하향 수직 핸드오버 상황에서 송신자에 기반을 둔 TCP 혼잡 제어 기법

준회원 최 여 민*, 정회원 송 주 석**°

A Novel Sender-Based TCP Congestion Control for Downward Vertical Handover

Yeomin Choi* *Associate Member*, JooSeok Song**° *Regular Member*

요 약

본 논문에서는 셀룰러 망에서 무선 LAN으로 핸드오버를 수행하는 하향 수직 핸드오버 상황에서 TCP의 처리량 저하 문제를 해결하기 위한 송신자 기반의 새로운 혼잡 제어 기법을 제안한다. TCP는 하향 수직 핸드오버와 같이 링크의 특성이 급격히 변하게 되는 상황에 쉽게 적응하지 못하고 처리량이 저하되는 문제가 발생한다. 이 문제의 주된 원인은 셀룰러 망과 무선 LAN의 지연 시간 차이에 의해 생기는 패킷 재정렬에 의한 것으로, TCP는 이로 인해 필요하지 않은 혼잡 제어를 수행한다. 그 결과 TCP의 혼잡 윈도우의 크기가 줄어드는 것은 물론 무선 LAN의 대역폭마저 낭비된다. 본 논문에서는 이러한 현상을 방지하기 위해 송신자 측에서 하향 수직 핸드오버 발생 전까지 측정하던 셀룰러 망의 왕복 시간을 이용하여 재정렬로 인해 발생하는 중복된 ACK를 처리하는 한편, 중복된 ACK를 활용하여 TCP의 혼잡 윈도우 크기를 조절을 통해 TCP의 처리량을 향상 시키는 기법을 제안한다. 시뮬레이션을 통해 본 논문에서 제안한 기법이 하향 수직 핸드오버 수행 시 발생하는 재정렬에 의한 문제를 해결하고, TCP New Reno 및 기존에 제안되어 있는 nodupack 방식[1]에 비해 처리량을 향상시킴을 보인다.

Key Words : TCP, Vertical handover, Downward vertical handover, Reordering, Congestion control

ABSTRACT

In this paper, we propose a sender-based TCP congestion control scheme for downward vertical handover (DVHO), in which mobile node moves from a cellular network to a wireless LAN. DVHO can give rise to severe performance problems in TCP throughput because it causes a drastic change of link characteristics. Particularly, TCP executes falsely congestion control by packet reordering, which is occurred from link delay difference between a cellular link and a wireless LAN link. Therefore, the congestion window is reduced. And unnecessary retransmissions wastes bandwidth. To solve these problems, we propose a method using estimated round-trip time in cellular link to process duplicated ACKs from reordering. Furthermore, the duplicated ACKs are used to the control congestion window size. Simulation result shows that the proposed scheme can solve problems. Moreover, the proposed scheme can have better performance than TCP New Reno and nodupack[1].

※ "본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (IITA-2008-C1090-0801-0028)
* (주) 제이투엠소프트 (yeomin@j2m.co.kr)
** 연세대학교 정보통신연구실 (jssong@emerald.yonsei.ac.kr)(°:교신저자)
논문번호 : KICS2007-10-455, 접수일자 : 2007년 10월 8일, 최종논문접수일자 : 2008년 5월 19일

Ⅰ. 서 론

미래의 무선 네트워크 환경은 어디에서나 네트워크에 접속할 수 있는 셀룰러 망과 접근성은 제한되지만 빠른 속도를 지원할 수 있는 무선 LAN 등을 함께 사용할 수 있는 다중 무선 네트워크로 발전될 것으로 전망된다. 이에 따라 대부분의 모바일 기기가 이를 활용할 수 있도록 하기 위해 다중 무선 인터페이스를 갖추게 될 것이며, 수직적 핸드오버를 통해 필요에 따라 이종망의 무선 네트워크 환경을 끊김 없이 옮겨갈 수 있게 될 것이다.

