅된 특별한 용도의 패킷을 WZACK (Window size Zero ACKnowledgment packet) 이라고 이름 붙였다.

(그림 3)은 혼잡 제어 알고리즘을 갖는 Snoop의 흐름도를 보이고 있다. 기존의 Snoop 계층에서는 WZACK을 이용하여 송신측으로 하여금 버퍼가 가득찬 것으로 알리고 데이터 전송을 멎게 한다. 연속적인 에러가 발생하면 WZACK을 송신측쪽으로 전송하고 송신측은 이동 노드가 받을 수 없는 상태로 생각하고 이동 노드가 패킷을 받을 수 있는 상태가 될때까지 기다린다. Snoop 계층이 유지하고 있는 정보 중 연결된 호스트 정보에서 현재 연속적인 에러를 겪고 있다는 것과 현재 ACK의 번호를 추가한다.

그러나 송신측의 Persist 타이머에 의해 송신측에서 주기적으로 다시 수신측으로 버퍼가 비었는지 확인하는 메시지를 보낸다. 이 메시지는 Snoop 계층에서 가로채서 이동 노드로 전달하지 않고 다시 WZACK을 보내서 송신측 쪽으로 하여금 전송을 기다리게 한다.

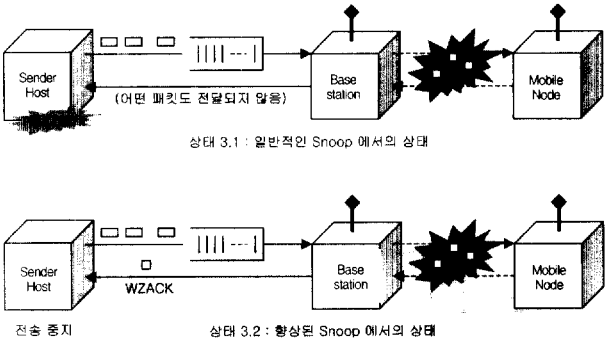
이렇게 연속적인 에러가 일어나는 동안 계속해서 Snoop 계층은 반복해서 국부적 재전송을 수행한다. 무선망에서의 에러가 없어지면 국부적 재전송에 의해 보내지지 않았던 패킷이 전송되고 이동 노드는 자연히 패킷에 대한 ACK을 생성해서 보내게 된다. Snoop 계층에서는 이동 노드가 전송한 ACK을 받게 되면 기존에 기억하고 있던 연결에서의 연속적인 에러 상황을 해제한다. 계속해서 ACK를 송신측으로 포워딩 해주게 되면 송신측에서는 예상하던 ACK을 받게 되어 전송 대기를 멈추고 다시 패킷을 전송하게 된다. 이를 간단히 설명하면,

① 연속적인 에러가 감지된다.
② WZACK 을 전송하면서 국부적 재전송을 계속 시도한다.
③ Persist 타이머에 의해 보내진 송신측 쪽의 확인 패킷을 가로챈다.
④ 송신측 쪽으로 다시 WZACK을 보낸다.
⑤ 에러 상태가 없어지면 정상적인 통신상태로 돌아간다.



(그림 3) 혼잡 제어 지연 알고리즘

(그림 4)는 간단히 연속적인 에러 발생시 기존의 Snoop과 향상된 Snoop의 차이점을 보이고있다. 기존의 Snoop은 연속적인 에러가 발생하면 그대로 송신측 쪽에서 타임아웃이 걸리게 되고 혼잡 제어가 일어나게 되나 향상된 Snoop에서는 송신측 쪽의 타임아웃을 멈추게 하고 혼잡 제어를 지연시킨다.



(그림 4) 향상된 Snoop과 기존 Snoop의 비교

이러한 방법은 이미 M-TCP에서 적용된 방법과 기본적으로 동일하다[8]. 이러한 방법으로 연속적인 에러에 따른 장시간의 연결 단절에도 급격한 속도저하를 막을 수 있다.

<표 2>는 프로토콜별 성능 비교를 보여주고 있다. 기존의 Snoop이 미비했던 점을 보완하여 다른 프로토콜에 비해 좋은 성능을 내는 것을 기대할 수 있다.

