I-TCP를 위한 이동성 지원 라우터에서의 버퍼 오버플로우 방지

(Prevention of Buffer Overflow in the Mobility Support Router for I-TCP)

김 창 호 † 최 학 준 †† 장 주 욱 †††
(Chang Ho Kim) (Hak Jun Choi) (Ju Wook Jang)

요 약 I-TCP의 이동성 지원 라우터(MSR)의 버퍼가 넘치는 현상을 방지하기 위한 혼잡제어 알고리즘을 제안한다. 무선망 환경에서의 높은 비트 오류율와 잦은 핸드오프로 인해 유무선이 혼재된 네트워크에서의 TCP Reno의 혼잡제어 방식은 유선으로만 이루어진 네트워크에서 보다 낮은 전송률을 보인다. 이를 해결하기 위해 하나의 TCP 연결을 유선과 무선부분 각각 두개의 TCP 연결로 나누어 처리하는 I-TCP가 제안되었다. 그러나 무선망의 비트 오류가 과다하게 발생하거나 핸드오프가 빈번하면 이동성 지원 라우터의 버퍼가 넘치는 현상이 발생할 수 있다. 이것은 MSR이 송신자에게 해달 ack를 보낸 패킷(MSR 버퍼에 있는)들이 수신자에게 전송되지 못하는 결과를 초래하여 TCP의 end-to-end semantic를 위반하게 된다. 본 논문에서는 송신자와 MSR 사이에 "흐름 제어" 기법을 도입하여 이동성 지원 라우터의 버퍼가 넘치는 현상을 방지하였다. 송신자와 MSR 사이의 advertised window를 MSR 버퍼의 남은 공간과 연동하여 MSR의 버퍼가 넘치기 전에 MSR로 전송되는 패킷의 양을 조절할 수 있다.

키워드 : 전송 제어 프로토콜, 간접 전송 제어 프로토콜, 무선망, 전송률, 흐름제어, 이동성

Abstract A congestion control algorithm to prevent buffer overflow in MSR(Mobility Support Router) for I-TCP is proposed. Due to high bit error rate and frequent hand-offs over wireless environment, the current congestion control scheme in TCP Reno over mixed(wired and wireless) network exhibits lower throughput than the throughput achieved over wired only network. I-TCP has been proposed to address this by splitting a TCP connection into two TCP connections over wired section and wireless section, respectively. However, buffer overflow in MSR may occur whenever there are excessive bit errors or frequent hand-offs. This may lead to the loss of packets acked by MSR(resident in buffer) to the sender, but not received by the receiver, breaking TCP end-to-end semantics. In this paper, a new scheme is proposed to prevent the MSR buffer from overflow by introducing "flow control" between the sender and the MSR. Advertised window for the TCP connection between the sender and the MSR is tied to the remaining MSR buffer space, controlling the flow of packets to the MSR buffer before overflow occurs.

Key words : TCP, I-TCP, Wireless Network, Throughput, Flow Control, Mobile

1. 서 론

최근 들어 무선 인터넷 및 고속 데이타 전송 등 무선망에 대한 수요가 급증하고 있다. 따라서 미래의 네트워크에서 무선망의 역할이 더욱 더 중요해질 것으로 예상된다. 그러므로, 무선망에서의 다양한 서비스를 효율적으로 제공하기 위해서는 무선 환경에서의 TCP/IP성능을 개선하는 것이 중요한 문제로 대두되고 있다.

이러한 무선망에서의 성능 개선을 위해 여러 기법들이 제안되었다. I-TCP[1], 스누프(Snoop)[2], Ihack[3], SMART[4], M-TCP[5] 등이 그것이다.

그러나, I-TCP의 경우 무선망에서의 패킷 손실이 많을 경우MSR(Mobility Support Router)의 버퍼에 오버플로가 발생함으로써 송신단의 혼잡제어로 인해 전송률

· 본 연구는 (주)엔지비의 지원을 받았음
† 정 회 원 : LG전자 연구원
yourdream@lge.com
†† 비 회 원 : 서강대학교 전자공학과
hakjunc@eecal.sogang.ac.kr
††† 종신회원 : 서강대학교 전자공학과 교수
jjang@mail.sogang.ac.kr
논문접수 : 2000년 12월 18일
심사완료 : 2003년 10월 22일

이 감소하는 단점이 있다. 스누프 역시 스누프 에이전트에서의 버퍼 부담이 존재하고, 무선망의 패킷 손실 시 스누프 에이전트의 재전송 시 휴지기가 존재하는 등의 단점이 있다.