그러나 기존의 TCP는 유선망에서 사용하는 것을 전제로 설계되었기 때문에 모바일 기기의 이동성이나 수직적 핸드 오버 등으로 인해 발생하는 링크의 특성의 급격한 변화에 제대로 대응하지 못하는 문제를 안고 있다. 따라서 최근 TCP에서 이루어지는 연구도 무선망에서 모바일 기기의 이동성과 수직적 핸드오버를 지원하는 것에 중점을 두고 있다. 본 논문에서는 셀룰러 망에서 무선 LAN으로 핸드오버를 수행하는 하향 수직 핸드오버 상황에 초점을 맞추어 이 상황에서 TCP의 처리량이 저하되는 원인을 분석하고, 송신자 기반의 이를 해결하기 위한 기법을 제안한다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서 하향 수직 핸드오버 상황에서의 TCP 성능 문제와 이를 해결하기 위한 관련 연구들을 소개하고, Ⅲ장에서 본 논문에서 제안하는 기법을 상세히 설명하며, Ⅳ장에서는 수식적인 분석과 함께 시뮬레이션을 통해 본 논문에서 제안한 기법에 대한 성능을 분석하고, Ⅴ장에서 마무리한다.

Ⅱ. 하향 수직 핸드오버시 TCP 성능 저하 및 관련 연구

2.1 하향 수직 핸드오버시 TCP에서 발생하는 문제점

TCP가 동작하는 동안에 하향 수직 핸드오버시 발생하는 문제는 재정렬[1]에 의한 것이다. 하향 수직 핸드오버가 일어날 때 무선 LAN의 전송 속도가 셀룰러 망의 전송 속도보다 빠르기 때문에 송신자 측에서는 나중에 보냈지만 무선 LAN을 통해 전달되는 데이터 패킷이 셀룰러 망을 통해 전달되는 데이터 패킷보다 수신자에게는 먼저 도착한다. 따라서 수신자는 받을 패킷의 순서가 잘못되었기 때문에 받아야 할 차례의 패킷 번호를 가진 중복된 ACK를 송신자에게 보낸다. 송신자는 이렇게 재정렬로 인해 발생된 중복된 ACK를 TCP의 dupThresh 값 (일반적으로 3) 만큼 연속으로 받게 되면 셀룰러 망을 통해 전달되고 있는 데이터 패킷이 손실된 것으로 가정하고 재전송 및 빠른 복구를 통한 혼잡 제어를 수행한다. 그러나 이 상황에서 셀룰러 망을 통해서 전달되고 있는 데이터 패킷은 손실되지 않은 상황이기 때문에 결과적으로 송신자는 필요하지 않은 재전송과 혼잡 제어를 수행한 것이며, 이로 인해 무선 LAN의 대역폭이 낭비되는 것은 물론 송신자 측 TCP의 혼잡 윈도우의 크기가 줄어들기 때문에 TCP의 처리량마저 저하되는 결과를 가져온다.

만약 하향 수직 핸드오버 과정에서 재정렬을 적절한 방법을 통해 처리하였다고 해도 하향 수직 핸드오버가 끝나는 시점에 송신자가 여러 개의 패킷을 한 번에 전송하려 시도하는 버스트가 발생한다. 이는 하향 수직 핸드오버가 진행 중일 때 수신자가 무선 LAN을 통해 이미 전달 받은 데이터 패킷들 때문이다. 그림 1에서 버스트가 일어나는 상황을 보여

