연속적인 패킷 손실 제어와 제로 윈도우 제어를 이용한 무선 TCP 전송 성능 향상 연구

김 성 찬[†] · 전 문 석[††]

요 약

기존의 가장 널리 쓰이는 전송계층 프로토콜인 TCP는 패킷 손실의 원인이 망의 혼잡 때문이라는 가정 하에 설계된 프로토콜로서 유선망과 고정 호스트로 이루어진 전통적인 네트워크에는 적합하지만 페이딩, 잡음, 간섭 등의 전송 에러가 발생하는 무선 환경에서는 전송 프로토콜로서 적용하기에 비효율적이다. 이것은 무선망에서의 비트 에러에 의한 패킷손실 역시 혼잡으로 간주하여 불필요한 전송 제어가 발생하기 때문이다. 본 논문에서는 무선 TCP 패킷의 전송성능을 향상시키기 위하여 연속적인 패킷 손실에 대한 제어와 TCP 윈도우 제어를 하여 불필요한 혼잡제어 알고리즘을 수행하지 않게 하는 개선된 SNOOP 프로토콜을 제안 하였고 개선된 모듈의 성능을 모의실험을 통하여 확인 하였다.

키워드 : SNOOP, TCP, 혼잡제어, Mobile Network

A Study of efficient Wireless TCP Transmission Using Consecutive Packet Loss and Zero Window Control

Sung-Chan Kim[†] · Moon-Seog Jun[††]

ABSTRACT

The conventional transport layer protocol TCP is designed to work under condition of packet loss is due to the network congestion, so that it's suitable in the traditional wired network with fixed hosts but it's inefficient on the wireless network where the environment of fading, noise, and transmission error comes from interference. This result from the needless transmission control of the bit error is due to treats the packet loss as a packet congestion control in the wireless network. In this paper, we propose the advanced SNOOP protocol with the consecutive packet loss and TCP window control to avoid the needless congestion management algorithm in wireless network for the wireless TCP packet transmission enhancement. We verify the performance of the advanced module from the simulation experiment result.

Key Words : SNOOP, TCP, Congestion Control, Mobile Network

1. 서 론

최근 무선 통신 기술의 발달로 모바일 컴퓨팅 환경에 대한 관심이 나날이 높아지고 있다. 특히, 무선 통신 이용자가 장소에 구애받지 않고 어느 때나 원하는 정보를 주고받을 수 있다는 것은 상당히 매력적이다. 오늘날의 인터넷은 대부분이 유선망에 기반을 두고 있으며 이러한 유선망에 무선망을 도입한 유·무선 통합망의 구현이 현실화되고 있다. 유·무선 통합 망에서 우선적으로 직면하는 문제는 단말기의 이동성이며, 이에 따라 어드레싱과 라우팅 기능을 보강하여 단말기의 이동성에 대응할 수 있는 이동 네트워크에 대한 연구가 최근 깊이 이루어져 왔다. 그러나 이러한 네트워크 계층에 대한 문제 이외에 신뢰성 있는 연결을 보장하기 위한 TCP(Transmission Control Protocol)의 유효성에 대한 검증은 해결해야 할 과제로 남아 있다. 현재 통신 환경에서 널리 사용되고 있는 TCP는 유선망과 고정 호스트(FH: Fixed Host)로 이루어진 전통적인 네트워크에 적합하며 유선망의 특성을 적용한 것이기 때문에, 무선 구간의 제한된 대역폭, 높은 지연, 산발적인 비트에러, 일시적인 연결 두절 및 핸드오프 등과 같은 특징을 가진 무선망에 이를 그대로 적용하게 되면 불필요한 메커니즘의 호출로 인해 성능 저하를 가져온다[1]. 이러한 성능 저하는 TCP가 유선망의 낮은 비트 에러율 때문에 통신상에서 발생하는 패킷 손실을 기본적으로 혼잡에 의한 것으로 생각하여 혼잡 제어 메커니즘으로 패킷 손실을 처리하기 때문이다. TCP 송신자는 패

† 정 회 원:(주)유코레일 전산실 차장
†† 종신회원:숭실대학교 정보과학대학 정교수
논문접수:2006년 9월 1일, 심사완료 : 2006년 11월 17일

킷 손실을 발견하면 먼저 전송 윈도우 크기를 줄이고 손실된 패킷을 재전송한다. 또한 혼잡 제어나 회피 메커니즘을 초기화 하고 재전송을 위한 타이머의 값을 증가시킨다. 하지만 무선망에서의 패킷 손실은 대부분 혼잡에 의해서가 아니라 핸드오프 시 발생하는 이동 호스트(MH: Mobile Host)와 기지국(BS: Base Station)간의 연결 끊김과 지연 시간, 그리고 무선망 자체의 비트 에러율에 의해서 발생한다. 그러므로 앞에서 언급한 TCP의 패킷 손실 메커니즘을 그대로 무선 네트워크에 적용시키면 오히려 불필요한 함수 호출로 인해 처리율의 성능 저하를 가져오게 된다[2].

본 연구에서는 이러한 무선망에서의 TCP의 패킷 처리 메커니즘을 향상시키기 위하여 지역 재전송 방식인 SNOOP 프로토콜을 이용하였고, 기존 연구에서 제기된 SNOOP 프로토콜의 단점으로 지적되어 왔던 연속적인 패킷 손실 발생 시 지속적인 타임아웃이나 재전송 타임아웃이 발생할 때 까지 패킷을 재전송할 수 없는 단점과 불필요한 혼잡제어로 인한 망의 효율을 떨어뜨리는 단점을 연속적인 패킷 손실에 대한 제어와 TCP 윈도우 제어를 이용하여 전송 성능을 향상시키도록 SNOOP 프로토콜을 수정하였고, IP 커널 단계에서 구현함으로써 기존의 SNOOP 프로토콜보다 성능을 향상 시켰다. 본 논문의 구성은 다음과 같다. 2장에서는 본 연구의 기초가 되는 관련연구에 대하여 기술하고, 3장에서는 본 논문에서 제안한 연속적인 패킷 손실 처리와 TCP 윈도우 제어를 이용한 개선된 무선 패킷 처리 기법 설계와 시스템 구현에 대하여 기술한다. 4장에서는 구현된 시스템에 대한 사용 결과로서 무선 패킷 메시지와 속도를 분석하고 모의실험으로 성능 평가를 한 후, 5장에서 결론을 맺는다.