그 밖의 기법들 역시 전송률 증가에는 한계가 있으며 M-TCP는 핸드오프 상황 하에서의 전송률 증가에 초점을 맞춘 기법이므로 본 논문에서는 비교 대상에서 제외하였다.

본 논문에서는 기존 기법의 단점인 버퍼링 부담을 줄임으로써 전송률을 향상시킬 수 있는 새로운 기법을 제안한다. 또한 이를 기존의 기법들과 비교하는 시뮬레이션을 수행하여 성능 향상을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 제안된 무선망에서의 TCP 전송률 향상 기법에 대해 살펴본다. 3장에서는 제안하는 TCP 전송률 향상 기법에 대해 설명한 후, 4장에서 시뮬레이션 수행 결과를 통해 그 성능을 분석한 다음, 5장에서 결론을 맺도록 한다.

2. 기존에 제안된 무선망에서의 TCP 전송률 향상 기법

기존의 TCP는 유한한 네트워크 자원을 효율적으로 공유하기 위하여 몇 가지 데이타 전송 제어 기법을 사용한다[6,7]. 그러한 방법 중 슬로우 스타트(slow start), 빠른 재전송(fast retransmission), 빠른 회복(fast recovery) 등이 있다. 이러한 방법들은 현재 널리 사용되고 있는 Tahoe, Reno, New-Reno 버전에 적용되고 있다. 그러나 그러한 기법들은 유선 네트워크 상황에서는 네트워크 자원을 최대한 효율적으로 이용하기 위한 적절한 기법이지만, 무선 네트워크 상황에서는 패킷 손실 후 전송률만 감소시켜 단점을 갖는다. 따라서 이러한 단점을 극복하기 위해 여러 가지 기법들이 제안되었고, 앞으로 이러한 기법들에 대해 살펴보겠다[8,9].

무선망에서의 TCP 전송률 향상 기법의 핵심은 네트워크 혼잡과 무관한 패킷 손실 시에는 송신 단에서 혼잡 제어를 수행하지 않도록 하는 것이다. 이러한 TCP 전송률 향상 기법은 크게 두 가지로 나눌 수 있다. 종단 간 연결을 유지하는 기법과 연결을 분리하는 기법이 그것이다. 종단 간 연결을 유지하는 대표적 기법은 스누프가 있고, 연결을 분리하는 대표적 기법은 I-TCP를 들 수 있다.

2.1 스누프

스누프는 종단 간 연결을 유지하는 방법 중의 하나이다[2]. 유선망과 무선망의 접속점에 위치한 중간 호스트에 스누프 모듈이 존재한다. 스누프 모듈이 무선망에서의 패킷 손실을 송신 단에 알리지 않고 지역 재전송을 함으로써 송신 단에서 혼잡 제어를 수행하지 않도록 하여 전송률 손실을 방지하는 것이다.

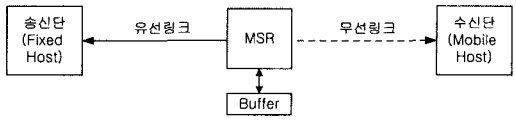

그림 1 스누프의 구조

스누프는 종단 간 의미 구조(semantics)가 파괴되지 않으며, 기존 TCP를 수정하지 않고 스누프 모듈만 추가함으로써 구현할 수 있다는 장점이 있다.

그러나, 무선망에서의 패킷 손실이 많거나 핸드오프(hand-off) 등으로 인한 연결 끊김이 자주 일어날 경우, 스누프 모듈에서 지역 재전송을 수행하는 동안 송신 단은 휴지기에 들어가거나 타임아웃(time out)이 일어나 전송률이 상당히 감소하는 단점이 있다.

2.2 I-TCP(Indirect TCP)

I-TCP는 연결을 분리하는 기법 중의 하나이다[1]. 즉, 종단 간 연결을 유선망과 무선망의 두 부분으로 나누고 그 중간 지점의 MSR(Mobile Support Router)에서 연결을 관리한다. MSR은 무선 수신 단으로 패킷을 재전송 해 주는 역할을 하고, 수신단의 이동으로 인해 핸드오프가 발생 시, 다른 MSR로 무선 수신 단에 대한 정보를 넘겨줌으로써, 기존의 연결을 유지시켜주는 역할을 한다.