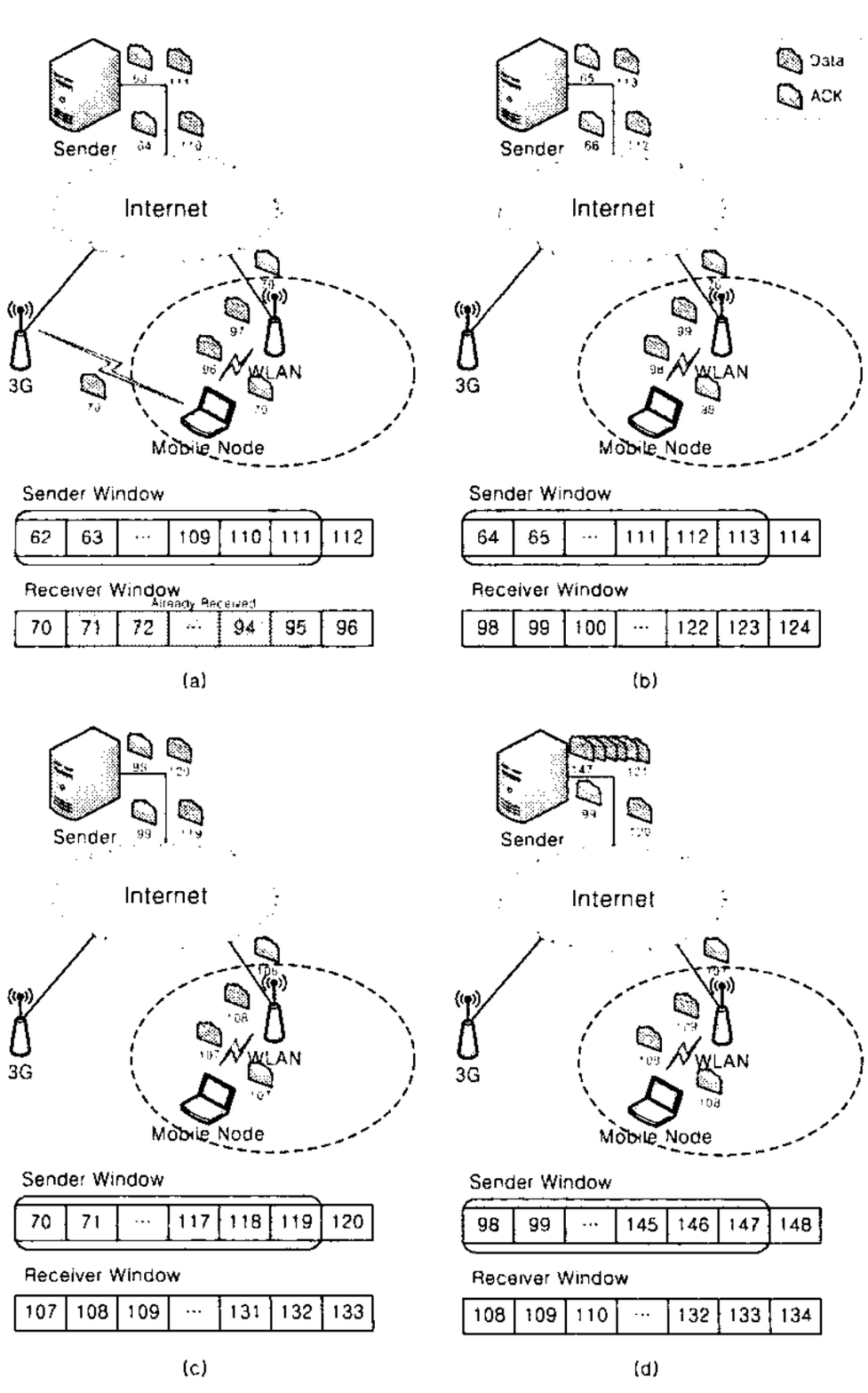


그림 1. 재정렬 처리 이후 버스트가 발생하는 상황
Fig. 1. Burst after resolution of reordering

주고 있는데, 이 그림은 TCP의 동작을 단순화해서 표현한 그림이며, 재정렬로 인해 발생한 중복된 ACK 또한 표시되어 있지만 이는 송신자가 성공적으로 처리하고 있는 것을 가정한다. 그림 1a에서 보여 지듯이 하향 수직 핸드오버 가 진행 중인 상황에서 셀룰러 망을 통해서 70번 데이터 패킷이 전송 되고 있지만 수신자는 이미 무선 LAN을 통해서 71번부터 95번 데이터 패킷을 모두 받은 상황에서 70번 ACK를 전송하고 있다. 이 때 그림 1b와 같이 수신자가 셀룰러 망을 통해 70번 데이터 패킷을 전송받고, 그 동안 무선 LAN을 통해 96, 97번 데이터 패킷을 전송받은 상태라면 수신자 측의 TCP에서는 70번부터 97번 데이터 패킷을 모두 받았기 때문에 다음에 받을 데이터 패킷인 98번 ACK를 송신자에게 보내게 된다. 이후 그림 1c와 같이 송신자가 98번 ACK를 받기 직전의 상황에서는 송신자가 직전에 받은 ACK의 번호가 70번이었기 때문에 송신자 측 TCP의 혼잡 윈도우는 70번부터 시작하고 있다. 그러나 그림 1d와 같이 송신자가 98번 ACK를 전송받았다면 송신자의 혼잡 윈도우는 뒤로 밀려나서 98번부터 시작하게 된다. 이로 인해 송신자 측 TCP에서는 혼잡 윈도우에 새로 들어오는 121번부터 147번 데이터 패킷에 대한 전송을 시도한다. 이렇게 한꺼번에 전송을 시도하는 패킷들 때문에 버스트가 발생하는데, 이로 인해 무선 LAN AP에서는 버퍼 넘침으로 인해서 패킷이 버려질 수 있다.. 따라서 하향 수직 핸드오버 상황에서 TCP 성능 향상을 위해서는 재정렬 문제 해결과 함께 버스트로 인해 발생하는 버퍼 넘침 현상으로 인해 패킷이 버려지는 문제가 발생할 위험을 함께 고려해야 한다.

