무선 링크에서의 TCP 성능 평가
(Performance Evaluation of TCP over Wireless Links)

박 진 영 † 채 기 준 ††
(Jinyoung Park) (Kijoon Chae)

요 약 현재 가장 널리 쓰이는 수송계층 프로토콜인 TCP는 패킷 손실이 망의 혼잡 때문에 발생하는 기존의 망들에 적합하도록 되어 있다. TCP는 망의 패킷 에러율이 낮다는 가정 하에 종단간 신뢰성 있는 패킷 전송을 수행한다. 그러나 무선 링크가 있는 망에서는 높은 에러율과 핸드오프에 의한 패킷 손실이 발생하게 된다. TCP는 이런 경우에 발생되는 모든 패킷 유실에 대해서 혼잡 제어와 회피 알고리즘을 작동시키게 된다. 따라서 무선 링크에서의 높은 에러율에 의해 TCP의 불필요한 혼잡제어 알고리즘이 작동되고 망의 대역폭을 비효율적으로 사용하게 하여 무선 환경과 연결된 망에서의 종단간 성능을 저하시키게 된다. 이러한 무선 링크에서의 TCP 성능 향상을 위한 많은 방안들이 연구되고 있다. 본 논문에서는 무선 링크에서의 TCP 성능 향상을 위한 기존의 기법들을 비교 분석하고 각 방법의 성능을 평가하여 유무선 링크가 연결된 망에서 TCP의 성능 향상에 적합한 모델을 제시하고자 한다. 이는 기지국에서의 캐쉬 기법과 함께 TCP 종단간 SACK 옵션에 의한 수신자 정보를 사용하는 방법으로 시뮬레이션을 통하여 제시한 모델의 유효 처리율이 타 방식에 비해 향상되었고 무선 링크로의 중복된 재전송의 비율이 줄었음을 알 수 있었다.

Abstract Nowadays, most widely used transport protocol, TCP is tuned to perform well in traditional networks where packet losses occur mostly because of congestion. TCP performs reliable end-to-end packet transmission under the assumption of low packet error rate. However, networks with wireless links suffer from significant losses due to high error rate and handoffs. TCP responds to all losses by invoking congestion control and avoidance algorithms, resulting in inefficient use of network bandwidth and degraded end-to-end performance in that system. To solve this problem, several methods have been proposed. In this paper, we analyse and compare these methods and propose appropriate model for improving TCP performance in the network with wireless links. This model uses TCP selective acknowledgement (SACK) option between TCP ends, and also uses caching method at the base station. Our simulation results show that using TCP SACK option with base station caching significantly reduces unnecessary duplicate retransmissions and recover packet losses effectively.

1. 서 론

이동 통신은 최근 빠르게 성장하고 있는 통신 시장으로 고속의 무선 LAN 상품이 등장해 사용되고 있으며 WAN에서도 저속의 무선 서비스가 가능하다. 또한 다양한 형태의 망을 연결한 인터넷이 계속 팽창해 나가고 있으며 향후 이동성을 갖는 인터넷 호스트들이 증가할 전망이다. 무선 링크는 반드시 유선 케이블을 이용하여야 하는 유선 링크에 비하여 확장성, 유연성, 설치 및 유지 보수의 용이성 등과 같은 장점을 제공하며 투자 대비 성능이 뛰어나다.

이러한 무선 환경은 유선보다 높은 비트 에러율을 나타내며 핸드오프에 의한 통신 중단이나 패킷 손실이 발생하게 된다. 기존의 망에서 널리 쓰이고 있는 수송 계층 프로토콜인 TCP (Transmission Control Protocol)는 패킷 유실이 주로 망 내부의 혼잡에 의해서만 일어난다고 가정하고 전송한 패킷에 대해 타이머를 두어 일

† 비 회 원 : LG정보통신 차세대통신연구소 연구원
jinyoungp@lgic.co.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
kjchae@ewha.ac.kr
논문접수 : 1999년 1월 29일
심사완료 : 2000년 1월 31일

정 시간 동안 그에 대한 ACK (Acknowledgement)이 오지 않으면 혼잡 제어 알고리즘을 작동시키게 된다. 이는 무선망에서의 높은 에러율과 핸드 오프에 의한 패킷 손실의 특성을 고려하지 않고 혼잡 제어와 회피 알고리즘을 작동시키게 되고 망의 대역폭을 비효율적으로 사용하도록 하여 무선 환경과 연결된 망에서의 종단간 처리율을 저하시키게 된다. 또한 이렇게 망의 혼잡과 관련없는 패킷 유실에 대해 재전송 타이머에 의존하여 재전송을 하는 경우 유실된 패킷을 회복하는 데에 오랜 시간이 걸리게 된다. 따라서 이러한 무선 링크에서의 TCP 성능 향상을 위한 많은 방안들이 연구되고 있다. 무선망을 통해서 데이타를 전달하는 경우 대부분 유선망을 통해서 기지국까지 연결되고 기지국에서 이동 호스트까지의 마지막 링크만 무선 링크이다. 이 경우 유선망을 통하여 오류없이 전달된 데이타가 마지막 무선 링크의 높은 비트 에러율에 의해 전달되지 못하면 무선 링크에 의해 전체 연결의 성능이 저하되어 데이타를 전달하는데 사용한 유선망의 대역폭 낭비와 종단간 TCP 처리율의 저하를 가져오게 된다.

본 논문에서는 무선 링크에서의 TCP 성능 향상을 위한 여러 가지 기법들에 대해 알아보고 이를 비교 분석, 성능 평가를 수행하여 유·무선 링크가 연결된 망에서 적합한 모델을 제시하도록 한다. 이 방안은 기지국에서의 캐쉬 기법을 사용하여 무선 링크에서 유실된 패킷을 기지국에서 재빨리 재전송 할 수 있게 하며 TCP 종단간 TCP SACK 옵션을 사용하여 수신자가 받은 데이타가 무엇인지를 알려주는 정보를 ACK에 포함시켜 보냄으로써 기지국에서 이 정보를 사용하여 유실된 패킷만 재전송 해주는 방안이다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존의 TCP 매커니즘들을 알아보도록 한다. TCP의 기본 작동 메커니즘과 현재 제시되고 있는 확장된 TCP 혼잡 제어 기법들인 TCP Tahoe, TCP Reno, TCP New-Reno, TCP SACK 옵션을 사용한 TCP에 대하여 설명한다. 또한 무선 링크에서의 TCP의 고려사항 및 기존의 제안된 방법들에 대해 알아보고 각 방법의 특성과 문제점을 분석해 보도록 한다. 3 장에서는 본 논문에서 무선 링크에 적합한 모델로서 제시하는 방법을 설명한다. 4장에서는 이러한 여러 방안들의 성능을 시뮬레이션을 통하여 비교 분석해 보도록 하고 마지막으로 5 장에서는 본 논문의 결론과 향후 연구 계획에 대하여 기술한다.

2. 기존의 TCP 메커니즘들

2.1 확장된 TCP 혼잡 제어 기법

현재 TCP의 공식적인 버전은 RFC 793에 정의되어 있으며 이후로 확장된 방안들이 제시되고 있다. 현재 많이 쓰이고 있는 TCP 버전인 TCP Tahoe, TCP Reno 및 그의 확장된 버전인 TCP New-Reno, TCP SACK 옵션을 사용하는 경우의 TCP에 대해 간략히 설명한다. 각각을 살펴보면 다음과 같다.

• TCP Tahoe

Slow Start와 Congestion Avoidance는 TCP Tahoe에 최초로 구현되었다[1]. 1989년 발표된 RFC 1122에서는 TCP가 반드시 Slow Start와 Congestion Avoidance를 구현해야 한다고 명시하고 있다. 그리고 Fast Retransmit 알고리즘도 패킷 유실시 재전송 타임아웃을 기다리는 비효율성을 막기 위해 TCP Tahoe에 포함되어 있다.

• TCP Reno

TCP Reno구현에서 Fast Recovery 알고리즘이 구현되었다[2]. 이 알고리즘을 추가함으로써 중복 ACK에 의한 Fast Retransmit 이후에 CWND를 1 세그먼트부터 시작하여 Slow Start를 수행하는 것이 아니라 바로 윈도우 크기를 재전송 이전의 반의 크기로 설정하고 Congestion Avoidance 단계로 들어갈 수 있게 하였다.

• TCP New-Reno

만약 두 개 또는 그 이상의 패킷이 같은 전송 윈도우 내에서 유실될 경우 Fast Retransmit과 Fast Recovery 알고리즘도 재전송 타임아웃을 기다리지 않고는 여러 유실들로부터 회복할 수 없다. TCP New-Reno는 윈도우 당 한 개 이상의 패킷 유실시 각각의 유실된 패킷의 재전송 타임아웃을 기다리지 않고 한 RTT당 한 개씩 유실된 패킷을 재전송 할 수 있도록 했다. 송신자는 Fast Retransmit이 작동될 때 보낸 가장 큰 일련 번호를 기억하여 RECOVER라는 변수에 저장한다. 그리고 재 전송된 패킷에 대한 ACK이 오면 이 ACK이 RECOVER를 포함하여 ACK을 했는지를 검사한다. 만약 그렇다면 새로운 ACK이므로 Fast Retransmit 단계에서 Congestion Avoidance 단계로 이동한다.