연결을 두 개로 나눔으로써 TCP 연결 입장에서 보면 두 가지 변화가 생긴다. 실질적인 RTT(Round Trip Time)와 패킷 손실률이 줄어들게 된다[1].

수신 단이 이동하여 핸드오프현상이 발생, 순간적인 끊김이 있거나, 무선 연결의 특성으로 인해 무선망에서 패킷 손실이 일어난다 하더라도 고정 호스트에서 재전송하지 않고 MSR에서 duplicate ACK나 타임아웃을 감지하여 재전송을 수행함으로써 무선망에서의 패킷 손실이 송신 단에서는 혼잡 상황으로 인식하지 않기 때문에, 전송률을 개선할 수 있다[1].

이는 기존 TCP의 구조를 변경하지 않는 장점이 있으나, 종단 간 의미구조가 깨진다는 단점이 있다. 즉, 실제로 수신 단에서 데이타를 수신하지 않았음에도 불구하고 송신 단은 ACK를 받을 수 있는 것이다. 이러한 단점은 패킷 손실률이 높거나 장시간의 연결 단절 시에는 큰 문제가 될 수 있다.

또한, 무선망에서의 패킷 손실이 많을 경우 지연시간

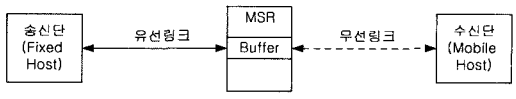

그림 2 I-TCP의 구조

이 증가한다. I-TCP의 MSR에서 계산하고 있는 버퍼 사이즈는 BW×Delay이다. 이 경우 추가적인 지연시간에 의해서, 순간적으로 MSR 버퍼의 수신 가능 용량보다 더 많은 양의 데이타가 들어올 수 있으므로, 버퍼가 넘치는 현상이 발생하게 된다. 이런 경우 송신 단에서 오버플로에 의한 패킷 손실을 혼잡에 의한 손실로 판단하므로, 전송률이 급격히 감소한다.

2.3 그 밖의 기법들

그 밖의 무선망에서의 TCP 전송률을 향상시키기 위한 기법으로는 링크 계층(link layer)에서의 구현과 SACK(Selective Acknowledgement)[10]을 사용하는 기법들이 있다.

연결 계층에서의 구현 기법으로는 FEC(Forward Error Correction)와 ARQ(Automatic Repeat Request)를 사용하여 패킷 손실을 복구하는 기법[11]이 있으나 많은 패킷 손실이나 장시간 연결 끊김 등으로 TCP 타임아웃 등이 발생할 때의 전송률 감소는 막을 수 없다는 단점이 있다. 따라서 연결 계층에서의 전송률 향상 기법은 전송 계층(transport layer)에서의 전송률 향상 기법과 병행하여 적용될 때 최상의 결과를 낳을 수 있다.

또 다른 방법으로 SACK를 사용하는 기법을 들 수 있다. 그 중 대표적인 기법이 SMART이다. 이 기법은 기존 TCP처럼 여러 개의 패킷이 손실되었을 때 타임아웃을 발생시키지 않고 효과적으로 손실된 패킷을 재전송할 수 있다. 그러나 이 기법은 유선망에서는 효과적이지만 네트워크 혼잡과 무관한 손실이 있을 때의 전송률 감소를 막을 수 없다. 즉, 무선망에서의 근본적 해결책이 될 수는 없다.

3. 제안하는 TCP 전송률 향상 기법

3.1 제안하는 전송률 증가 기법

TCP 전송률의 증가를 위해 본 논문에서는 I-TCP의 알고리즘을 사용한다. 그러나 I-TCP는 앞에서 살펴본 것처럼 무선망에서의 패킷 손실 시 버퍼가 넘칠 수 있는 단점이 있다.

I-TCP에서 MSR의 버퍼 크기는 링크의 최대 이용률인 BW×delay로 정하는 것이 적당한데, 무선망에서의 패킷 손실로 인해 MSR의 버퍼에 데이타가 계속 쌓이게 되면 버퍼의 오버플로로 인해 패킷 드롭(drop)이 생기게 된다.

MSR의 기본 버퍼 크기를 BW×delay로 설정하였을 때, 무선 수신 단이 장시간 끊기거나 지속적인 패킷의 유실이 일어날 경우, 추가적인 지연시간 α가 생기게 된다. 결과적으로 BW×(delay + α)의 패킷을 MSR의 버퍼에 저장하여야 하지만, 버퍼의 크기가 일정하기 때문에 BW×α만큼의 패킷이 오버플로로 인해서 유실되게 된다.