2.2 하향 수직 핸드오버 상황에서 TCP 성능 향상 연구

하향 수직 핸드오버 상황에서 TCP의 성능 향상을 위해 다양한 관점에서 연구되어 왔다. Daniel 등에 의해 제안된 RTO-conv[2]는 무선 LAN의 지연시간을 활용하는 것에 초점을 맞추고 있다. 이 기법에서는 하향 수직 핸드오버 수행 과정에서 송신자가 무선 LAN의 왕복 시간을 측정하여 하향 수직 핸드오버가 완료된 직후 TCP의 RTO 값을 측정된 무선 LAN의 왕복 시간을 이용하여 조절한다. 그러나 실제로 패킷 손실이 발생한 경우 RTO까지 기다리지 않고 중복된 ACK를 이용한 빠른 복구를 통해서 해결되기 때문에 이 기법은 TCP의 성능을 향상시키지는 못한다. Gou 등[3]은 수신자가 셀룰러 망에서는 awnd 값의 제한을 통해 데이터 전송을 제한하고 있다가 하향 수직 핸드오버가 발생했을 때 awnd 값을 증가시켜서 빠른 복구를 빠르게 발생시켜서 무선 LAN의 BDP (Bandwidth delay product)를 활용하려 한다. 그러나 이 기법에서는 셀룰러 망에서의 처리량이 제한되는 것은 물론 빠른 복구로 인한 성능 저하나 대역폭 낭비 문제는 그대로 남아 있다. Sarolathi 등[4]은 하향 수직 핸드오버가 일어날 때 Quick-Start[6]를 이용하여 새로운 경로의 최대 대역폭을 알아내고, 이를 이용해서 송신자의 cwnd와 ssthresh 값를 조절하는 기법을 제안하였다. 이를 통해 무선 LAN의 대역폭을 활용할 수 있으나 하향 수직 핸드오버 상황에서 발생하는 재정렬은 고려되지 않았기 때문에 큰 성능 향상을 보여주지는 못한다. Matsushita 등의 연구[5]에서는 수신자가 주도적으로 TCP 혼잡 제어를 수행하는 기법을 제안하였다. 하향 수직 핸드오버 환경에서 수신자는 무선 LAN의 BDP를 계산하고, ACK를 보내는 시간 조절을 통해 송신자의 혼잡 윈도우를 조절한다. 그러나 수신자 측에서 송신자의 혼잡 윈도우 상황 등을 알 수 없기 때문에 무선 LAN의 BDP를 제대로 활용하기 어려운 문제가 있다.

위의 연구들은 모두 reordering으로 인해 발생하는 문제를 고려하지 않았기 때문에 이로 인한 TCP 처리량 감소 및 대역폭 낭비의 문제 또한 해결하지 못하고 있다. 반면 Hansmann 등의 연구[11]에서는 하향 수직 핸드오버시 발생하는 재정렬 문제를 해결하기 위해 nodupack을 제안하고 있다. 이 기법은 수신자 측에서 재정렬로 인해 발생하는 중복된 ACK를 보내지 않게 하는 것에 바탕을 두고 제안되었다. 그러나 제안된 기법의 자세한 구현에 대해서는 모호하게 기술하고 있으며 nodupack을 적용하기 위해서는 수신자가 받은 패킷이 어떤 망을 통해 전달되었는지 알아야 하기 때문에 수신자 측에서는 TCP 뿐만 아니라 IP 등 하위 계층에서의 정보를 가져올 수 있는 cross layer에 기반을 둔 설계가 필요하다는 한계가 있다. 본 논문에서는 하향 수직 핸드오버시 발생하는 재정렬 문제의 해결에 초점을 맞추고 있는 만큼 nodupack과의 하향 수직 핸드오버 발생 시 전송되는 처리량 등의 비교를 통해 본 논문에서 제안한 기법의 성능 평가를 할 것이다.