• TCP SACK 옵션의 사용

SACK 옵션을 사용한 TCP[23,26,27,28]는 TCP Reno의 특성에 SACK 옵션을 첨가한 것이 다르다. TCP SACK 옵션을 사용할 경우의 ACK에는 수신측이 이미 받은 연속적인 데이타 블록들의 리스트가 포함된다(그림 1). 송신자는 이 SACK 정보를 사용하여 보내어졌으나 아직 ACK되지 않은 세그먼트들에 대한 테이블을 유지한다. 한 윈도우 내에서 여러 개의 패킷 유실

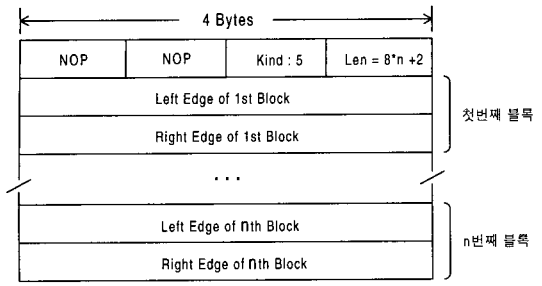

그림 1 TCP SACK 옵션의 포맷

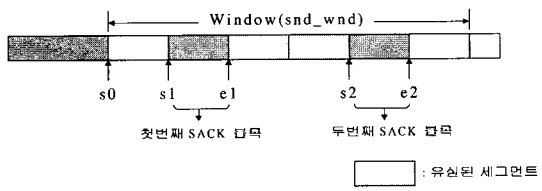

그림 2 SACK 옵션의 사용 예

이 있을 때 TCP Tahoe, TCP Reno는 첫 유실된 패킷의 재전송 이후의 나머지 유실된 패킷들에 대해 재전송 타임아웃에 의지하여 재전송을 하게 되고 재전송 타임아웃이 만료될 때마다 재전송 타이머를 후퇴 (back off) 시킨다. 그러한 경우 망의 효율이 심각하게 떨어지게 된다. TCP New-Reno의 경우에는 재전송 타임아웃에 의지하지 않고 한 RTT당 한 개의 유실된 패킷을 재전송할 수 있지만 유실된 여러 패킷들을 한 RTT 이내에 회복할 수는 없다. 그러나 SACK TCP의 경우에는 SACK 정보를 사용하여 한 RTT 이내에 여러 개의 유실된 패킷을 회복할 수 있다. 그리고 이 경우 유실된 패킷만을 전송하므로 중복되는 전송을 막을 수 있다. [그림 2]는 SACK 옵션의 작동 예로 수신자 큐 (reassembly queue) 안의 세그먼트들을 나타낸다. 이는 송신자가 시작 일련 번호 (sequence number)가 s0이고 총 길이 e2 - s0인 다섯 개의 세그먼트들을 보낸 경우이다. 이때 흐린색으로 표시된 세 개의 세그먼트가 유실되어 수신자는 송신자에게 TCP ACK 필드는 일련 번호 s0+1이고 SACK 옵션 필드는 s1 (SACK 포맷에서 left edge에 해당)과 e1 (right edge에 해당)인 첫 번째 SACK 블록과 s2와 e2인 두 번째 SACK 블록을 포함한 SACK을 생성하여 보내게 된다. 그림에서 진한 블록으로 표현된 곳은 수신자 큐에 순서대로 잘 도착한 세그먼트들을 나타낸다. 현재 SACK TCP가 완전히 표준화가 되어 있지 않으므로 다양한 SACK TCP의 구현들이 제시되고 있다[10]. 본 논문에서는 RFC 1323[8], RFC 2018[23]과 RFC 1072[28]에 명시된 SACK TCP 방식을 따르고 네트워크 시뮬레이션 툴인 ns v.2.0[31]에서 구현한 SACK TCP의 구현 방법을 참조한 SACK TCP 구현을 사용한다.

2.2 무선 링크에서의 TCP의 고려사항 및 기존의 제안된 방법

무선 데이타 통신을 위해 사용되는 대표적인 수송 계층 프로토콜은 기존의 유선망에서 사용되는 TCP이다. 그러나 TCP는 낮은 오류의 유선망의 환경에 맞게 고안되었으므로, 매우 다른 오류 환경을 가진 무선 링크에서는 제대로 동작하지 않는다. 이렇게 혼잡에 의하지 않은 패킷 유실에 대해서도 TCP는 혼잡 제어 메커니즘을 작동시키고 윈도우 크기를 줄이며 재전송 타이머를 후퇴시킴으로써 불필요하게 망의 이용률을 저하시킨다[30]. 최근 이렇게 망 내의 혼잡과 관련 없는 패킷 유실이 TCP 성능에 미치는 영향을 완화하기 위해 I-TCP를 사용하는 방법, Snoop 프로토콜을 사용하는 방법, 종단간 TCP SACK 옵션을 지원하는 TCP를 사용하는 방법 그리고 SMART 프로토콜을 사용하는 방법 등이 제안되고 있다. 이 메커니즘들에 대해 간략히 살펴보면 다음과 같다.

2.2.1 I-TCP (Indirect TCP)

I-TCP는 TCP 연결을 고정 호스트와 기지국 사이의 유선 연결, 기지국과 이동 호스트사이의 무선 연결로 나누어서 각각의 연결에 적합한 TCP 정책을 적용함으로써 전송 성능의 향상을 이루고자하는 방법이다[6]. 그러나 이 경우 매 패킷이 TCP 프로토콜 프로세싱을 두 번씩 하게 되는 오버헤드가 발생하게 될 뿐 아니라 실제로 패킷이 이동 수신자에 도착하기 전에 ACK이 먼저 송신자에게 도착하는 경우가 발생할 수 있으므로 TCP의 종단간 작동 의미 (end-to-end semantics)에도 어긋난다. 또한 이 방법은 핸드오프 시에 더 큰 지연을 갖는다고 알려졌다[6].

2.2.2 Snoop 프로토콜

Snoop 프로토콜[3]은 이동 호스트와 고정 호스트 사이에서 패킷 송수신을 책임지는 중간 호스트인 기지국에서 Snoop 에이전트를 통해 전송 성능의 향상을 이루어 내는 프로토콜이다. Snoop 에이전트는 기지국을 통해 전송되고 있으며 아직 수신자에게 ACK이 안된 세그먼트를 캐쉬에 저장했다가 중복되는 ACK이나 지역 타임아웃에 의해 유실된 패킷이 있다는 것을 알게 되었을 때 캐쉬에 있는 유실된 패킷을 재전송한다. 그리고

반복되는 ACK이 송신자로 가는 것을 막아 송신자로부터의 중복되는 재전송을 막는다. 캐쉬에 저장된 세그먼트는 올바른 ACK이 오면 삭제된다.

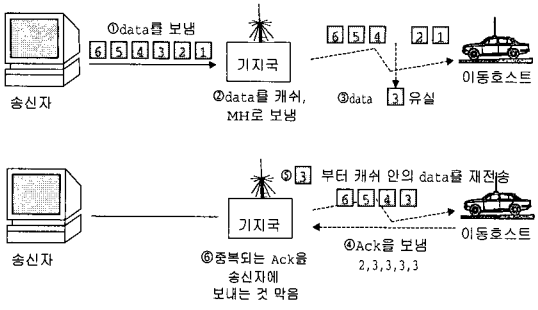

그림 3 TCP Reno에서 snoop만 사용한 경우의 예

기존의 TCP 메커니즘만을 사용한 Snoop 에이전트의 경우에는 패킷 유실시 송신측에 중복되는 ACK을 보내는 동안 수신측이 받은 패킷들에 대한 정보가 없으므로 기지국은 수신측이 이미 받은 패킷을 중복하여 보내는 경우가 발생하게 된다. [그림3]에서 보듯이 데이타 3이 유실된 경우 데이타 3부터 재전송을 수행하여 이미 수신측에 잘 도착한 데이타인 4, 5, 6도 재전송 하게 된다.

2.2.3 TCP SACK 옵션의 사용

송신자와 수신자의 종단간 TCP SACK 옵션을 지원하는 TCP를 사용하게 되면 무선 링크에서 생긴 버스티한 패킷 유실에 대해서 TCP Reno나 New-Reno를 사용할 경우보다 더 좋은 성능을 나타낸다는 보고가 있다[6]. 그러나 종단간 TCP SACK 만을 사용할 경우에는 무선 링크에서 발생한 패킷 유실의 재전송을 위해 종단간 재전송을 실시하여 유선 링크까지 영향을 받게 된다. 즉, 고정 호스트에서 기지국까지 잘 도착하고 무선 링크에서 유실된 데이타의 재전송을 위해서 다시 유선 링크에서의 대역폭이 사용되게 된다.

2.2.4 SMART 프로토콜

SMART(Simple Method to Aid Retransmissions) 프로토콜[20]에서는 ACK을 유발시킨 패킷의 일련 번호를 ACK에 포함시켜 수신자가 지금까지 연속적으로 손실 없이 받은 패킷의 마지막 일련 번호와 그 ACK을 발생시킨 패킷의 시작 일련 번호를 비교하여 같지 않으면 송신자가 그 두 번호 사이의 패킷들이 유실된 것으로 보고 재전송을 한다. SACK TCP에 비해 ACK을 만들거나 전송하는데 오버헤드가 적다는 장점이 있으나 망 내에서 패킷들의 순서가 뒤바뀌어 전송된 경우에 대해서도 유실된 것으로 보고 재전송을 하므로 융통성이 떨어진다. SACK TCP의 경우에는 중복 ACK이 3개 도착할 때까지는 재전송을 하지 않으므로 그 동안 도착한 패킷들의 순서가 뒤바뀌어 있어도 SACK 정보를 통해 이를 재전송하지 않을 수 있다.