〈표 2〉 다른 Mobile TCP와의 비교

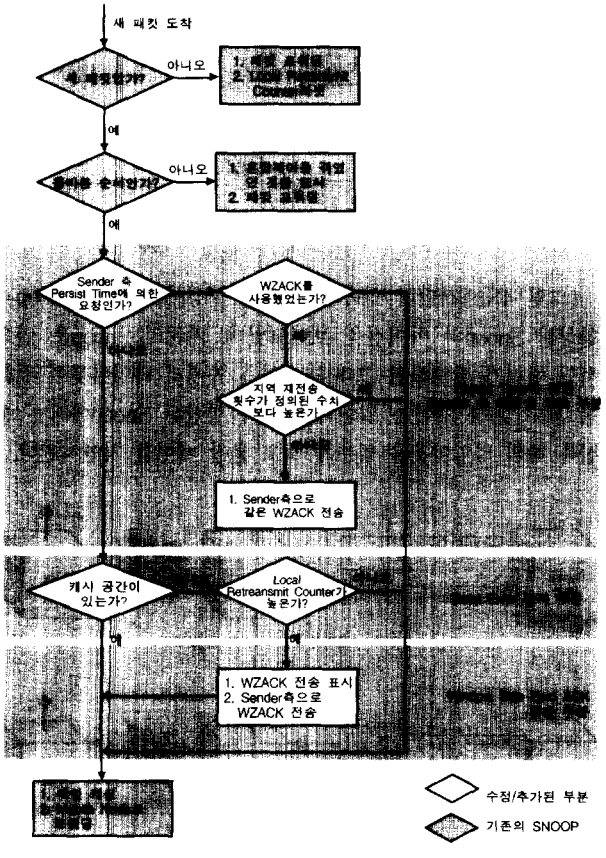
	I-TCP & MTCP	M-TCP	Snoop	향상된 Snoop
End-to-End Semantics		○	○	○
빈번한 에러에대한 처리	○	○	○	○
장시간 연결 단절		○		○
잦은 연결 단절	△	○		○
Selective ACK 이용			○	○

4. 구 현

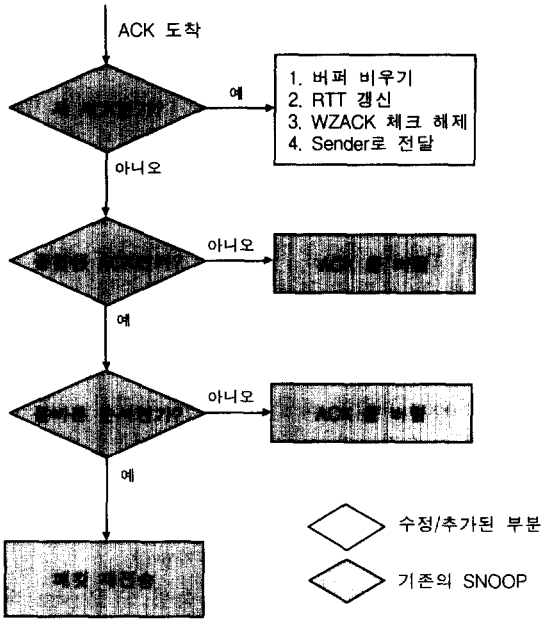
4.1 향상된 Snoop

본 논문의 구현은 기존의 Snoop 개발자와는 별도로 본 논문에서만 언급되는 기능향상을 위한 독자적인 결과이다.

현재 Snoop 프로토콜은 http : //www.cs.berkeley.edu/~hari/software/snoop/ 에서 BSD 2.X 또는 3.X 버전에서 동작하는 C 소스를 배포하고 있다. 배포되고 있는 Snoop 소스는 snoopsrc.tar.gz 라는 파일명으로 배포되고 있다. 소스에서 inet 디



(그림 5) 향상된 Snoop 흐름도 : 데이터 패킷 전달 처리



(그림 6) 향상된 Snoop 흐름도 : ACK 패킷 처리

렉토리에 있는 snoop.h와 snoop.c에 설계에서 설명한 알고리즘을 적용하여 코딩 하였다.

그러나 실제적으로 구현할 수 있는 이동망 환경이 구축이 되어 있지 않아 구현에서는 Snoop 소스 상에서 수정된 부분과 추가된 코드에 대한 설명으로 마치고 실험결과는 Network Simulator(NS)의 시뮬레이션을 통해 산출해내었다.