Ⅲ. 제안하는 TCP sender 기반 혼잡 제어 기법

본 논문에서 제안하는 기법은 셀룰러 망에서 무선

표 1. 변수 설명

변수 이름	설명
A	송신자가 수신자로부터 받은 ACK의 번호
T_i	송신자가 i번 데이터 세그먼트를 보낸 시간
T_{now}	현재 시간
F	송신자의 혼잡 윈도우의 첫 바이트 번호
S	TCP 세그먼트 크기
cwnd	송신자의 혼잡 윈도우 크기 (세그먼트 수)
RTT	송신자측 TCP에서 측정하는 세그먼트의 왕복 시간

LAN으로의 하향 수직 핸드오버 상황에서 재정렬 문제에 대해 초점을 맞추고, 이를 통해 TCP의 동작 효율을 개선하는 것을 목적으로 한다. 본 논문에서는 수신자가 셀룰러 망에서 무선 LAN으로 하향 수직 핸드오버를 수행하는 과정에 대해서는 고려하지 않으며, 수신자가 무선 LAN와 연결된 직후 핸드오버 알림 메시지를 보내서 송신자에게 이를 알리는 것으로 가정한다. 본 논문에서 사용되는 변수들은 표 1과 같으며 TCP 내에서의 동작만을 표현하기 위해 패킷과 구분하여 세그먼트로 표기하였다.

본 기법에서 수신자는 핸드오버 알림 메시지를 보내는 것 외에는 다른 어떠한 역할도 하지 않는다. 따라서 수신자 측에서는 재정렬로 인해 발생하는 중복된 ACK를 그대로 전송하며 이 문제에 대한 처리는 전적으로 송신자가 담당한다. 이는 셀룰러 망과 무선 LAN의 지연 시간 차이에 의한 것으로 송신자는 이를 처리하기 위해 셀룰러 망에서 동작 중 측정해오던 왕복 시간을 이용한다. 이를 위해 송신자 측 TCP에서는 수신자가 보낸 핸드오버 알림 메시지를 받으면 그 시점까지 측정해오던 왕복 시간을 ERTT (Expected Rount-Trip Time) 값으로 저장한다. ERTT 값의 의미는 핸드오버 알림 메시지를 받은 시점까지 측정해오던 왕복 시간으로 셀룰러 망을 통해서 전달되는 세그먼트에 대해 정상적인 응답이 오는데 걸릴 것으로 기대되는 시간이다. 송신자는 또한 핸드오버 알림 메시지를 받은 시점에서 혼잡 윈도우의 마지막 바이트 번호를 HEnd 값으로 저장한다. 이는 이 바이트 번호 이후의 세그먼트들은 모두 무선 LAN을 통해서 전달될 것이므로 이보다 큰 번호를 가진 ACK를 받았다면 수신자가 셀룰러 망을 통해 전송되던 데이터를 모두 받았다는 것으로 송신자 측에서 하향 수직 핸드오버 수행이 완료되었다는 것으로 판단할 수 있다.

송신자는 이 과정을 거친 후 수신자로부터 받은 ACK에 대해 다음과 같이 처리한다.

1) F = A 이며, $T_{now} - T_i \leq$ ERTT인 경우, 송신자는 셀룰러 망으로 전달된 패킷에 대해 정상적인 응답이 올 것으로 예상한 시간이 지나지 않았기 때문에 이 중복된 ACK가 무선 LAN link를 통해 전달된 데이터에 의해 발생한 것으로 판단한다. 그러나 ACK가 도착했다는 것은 무선 LAN을 통해서 데이터가 성공적으로 전달되었다는 것을 의미하기 때문에 송신자는 cwnd = cwnd + 1로 조절해서 새로운 데이터 세그먼트를 보낸다.
2) F = A 이며, $T_{now} - T_i >$ ERTT인 경우, 송신자는 셀룰러 망으로 전달된 패킷에 대해 정상적인 응답이 올 것으로 예상한 시간이 지났기 때문에 이 ACK는 정상적으로 발생한 중복된 ACK로 판단한다. 따라서 송신자는 연속적으로 받은 중복된 ACK의 개수를 세어보면서 이것이 dupThresh 값 (일반적으로 3) 가 되는 시점에서 해당 번호의 데이터 세그먼트를 재전송한다. 전송한 데이터가 손실되었을 위험이 있기 때문에 이 과정에서는 송신자는 cwnd의 크기를 조절하지 않는다.
3) F < A인 경우 송신자는 F번부터 A - 1번 바이트 번호를 갖는 데이터가 모두 정상적으로 전송되었다는 것을 알 수 있다. 이 때 버스트의 발생을 방지하기 위해 송신자는 1개의 새로운 데이터 세그먼트만을 새로 전송하게 한다. 이를 위해 cwnd = cwnd - (A - F) / S + 1로 조절하여 한 개의 새로운 데이터 세그먼트를 전송할 수 있게 한다.
4) F > HEnd인 경우 송신자는 수신자가 셀룰러 망을 통해 전달되던 데이터를 모두 받았다는 것을 알 수 있다. 따라서 하향 수직 핸드오버 처리를 종료하며 정상적인 TCP 혼잡 제어 상태로 돌아간다.