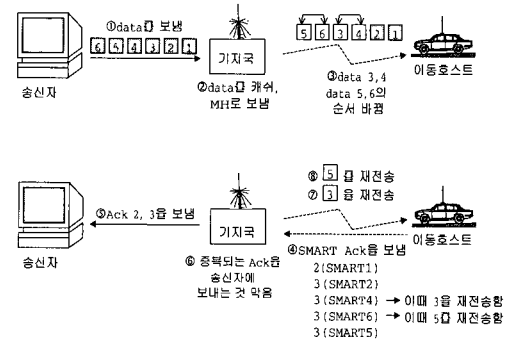

그림 4 SMART 프로토콜과 Snoop을 이용하는 경우의 중복 재전송 예

2.2.5 무선 링크에서 SMART 프로토콜과 Snoop을 이용하는 경우

무선 링크에서 SMART 프로토콜을 사용하고 기지국에 Snoop 모듈을 두어 기지국에서 빠른 지역 재전송을 실시하고자 하는 방안이 연구되었다[6]. 이 경우 기존의 종단간 SMART 프로토콜만 사용한 경우와 TCP Reno와 Snoop 프로토콜을 사용한 경우에 비해 성능이 뛰어남이 알려져 있다[5]. 그러나 SMART 프로토콜을 사용하는 경우에도 이동 호스트가 송신자인 경우에 기지국에 제대로 도착하지 못한 데이타의 정보를 NACK (Negative Acknowledgement)으로 만들어 보내기 위해서 TCP SACK 옵션을 사용할 것을 제안하고 있다. 여기서의 NACK은 앞서 말한 데이타 링크 계층에서의 NAK 패킷과 다른 수송계층 패킷을 의미한다. 이렇게 될 경우 TCP SACK 옵션 이외에 SMART를 위한 필드가 더 첨가되어져야 하고 TCP SACK 옵션 보다 간단한 처리 과정과 작은 헤더 크기를 사용한다는 장점이 사라지게 된다. 또한 SMART 프로토콜을 사용할 경우에는 망 내에서의 패킷들이 순서대로 올 경우만 고려하므로 패킷들의 순서가 조금 바뀐 경우에 대해서도 융통성이 없이 바로 재전송을 실시하게 된다. [그림 4]에서 데이타 3과 5는 수신자인 이동 호스트가 제대로 받았으나 SMART 프로토콜에 의해 중복 재전송이 실시된다. 그리고 무선 링크에서 SMART 프로토콜을 사용하고 Snoop 모듈을 사용하는 경우에 있어서 기지국의 캐쉬

가 다 차서 더 이상 캐쉬에 저장 할 수 없을 때 무선 링크에서 에러가 발생한다면 전체 연결의 효율이 떨어지게 된다.

그림 5 TCP SACK을 종단간 사용하고 기지국에서 SACK_Cache 쓰는 경우

3. 무선 링크에서 TCP 성능 향상에 적합한 모델

본 장에서는 무선 링크에서의 TCP의 성능 향상에 적합하고 무선 링크가 차세대 통신망의 근간이 될 ATM 망이나 위성 통신망과 연동될 경우에도 TCP의 효율을 높일 수 있는 모델로서 TCP SACK 옵션과 기지국에서의 캐쉬 기법을 함께 사용한 방법을 설명하고자 한다. 현재 인터넷의 백본 망으로 자리 잡아가고 있는 ATM 망에서의 TCP 성능에 관해 살펴보면 TCP SACK을 사용하였을 경우에 TCP Tahoe나 TCP Reno, TCP New-Reno를 사용하였을 경우보다 높은 성능을 나타낸다는 실험 결과가 많이 나오고 있다[7]. 그리고 지상의 호스트와 통신하는 위성 망에서의 TCP 성능 역시 TCP SACK에서 좋은 성능을 보이고 있다. 이는 대역폭이 큰 ATM 망이나 지연이 긴 위성 망과 같이 대역폭과 지연의 곱이 큰 망에서는 패킷 유실시 이를 발견하기 위해 재전송 타임아웃을 기다리는 동안 낭비되는 대역폭에 의해 망의 효율 저하가 크지만[8], TCP SACK은 이러한 망에서 여러 개의 패킷이 유실될 경우라도 잦은 재전송 타임아웃을 유발시키지 않고 빠른 시간 내에 유실된 패킷들을 회복할 수 있기 때문이다.

본 논문에서는 지연과 대역폭이 큰 WAN 환경과 패킷 유실이 잦은 무선 환경에서 좋은 성능을 나타내는 수송 계층 프로토콜인 TCP SACK을 송신자와 수신자 종단간에 사용하고자 한다. 또한 무선망에서의 이동성과 잦은 패킷 유실에 의한 영향을 유선망이 적게 받도록 하기 위하여 기지국에서 양방향으로 지나는 데이타를 캐쉬에 저장하면서 SACK에 담긴 수신자 정보를 이용하여 지역 재전송을 제공해 주는 방법을 무선망에 적합한 모델로 여기고 이를 타 방법과 성능 비교하고자 한다(그림5). 이 모델에서는 기지국에서 캐쉬 내의 데이타를 이용하여 지역 재전송을 할 경우에 SACK 정보를 사용하여 유실된 패킷만 재전송 하므로 중복되는 재전송에 의한 무선 링크의 불필요한 낭비를 막을 수 있다. 무선 링크에서 SMART 프로토콜을 사용하고 Snoop 모듈을 사용하는 경우에는 캐쉬가 다 찼을 때 효율이 심하게 떨어지게 되지만 이 경우에는 종단간 SACK 정보를 사용하여 종단간 선택적 재전송도 가능하게 된다. 그리고 패킷의 순서가 조금 뒤바뀌어 도착하는 경우에도 3번의 (또는 지역 재전송 임계값만큼의) 중복 ACK이 올 때까지 재전송을 수행하지 않으므로 더 융통성 있게 대처할 수 있다. [그림 6]에서 보듯 3과 4, 5와 6의 순서가 바뀌어 전송되었으나 SACK 정보를 포함한 경우에는 중복되는 3번의 ACK을 받기 전에는 재전송을 하지 않으므로 수신자가 이미 받은 데이타인 3과 5에 대해 [그림 4]의 SMART 프로토콜과는 달리 재전송을 하지 않는다.

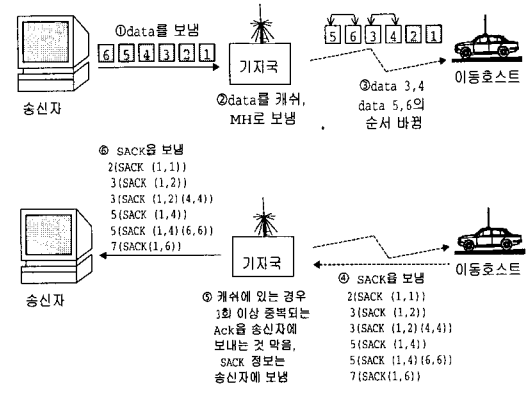

그림 6 종단간 TCP SACK과 기지국에서의 캐쉬를 이용하는 경우의 작동 예

3.1 기지국에서의 작동 메커니즘

기지국에서 SACK 정보를 사용하여 캐쉬내의 패킷을 지역 재전송 하기 위해서 SACK_Cache 모듈을 제안하고자 한다. 이 모듈은 크게 두 개의 프로시져 SACK_Cache_Data와 SACK_Cache_Ack으로 나누어 볼 수 있다.

SACK_Cache_Data는 고정 호스트 (Fixed Host: FH)에서 이동 호스트 (Mobile Host: MH)로 데이타를 전송할 경우의 처리를 해 주는 프로시져이고 SACK_Cache_Ack은 이동 호스트가 받은 데이타에 대한 ACK을 보냈을 경우 처리해 주는 프로시져이다. 프로시져들의 작동 원리를 이해하기 위하여 송신자는 고정 호스트

이고 수신자는 이동 호스트로 생각한다. 이 프로시져들의 작동 메커니즘은 다음과 같다.

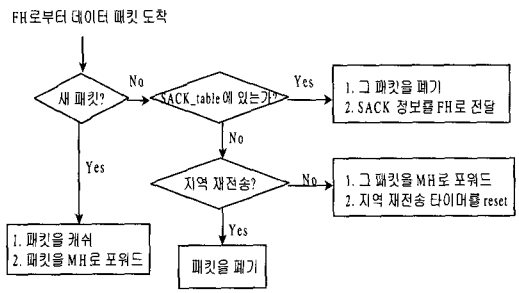

그림 7 SACK_Cache_Data의 작동 메커니즘

3.1.1 SACK_Cache_Data의 작동

SACK_Cache 모듈에서 SACK_table은 기지국 내에서 SACK 정보를 유지하고 있는 테이블이다. 여기에는 그 SACK을 유발시킨 패킷의 일련 번호와 수신측이 받은 데이타 블록의 리스트가 범위 형태로 들어 있다. 그러나 TCP 헤더 크기의 제약에 의해 SACK 옵션에 포함될 수 있는 SACK 블록은 3개에서 4개가 된다. 이 테이블은 새로운 SACK을 받을 때마다 수신측의 버퍼에 어떤 패킷이 있는지 최근 정보로 갱신된다.

SACK_Cache_Data의 작동을 살펴보면 먼저 고정 호스트로부터 기지국으로 데이타 패킷이 도착하면 그 패킷이 새로 도착한 패킷인지 송신자가 재전송한 패킷인지를 살펴본다. 만약 새로 도착한 패킷이라면 이를 캐쉬에 넣고 그 패킷을 이동 호스트로 포워드 한다. 만약 송신자가 재전송한 패킷이라면 그 패킷이 SACK 되었는지를 기지국의 SACK_table을 살펴보아 SACK 되었으면 그것은 수신자가 이미 받은 패킷이므로 폐기시키고 송신자에게 SACK을 보내어 수신자 측의 올바른 정보로 재전송 큐의 정보를 갱신하도록 한다. 이렇게 해줌으로써 수신자가 데이타를 받았으나 기지국에서 송신자로 ACK이 가던 도중에 유실된 경우와 같은 때에 무선 링크로의 불필요한 재전송을 막을 수 있다. 이러한 경우는 고정 호스트에서 이동 호스트로 보낸 패킷이 제대로 도착하여 이동 호스트가 그에 대한 ACK을 기지국에 보냈는데 이 ACK이 기지국에서 고정 호스트로 가던 중에(즉 유선망에서) 유실되어 고정 호스트에서 패킷을 재전송한 경우에 해당된다. 기지국은 이미 이것이 이동 호스트에 의해 SACK 되었음을 알게 되고 불필요하게 무선망으로 재전송을 수행하지 않고 고정 호스트 쪽으로 SACK 정보를 보내어 고정 호스트의 SACK 테이블 갱신에 사용하도록 한다. 만약 기지국의 SACK_table에 없다면 기지국에서 지역 재전송한 패킷인지 살펴보아 맞으면 이미 기지국이 이동 호스트로부터의 중복되는 ACK에 의해 지역 재전송을 수행하고 있는 중이므로 곧 ACK이 도착될 것이 예상되므로 그 패킷을 폐기하고 아니면 그 패킷을 이동 호스트로 포워드 시키고 지역 재전송 타이머를 리셋 시킨다.