일정한 버퍼 크기 하에서 드롭이 자주 또는 많이 발생하게 되면 송신 단에서 그 원인을 혼잡으로 인식하기 때문에 윈도우 크기를 줄이거나 타임아웃이 일어나 TCP 전송률이 상당히 감소하는 문제점이 생긴다. 따라서 이러한 드롭을 예방하는 것이 필요하다. 본 논문에서는 이러한 드롭을 예방하기 위해서 MSR에서 송신 단으로 ACK를 보낼 때 ACK 패킷의 애드버타이즈(Advertised) 윈도우 크기를 MSR의 버퍼가 비어있는 크기만큼 수정해서 보냄으로써 송신 단에서 MSR의 비어있는 버퍼 크기만큼의 패킷을 송신하게 하여 MSR에서의 버퍼 오버플로를 예방하였다. MSR에서의 버퍼 오버플로는 유선망에서의 패킷 혼잡 때문이 아니므로, MSR에서 미리 애드버타이즈 윈도우의 크기를 줄여서 버퍼에서의 드롭을 예방함으로써 전송률을 증가시키는 것이다.

실제로 본 논문에서는 애드버타이즈 윈도우 크기를 MSR의 비어있는 버퍼 크기로 하여 송신 단에 전송하게 구현하였다. 이러한 방법은 전체 TCP 알고리즘을 수정할 필요 없이 MSR에서 ACK 패킷만 조금 수정하게 하면 되므로 구현이 간단하다는 장점이 있다.

시간 t에 비어 있는 버퍼 공간을 $Q_f(t)$라고 하면,

$$Q_f(t) = Q_t - Q_u(t) \quad (1)$$

(Q_t : 총 버퍼 공간, $Q_u(t)$: 시간 t에 버퍼 사용 공간)이다.

이 때, 애드버타이즈 윈도우의 크기는 비어있는 버퍼 공간에 비례해야 한다. 즉, $f(Q_f(t))$가 되어야 하는 것이다.

따라서, $W_r(t)$를 시간 t에 수신단에서의 애드버타이즈 윈도우 크기라고 하고, $W_{MSR}(t)$를 MSR에서 송신 패킷을 받은 후 바로 송신 단으로 보내주는 ACK의 애드버타이즈 윈도우 크기라고 하면

$$W_{MSR}(t) = \max(\min(W_r(t), Q_f(t)), MSS) \quad (2)$$

이 되어야 한다[12].

여기서, 애드버타이즈 윈도우 크기가 MSS(Maximum Segment Size)보다 커야 하는 이유는 TCP 연결 확립 초기의 short-term deadlock 현상을 방지하기 위해서이다. 연결 초기에 SYN 패킷처럼 1 MSS보다 크기가 작은 패킷 하나를 전송할 경우 수신 단에서 delayed ACK를 사용하는 경우 또 다른 패킷이 도착할 때까지 기다리는 short-term deadlock 현상이 발생하기 때문에 전체적 전송률이 상당히 감소한다. 따라서 이를 막기 위하여 애드버타이즈 윈도우 크기는 항상 MSS보다는 커야 한다.

3.2 제안하는 MSR의 구조와 기능

본 논문에서 제안하는 알고리즘의 전체적인 네트워크 구현도가 그림 3과 같다. Access pointer를 중심으로

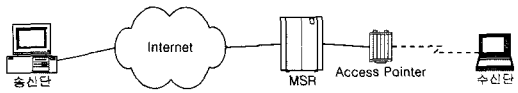

그림 3 제안하는 기법의 전체적인 네트워크 구현도

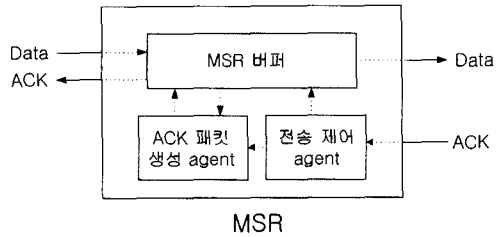

그림 4 MSR의 구조

유선망과 무선망으로 나뉘므로 access pointer와 MSR이 바로 연결되어 MSR이 유선망과 무선망의 연결을 이어주는 역할을 한다.