본 논문에서 제안하는 기법에 대한 간략한 흐름도는 그림 2와 같다. 본 논문에서 제안한 기법은 이 과정들을 통해 하향 수직 핸드오버 상황에서 발생하는 중복된 ACK로 인한 잘못된 혼잡 제어를 피할 뿐만 아니라 ACK를 받을 때마다 1개의 새로운 데이터 세그먼트를 추가로 전송하기 때문에 핸드오버가 발생

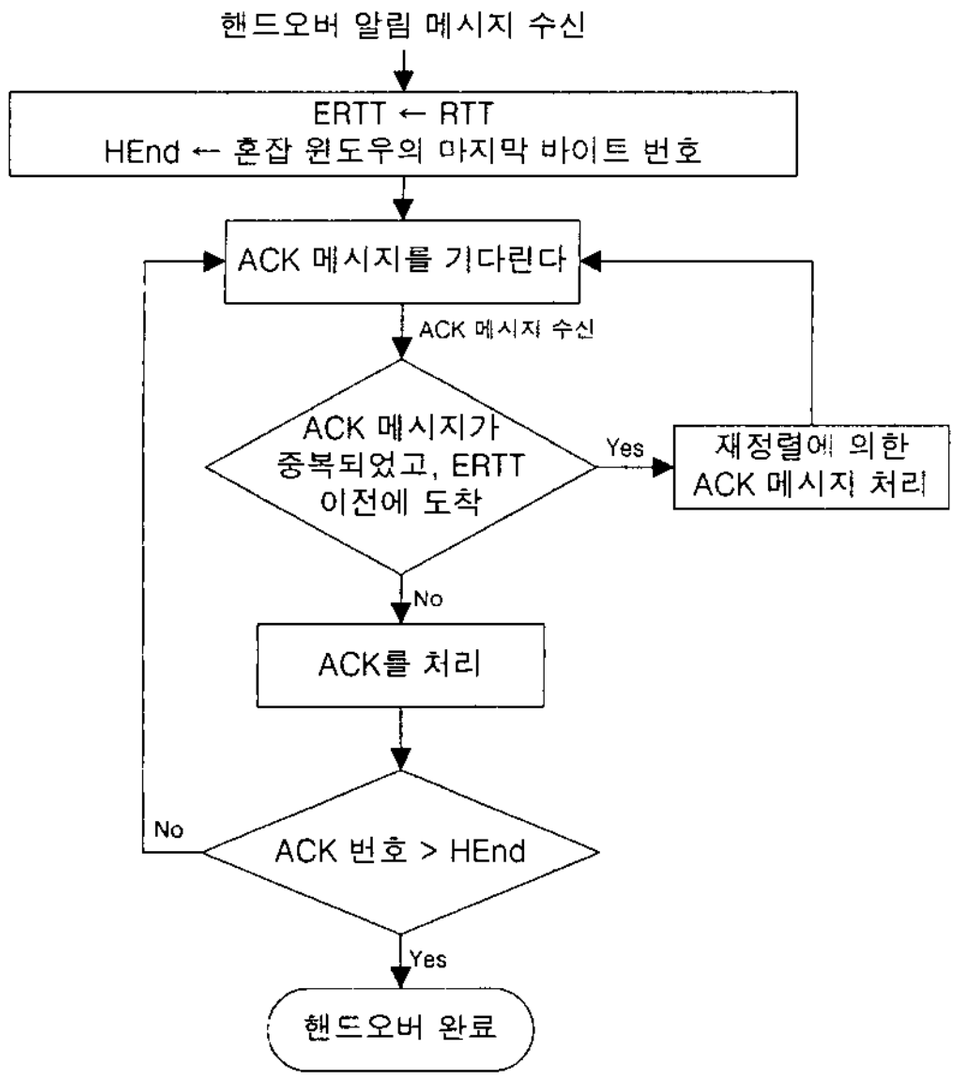


그림 2. 제안하는 기법의 흐름도
Fig. 2. Flow chart of proposed scheme

한 시점에서 네트워크상에서 전송되고 있는 데이터 패킷의 양을 그대로 유지하게 되며, 이를 통해 핸드오버가 완료되었을 때 발생할 수 있는 버스트를 방지하기 때문에 버스트로 인해 발생할 수 있는 버퍼 넘침으로 인한 패킷 손실을 막을 수 있다.

Ⅳ. 성능 평가