3.1.2 SACK_Cache_Ack의 작동

SACK_Cache_Ack의 작동을 살펴보면 이동 호스트로부터 SACK 정보를 담은 ACK이 도착하면 그 ACK이 중복된 ACK인지 새로운 ACK인지를 살펴본다. 반복되는 ACK인지 확인하는 부분은 새로운 번호이고 중복되지 않는 ACK을 받더라도 이미 더 큰 일련번호의 ACK을 받은 이후라면 새로운 ACK으로서의 의미가 없으므로 이러한 경우에 그 ACK을 폐기하기 위해서이다. 만약 새로운 ACK이면 기지국의 SACK_table을 갱신하고 버퍼내의 패킷을 제거한다. 그리고 지역 RTT 추정치를 갱신하고 고정 호스트로 SACK을 전달한다. 만약 중복된 ACK이면 일단 그 ACK에 새로 포함된 SACK 정보로 기지국의 SACK_table을 갱신하고 그 중복 횟수가 지역 재전송 임계 회수보다 크면 그 해당 데이타가 캐쉬에 있는지 살펴본다. 만약 캐쉬에 있으면 해당 데이타를 재전송하고 SACK 정보만을 추출하여 송신자에게 보내어 송신자의 재전송 타이머가 만료되는 것을 막으면서 동시에 송신자 측의 SACK 테이블의 내용을 갱신할 수 있도록 해준다. 이는 기지국에서 SACK 정보를 추출하여 고정 호스트로 보낼 수 있도록 하는 것으로서 실제 이동 호스트가 보내는 SACK에는 기존의 ACK에 포함되어 있는 ACK의 일련 번호와 SACK 블록 정보가 포함되어 있다. 그러나 만약 기지국에서 이동 호스트가 생성하는 것과 같은 SACK을 생성한다면 이는 TCP가 종단간 프로토콜이라는 점을 위배하게 되므로, 이를 위배하지 않으면서 SACK 정보를 효율적으로 사용하여 무선 구간에서의 불필요한 대역폭 낭비를 없애기 위해서 기지국에서 고정 호스트로 SACK 정보만을 추출하여 보내게 된다. 이 정보를 받으면 고정 호스트는 아직 ACK 되지 않은 해당 데이타에 대한 재전송 타이머를 연장시키게 되고 자신의 SACK 테이블 정보를 갱신하게 된다. 만약 그 재전송 타이머가 만료될 때까지도 실제 ACK이 오지 않게 되면 TCP SACK의 종단간 재전송 메커니즘을 수행하게 된다. 그리고 데이타가 캐쉬에 없을 경우는 그 ACK을 고정 호스트에 전달한다.

위의 모듈의 작동 원리를 예를 들어서 살펴보면 다음과 같다.

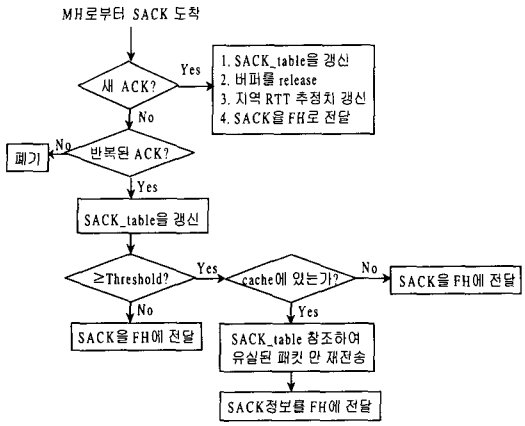

그림 8 SACK_Cache_Ack의 작동 메커니즘

[그림 9]에서는 data8과 data9가 고정 호스트에서 기지국까지는 잘 도착하였으나 기지국에서 이동 호스트로 가던 중에 유실된 경우이다. SACK_Cache를 사용할 경우에는 기지국에서 이동 호스트로 SACK 정보를 이용하여 유실된 것만 지역적으로 빨리 재전송을 해 줄 수 있다. 위와 같은 경우에 기존의 TCP를 지원하는 Snoop 프로토콜을 사용한다면 기지국에서는 이미 이동 호스트로 잘 도착한 패킷인 data10, data11, data12 까지 다시 전송하게 되므로 불필요한 재전송에 의해 무선 링크의 낭비가 발생하게 된다. 그리고 위와 같은 경우에 기지국에서의 특별한 기능을 두지 않고 종단간 TCP SACK만을 사용하는 경우에는 이동 호스트에서 보내는 SACK 정보를 포함한 중복 ACK이 고정 호스트까지 도착하여야 고정 호스트가 재전송을 하게 되므로 그 데이타가 재전송되기까지 많은 시간이 소요되게 된다 (그림 10).

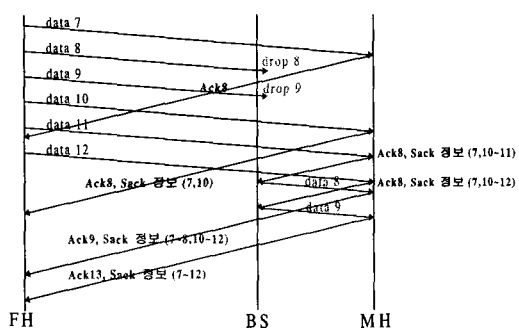

그림 9 SACK_Cache의 작동 예

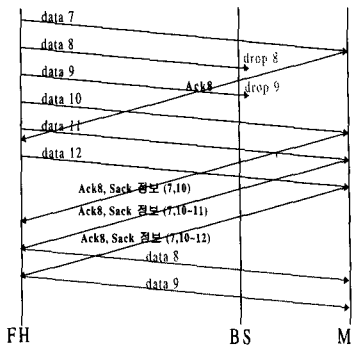

그림 10 종단간 SACK TCP만 사용한 경우의 작동 예

3.1.3 이동 호스트가 전송자일 경우의 작동

이동 호스트에서 고정 호스트로 데이타를 전송하는 경우에는 이동 호스트에서 기지국까지 데이타가 오던 도중에 유실될 확률이 높기 때문에 기지국에서 고정 호스트로의 지역적인 재전송은 그다지 큰 의미가 없게 된다. 따라서 이 경우에는 기지국의 SACK_Cache 모듈이 송신원으로부터 받은 데이타를 저장한 캐쉬 내의 불연속적인 일련번호를 찾아 자신이 받지 못한 데이타에 대한 NACK을 이동 호스트에 보내어 보다 빠른 재전송을 요구하도록 한다. 현재 TCP 구현에는 NACK에 대한 구현이 포함되어 있지 않다. 이러한 NACK의 구현 시에도 SACK 옵션 필드를 사용하여 할 수 있다[5]. NACK의 작동 예는 [그림 11]과 같다. 이동 호스트에서 data8을 보냈으나 기지국에 도착하기 전에 유실되었다. 이 경우 기지국에서는 어떤 지역 재전송 임계값 (위에선 2번) 만큼 동안 원하는 순서의 데이타가 오지 않으면 재전송 요구를 이동 호스트에 보내게 된다. 이렇게

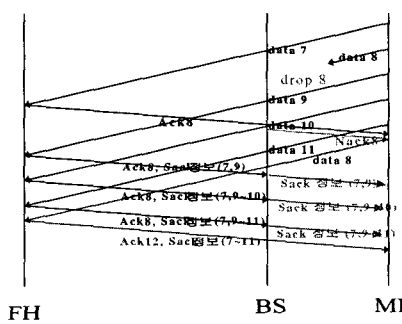

그림 11 NACK의 작동 예

해줌으로써 고정 호스트로 유실된 데이타가 더 빨리 재전송 될 수 있다.

이러한 SACK_Cache를 지원하기 위해 기지국에서 필요한 자료구조는 [그림12]와 같다. 각 SACK 블록들은 그 블록의 시작과 끝을 포함하고 있고 이 SACK 블록들의 리스트를 관리함으로써 수신자 버퍼에 대한 정보를 이용할 수 있다. 이를 구현하기 위해 기지국에서는 이동 호스트가 보낸 SACK 블록들이 들어오면 SACK_table 내의 적합한 위치에 삽입시키고 만약 그 블록이 기존의 SACK 블록과 오버랩 되거나 기존의 SACK 블록에 포함될 경우 이를 합치거나 삭제한다. 이렇게 생성된 SACK_table 내의 SACK 정보를 사용하면 유실된 패킷만 재전송 하므로 이동 호스트로의 불필요한 재전송을 막을 수 있다. 그리고 이동 호스트가 자신이 받은 데이타에 대한 ACK을 기지국에 보냈고 아직 그 ACK이 고정 호스트까지 도착하지 않은 상태로 고정 호스트가 데이타를 재전송하여 기지국에 다시 도착한 경우에는 이를 기지국에서 자신의 SACK 정보를 살펴보고 그 데이타 패킷을 폐기시킴으로써 무선 대역폭의 낭비를 막을 수 있다.