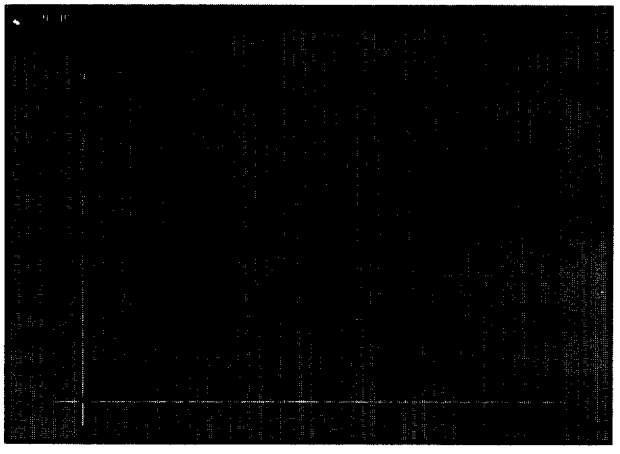
(그림 5)와 (그림 6)은 각각 기존의 Snoop 알고리즘에서 추가/수정된 부분에 대한 설명을 흐름도를 통해 보여주고 있다.

5. 실험 결과

실험은 네트워크 시뮬레이션 툴로 많이 사용되는 Network Simulator(NS)를 이용하여 에러 상황을 시뮬레이션했다.

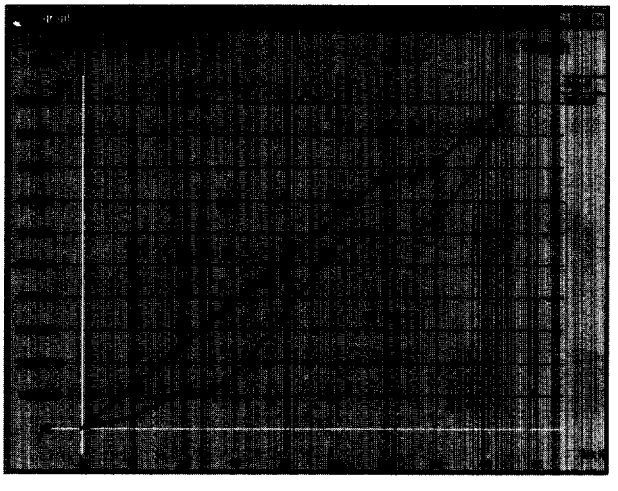
실험 환경은 Linux Kernel 2.2.13 상에서 NS2 1.5 b를 설치하여 컴파일하였다. 전송 속도를 250kb/sec으로 두고 TCP의 최대 세그먼트 크기(MSS : Maximum Segment Size)를 512Bytes로 두었다. 기본적인 전송지연은 50ms로 하였고 Snoop의 캐시 버퍼의 크기는 32KBytes로 사용하여 1M Bytes의 데이터를 전송하는 것으로 했다. 반송되는 쪽의 ACK이 손실되는 경우는 본 논문에서 고려하지 않은 부분이기 때문에 반송되는 쪽에 대한 패킷은 손실되는 경우가 없는 것으로 가정하였다. 그리고 처음 3.5초간 전송을 중단시키고 두 번째로 7초간 중단시킴으로써 연속적인 에러의 상태를 시뮬레이트 하였다.

실험 결과는 기존의 Snoop과 향상된 Snoop의 Sequence number와 송신측 윈도우 크기를 Xgraph로 출력한 결과 그래프로 표현하였다. (그림 7)는 기존의 Snoop을 그대로 적용한 경우의 그래프를 보이고 있고 (그림 8)은 향상된 Snoop을 이용한 경우의 그래프를 보이고 있다.



(그림 7) 기존의 Snoop 동작

붉은선(진한색선)이 전송된 데이터의 바이트수를 나타내고 녹색선(옅은색선)은 송신측의 윈도우 크기의 증감을 보여주고 있다. (그림 7)에서 나타나듯 기존의 Snoop의 경우 처음 3.5초 간의 에러는 국부적 재전송으로 처리하여 별다른 윈도우 사이즈의 크기 변화 없이 안정적인 데이터 전송을 수행하지만 7초간의 연속적인 에러의 경우 일정 수준까지는 윈도우 크기를 유지하지만 곧 송신측 측의 혼잡 제어로 인해 절반씩 줄어들게 된다. 그래서 혼잡 제어로 인해 절반으로 줄어든 윈도우 사이즈 때문에 이후 전송속도가 떨어지는 것을 확인할 수 있다.