2. 관련연구

무선망에서의 패킷 처리 성능을 향상시키기 위한 여러 가지 연구가 이루어지고 있는데, 이런 연구의 결과로서 다음과 같은 기법들이 있다.

2.1 지역 재전송 방안[3, 11]

지역 재전송 방안에는 TULIP(Transport Unaware Link Improvement Protocol)[3]과 Delayed DupAck 기법[3]이 있다. TULIP은 캘리포니아 대학에서 설계 되고 구현 되었으며 전송계층 이나 네트워크 계층 프로토콜의 별도 수정이나 계산이 필요 없고, 무선 링크 위에 TULIP을 올려 TCP 전송 성능을 향상 시키는 방법이다. TULIP에서는 이동 호스트의 TULIP 모듈에서 패킷의 순서를 맞춰 줌으로써 무선망에서 비트 에러로 인한 패킷 전송 순서 오류의 발생을 막아준다. 즉 기존의 방안에서 TCP계층으로 전송해 주기 때문에 패킷이 전송 순서 오류 일 때 패킷 손실을 즉시 알아차리게 되지만, TULIP에서는 순서가 제대로 될 때 까지 기다려 TCP 계층에서는 항상 정상 순서의 패킷 데이터만 전송되어 효율적인 결과를 가지게 한다. TULIP 프로토콜의 장점은 중간에 별도로 프록시가 필요하지 않으며 높은 비트 에러 시에 효율적으로 처리하여 좋은 처리량을 보여준다. 또한 링크 계층에서 구현되기 때문에 TCP, UDP에서 모두 사용될 수 있다. 단점으로는 링 계층의 재전송이 반 이중 전송(Half Duplex radio channel)에서만 최적화 되어 있다[4].

Delayed DupAck은 SNOOP 프로토콜과 유사하나 베이스 스테이션의 링크 계층이 TCP 프로토콜의 상태정보를 알 필요가 없이 링크 계층 재전송을 수행하는 방식이다. 동작 원리를 살펴보면, 재전송은 베이스 스테이션에서 수행하고 이동 단말기에서 중복 Ack 패킷을 지연 전송함으로써 TCP와 링크 계층 재전송간의 간섭을 줄인다. 이동 단말기는 비정상적 순서의 패킷을 수신할 때 일정 시간동안 3번째 DupAck을 지연시킨다. 이와 같이 3번째 DupAck을 지연시키는 이유는 그 시간 동안 링크 레벨의 재전송이 성공되기를 기다리며 고정 호스트의 TCP단계의 재전송을 막기 위한 것이다. 따라서 지연된 DupAck은 TCP 고정 호스트로부터 반응을 초래하지 않고 전송 에러를 복구 시킬 수 있다. 지연된 DupAck은 일정 시간간격 동안 손실 패킷이 수신되지 않으면 고정 호스트로 3번째 DupAck이 바로 전송되어 빠른 재전송을 시작하도록 한다. Delayed DupAck의 지연시간의 적정 값은 무선 링크의 속성에 좌우되므로 지연시간 값을 자동으로 선택할 수 있는 메커니즘이 필요하다. 또한 통신망의 과부하로 인한 패킷 에러 발생 시에 이에 대한 복구가 지연되는 단점을 갖는다[5].

2.2 단 대 단 빠른 재전송(End to End Fast retransmission)[4, 11]

기존의 TCP 에서는 핸드오프 시 패킷 손실이 있으면 네트워크의 상태가 좋아질 때 가지 일정시간(재전송 타임아웃)동안 기다려야 했다. 하지만 패킷 손실의 원인이 혼잡에 의한 것이 아니므로 재전송 타임아웃동안 기다리는 것은 무의미하기 때문에 핸드오프가 완성된 후 곧바로 고정 호스트는 손실된 패킷을 이동 단말기에게 재전송함으로써 지연 시간을 줄이고자 했다. 이미 TCP Tahoe 등에서는 빠른 재전송을 구현하고 있지만, 이 프로토콜에서는 단지 이전의 TCP와 이동 IP에 약간의 수정을 가하여 핸드오프가 완성되었음을 고정 호스트에게 알리고, 고정 호스트는 그 즉시 빠른 재전송을 호출하게 된다. 그러므로 지연 시간이 길어질수록 기존의 TCP보다 더 나은 성능을 가져온다. 즉, 패킷 손실이 혼잡에 의한 것인지 이동에 의한 것이지를 구분하여, 혼잡에 의한 것이면 이전의 프로토콜을 적용하고, 이동에 의한 것이면 이에 해당하는 프로토콜을 적용하여 이동 컴퓨팅 환경에 더 나은 성능을 제공한다[6].

2.3 연결 분할 방안(Split Connection)[4, 11]

연결 분할 방안의 대표적인 방법으로 I-TCP(Indirect TCP) 기법이 있다. 이동 단말기가 고정 호스트와 통신을 하는 경우 무선 네트워크에서는 이동 단말기의 이동성과 무선 링크의 높은 데이터 손실률에 의해 전송 성능 저하가 발생하게 되고 유선 네트워크에서는 데이터의 폭주에 의해 전송 성능저하가 발생한다. 이와 같이 유선네트워크와 무선네트