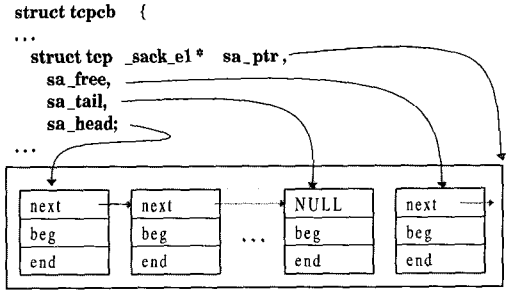

그림 12 송신자 측의 SACK을 위한 자료구조

이러한 SACK을 지원하기 위해서 사용되는 버퍼의 양과 SACK의 처리에 대한 오버헤드에 대하여서도 다양한 윈도우 크기와 패킷 유실률에 대해서 수신자 큐 안의 데이타 블록의 수에 관한 분포 함수 (distribution function)를 계산한 결과 큐의 길이는 길 필요가 없고 10개 정도의 원소로도 대부분의 크기의 윈도우에서 적합한 것으로 나타났다[9]. 이는 SACK을 위해 사용하는 큐의 길이를 상수 크기로 제한하므로 SACK의 처리에 대한 오버헤드에 상한을 지정하게 된다. 따라서 SACK을 사용하지 않는 경우의 중복되는 재전송에 의한 불필요한 자원의 낭비나 유실된 여러 개의 세그먼트를 재전송 타임아웃에 의존해 재전송 할 경우에 걸리는 긴 지연에 의한 대역폭 낭비를 고려해 본다면 큰 오버헤드는 아니다.

3.1.4 핸드오프시의 작동

핸드오프로 인한 패킷 손실은 핸드오프로 인한 지연을 줄임으로써 그 영향을 최소화할 수 있다. 이를 위해서 이동 호스트가 현재 있는 셀을 관장하는 기지국 외에 근처의 핸드오프가 예상되는 기지국으로도 데이타를 전송하는 방법을 사용한다. 즉 현재 속한 셀과 그 주변 셀의 기지국들이 하나의 멀티캐스트 그룹에 가입하고 송신자는 데이타를 이 멀티캐스트 그룹으로 전송한다. 핸드오프가 일어난 경우, 새로 이동한 셀의 기지국에 있는 SACK_Cache 모듈은 캐쉬의 데이타를 전송하게 된다. 즉 기지국간에 핸드오프와 관련된 정보가 직접적으로 전달되지 않고, 이미 캐쉬에 저장한 데이타를 이용해서 데이타 전송을 계속하므로 핸드오프 지연시간을 줄일 수 있게 된다. 이 기법은 기존의 종단간 TCP Reno를 사용하고 기지국에서 캐쉬를 사용하는 Snoop 기법에서 적용되는 방법과 동일하다. 종단간 TCP Reno를 사용할 경우에는 새로 이동해간 셀의 기지국이 캐쉬에 가지고 있는 데이타와 이동 호스트가 현재까지 수신한 데이타 사이에 차이가 생길 수 있다. 그러나 SACK_Cache를 사용하는 경우에는 이러한 경우에 수신자인 이동 호스트가 자신이 어느 패킷을 받고 어느 패킷을 받지 못하였는가 하는 정보를 포함하는 SACK을 생성하여 새로운 기지국에 보내줌으로써 그 정보에 의해 기지국에서 SACK 테이블을 갱신하여 중복되는 재전송을 막을 수 있다.

3.2 제시하는 모델의 특징

현재 TCP SACK 옵션은 아직 시험적인 구현의 단계이고 다양한 변형들이 구현되고 있다[10]. 그러나 앞으로 ATM 망이나 위성 통신망을 널리 사용하게 될 경우에 기존의 TCP 만으로는 만족스러운 성능을 나타낼 수 없다. 이러한 망에서 TCP SACK 옵션을 사용할 경우 더 좋은 성능을 보인다고 알려져 있다[21,22]. 그리고 무선 링크에서의 높은 패킷 에러율과 핸드오프 시의 지연에 의한 버스티한 패킷 유실 등에 대해서도 기존의 TCP의 혼잡 제어 메커니즘만으로는 망을 효율적으로 사용할 수 없다. 따라서 차세대 통신망에서는 TCP SACK이 널리 쓰이게 될 것으로 예상된다. 이미 TCP SACK 옵션이 구현된 윈도우즈 95용 FTP 프로그램이 개발되어 상용화되었고 마이크로소프트사의 윈도우즈 98에서도 SACK 옵션이 디폴트로 설정되어 있다[10]. 따라서 ATM 망과 위성 통신망, 그리고 개인 이동 통신망이 함께 사용될 차세대 통신망에서 위의 메커니즘

은 TCP 효율을 높이는 데 사용될 수 있다.

4. 시뮬레이션 및 결과 분석

본 장에서는 위에서 구현한 SACK_Cache 모듈을 사용하였을 경우의 TCP 성능을 분석하기 위하여 수행한 시뮬레이션에 대하여 기술한다. 시뮬레이션은 기지국에서 특별한 역할을 하지 않고 종단간 TCP Reno를 사용하는 경우, 기지국에서의 캐쉬 기법을 사용하고 종단간 TCP Reno를 사용한 경우(기존의 TCP Reno를 쓰는 Snoop 프로토콜의 메커니즘), 기지국에서 특별한 역할을 하지 않고 종단간 TCP SACK을 사용한 경우 마지막으로 기지국에서의 캐쉬 기법을 사용하고 종단간 TCP SACK 옵션을 사용하며 기지국에서 SACK 정보를 이용하는 경우의 성능을 비교하도록 한다.

4.1.1 망 모델 구조

일반화된 개인 이동 통신 환경을 나타내는 간소화된 구조로서 [그림13]에 도시한 것과 같은 유무선 링크의 연결을 고려한다. 즉, 고정 호스트는 유선망을 통해서 기지국에 연결되어 있고 이동 호스트는 기지국에 무선 링크를 통해서 연결되어 있다. 이 경우에 SNR의 변화에 따르는 무선 링크의 패킷 에러율에 대해서 위의 네 가지 메커니즘에서 얻을 수 있는 TCP의 유효 처리율을 비교해 보도록 한다. 망에서의 트래픽은 고정 호스트에서 이동 호스트로 데이타가 이동하고 이동 호스트에서 고정 호스트로 ACK이 이동하는 단방향 트래픽으로 한다.

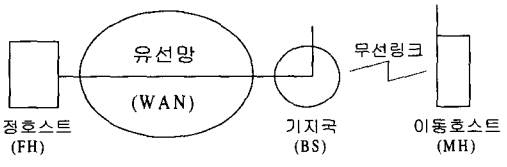

그림 13 개인 이동 통신 환경에서의 유무선 링크의 연결

4.1.2 무선 채널 모델링

무선 환경에서는 전송파가 여러 경로를 통해 수신기에 전달되므로 수신된 신호들의 위상차에 의해 신호가 감쇄되는 페이딩 현상이 발생하게 된다. 이때 직접 파가 전달되는 경우에는 포락선이 라이시안 분포를 가지기 때문에 라이시안 페이딩이라 하고 직접파가 없거나 약하고 반사파가 대부분인 경우는 레일리 분포를 가지기 때문에 레일리 (단기간) 페이딩이라 한다. 그리고 지형에 따라 천천히 신호의 세기가 변하는 것과 음영(shadow) 페이딩은 대수 정규 (장기간) 페이딩으로 설명된다[11]. 실제 모델을 적용할 경우에는 무선망의 셀의 크기에 따라 위의 세 가지 모델을 적용할 수 있다. 매크로 셀과 마이크로 셀의 경우에는 대수 정규 페이딩과 레일리 페이딩으로 모델링 될 수 있다. 하지만 마이크로 셀의 경우에는 셀의 크기가 작으므로 위치에 따라 직접파가 존재하는 경우와 존재하지 않는 경우가 있으므로 라이시안 페이딩과 레일리 페이딩의 성격을 다 가지고 있다. 피코 셀에서는 직접파가 존재하므로 라이시안 페이딩으로 모델링된다. 본 논문에서는 마이크로 셀 환경을 고려하므로 레일리 페이딩 모델로 설명할 수 있다.

본 논문에서는 TCP가 무선 링크에 의해 어떤 영향을 받는지가 관심사이므로 무선 채널은 레일리 페이딩 하에서 패킷을 보낼 때 그 패킷이 오류인가 아닌가를 나타낼 수 있으면 된다. 이런 오류 패턴은 레일리 페이딩 신호를 발생시켜 비트 오류 확률과 패킷 오류 확률을 구함으로써 얻을 수 있다.

우선 레일리 페이딩 신호는 Jakes가 기술한 방법에 의해 발생시킬수 있다[12]. 이 방법은 수신기에 도착하는 신호들의 도착각이 균일 분포를 갖는다 (즉, 파워 스펙트럼이 대칭이다)는 가정을 하여 복소수 가우시안 랜덤 프로세스를 발생시킨다. 발생된 신호의 절대값을 취하면 레일리 페이딩 신호가 되며 이 신호는 평균 파워가 0dB이고 파워 스펙트럼의 상관 분산은

$$\rho = J_0(2\pi f_d T) \tag{1}$$

이다. 여기서 f_d는 최대 도플러 주파수를 의미하고 J_0는 1종 0차 수정 베셀 함수이다. 페이딩 프로세스의 상관 관계는 정규화된 도플러 대역폭인 f_dT와 관계가 있다. T초 만큼 떨어져 있는 두 표본에 대해 f_dT값이 작을 때 (<0.1) 느린 페이딩이라 하고 f_dT값이 클 때 (>0.2) 빠른 페이딩이라 한다.