MSR 모듈의 대략적 구조는 그림 4와 같다. MSR 버퍼는 FIFO(First In First Out) 구조이다. 송신 단에서 전달되는 데이타는 MSR 버퍼에 쌓이게 되고 MSR은 데이타를 받으면 앞에서 살펴본 식 2인 $W_{MSR}(t)$의 크기만큼 애드버타이즈 윈도우를 수정한 ACK 패킷을 송신단에 바로 보낸다. 이 때, 현재 비어 있는 버퍼 공간과 전송 제어 에이전트에서 받은 수신단의 애드버타이즈 윈도우 크기와 MSS, 이 세 가지를 비교하여 윈도우 크기를 결정하게 된다. 또한, 버퍼에 쌓인 데이타는 수신단으로 전송하고 ACK를 받았을 때 버퍼에 쌓인 데이타를 삭제한다. 또한, duplicate ACK나 타임아웃 등으로 데이타의 손실이 감지되었을 때는 전송 제어 에이전트가 데이타의 재전송을 수행한다.

4. 실험 및 결과

본 논문에서 제안한 기법의 성능을 확인하기 위하여 시뮬레이션을 통해 구현한 후 실험하였다. 시뮬레이션 도구로는 대표적 TCP/IP 네트워크 시뮬레이션 도구인 버클리 대학의 ns-2 시뮬레이터를 사용하였다[13].

그림 5는 기존 TCP, 스누프 및 I-TCP와 제안된 기법 간의 성능 비교를 위한 시뮬레이션 환경이다.

송신 단에서 수신 단까지 기존의 TCP 연결을 사용한 경우는 MSR이 패킷 포워딩만을 하게 되고, 그 이외의 경우는 MSR이 Spilit Connection의 중계점 역할을 하게 된다.

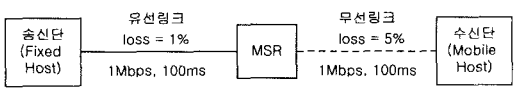

그림 5 시뮬레이션 환경

사용한 TCP 버전은 현재 가장 널리 사용되고 있는 TCP Reno이다.

각 링크는 1Mbps의 대역폭, 100msec의 지연 시간에 3 %의 패킷 손실률을 가지도록 했다. 또한, Wnd_{max} = 28이다. 기존 TCP의 RTT는 400msec, 제안 기법의 RTT는 실질적으로 200msec가 된다. 또한, MSR의 버퍼 크기는 BW×delay = 1Mbps×100msec = 24 packets가 된다. 1초에서 10초까지 FTP를 이용한 데이타 전송을 수행하였을 때의 시간에 따르는 송신 단에서의 전송 패킷의 순서 번호는 그림 6~9와 같다. 그림에서 가로축은 진행 시간을 나타내고, 세로축은 송신 단에서 송신한 패킷의 순서 번호를 나타낸다.

TCP의 경우 무선 송신 단에서의 높은 손실률에 의해 재전송상황이 많이 발생하여, 전송 상황이 상당히 열악하며, 이 상황은 시간이 갈수록 더 악화되어 감을 알 수 있다.

스누프의 경우, 무선 링크에서의 높은 손실률에도 불구하고 송신 단에서의 패킷 재전송과 그에 따른 혼잡제어는 거의 없는 것을 알 수 있다. 그것은 무선 링크에서 손실이 있을 때 MSR에서 duplicate ACK를 버리고 버퍼에 있는 패킷을 재전송하기 때문이다. 따라서 MSR에서 재전송이 있을 경우 송신 단은 그에 따른 ACK를 기다리기 위해 잠깐 휴지기에 들어감을 알 수 있다. 그림 7에서 약 4.5초, 6초에서 MSR의 재전송에 따라 송신 단이 잠시 휴지기에 들어감을 볼 수 있다. 즉, 이런 휴지기 때문에 전송률의 증가에 한계가 있다. 스누프의 경우 전송률은 28.3 packets/sec이다.