워크의 성질이 다른 점에 착안하여 유선네트워크와 무선네트워크의 데이터 전송정책을 분할하여 적용하려는 연구가 진행되어 왔다. I-TCP에서는 베이스 스테이션(BS)을 MSR(Mobile Support Router)로 언급하고 있다. 이동 단말기는 MSR과 무선 TCP를 이용해서 연결하고, MSR은 고정 호스트와 기존의 TCP를 이용하여 연결한다. 이러한 I-TCP의 장점은 이동 단말기가 핸드오프 시 MSR간의 소켓정보 전송을 통해 소켓의 재설정이 필요하지 않고, 무선 링크에 적합한 TCP를 사용함으로써 전송 성능을 향상시킬 수 있다는 점이다. 그러나 I-TCP의 치명적인 단점은 전송 계층의 연결 분리로 인해 TCP의 종단 간 작동의미(End-To-End semantic)에 위배된다는 것이다. 이것은 고정 호스트에서 이동 단말기로 TCP 패킷을 전송할 때 TCP ACK이 이동 단말기가 아닌 MSR에서 발생함으로써 MSR의 고장이나 오랜 시간의 연결두절 시에는 중대한 문제가 초래된다. 또한 I-TCP는 새로운 소켓 인터페이스가 적용되므로 기존의 인터넷 어플리케이션은 재 컴파일이 요구되며, 이동 단말기의 핸드오프가 발생하면 이전 MSR과 현재 MSR의 전송층간에 연결정보를 유지하고 전송해야 하는 오버헤드도 존재한다. 그러므로 FTP, WWW등의 어플리케이션 계층에서 ACK를 제공하는 경우에만 I-TCP가 적합하고, TELNET등의 전송계층에서 ACK를 사용하는 것은 기존의 일반적인 TCP가 적합하다[7, 8].

3. 연속적인 패킷 손실 제어와 TCP윈도우 제어 설계

패킷 Snooping기법은 TCP의 종단간의 연결 개념을 유지할 수 있다는 장점이 있으나, SNOOP이 무선망에서 발생한 패킷 손실을 지역 재전송에 의해 복구하는 동안에 TCP의 고정 호스트는 Ack 패킷을 기다려야 한다. 이로 인해 고정 호스트에서 휴지 시간이 발생하여 유선망의 자원을 낭비하게 되며, 연속적인 패킷 손실이 발생했을 경우 기지국에서의 지역적 패킷 손실 복구에 걸리는 시간이 지나치게 길어질 때 무선망의 패킷 손실이 숨겨지지 못하고 지속적인 타임아웃과 재전송 타임아웃을 기다려야 하는 문제점이 발생한다. 연속적인 패킷 손실 발생 시 지속적인 타임아웃을 기다려야 하는 문제와 고정 호스트에게 망의 상태를 타임아웃으로부터 완전히 숨길 수 없는 단점을 개선하기 위해서 CPLC(Consecutive Packet Loss Control) 기법[9]과 ZWC(Zero Window Control)[10]을 고려하였다.

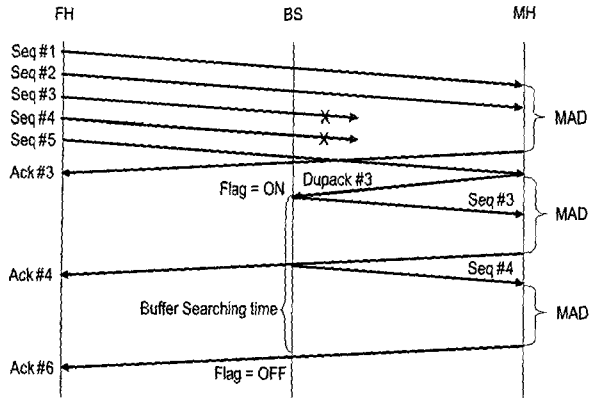
3.1 연속적인 패킷 손실 제어(CPLC: Consecutive Packet Loss Control)

연속적인 패킷 손실 제어는 현재 버퍼에 캐싱 되어 있는 패킷중 가장 앞에 있는 패킷에 대해 중복 Ack 패킷을 받은 상태인지 아닌지에 무선 네트워크의 전송 상태를 비정상과 정상 상태의 두 가지로 분류하여 패킷을 처리하도록 한다. 정상 상태에서는 정상적인 패킷 Snooping을 하도록 하고, 비정상 상태에서 베이스 스테이션의 버퍼에 캐싱된 패킷들에 대하여 연속적인 패킷 손실 제어를 한다. (그림 1) 은 베이스 스테이션에서의 연속적인 패킷 손실에 대한 처리 알고리즘을 보여준다.