예로 반송파 주파수 (f_c)가 915MHz, 링크 용량 (C)이 2Mbps, 전파 속도 (c)가 3×10^8m/s이고 20km/h의 속도 (V)로 움직이는 경우 느린 페이딩으로 볼 수 있는 최대 패킷길이 L_{max}는 정규화된 도플러 대역폭이

$$f_dT = V\frac{f_c}{c}\frac{L}{C} < 0.1 \tag{2}$$

라는 관계식을 이용하면[13], 약 11803 bit (약 1475 byte)의 패킷 길이가 되는 것을 알 수 있다. 따라서 본 논문에서는 1200 byte의 패킷 크기를 가지고 실험하므로 이러한 환경에서는 대부분 느린 페이딩으로 볼 수

있고 따라서 같은 비트 오류 확률을 한 패킷 내에 적용할 수 있다.

이 때 패킷 길이를 L비트라 하면 링크 계층에서의 어떠한 에러 수정 기법을 사용하지 않은 경우의 한 패킷의 에러 확률은

$$PER(\overline{\gamma_b}) = 1-[1-P_2(\overline{\gamma_b})]^L \quad (3)$$

이다[17]. 여기서 P_2는 비트 에러 확률을 나타낸다.

비트 에러 확률은 변조 방식에 따라 달라지는데 많이 사용하는 이진 PSK인 경우

$$P_2(\overline{\gamma_b}) = Q(\sqrt{2\overline{\gamma_b}}) \quad (4)$$

와 같이 수신된 평균 SNR 즉 $\overline{\gamma_b}$의 함수로 나타낼 수 있다[16]. SNR은 평균 비트 에너지 대 잡음 전력비(signal-to-noise ratio)이며 다음 식과 같다.

$$\overline{\gamma_b} = \frac{\varepsilon_b}{N_0} E(\alpha^2) \quad (5)$$

여기서 α는 정규화된 레일리 분포를 따르며 $E(\alpha^2)$는 α^2의 평균값이다.

함수 Q(x)는

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt \quad (6)$$

로 정의되어 있다[17].

본 논문에서는 f_dT를 위의 (식 2)에서 계산된 값인 0.08133으로 고정시켜 두고 SNR을 변화시킴에 따라 달라지는 비트 에러율을 계산하고 그에 따른 패킷 에러 확률을 구한다. 그리고 난수를 발생시켜 그 패킷 에러 확률에 대해 각각의 패킷이 에러가 발생하였는지를 판단한다[15].

위에서 설명한 모델보다 더 일반적인 모델을 얻기 위해 [13][14]의 연구들을 바탕으로 한 1차 마코비안 오류 모델에 대해 검토해 본 결과 페이딩 주파수가 낮아질수록 마코비안으로 근사화 시킬 때 군집 오류 길이 분포가 잘 맞지 않으므로 TCP 패킷이 무선 링크에서 겪는 지연을 정확히 반영할 수 없다는 연구가 있다[15]. 따라서 1차 마코비안 모델은 링크 계층의 처리율을 연구하는데는 적합하지만 TCP의 성능을 연구하는 데는 적합하지 않으므로 여기서는 위의 단계를 거쳐 얻어지는 패킷 에러 패턴을 사용하도록 하겠다.

4.2 시뮬레이션 파라미터

시뮬레이션에 사용된 고정 파라미터는 [표 1]과 같다. SNR을 네 가지로 변화시킴에 따라 무선 링크에서의 패킷 에러 확률을 변화시키면서 실험을 실시하였다. 그리고 TCP의 최대 윈도우 크기는 8Kbyte인 경우와 26Kbyte인 경우 두 가지로 나누어서 실험을 하였다. TCP 패킷의 크기는 1200 byte로 설정하였다. 그리고 TCP 송신자의 응용 프로그램에는 윈도우가 허락하는 한 계속 보낼 데이타가 있는 것으로 가정하였다[7]. 즉, 시뮬레이션 시간 동안 보낼 데이타는 계속해서 기다리고 있으며 윈도우 크기에 따라 그 만큼의 데이타가 전송되어 진다. 기지국에서의 캐쉬 기능을 사용하는 경우의 캐쉬 크기는 25 세그먼트(30Kbyte)로 설정하였다.

표 1 시뮬레이션 고정 파라미터 값

유선 링크의 지연	5 ms/Km
무선 링크의 지연	7 ms/Km
고정 호스트에서 기지국까지의 거리 (유선 링크의 거리)	10 Km
기지국에서 이동 호스트까지의 거리 (무선 링크의 거리)	500 m
TCP 데이타 패킷 크기	1200 Byte
ACK 패킷 크기	40 Byte
무선 링크 대역폭	2 Mbps
유선 링크 대역폭	10 Mbps
TCP 타이머 소밀도	100 ms

이 시뮬레이션에서 위의 네 가지 TCP 메커니즘에 대해 비교하고자 하는 값은 SNR의 변화에 따른 TCP 유효 처리율과 수신지에서 받은 패킷 중 유효한 패킷의 비율, 수신지에서 받은 패킷 중 중복된 패킷의 비율, 그리고 동일 윈도우 내에서 하나 이상의 패킷 유실이 일어날 경우의 TCP 유효 처리율이다. 여기서 유효 처리율이란 단위 시간 동안 TCP 수신자가 받은 데이타중 순서가 어긋나 연속되지 않거나 중복된 패킷을 제외한 실제 유효한 데이타의 양을 링크 대역폭으로 나눈 값을 의미한다[19].

4.3 시뮬레이션 결과

본 절에서는 앞의 모델을 사용하여 수행한 시뮬레이션 결과를 분석한다. 기지국에서 특별한 역할을 하지 않고 종단간 TCP Reno를 사용하는 경우를 Reno라고 표기하고, 기지국에서의 캐쉬를 사용하고 종단간 TCP Reno를 사용한 경우를 Reno Cache, 기지국에서 특별한 역할을 하지 않고 종단간 TCP SACK을 사용한 경우를 SACK, 기지국에서의 캐쉬를 사용하고 종단간 TCP SACK 옵션을 사용하며 기지국에서 SACK 정보를 이

용하는 경우를 SACK Cache라고 표기한다. 각각의 경우에 대한 TCP 유효 처리율 대 대역폭의 비, 수신측이 받은 패킷 중 중복되어 받은 패킷의 비율, 수신측이 받은 패킷 중 유용한 패킷의 비율과 한 윈도우 내에서 한 개 이상의 패킷 유실시의 유효 처리율을 비교하여 전체적인 종단간 TCP의 성능을 비교하도록 한다. 여기서 TCP 유효 처리율이라 함은 단위 시간 동안 실제 수신측이 받은 데이타 패킷 중에 중복되거나 연속되지 않아 실제 상위 계층에서 사용하지 못하는 데이타를 제외한 유효한 패킷의 양을 의미한다.

4.3.1 SNR 변화에 따른 TCP 유효 처리율

네 가지의 TCP 메커니즘에서 SNR을 변화시킴으로써 무선 링크의 패킷 에러율을 변화 시켰을 경우의 TCP 유효 처리율을 알아보았다. 여기서 유효 처리율이란 단위 시간당 수신측에서 받은 데이타 중 중복되거나 연속적이지 않은 데이타를 제외한 실제 상위 계층에서 유용한 데이타의 양을 말한다. 이 그래프에서는 그러한 데이타의 양을 링크의 대역폭으로 나눈 값을 나타내었다. 그리고 윈도우 크기를 8Kbyte와 26Kbyte로 변화시키면서 TCP의 유효 처리율을 비교하여 보았다.

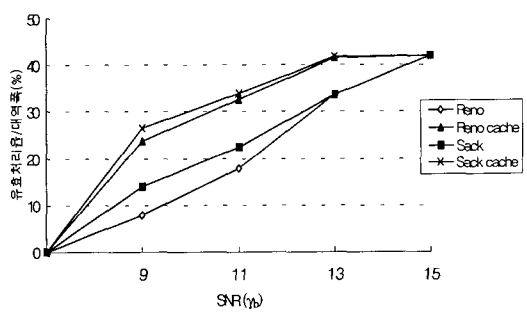

그림 14 윈도우 크기 8Kbyte인 경우의 TCP 유효 처리율

[그림 14]는 윈도우 크기가 8Kbyte일 경우이고 [그림 15]는 윈도우 크기가 26Kbyte일 경우이다.

SNR이 작을 경우 윈도우의 크기에 상관없이 종단점에서만 재전송을 하는 TCP Reno와 TCP SACK에 비해 기지국에서 캐쉬를 이용하여 재전송을 실시하는 TCP Reno Cache와 TCP SACK Cache를 사용할 경우에 TCP 유효 처리율이 더 높은 것으로 나타났다. 그리고 윈도우의 크기가 커질수록 더 TCP 유효 처리율이 좋아진 것으로 나타났다. 이는 윈도우 크기가 클수록 더 많은 양을 한꺼번에 전송 할 수 있으므로 망의 대역폭을 더 빨리 효율적으로 이용할 수 있기 때문이다.

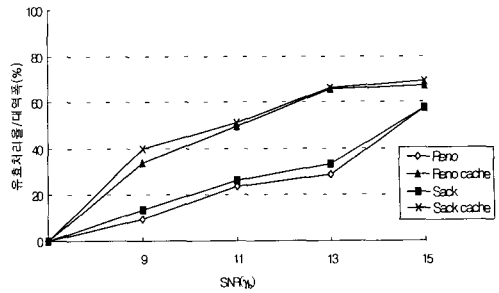

그림 15 윈도우 크기 26Kbyte인 경우의 TCP 유효 처리율

그리고 SNR이 작을 경우에는 한 윈도우 내에서도 여러 개의 패킷 유실이 발생하게 된다. 위의 실험 결과를 살펴보면 SNR이 작은 경우에 (즉, 무선 채널의 상태가 더 나쁜 경우에) SACK Cache와 Reno Cache의 성능이 캐쉬를 사용하지 않은 경우에 비해 향상되었음을 알 수 있다. 이는 한 윈도우 내에서 여러 개의 패킷 에러가 발생한 경우에도 지역 재전송을 사용하여 더 효율적으로 처리해 주기 때문이다.