본 논문에서 제안한 기법과 기존의 TCP New Reno, nodupack의 하향 수직 핸드오버가 발생하여 진행 중인 상황에서의 성능 평가를 수행하였다. 성능 평가를 위한 네트워크 모델은 그림 3과 같다.

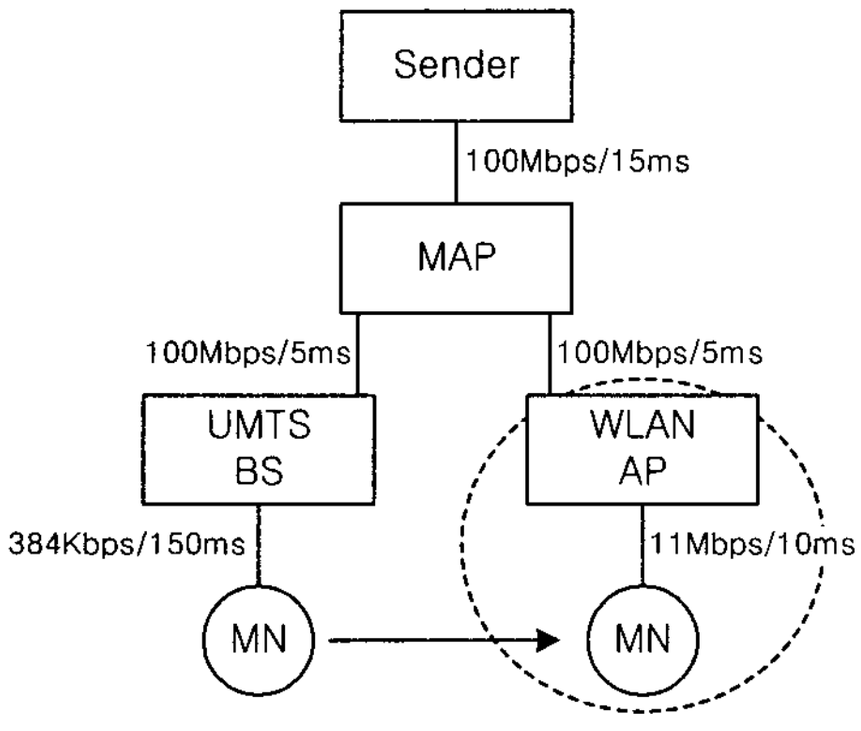


그림 3. 네트워크 모델
Fig 3. Network model

4.1 수식적인 성능 평가

수신자가 하향 수직 핸드오버를 수행하는 동안의 처리량을 수식적으로 모델링하여 비교하였다. 이를 위해 송신자와 수신자가 보낸 패킷이 도착하는 시간은 패킷이 통과하는 링크의 대역폭과 지연 시간에만 영향을 받으며, 유선 구간의 대역폭은 상대적으로 크기 때문에 무시할 수 있는 것으로 가정하였다. 또한 중간에 위치한 라우터 등의 처리 시간은 없는 것으로 가정하였으며, 이에 따라 송신자가 핸드오버 알림 메시지를 받은 시점에서 네트워크상에 존재하는 패킷의 개수는 송신자의 혼잡 윈도우 크기인 cwnd와 같다.