처음 Ack 패킷이 베이스 스테이션에 도착하면 새로운 Ack 패킷인가를 확인하고 새로운 Ack 패킷이고 정상 상태라면 기존의 패킷 Snooping과 마찬가지로 해당 데이터 패킷을 버퍼에서 삭제하고 해당 무선 네트워크의 RTT를 계산한다. 이후 정상적으로 수신한 Ack 패킷을 FH로 전송한다. (그림 1) 알고리즘에서 FLAG는 연속적인 패킷 손실에 대한 처리 여부를 나타낸다. 이동 단말기에서 전송된 Ack 패킷이 새로운 Ack패킷이 아니라면 우선 버퍼에서 이미 삭제된 데이터 패킷의 Ack 패킷인지 확인하고 맞으면, 전송된 Ack 패킷은 늦게 도착한 패킷이기 때문에 그냥 버린다. 삭제된 데이터 패킷의 Ack 패킷이 아니면, 첫 번째 중복 Ack 패킷인가를 확인하고 첫 번째 중복 Ack 패킷이면 손실된 데이터 패킷을 이동 단말기로 지역 재전송 해주고 이후 중복 Ack의 수를 계산한다. 그리고 이 중복 Ack 패킷은 버린다. 현재상태가 연속적인 패킷 손실에 대한 제어상태임을 표시하기 위해 FLAG값을 ON으로 하여 비정상 상태로 표시하고 last_seq 변수에 현재 버퍼에 캐싱된 마지막 데이터 패킷의 seq 번호를 저장한다. 전송된 ack 패킷에 대해 버퍼를 검색하고 재전송함으로써 연속적인 손실에 대비하도록 하고, 무선 네트워크 전송 상태가 비정상상태(FLAG=ON)에서는 정상 상태(FLAG=OFF)가 될 때까지 Ack패킷에 대한 즉각적인 로컬 재전송이 발생한다. 이는 Ack 패킷이 도착할 때마다 FLAG=ON 시점에서 기억된 last_seq 값을 현재 전송된 Ack패킷의 seq보다 작은지를 비교하고, 작을 때, 정상상태(FLAG=OFF)로 바꾸어 준다.

```
ACK PACKET ARRIVES
IF ACK PACKET == NEW ACK PACKET THEN
    IF ACK PACKET IS ABNORMAL && FLAG = ON THEN
        IF FIRST ACK SEQ NO > LAST ACK SEQ NO THEN
            FLAG =OFF; ACK PACKET IS NORMAL;
             WIRELESS RTT UPDATE;
            CLEAR SNOOP BUFFER;
            DELIVER THE ACK PACKET TO SENDER;
        ELSE LOSS PACKET LOCAL RECOVERY;
              WIRELESS RTT UPDATE;
              CLEAR SNOOP BUFFER;
              DELIVER THE ACK PACKET TO SENDER;
        ENDIF;
    ELSE WIRELESS RTT UPDATE;
          CLEAR SNOOP BUFFER;
          DELIVER THE ACK PACKET TO SENDER;
    ENDIF;
ELSE ENDIF;
IF ACK PACKET IS NOT IN SNOOP BUFFER THEN
     DROP THE DELIVERED ACK PACKET;
ELSE ENDIF;
IF ACK PACKET == FIRST DUPACK PACKET THEN
  LOSS PACKET LOCAL RECOVERY;
  CALCULATE THE NUBER OF DUPACK PACKET;
  DROP THE DUPACK PACKET;
  LAST SEQ NO = LAST SEQ NO IN CASHED PACKET IN BUFFER;
ELSE DROP THE DELIVERED ACK PACKET;
ENDIF;
RETURN;
```

(그림 1) BS에서의 연속적인 패킷 손실제어 알고리즘



(그림 2) 연속적인 패킷 손실에 대한 제어[11]

연속적인 패킷의 손실이 발생했을 때 손실되는 패킷에 대한 처리를 고정 호스트와 와 베이스 스테이션, 이동 단말기의 패킷 흐름을 통해 설명하면 (그림 2) 와 같다. 고정 호스트에서 데이터 패킷이 1번부터 5번까지 보내졌고, 이 패킷들이 베이스 스테이션을 지나 이동 단말기로 가는 도중 3번과 4번 패킷이 유실되었다고 하자. 일반적으로 TCP는 지연 Ack 방식을 사용하기 때문에 이동 단말기에서는 MAD(Maximum Ack Delay) 시간 이후에 Ack 패킷을 보내게 된다. 그러므로 이동 단말기에서는 데이터 패킷1과 2를 받고 데이터패킷 3번을 요청하는 Ack 패킷 3번을 보낸다. Ack 패킷 3번이 베이스 스테이션에 도착하면 베이스 스테이션에서는 새로운 Ack 패킷인지에 대해서 확인을 하고 버퍼에 저장된 1번과 2번 데이터를 삭제한 후 송신자에게 전달한다. 이후 이동 단말기에 데이터 패킷 5번이 도착하면 순서에 어긋난 패킷임을 감지하고 중복 Ack 패킷 3번을 보내게 된다. 중복 Ack 패킷 3번이 베이스 스테이션에 도착하면, 중복 Ack 패킷임을 감지하고 무선망이 비정상 상태임을 알리는 FLAG를 ON 시키고, 동시에 버퍼에 들어있는 데이터 패킷 3번을 지역 재전송 하게 된다. 이후에 베이스 스테이션에 들어오는 Ack 패킷에 대해서는 FLAG ON 되어있는 동안에는 즉각적인 지역 재전송을 하도록 한다. 연속적인 패킷 손실에 대한 처리에서 중요한 부분은 첫 번째 중복 Ack 패킷에 대한 고려이다. 기존의 TCP 패킷 Snooping에서는 고정 호스트에서 전송하는 패킷이 없어 교착 상태가 되었을 때, Ack 패킷을 받지 않는 데이터 패킷이 Snoop 버퍼에 캐시 되어 있어도 지속적인 타임아웃이나 타임아웃이후에나 재전송이 가능했다. 이러한 불필요한 대기시간을 줄이기 위해, 연속적인 패킷 손실 제어 에서는 첫 번째 중복 Ack 패킷을 받았을 때 Snoop 버퍼에 있는 데이터 패킷을 검색할 수 있도록 한다. 이것은 FLAG의 설정을 통해 버퍼의 검색 여부를 가름 하도록 하였다.