즉 기지국에서의 지역 재전송을 실시하는 것이 TCP 패킷의 에러를 송신자의 재전송 타임아웃 없이 빨리 회복하여 유효 처리율을 향상시킴을 알 수 있다.

4.3.2 수신원에 중복되어 도착한 패킷의 비율

본 절에서는 캐쉬 기능을 사용한 두 TCP 메커니즘에 대해 수신원에 중복되어 도착한 패킷의 비율을 비교하고자 한다. SACK 옵션을 사용하지 않는 경우에 TCP 송신자가 패킷 유실을 감지하게 되면 TCP가 Go-back-N 방식의 재전송을 하게 된다. 즉, 유실된 패킷 이후의 패킷들을 다 재전송 하게 된다. 송신자는 유실된 패킷에 대한 중복 ACK을 계속 받게되고 3번의 중복 ACK을 받으면 재전송을 실시한다. 그리고 그 이후로 중복 ACK을 계속 받는 동안에는 재전송한 패킷 이후에 어떤 패킷을 재전송 해야 하는지 또는 새로운 패킷을 보내야 하는지 모른다. SACK 옵션을 사용하게 되면 이러한 경우에 먼저 유실된 패킷들을 다 재전송하고 나서 새로운 패킷들을 전송하게 된다. 이러한 SACK의 메커니즘을 기지국에서 재전송하는 경우에 SACK 정보를 이용하는 SACK Cache와 SACK 메커니즘을 쓰지 않는 Reno Cache의 두 가지 경우에 대해 수신자가 받은 패킷 중에 중복된 패킷의 비율을 구하여 비교하였다.

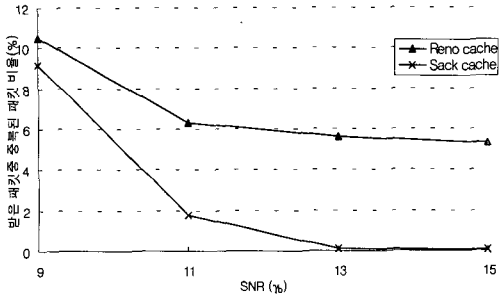

그림 16 윈도우 크기 8Kbyte인 경우의 중복된 패킷 비율

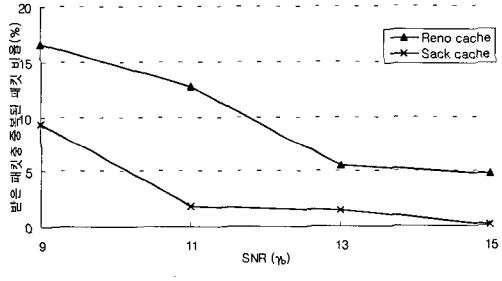

그림 17 윈도우 크기 26Kbyte인 경우의 중복된 패킷 비율

[그림 16]은 윈도우 크기 8Kbyte인 경우, [그림 17]은 윈도우 크기 26Kbyte인 경우의 중복된 패킷 비율이다. Reno Cache의 경우에는 윈도우 크기가 큰 경우에 더 현저하게 중복되어 받은 패킷의 수가 늘어났다. 그러나 SACK Cache의 경우에는 중복되어 받은 패킷의 비율이 윈도우 크기에 더 적은 영향을 받는 것으로 나타났다. 그리고 Reno Cache의 경우에 비하여 모든 SNR에서 매우 낮은 중복 비율을 나타내고 있음을 알 수 있다. SACK Cache의 경우에는 SACK 정보를 사용하여 중복되는 전송률을 낮추어 무선 링크의 효율을 높일 수 있다.

4.3.3 수신원에 도착한 패킷 중 유효한 패킷의 비율

[그림 18]과 [그림 19]는 수신원에 도착한 총 패킷 중에 유효한 패킷의 비율을 나타낸다. 수신원이 받은 총 패킷은 처리율과 관련 있고 유효한 패킷은 유효 처리율과 관련이 있다. 즉, 이는 무조건 얼만큼의 패킷을 받았느냐가 아니라 받은 것 중 유효한 것이 어느 정도인가를 나타내는 것이다. [그림 18]의 그래프를 살펴보면 SNR이 작은 경우와 큰 경우 모두에서 SACK Cache 메커니즘을 사용한 경우에 가장 좋은 성능을 보였다. 그리고 여기서 SNR이 낮아서 패킷 에러율이 상대적으로 높은 경우에는 Reno Cache가 종단간 SACK TCP를 사용한 경우보다 좋은 성능을 보이는데 비해 패킷 에러율이 낮은 경우에는 Reno Cache와 종단간 SACK TCP를 사용한 경우와 거의 비슷한 정도의 효율을 나타냈다. 이는 비록 종단간 SACK TCP를 사용할 경우에 캐쉬를 사용하여 기지국에서 재전송 하는 경우보다 지연은 더 생기지만 SACK 정보를 사용하여 필요한 것만 재전송하므로 수신원이 받은 총 패킷 수에서 유효한 패킷의 비율이 높게 나타난 것이다. 이는 윈도우 크기가 더 커진 경우인 [그림 19]에서 더 잘 나타나 있다. 이는 송신자의 윈도우 크기가 큰 경우에 더 많은 양의 패킷을 한 번에 전송하게 되므로 이때 SACK을 사용하지 않고 Go-back-N 방식으로 재전송을 하게 되면 재전송해야하는 패킷의 수가 많아져서 발생하는 현상이다. 그

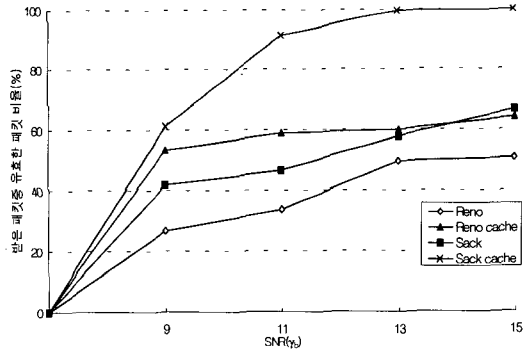

그림 18 윈도우 크기 8Kbyte인 경우 수신원에 도착한 패킷 중 유효한 패킷의 비율

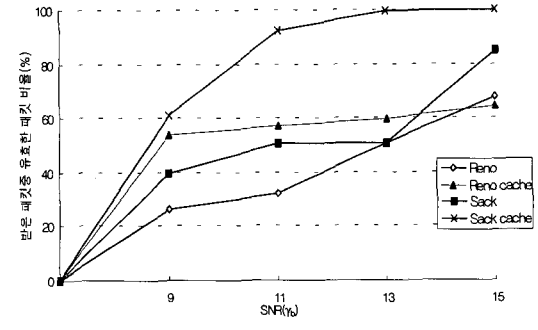

그림 19 윈도우 크기 26Kbyte인 경우 수신원에 도착한 패킷 중 유효한 패킷의 비율

리고 SNR이 낮아 패킷 에러율이 높을 때에 Reno Cache가 종단간 TCP SACK만을 사용한 경우에 비해 성능이 더 좋은 이유는 TCP SACK만을 사용한 경우에는 무선에서 발생한 패킷 에러에 대해 지역적으로 빨리 해결하지 않아 종단간 패킷 유실을 알리는데 걸리는 지연시간 동안 계속 패킷을 보내게 되므로 발생하는 현상으로 여길 수 있다.

4.3.4 동일 전송 윈도우 내에서 한 개 이상의 패킷 유실시 TCP 유효 처리율

[그림 20]은 동일 전송 윈도우 내에서 한 개 이상의 패킷이 유실되는 경우의 TCP 유효 처리율을 나타낸다. 이 시뮬레이션 결과는 윈도우 크기가 26Kbyte인 경우에 한 윈도우 내에서 인위적으로 2개씩의 패킷 유실을 발생시킨 경우를 실험 한 결과이다. [그림 20]은 각각의 TCP 메커니즘의 유효 처리율을 TCP Reno의 유효 처리율로 나눈 값을 나타낸 것이다. 종단간 TCP SACK만을 사용한 경우 TCP Reno를 사용한 경우보다 유효 처리율이 2배, 그리고 Reno Cache를 사용한 경우에는 5배 그리고 SACK Cache를 사용한 경우에는 6배 정도로 높은 유효 처리율을 나타내었다. 이는 TCP Reno의 경우에는 한 윈도우 내에서 여러 패킷이 유실된 경우에 재전송 타임아웃을 기다리는 시간 동안 대역폭의 낭비가 심하기 때문으로 볼 수 있다. 그리고 Cache를 기지국에 두어 지역 재전송을 실시 할 경우에 유효 처리율이 좋은 것을 볼 수 있다.

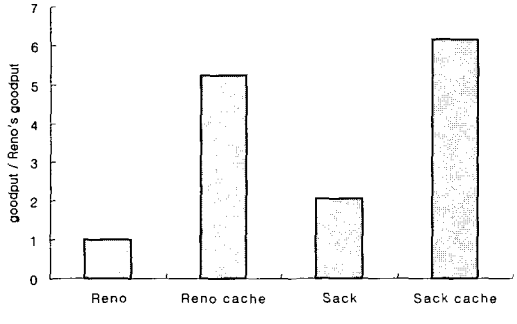

그림 20 동일 전송 윈도우 내에서 한 개 이상의 패킷 유실시 TCP 유효 처리율

5. 결론 및 향후 연구 과제

본 논문에서는 이동 통신에서 사용되는 무선 링크에서의 여러 가지 TCP 메커니즘들에 대해 비교하였다. 이러한 무선 환경은 유선보다 높은 에러율을 나타내며 핸드오프에 의한 단절과 패킷 유실이 발생하게 된다. 그러나 기존의 망에서 널리 쓰이고 있는 수송 계층 프로토콜인 TCP는 패킷 유실이 주로 망 내부의 혼잡에 의해서만 일어난다고 가정하고 작동하기 때문에 이러한 무선의 특성을 고려하지 않고 모든 패킷 유실에 대해서 혼잡 제어와 회피 알고리즘을 작동시키게 되고 이는 무선 환경과 연결된 망에서의 종단간 처리율을 저하시키게 된다.