I-TCP의 경우는 무선 링크에서의 패킷 손실이 자주 생김으로써, 패킷 재전송을 수행하는 MSR의 버퍼에 데이타가 많이 쌓이게 되고, 버퍼에 오버플로가 생기게 되면 패킷 손실이 생겨서 송신 단 측에서는 패킷 재전송과 그에 따르는 혼잡 제어를 수행함으로써 전송률이 감

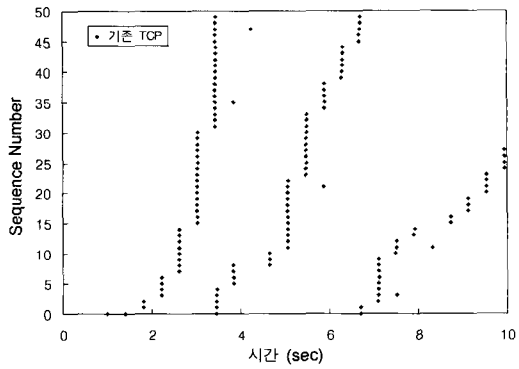

그림 6 기존 TCP의 성능

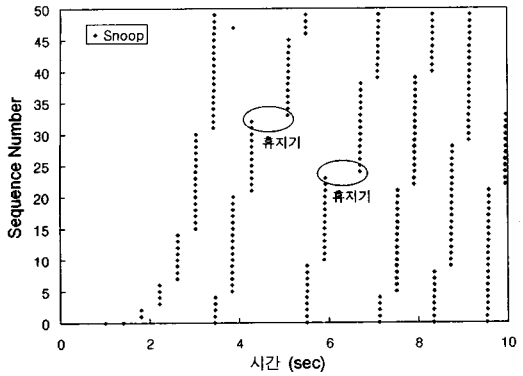

그림 7 스누프의 성능

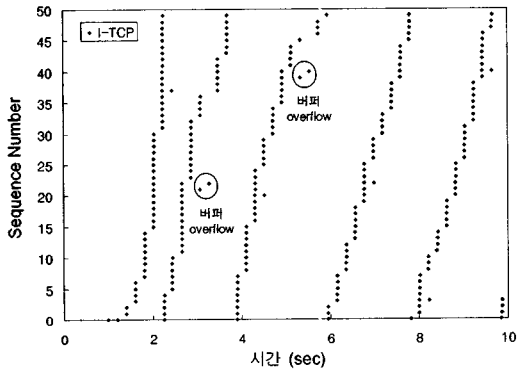

그림 8 I-TCP의 성능

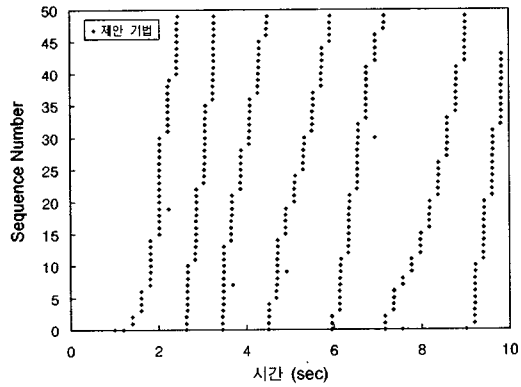

그림 9 제안하는 기법의 성능

소하게 된다. 그림 8에서도 버퍼 오버플로가 일어나는 것과 그로 인해 전송률이 감소한 것을 확인할 수 있다. I-TCP의 경우 전송률은 25.3 packets/sec이다.

그러나, 제안한 기법의 경우 무선 링크에서의 패킷 손실이 자주 생겨서 MSR의 버퍼에 데이타가 많이 쌓이게 되도, MSR에서는 비어있는 버퍼의 양만큼의 애드버타이즈 윈도우를 송신 단으로 보내므로 버퍼에 오버플로가 생기지 않고 따라서 송신 단에서의 전송률이 감소하지 않는 것을 알 수 있다. 이는 그림 9에서 확인할 수 있다. 제안 기법의 경우 전송률은 30.9 packets/sec이다.