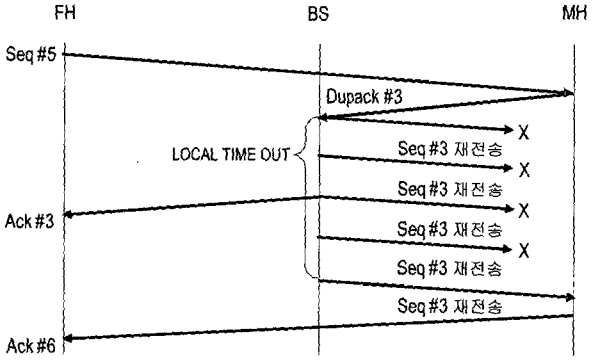
3.2 제로 윈도우 제어

본 연구에서 사용하는 TCP 패킷 Snooping 기법은 무선망에서 발생하는 패킷 지연에 대해서 유선망에서는 네트워크 상태가 좋아도 무선망에서 발생하는 혼잡에 대한 타임아웃을 막을 수 없는 단점이 있다. 고정 호스트에서는 무선망에서의 지연에 의해 오랫동안 Ack 패킷을 받지 못하고, 결국 타임아웃이 되어 TCP 윈도우 크기를 줄이고 불필요한 혼잡제어를 수행하게 된다. 또한 지역 재전송 중에도 불구하고, 고정 호스트로부터의 중복된 재전송이 일어나는 간섭의 문제가 발행한다. 이러한 문제점을 해결하기 위해 제로 윈도우 제어 기법을 도입하였다.

```
PACKET DELIVERY TIME OUT
SNOOP ACK PACKET RETRANSMITTION
IF CURRENT NO OF RETRANSMITTION < MAX NO OF RETRANSMITTION
    THEN LOSS PACKET LOCAL RECOVERY TO MH;
  IF MAX ZWC < ZWC < MIN ZWC
    THEN WINDOWS SIZE=0; DELIVER THE ZWC ACK PACKET;
    ZWC SEQUENCE UPDATE;
    IF ACK PACKET == FIRST ACK PACKET
      THEN STORE THE ACK PACKET IN ZWC SKB;
    ELSE ZWC SEQUENCE UPDATE;
        SNOOP ACK TRANSMITTION;
    ENDIF;
  ELSE SNOOP ACK TRANSMITTION;
  ENDIF;
ELSE SNOOP ACK TRANSMITTION;
ENDIF;

RETURN;
```

(그림 3) 제로 윈도우 제어기법



ZWC(ZERO WINDOW CONTROL): FH의 불필요한 간섭인 TIME OUT 재전송을 방지

(그림 4) 제로 윈도우 제어 시 패킷 흐름[12]

이 기법은 무선 상에서 높은 비율의 에러를 감지했을 경우 베이스 스테이션에서 수신 윈도우 크기가 0임을 알리는 Ack 패킷을 고정 호스트로 전송해 준다. 이것을 통해 고정 호스트에서는 데이터 전송을 보류하고 패킷의 타이머를 중지한다. 이는 타임아웃에 의한 불필요한 혼잡제어와 간섭을 줄여주는 효과가 있다. 데이터 패킷에 대한 Ack 패킷의 전송이 늦어질 경우 고정 호스트에서 타임아웃이 발생하게 되고, 이때 타임아웃이 발생한 패킷에 대해 지역 재전송이 이루어지도록 한다. 먼저 각 패킷 마다 재전송 해 줄 수 있는 횟수가 정해져 있기 때문에 재전송 횟수를 확인하고, 이 횟수가 넘지 않을 때에는 재전송을 처리해주고 이 횟수가 넘게 되면 재전송 해주지 않고 다음 단계로 넘어간다. 다음 단계에서는 윈도우 크기를 0으로 적용할 것인지에 대한 여부를 결정한다. 타임아웃이 최소값을 넘게 되면 윈도우크기가 0이라는 Ack 패킷을 고정 호스트에게 보내주게 되면 고정 호스트에서는 더 이상 패킷을 보내지 않고 타이머를 중지 시킨 채 기다리는 상태가 된다. 이때 정상적인 Ack 패킷

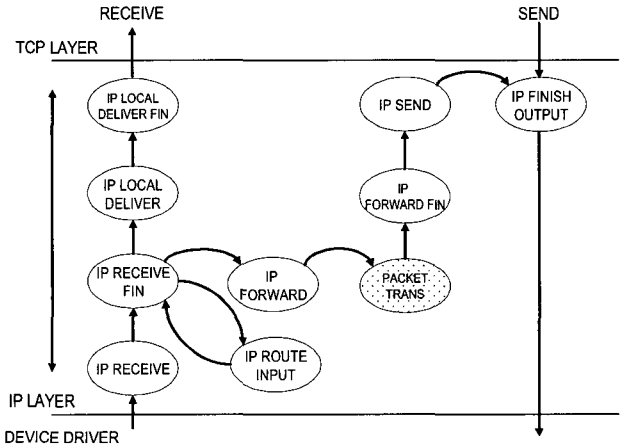
을 받게 되면 윈도우 크기가 0인 상황이 자동으로 해제된다. 제일 중요한 기본 아이디어는 TCP헤더 필드인 윈도우 필드의 값을 0으로 바꾸고, 고정 호스트에게 수신 버퍼가 0임을 알려줄 수 있도록 한다. 그림 4의 패킷 흐름을 보면 중복 Ack 패킷이 3번 도착하였을 때 Snoop의 동작에 의해 베이스 스테이션에서는 지역 재전송을 해주고, 이에 대한 Ack 패킷을 기다리는 타이머를 설정한다. 로컬 타이머가 타임아웃 되면, 베이스 스테이션에서는 패킷 손실이라 생각하고 다시 전송한다. 이때 2번의 로컬 타임아웃이 발생될 때 까지 해당 패킷에 대한 Ack 패킷을 받지 못하면 이동 단말기의 윈도우 크기가 0임을 알리는 Ack 패킷을 고정 호스트에 보내게 되고, 이로 인해 고정 호스트에서의 패킷 재전송을 막음으로써, 고정 호스트와 베이스 스테이션간의 혼잡을 막을 수 있게 된다. 만약 베이스 스테이션과 이동 단말기사이에 일정 횟수 이상 재전송이 일어날 때까지 해당 Ack 패킷을 받지 못하면 자동으로 윈도우 크기가 0인 상태를 벗어나게 함으로써 유선망의 활용도를 높이도록 하였다.

3.3 설계 및 구현

본 연구에서 제안한 방법을 베이스 스테이션 내에서 구현하기 위해서는 운영체제의 커널 내에서 패킷 처리 함수를 호출하는 함수를 구현 해 주어야 한다. 개선된 Snoop 프로토콜[4, 8, 10]을 탑재한 타깃 시스템 사양과 도구는 표 1과 같고, 패킷이 송신단에서 수신단에 도착하여 IP 계층을 거쳐 TCP 계층에서 처리되는 과정을 상태도로 나타내면 (그림 5)와 같다.