본 논문에서는 이러한 무선망에 적합한 TCP 모델을 찾기 위해 기존에 제시된 방안들에 대해 알아보고, 각 방법의 특성과 문제점을 분석하였으며, 시뮬레이션을 통하여 성능을 비교하였다. 시뮬레이션 결과 종단간 SACK 옵션을 사용하는 TCP를 사용하고 기지국에서 이 무선에서의 패킷 에러로 인한 패킷 유실을 SACK 정보를 이용하여 기지국의 캐쉬로부터 지역 재전송하는 모델이 가장 좋은 성능을 나타냄을 알 수 있었다.

여러 메커니즘들의 성능을 종단에서의 TCP 유효 처리율과 수신지에서 중복되어 받은 패킷의 비율, 수신지에서 받은 패킷 중 유효한 패킷의 비율, 동일 전송 윈도우 내에서 하나 이상의 패킷 유실이 일어난 경우의 유효 처리율을 기준으로 하여 비교하였다. 각각의 실험에 대해 SNR을 변화시켜 패킷 에러율을 달리 해보고 윈도우 크기를 변화시켜 보면서 성능을 평가하였다.

종단에서의 TCP 유효 처리율의 경우 윈도우 크기가 큰 경우가 더 높은 유효 처리율을 나타내었고 SNR이 작을수록, 즉 패킷 에러 확률이 높을 수록 SACK 옵션을 사용한 경우와 그렇지 않은 경우의 성능 차이가 큰 것으로 나타났다. 이는 종단간 TCP SACK 옵션을 사용한 경우와 TCP Reno를 사용한 경우를 비교했을 때나 기지국에서의 캐쉬 기법을 사용하는 SACK Cache와 Reno Cache의 성능을 비교했을 경우에 있어서 마찬가지로 나타났다. 그리고 기지국에서의 캐쉬 기법을 사용한 경우가 그렇지 않은 경우보다 유효 처리율이 높으며 SACK 정보를 이용하는 경우가 그렇지 않은 경우보다 높은 것으로 나타났다. 본 논문에서 무선 링크를 포함한 망에서 적합한 TCP 모델로 제시하는 종단간 TCP SACK 옵션을 사용하며 기지국에서 캐쉬 기법을 사용하여 지역 재전송을 하는 경우에 성능이 가장 좋은 것으로 나타났다. 중복되어 받은 패킷의 비율의 경우 SACK 옵션을 사용한 캐쉬 기법이 SACK 옵션을 사용하지 않은 캐쉬 기법에 비해 월등히 낮은 것으로 나타났다. 이는 SACK 정보를 사용하여 기지국에서 중복되는 재전송 횟수를 줄였기 때문이다. 동일 전송 윈도우 내에서 하나 이상의 패킷이 유실된 경우에는 기존의 재

전송 타임아웃에 의지해야 하는 TCP Reno에 비해 TCP SACK이 더 좋은 성능을 나타내었으며 기지국에서 캐쉬를 사용하지 않아 직접 송신자가 재전송을 행해야 하는 TCP SACK에 비해 기지국에서 캐쉬를 사용하는 Reno Cache와 제안한 방법인 SACK Cache가 더 높은 성능을 보였다.

시뮬레이션을 통하여 제안한 모델의 유효 처리율이 타 방식에 비해 향상되었고 무선 링크로의 중복된 재전송의 비율이 줄었음을 알 수 있었다. 향후 연구 과제로는 본 논문에서 제시한 모델을 사용하여 ATM 망과 무선 링크의 연동으로 생각하였을 경우의 성능 평가를 들 수 있다. 차세대 통신망의 기반이 될 ATM 망이 유선망을 구성하고 개인 휴대 통신에 필수 구성 요소인 무선 링크가 종단에 연결된 경우의 TCP 성능을 UBR 서비스를 사용하였을 경우 ATM 스위치에서의 다양한 패킷 폐기 기법을 사용하였을 경우에 대해 평가해 볼 수 있으며 또한 ABR 서비스를 사용하였을 경우에 ATM 계층에서의 다양한 트래픽 제어 기법과의 상호 작용을 연구할 필요도 있다. 그리고 유·무선 링크간 대역폭의 비대칭적 요소에 대해서도 지속적인 연구가 필요할 것이다.

참 고 문 헌

[1] V. Jacobson, "Congestion Avoidance and Control," in *Proc. SIGCOMM '88*, ACM, USA, pp. 314-329, 1988.

[2] W. R. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmission, and Fast Recovery Algorithms," IETF, RFC 2001, Jan. 1997.

[3] E.Amir et al. "Efficient TCP over networks with wireless links," in *Proc. HotOS-V*, May 1995.

[4] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," In *Proc. 15th International Conf. on Distributed Computing Systems (ICDCS)*, May 1995.

[5] H. Balakrishnan, S. Seshan, and R.H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," ACM Wireless Networks, 1(4), Dec. 1995.

[6] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, R. H. Katz, "A Comparison of Mechanisms for Improving TCP performance over Wireless Links," IEEE/ACM Transactions on Networking, Dec. 1997.

[7] R. Goyal, R. Jain, S. Kalyanaraman,"TCP Selective Acknowledgments and UBR Drop Policies to Improve ATM-UBR Performance over Terrestrial and Satellite Networks," in *Proc. ICCCN97*, Sep. 1997.

[8] V. Jacobson, R. Braden, D. Broman, "TCP Extensions for High Performance," RFC 1323, May 1992.

[9] Luigi Rizzo , "Issues in the implementatioin of selective acknowledgemets for TCP," Draft, Jan. 1996.

[10] "Experimental TCP Selective Acknowledgment Implementations," Pittsburgh Supercomputing Center (PSC), http://www.psc.edu/networking/all_sack.html

[11] A. DeSimone et al., "Throughput performance of transport-layer protocols over wireless LANs," in *Proc. IEEE GLOBECOM '93*, pp. 542-549, 1993.

[12] W. C. Jakes, Jr., "Microwave mobile communications," John Wiley & Sons, 1974.

[13] M. Zorzi et al., "On the accuracy of a first-order markov model for data transmission on fading channels," in *Proc. ICUPC '95*, 1995.

[14] H. S. Wang, "On verifying the first-order markovian assumption for a rayleigh fading channel model," in *Proc. ICUPC '94*, pp. 160-164, 1994.

[15] 전병곤, 이병기, "PCN 환경에서의 TCP 성능 분석", 한국통신학회 논문지, 제 23권 2호, 1998. 2.

[16] John G. Proakis, "Digital Communications," Third Edition, McGRAW-HILL, 1995.

[17] Young Yong Kim, San-qi Li, "Modeling Fast Fading Channel Dynamics for Packet Data Performance Analysis," in *Proc. IEEE INFOCOM '98*, pp. 1292-1300, 1998.

[18] K. Y. Wang, S. K. Tripathi, "Mobile-End Transport Protocol : An Alternative to TCP/IP Over Wireless Links," in *Proc. IEEE INFOCOM '98*, pp. 1046-1053, 1998.

[19] Romanow, A., and Floyd, S., "Dynamics of TCP Traffic over ATM Networks," in *Proc. IEEE JSAC*, pp.633-641, May. 1995.

[20] S. Keshav, S. Morgan, "SMART Retransimssion: Performance with Overload and Random Losses," in *Proc. INFOCOM '97*, 1997.

[21] William Stallings, "High-Speed Networks: TCP/IP and ATM design principles," 1998.

[22] P. Johansson, E. Wedlund, J. M. Karlsson, "Interaction Between TCP Flow Control and ABR Rate Control," Proceedings of IEEE ATM '97 , 1997.

[23] M. Mathis, J. Madhavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options," IETF, RFC 2018, Oct. 1996.

[24] W. Richard Stevens, "TCP/IP Illustrated, Volume 1 : The Protocols," Addison-Wesley, Dec. 1993.

[25] Gary R. Wright, W. Richard Stevens, "TCP/IP Illustrated Volume 2 : The Implementation," Addison-Wesley, Jan. 1995.

[26] Fall K., Floyd S., "Comparisons of Tahoe, Reno, and Sack TCP," ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z, Dec. 1995.
[27] Floyd S., "Issues of TCP with SACK," ftp://ftp.ee.lbl.gov/papers/issues_sa.ps.Z, Jan. 1996.
[28] Jacobson, V. , R. Braden, "TCP Extensions for Long-Delay Paths," IETF, RFC 1072, Oct. 1988.
[29] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification," IETF, RFC 793, DARPA, Sep.1981.
[30] Ramon Caceres, Liviu Iftode,"Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE Journal on Selected Areas in Communications, June. 1996.
[31] UCB/LBNL Network Simulator : ns-v.2.0, http://www-mash.cs.berkeley.edu/ns.

박 진 영
1997년 2월 이화여자대학교 컴퓨터학과 공학사. 1999년 2월 이화여자대학교 컴퓨터학과 공학석사. 1999년 3월 ~ 현재 LG 정보통신 차세대통신연구소 연구원. 관심분야는 IMT-2000, IP 기반의 이동통신망 프로토콜

채 기 준
1982년 연세대학교 수학과 학사. 1984년 미국 Syracuse University 전자계산학과 석사. 1990년 미국 North Carolina State University 컴퓨터공학과 박사. 1990년 ~ 1992년 미국 해군사관학교 전자계산학과 조교수. 1992년 ~ 현재 이화여자대학교 컴퓨터학과 부교수. 관심분야는 고속통신망, 무선통신망, 정보보호시스템, 망관리, LAN, 성능평가