(그림 8) 향상된 Snoop 동작

(그림 8)은 향상된 Snoop을 이용한 결과치를 보여준다. (그림 7)에서는 7초간 발생한 연속적인 에러에서 급격한 윈도우 크기의 하락을 보여주는 반면 향상된 Snoop은 WZACK을 이용해서 송신측 쪽의 전송을 중단시킴으로써 혼잡 제어를 지연시켜 장시간의 연속적인 에러에도 윈도우 크기가 줄어드는 것을 억제하였다. 수 초후 에러가 나지 않는 시점에서는 기존의 윈도우 크기가 그대로 유지되기 때문에 계속해서 빠른 속도로 전송이 가능함을 확인할 수 있다.

<표 3>은 실제적으로 1MBytes의 데이터를 전송하는데 걸리는 시간을 보여주고있다. 연속적인 에러가 발생하지 않는 경우에는 별다른 차이가 없지만 연속적인 에러가 발생하는 경우에는 향상된 Snoop이 기존의 Snoop 비해 보다 빠른 전송 효율을 보임을 확인할 수 있었다.

〈표 3〉 전송시간 비교표 (1Mbytes 전송, 250kb/s)

종 류	3.5초/7초간 연속적 에러		15초간 연속적 에러	
	평균 전송 속도(kb/s)	전송시간(초)	평균 전송 속도(kb/s)	전송시간(초)
기존의 Snoop	166.84	49.10 ± 0.71	148.28	55.27 ± 0.46
향상된 Snoop	190.78	42.94 ± 0.42	173.05	47.34 ± 0.56

위의 결과를 종합해볼 때 연속된 에러 감지 알고리즘과 혼잡 제어 지연 알고리즘을 이용한 향상된 Snoop은 기존의 Snoop에 비해 연속적인 에러가 일어나는 상황에서 보다 나은 성능을 보여주는 결과를 확인할 수 있었다.

6. 결론 및 향후과제

본 논문은 이동망 환경에서 향상된 성능을 보이기 위해 기존의 Snoop이라고 하는 프로토콜을 변형하여, 기지국과 이동 노드에서의 최소한의 변경만을 하는데 목표를 두었다. 고정된 호스트쪽의 하부 TCP 라이브러리의 변경없이 성능 향상을 꾀했던 Snoop의 장점은 그대로 살린 채 연속적인 에러에 대한 방안을 제안하였다. 이 방법은 기존의 망에 대한 어려운 작업 없이 바로 이동망 환경에 적용할 수 있는 기술로 이용될 수 있다.

앞으로 이동망 환경이 컴퓨팅 환경에서 점점 더 많은 비중을 차지하게 되어감으로써 관련된 성능향상에 대한 연구들이 지속적으로 진행이 될 것으로 생각된다. 그리고 각 부분에서 고정된 호스트쪽도 많은 개선이 이루어지게 되고 새로운 버전의 TCP에서 더 나은 부분들을 이룩할 수 있게 될 것이다.

향후 연구 과제로는 여전히 문제점으로 자리잡고 있는 근본적인 이동망 환경에 대한 대안이 지속적으로 연구되어야 할 것이다.

참 고 문 헌

[1] 강현국, 김대영, 안상현, 한선영, Comer TCP/IP 인터네트워킹, 그린출판사, Mar, 1997.

[2] Jorge, A. Cobb., and Prathima. Agrawal., Congestion or Corruption? A Strategy for Efficient Wireless TCP Sessions, IEEE Symposium on Computers and Communications, 1995.

[3] Hari Balakrishnan, Venkat Padmanabhan, Srinivasan Seshan, and Randy H. Katz., A Comparison of Mechanisms for Improving TCP Performance over Wireless Links, IEEE/ M Transactions on Networking, December 1997.

[4] Ray. avatkar., and Namrata. Bhagawat., Improving End-to-End Performance of TCP over Mobile Internetworks, Proceedings IEEE Workshop on Mobile Computing Systems and Applications, December 1994.

[5] Ajay V. Bakre., and B. R. Badrinath., I-TCP : Indirect TCP for Mobile Hosts, Proc. 15th Int'l Conf. on Distributed Computing Systems, May 1995.