〈표 1〉 베이스 스테이션 구현 환경

	Base Station
CPU	Pentium 500 Mhz
Memory	128 M byte
OS	Linux 2.4.30
Tool	GCC
S/W	Berkeley Snoop protocol Software



(그림 5) 패킷 처리 흐름도

```
PACKET ARRIVES AND MEMORY ALLOCATION
IF CONNECTION ID == CONNECTED THEN
  IF CONNECTION ID == SYN THEN
    SNOOP CONNECTION CLEAN;
    SNOOP CONNECTION INITAILIZATION;
  ELSE RETRUN TO TRANSACTION OF NORMAL TCP PACKET;
  ENDIF;
ELSE ENDIF;
IF CONNECTION ID == SYNC THEN
  IF CONNECTION IS FULL THEN
    RETURN TO TRANSACTION OF  NORMAL TCP PACKET;
  ELSE SNOOP CONNECTION INITIALIZATION;
ELSE ENDIF;
IF CONNECTION ID == RST FIN THEN
  SNOOP CONNECTION CLEAN;
ELSE ENDIF;
IF CONNECTION ID == DATA THEN
  SNOOP DATA PACKET TRANSACTION;
ELSE ENDIF;
IF CONNECTION ID == ACK THEN
  SNOOP ACK PACKET TRANSACTION;
ELSE ENDIF;
RETURN;
```

(그림 6) 패킷처리 알고리즘

인터넷에 연결된 호스트에서 패킷이 오면 드라이버는 이를 받아서 메모리에 저장한 후 소프트 IRQ를 발생시킨다. IRQ 핸들러에서 이 패킷을 받아 IP로 올릴지 아니면 ARQ, RARQ로 처리해야 할지를 결정한 후 IP로 올릴 데이터라면 IP 수신 함수를 호출하게 된다. 이 함수에서 몇 가지 처리를 한 후 IP 수신을 종료하고 패킷이 자신에게서 온 것인지 아니면 다른 호스트로 향하는 것인지를 결정하게 된다. 본 연구 에서는 무선망으로 향하는 패킷인 경우에 포워딩의 완료 이전에 Snoop 버퍼에 데이터 패킷 캐싱 작업이나 Ack 패킷에 대한 선별 작업을 하여 중복 Ack 패킷을 줄여주는 모듈을 구현 하였다. (그림 5)에서 볼 수 있듯이 IP 포워딩 다음 단계에 Packet_Transaction()이라는 모듈을 두어 본 연구의 3.1과 3.2에서 제안한 연속적인 패킷 손실에 대한 제어와 윈도우 크기를 조정하도록 한다. IP 포워딩 단계에서 패킷을 처리하기 위해 호출하는 Packet Transaction 모듈의 알고리즘은 (그림 6)과 같다. IP 포워딩 단계에서 새로운 패킷이 도착하게 되면, 기존에 연결된 상태에서의 정보인지를 확인하여 새로운 연결일 경우 새로운 Connection ID를 생성하게 하였다. SYN 패킷에 대해서는 연결 배열의 한 엔트리를 할당해 주고 기존에 있던 연결에 대한 SYN 패킷인 경우 기존 연결정보를 해제해 준다. 만약 연결 상태가 FULL 일 경우 연결 요청은 무시하고 기존의 TCP/IP 패킷 처리를 하도록 하였다. RST 패킷이나 FIN 패킷의 경우 기존 연결 정보를 해제 하도록 하고, 데이터 패킷이면 패킷 Snooping의 데이터 패킷 처리를, Ack 패킷일 경우 Ack 패킷 처리를 하도록 한다.

4. 결과 분석

본 연구에서 구현한 모델을 실제 환경에서 테스트를 하는 것은 어려운 점이 많다. 실제로 고정 호스트와 베이스 스테

```
============ Packet Type ============
This packet send from FH -> Data Packet
Packet sequence: 1470017853
Packet size: 193476543
Packet's source ip addr: 4127212341
Packet's destination ip addr: 4345782191
========== Current Butt Sate ==========
last_seq in snoop_buffer: 1781129477
last_ack in snoop_buffer: 1781129477
======= Connection array number ========
Connection id && array num: 0
======= Data packet handling ========
This packet is New data
In order data->perfect new data
data_len=31, update last_seq=1771121317
data packet retransmit -> time out
```

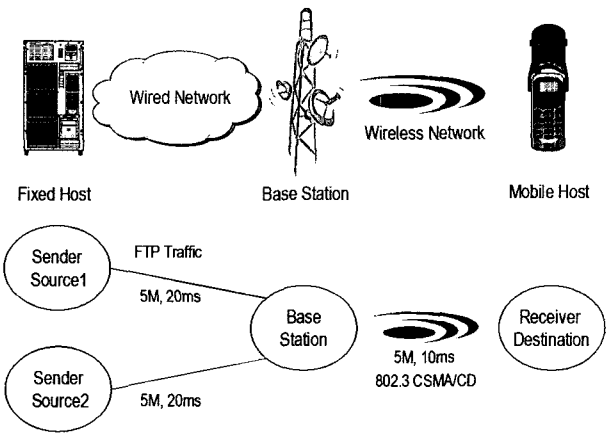
(그림 7) FH에서 MH로 전송되는 데이터 패킷

〈표 2〉 시뮬레이션 파라미터

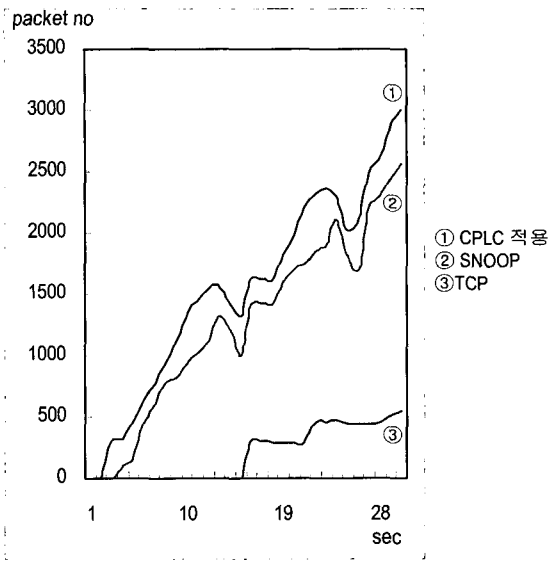
Parameter	Value
F1,F2 TCP Segment Size	1040 bytes
F1,F2 Ack Packet Size	40 bytes
Application Data type	FTP
Transport Protocol	TCP Reno/Sink
Buffer Queue Size	200
Program language	C++/Object Tcl
Simulator	LBNL NS2[13]