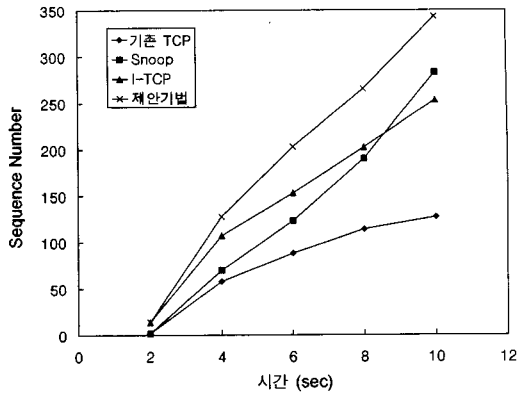

그림 10 각 기법간 성능 비교

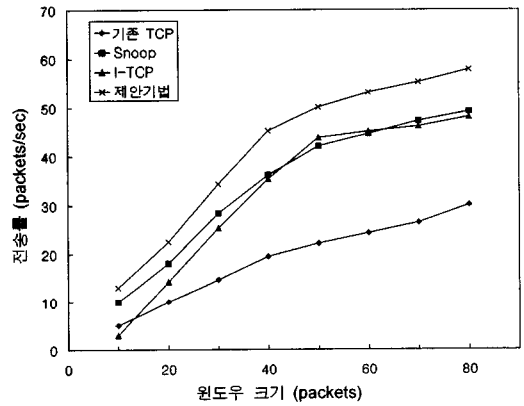

그림 11 윈도우 크기에 따른 각 기법의 End-to-End 전송률

기존 TCP, 스누프, I-TCP, 제안하는 기법의 시간에 따르는 전송 패킷의 시퀀스 번호를 함께 그래프로 나타내면 그림 10과 같다. 전송된 패킷의 양을 시간에 따라 비교해 보면 스누프와 I-TCP는 기존 TCP보다는 전송률이 높지만 제안하는 기법보다는 낮음을 확인할 수 있다.

TCP의 전송률은 윈도우 크기에 대해서도 제한되기 때문에 윈도우 크기에 따르는 각 기법의 전송률을 시뮬레이션해 보았다. 윈도우 크기를 제외한 나머지 조건들은 앞의 실험과 동일하다. 그 결과를 나타낸 것이 그림 11이다. 가로축은 윈도우 크기, 세로축은 전송률을 나타낸다.

그림 11을 보면, 윈도우 크기가 매우 작을 때는 I-TCP와 기존 TCP의 전송률이 거의 같음을 알 수 있다. 이는 윈도우 크기가 너무 작아서 무선 링크에서의 패킷

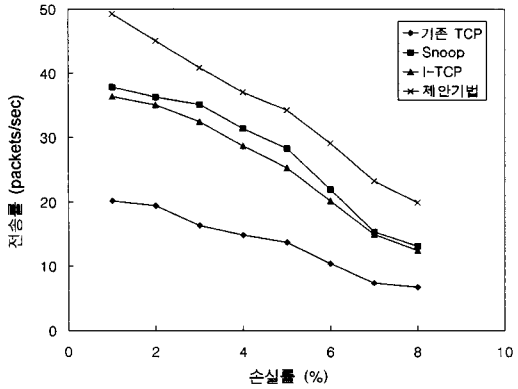

그림 12 무선 링크에서의 손실률에 따른 각 기법의 End-to-End 전송률

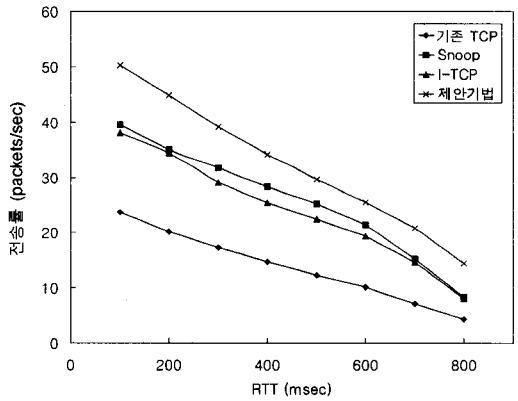

그림 13 RTT에 따른 각 기법의 End-to-End 전송률

손실 시에 MSR의 버퍼가 자주 넘치게 되고 이로 인해 송신단의 전송률이 급격히 감소하기 때문이다.

그러나, 윈도우 크기가 증가함에 따라 스누프, I-TCP, 제안 기법 모두 기존 TCP 보다 전송률이 큼을 알 수 있다. 스누프와 I-TCP는 전반적으로 비슷한 전송률을 보이고, 제안 기법은 가장 높은 전송률을 보였다.

윈도우 크기가 50 이상일 때는 무선 링크에서의 패킷 손실로 인해 전송률 증가 속도가 조금 둔해지는 것을 알 수 있다.

다음으로 무선 링크에서의 손실률에 따르는 전송률을 시뮬레이션 해 보았다. 시뮬레이션 결과는 그림 12에 나타내었다. 가로축은 무선 링크의 손실률, 세로축은 전송률을 나타낸다.

무선 링크에서의 손실률이 증가함에 따라 전체적으로 전송률이 감소하였다. 그러나 전체적으로 전송률 크기를 비교해 보면 제안 기법, 스누프, I-TCP, 기존 TCP 순으로 큼을 알 수 있다. 제안 기법은 MSR의 버퍼가 넘치지는 않았으나, 무선 링크의 손실률이 크면 재전송으로 인해 전송률이 감소하므로 다른 기법들과 마찬가지로 손실률 증가에 따라 전송률이 감소함을 알 수 있다.