[6] Hari. Balakrishnan, Srinivasan Seshan, and Randy H. Katz. Improving Reliable Transport and Handoff Performance in Cellular Wireless NetworksACM Wireless Networks, Vol.1, No.4, December, 1995.

[7] S. Paul, e. Ayanoglu, T. F. LaPorta, K. H. Chen, K. K. Sabnani, and R. D. Girlin., "An Asymmetric Link-Layer Protocol for Digital Cellular Communications," In Proc. InfoComm 95, 1995.

[8] Elan Amir, Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz., "Efficient TCP over Networks with Wireless Links," Proc. Fifth Workshop on Hot Topics in Operating Systems (HotOS-V), Orcas Island, WA, May, 1995.

[9] K. Brown and S.Singh, M-TCP : TCP for Mobile Cellular Networks, ACM Computer Communication Review, 27 May, 1997.

[10] V. Jacobson., and R. Braden., "TCP Extensions for Long-Delay Path," Request fro Comments RFC 1072, Network Working Group, October 1988.

[11] Sally, Floyd., "Issues of TCP with SACK," Very rough draft, March 9, 1996.

[12] W. Richard Stevens. TCP/IP Illstrated, Vol.The Protocols. Addison-Wesley. February, 1998.

[13] Bikram S Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan. Improving Performance of TCP over Wireless Network, Proceedings of the 17th International Conference on Distributed Computing Systems, 1997.

[14] Ramon Caceres and Liviu Iftode. Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments, IEEE Journal of Selected Areas in Communications, Vol.13, No.5, June, 1995.

[15] 황인용, TCP Timers. http : //ccl.chungnam.ac.kr/~iyhwang/project/misc/docs /tcptimer.html.

[16] Tom Goff, James Moronski, Dhananjay S. Phatak and Vipul Gupta. Freeze-TCP : A True End to End TCP Enhancement Mechanism for Mobile Environments, IEEE INFOCOM, 2000.

[17] NS 시뮬레이션 소스, http : //cclab.konkuk.ac.kr/~kimyong/mobiletcp/snoop/ns/.

김 용

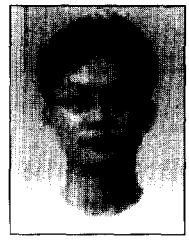
e-mail : kimyong@kimyong.pe.kr

1998년 건국대학교 전자계산학과 졸업 (학사)

2000년 건국대학교 대학원 컴퓨터정보 통신공학과 (공학석사)

2001년~현재 건국대학교 대학원 컴퓨터정보통신공학과 박사과정 재학중

관심분야 : IPv6, Mobile IP, 차세대네트워크



성 호 철

e-mail : bullyboy@kkucc.konkuk.ac.kr

1998년 건국대학교 전자계산학과 졸업 (학사)

2000년 건국대학교 대학원 컴퓨터정보통신공학과 (공학석사)

2001년~현재 건국대학교 대학원 컴퓨터정보통신공학과 박사과정 재학 중

관심분야 : IPv6, Mobile IP, 차세대네트워크

현 호 재

e-mail : hjhyun@cclab.konkuk.ac.kr

1997년 동국대학교 전자계산학과 졸업 (학사)

1999년 건국대학교 대학원 컴퓨터정보통신공학과 졸업 (공학석사)

1999년~현재 건국대학교 대학원 컴퓨터정보통신공학과 박사과정 재학 중

관심분야 : CORBA, Mobile IP, IPsec, Differential service

한 선 영

e-mail : syhan@cclab.konkuk.ac.kr

1977년 서울대학교 계산통계학과(학사)

1979년 한국과학기술원 전산학 석사

1988년 한국과학기술원 전산학 박사

1981년~현재 건국대학교 컴퓨터공학과 교수

1989년~1990년 미국 Maryland대 컴퓨터 과학과 객원부교수

1990년~현재 개방형 컴퓨터통신 연구회 이사

1990년~1997년 한국과학기술원 인공지능 연구센터 참여교수

1991년~1993년 한국 정보과학회 정보통신연구회 부위원장

1996년~1998년 개방형 컴퓨터통신 연구회 총무이사

1998년~1999년 미국 Maryland 대학교 컴퓨터 과학과 객원교수

2000년~현재 건국대학교 정보통신원 원장

관심분야 : Real-Time CORBA, Internet Caching, 차세대 인터넷 프로토콜, Mobile IP