이션 이동 단말기를 실험실 환경에서 무선 랜과 유선 랜에 연결하여 패킷 전송 성능을 확인해 본 결과 크게 차이가 나지 않았지만 개선된 Snoop 프로토콜이 동작하는 것은 패킷 캡처를 통해 (그림 7)와 같이 확인 할 수 있다. 커널 메시지 캡처를 통해 베이스 스테이션에 유입되는 패킷의 타입, 시퀀스 번호와 크기, 주소 정보를 확인할 수 있고 각 단계별 타임도 체크 할 수 있다. (그림 7)는 고정 호스트에서 데이터 패킷이 베이스 스테이션에 도착하여 그에 따른 핸들링을 수행하는 것을 커널 메시지로 보여준다. 패킷의 타입은 데이터 패킷이며 패킷의 시퀀스는 1470017853이고 패킷 사이즈는 193476543 이며 데이터 패킷의 핸들링을 통해 마지막 패킷 시퀀스 번호를 업데이트 하는 것을 확인 할 수 있다. 실제 상황에서의 성능 분석이 어려운 이유는 각 노드 간 연결은 1 홉 차이의 직접 연결로서 실제 사용되어질 환경에서 발생 할 수 있는 노이즈, 페이딩, 간섭등과 같은 에러 발생 환경을 임의적으로 줄 수 없었기 때문이다.

따라서 본 연구에서는 실제 상황에서 발생 가능성이 있는 노이즈, 페이딩, 간섭등과 같은 에러율이 높은 무선 환경 하에서 성능을 분석하기 위하여 NS(Network Simulator)를 사용하였으며 시뮬레이션 파라미터는 표 2와 같다. 기존의 구현되어 있는 Snoop 모듈[8, 10, 16]을 적용하여 Snoop TCP에 대한 시뮬레이션을 하였고, 본 연구에서 제안한 방법들은 기존의 Snoop 모듈부분을 변경하여 시뮬레이션 하였고, 시뮬레이션을 위한 네트워크 토폴로지는 아래 (그림 8)과 같다. (그림 8)에서 유선 구간은 LAN으로 설정하였으며, 고정 호스트는 베이스 스테이션과 1 홉으로 연하였고, 무선 구간은 역시 1 홉으로 베이스 스테이션과 이동 단말기를 연결 하였다. 무선구간에 대한 채널은 편의상 IEEE 802.3 CSMA/CD인 유선 이더넷으로 구현 하였다. 이 채널에 에러 모델을 삽입하고, 고정 호스트에서 베이스 스테이션에 연결이 될 때 베이스 스테이션의 LLSnoop Agent[14]로 연결이 되도록 구현 하였다. 베이스 스테이션과 이동 단말기의 링크상의 지연은 무시해도 좋은 10 ms 으로 하였고 2 개의 고정 호스트와 베이스 스테이션사이는 모두 5M bps 대역에 20 ms의 지연 값을 부여하였다. 전송을 위한 트래픽은 일반적으로 장시간 연결되어 많은 데이터를 전송하는 FTP 트래픽을 사용하였다. NS2 시뮬레이션에서 송신자에서 사용된 TCP 알고리즘은 Slow Start, Congestion Avoidance, Fast Retransmit, Fast Recovery가 적용된 에러율 10의 TCP Reno를 사용하였고, 수신자에서 사용된 TCP는 TCP Sink를 사용하였다[15][16]. Snoop 모듈의 베이스 스테이션에서의 패킷을 캐시하기 위한 버퍼는 큐 사이즈로 표현되는데, 본 실험에서는 200개로 설정하였다. NS2에서의 성능 측정은 각 데이터 패킷의 Acknowledgement 번호가 실험 종료 시까지 전송된 마지막 Sequence 번호이므로 이는 전송된 패킷수를 의미한다. (그림 9)는 시뮬레이션 결과를 X-Graph로 확인한 그림이다. 그림에서 X축은 시간 Y축은 Sequence 번호를 보여준다. 이것은 단위시간당 증가하는 패킷의 시퀀스 번호를 보여줌으로써 각 조건에서의 성능을 보여준다. 그림에서 ①, ②, ③의 성능을 비교해 보면 기존의 TCP를 적용했을 때보다 SNOOP 모듈 적용 시 성능이 현저히 좋아 졌으며, 본 연구에서 제안한 방법을 적용 했을 시에 ②, ③번 보다 그 성능이 현저히 좋아진 것을 볼 수 있다. X축의 10초를 기준으로 해서 비교해 보면 ③번의 경우 전송된 패킷이 없으며, ②번의 경우 940, ①번의 경우 1350개의 패킷이 전송되었다. 이는 연속적인 패킷 손실 제어 기법을 적용한 ①번 모듈이 기존의 SNOOP 모듈인 ②번에 비해서 약 1.6배 좋은 성능을 보이고 있음을 확인 할 수 있다.