마지막으로 RTT에 따른 각 기법의 전송률을 살펴보았다. 그 결과는 그림 13에 나타내었다. 가로축은 RTT, 세로축은 전송률을 나타낸다.

RTT가 증가함에 따라 윈도우 증가 속도 감소, 패킷 손실시 회복 속도 감소 등으로 인해 전체적으로 전송률이 감소하는 것을 알 수 있다. 그러나, 이 역시 제안기법, 스누프, I-TCP, 기존 TCP 순으로 전송률이 큼을 알 수 있다.

5. 결 론

본 논문에서는 기존에 제시된 무선망에서의 TCP 전송률 향상 기법들의 장단점에 대해 살펴보고, 단점을 보완하기 위한 기법을 제안하였다.

RTT와 에러 율의 실질적인 감소라는 측면에서 I-TCP가 전송률 향상에 더욱 유리하고, 본 논문에서는 I-TCP의 알고리즘을 사용하되, 그 단점을 보완하기 위한 기법을 사용하였다. I-TCP는 무선 링크에서의 패킷 손실률이 커지게 되면 MSR의 버퍼가 넘치게 될 수 있고, 그로 인해 송신 단에서의 전송률이 급격히 감소한다. 이를 방지하기 위해 MSR에서 송신 단으로 ACK를 보낼 때 ACK의 애드버타이즈 윈도우 크기를 MSR의 빈 버퍼 공간만큼 정해줌으로써 MSR의 버퍼 넘침을 방지하였다.

제안 기법의 성능을 확인하기 위해서 시뮬레이션을 수행하였고, 시뮬레이션 결과 제안 기법, 스누프, I-TCP, 기존 TCP 순으로 전송률이 큼을 확인할 수 있었다.

본 논문에서는 하나의 연결에 대해서만 제안 기법을 적용시켰지만 추후로는 단일 MSR 상에서 여러 개의 연결을 관리하는 경우의 버퍼 관리 문제, 연결 관리 문제 등에 대한 연구가 필요하다. 그리고, MSR의 버퍼 크기에 따라서 I-TCP와 제안된 기법의 성능 변화에 대한 추가적인 연구와 더불어, 실제적인 구현에 의한 실험이 따라야 할 것이다.

참 고 문 헌

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

[2] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," ACM Wireless Networks, Dec. 1995.

[3] J. A. Cobb and P. Agrawal, "Congestion or Corruption? A Strategy for Efficient Wireless TCP

sessions," IEEE Symposium on Computers and Communications, pp. 262~268, 1995.
[4] S. Keshav and S. P. Morgan, "SMART: Performance with Overload and Random Losses," Proceedings of IEEE Infocom. 97, April 1997.
[5] Kevin Brown and Suresh Singh, "M-TCP: TCP for Mobile Cellular Networks," CCR Proceedings, Oct. 1997.
[6] V. Jacobson, "Congestion Avoidance and Control," ACM SIGCOMM'88, August 1988.
[7] W. R. Stevens, "TCP/IP Illustrated," Vol. 1, Addison-Wesley, Nov. 1994.
[8] 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, 1996.
[9] Nihal K. G. Samaraweera and Godred FairHurst, "Reinforcement of TCP Error Recovery for Wireless Communication," CCR Proceedings, Oct. 1998.
[10] V. Jacobson and R. T. Braden, "TCP Extensions for Long Delay Paths," RFC 1072, Oct. 1988.
[11] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin, "AIRMAIL: A Link-Layer Protocol for Wireless Networks," ACM Wireless Networks, Feb. 1995.
[12] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," ACM SIGCOMM'98, 1998.
[13] Steven McCanne and Sally Floyd, NS(Network Simulator), http://www-mash.cs.berkeley.edu/ns, 1995.

김 창 호
1998년 서강대학교 전자공학과(학사) 2000년 서강대학교 전자공학과(석사). 현재 LG전자 연구원

최 학 준
2002년 서강대학교(학사). 현재 서강대학교 전자공학과 대학원

장 주 욱
1983년 서울대학교 전자공학과(학사) 1985년 한국과학기술원 전기및전자공학과(석사). 1993년 Univ. of Southern California 컴퓨터공학과(박사). 서강대학교 전자공학과 부교수