(그림 8) 시뮬레이션 네트워크 토폴로지



(그림 9) 패킷 처리량 비교

5. 결 론

최근 무선 환경을 이용한 수요가 증대하면서 무선망에서의 기존 TCP 성능을 향상시키기 위해 많은 연구가 이루어지고 있다. 그 중에서 가장 대표적인 성능 향상 기법으로 Snoop을 들 수 있다. Snoop의 특징은 고정 호스트에서 전송되는 패킷을 베이스 스테이션에서 캐싱해 두었다가 이동 단말기로 전송도중에 패킷이 손실되면 그 해당하는 패킷을 재전송 해주는 방식이다. 이처럼 Snoop은 간단하면서도 효율적인 기법이기 때문에 본 연구에서는 이를 기반으로 Snoop의 단점을 보완하고 고정 호스트로 부터의 혼잡 제어 알고리즘의 간섭을 막을 수 있도록 하는 방법을 제안하였다. 연속적인 패킷의 손실이 일어날 경우 Snoop은 결국 타이머의 지속적인 타임아웃이나 재2전송 타임아웃에 의해서만 해결될 수 있다. 이것은 버퍼에 있는 패킷에 대해 즉각적인 해결이 있지 않고 그 만큼의 시간을 기다려야 하는 휴지 시간이 발생한다는 것이다. 이것을 해결하기 위해 연속적인 패킷 손실 제어기법을 사용하였다. 이것은 버퍼에 캐싱된 데이터를 즉각 확인함으로써 재전송을 할 수 있게 하고, 연속적인 패킷 손실이 있을 경우 휴지 시간을 줄여 패킷을 효율적으로 처리 할 수 있게 한다. 또한 기존 Snoop은 지역 재전송 중에 패킷손실과 같이 무선망에서 패킷 손실 상태의 복구가 지연 될 경우 결국엔 고정 호스트에서의 재전송으로 인해 간섭이 발생하게 된다. 이러한 단점을 보완하기 위해 제로 윈도우 제어기법을 사용하였다. 이 기법은 무선망에서 일정 횟수의 타임아웃이 일어났을 경우 이동 단말기에서 윈도우 크기가 0이라는 ACK 패킷을 고정 호스트로 보낸다. 이것은 고정 호스트로 하여금 타이머의 동작을 멈추게 하여 타임아웃에 의한 간섭을 막을 수 있다. 이러한 기법들을 통하여 기존 TCP와 Snoop의 성능을 향상 시킬 수 있었다. 그리고 본 연구에서는 개선된 모듈을 TCP 계층이 아닌 하위 단계인 IP계층에서 구현함으로서 2차적인 성능 향상을 이루었다. 그리고 실험 결과에서처럼 에러율이 높을수록 개선된 Snoop 모듈이 더 많은 패킷을 처리하는 것을 확인 하였다.

참 고 문 헌

[1] V. Jacobson, M. J. Karels, "Congestion avoidance and control," *computer communication review*, 1998. 11.

[2] Y. J. Song, Y. J. Shun, "Rate control Snoop: A Reliable Transport Protocol for Heterogeneous Networks with Wired and Wireless Links," IEEE Communication Society, 2003.

[3] F. Sun, V. Li and S. C. Liew, "Design of SNACK Mechanism for Wireless TCP with New Snoop," IEEE Communication Society, WCNC 2004.

[4] 문영성, 강인석, "개선된 SNOOP 기법을 이용한 무선 TCP 성능 향상 방안," 정보과학회논문지, 제32권 1호 2005. 2.

[5] 조용범, 조성준, "AT-Snoop 프로토콜의 지역 재전송 시간에 관한 연구," 한국통신학회논문지 Vol.30, 4B, 2005. 4.

[6] A.Chockalingam and G.Bao, "Performance of TCP/RLP protocol stack on correlated fading DS-CDMA wireless links," IEEE Trans. on Vehicular Tech, Vol.49, pp.28-33, 2003.

[7] M. Zorzi, A. Chockalingam, and A.R.Rao, "Throughput analysis of TCP on channels with memory," IEEE Journal of Selected Areas in Communications, Vol.18, No.8, 2000. 7.

[8] 조용범, 원기섭 조성준, "무선 링크에서 TCP 처리율 향상을 위한 Enhanced Snoop 프로토콜," 한국통신학회논문지 Vol.30, 6B, 2005. 6.

[9] 조준상, 최명환, "전송 오류율이 높은 무선 환경에서의 TCP 성능 저하 극복방안," Telecommunications review, 제10권 제6호, pp.1220-1232, 2000. 11.

[10] 김명섭, 최명환, "네트워크 이동 환경에서의 TCP 성능 향상 기법", 정보처리학회논문지, 제13-C권 제3호, 2006. 6.

[11] 장재신, "전송오류가 큰 이동통신 환경에서 TCP 성능 개선에 관한 연구", 한국통신학회논문지 Vol.28, 9B, 2003. 9.

[12] 김윤주, 이미정, 안재영, "무선망에서의 TCP 성능 향상을 위한 제한적인 Indirect-ACK," 정보과학회논문지 정보통신 30권 2호, 2003. 4.

[13] A.Louri, and A.K.Kodi, "An Optical Interconnection Network and a Modified Snooping protocol for the Design of SMPs," IEEE Trans. on Parallel and Distributed Systems, pp.1093-1104, Vol.15, No.12, 2004. 12.

[14] D.Dutta, and Y.Jang, "An Active Proxy Based Architecture for TCP in Heterogeneous Variable Bandwidth Networks," IEEE Globecom, 2001. 12.

[15] P.Sinha, T.Nandagopal, N.Venkitaraman, P.Sivakumar, "WTCP: A Reliable Transport Protocol for Wireless Wide Area Network," Kluwer Academic Publishers, Wireless Networks, pp.301-316, 2002. 8.

[16] NS 시뮬레이션 소스 http://www.isi.edu/nsnam/ns/snoop/

김 성 찬

e-mail : viper72@chol.com

2002년 2월 숭실대학교 정보과학대학원 (석사)

2005년 2월 숭실대학교 컴퓨터학과 박사 수료

2000년 10월~현재 (주)유코레일 전산실 차장

관심분야 : 정보보호, 유, 무선 통신공학

전 문 석

e-mail : mjun@computing.ssu.ac.kr

1986년 2월 University of Maryland 전산학 (석사)

1989년 2월 University of Maryland 전산학 (박사)

1989년 Morgan Sate University 전산학과 조교수

1989년~1991년 New Mexico Sate University 부설 Physical Science Lab 책임연구원

1991년~현재 숭실대학교 정보과학대학 정교수

관심분야 : 네트워크 보안, 컴퓨터 알고리즘, 암호학