먼저 핸드오버가 완료되는데 걸리는 시간 T는 송신자가 핸드오버가 발생하기 전에 UMTS 망을 통해서 보낸 데이터 패킷들을 수신자가 모두 받고 이에 대한 ACK를 송신자가 모두 받는데까지 걸린 시간이며, 이는 한 패킷이 네트워크를 통해 전송되는데 패킷 크기 / 대역폭 + 지연시간의 시간이 걸리는 것을 이용해서 UMTS를 통해서 마지막으로 전달되는 데이터 패킷의 ACK가 전달되는데 걸리는 시간 T는

$$T = cwnd * S_{DATA} / B_{UMTS} + D_{UMTS} + S_{ACK} / B_{WLAN} + D_{WLAN} \quad (1)$$

와 같이 계산할 수 있다. 여기에서 S_{DATA}와 S_{ACK}은 각각 데이터와 ACK 패킷의 크기이며, B_{UMTS}와 B_{WLAN}은 각각 UMTS와 무선 LAN의 대역폭, D_{UMTS}와 D_{WLAN}은 각각 UMTS와 무선 LAN을 이용할 때의 지연 시간을 의미한다.

이를 이용해서 TCP New Reno의 경우를 살펴보면 하향 수직 핸드오버가 발생하면서 생긴 재정렬로 인해서 빠른 복구 과정을 수행하게 된다. 빠른 복구를 수행하게 된다. 이 과정에서 송신자의 혼잡 윈도우의 크기는 원래 크기의 절반 + dupThresh 으로 줄어들고, 중복되거나 새로운 ACK를 받을 때마다 패킷을 하나씩 새로 보내게 된다. 이 때 핸드오버가 시작된 이후 송신자가 UMTS를 통해서 받게 되는 ACK 메시지는 무선 LAN을 통해서 받은 ACK 메시지보다 오래된 정보를 담고 있기 때문에 빠른 복구 과정에서 그냥 버려지게 된다.

따라서 무선 LAN을 통해서 받은 ACK 메시지만을 이용하게 되는데, 핸드오버가 수행된 직후 데이터는 UMTS로 수신자에게 전달되고, 이에 대한 응답은 무선 LAN을 통해 송신자에게 전달되는 i번째

패킷이 처리되는데 걸리는 시간 T_i은 다음과 같다.

$$T_i = i * S_{DATA} / B_{UMTS} + D_{UMTS} + S_{ACK} / B_{WLAN} + D_{WLAN} \quad (2)$$

또한 위 시간 이후에 새로 보내게 되는 데이터 패킷은 모두 무선 LAN을 통해서 전달되기 때문에 송신자가 T_i 시간에 받은 ACK를 핸드오버가 끝날 때까지 새로 전송한 패킷들은 무선 LAN을 통해서 계속 전달되므로 T_i 시간에 받은 ACK로 인해서 핸드오버가 끝날 때까지 송신자가 새로 전송하게 되는 패킷의 수 V_i는 (1)과 (2)를 활용해서 다음과 같이 계산할 수 있다.

$$V_i = 1 + \frac{T - Ti}{(S_{DATA} + S_{ACK})/B_{WLAN} + 2 \times D_{WLAN}} \quad (3)$$

TCP New Reno의 빠른 복구 과정에서 혼잡 윈도우의 크기가 기존 혼잡 윈도우 크기의 절반 + dupThresh 값만큼으로 줄어들기 때문에 보낸 패킷의 개수는 $P_{NewReno}$는 (3) 의 수식을 활용하여

$$P_{NewReno} = \sum_{i=0}^{cwnd} V_i - (cwnd/2 - dupThresh) \quad (4)$$

이 됨을 구할 수 있다.

다음으로 nodupack의 경우는 수신자 측에서 재정렬로 인해서 발생하는 중복된 응답을 처리하지 않기 때문에 핸드오버가 수행되는 동안 송신자는 핸드오버가 발생하기 전에 보냈던 데이터 패킷들에 대한 응답을 모두 받게 된다. 송신자 측에서 볼 때 이 과정은 TCP의 혼잡 회피 상태에 속하기 때문에 nodupack에서 핸드오버가 수행되는 동안 송신자가 보낸 패킷의 개수 $P_{nodupack}$는

$$P_{nodupack} = cwnd + 1 \quad (5)$$

이 된다.

마지막으로 본 논문에서 제안한 기법은 TCP New Reno의 빠른 복구 과정과 유사한 과정을 수행한다. 그러나 TCP New Reno와 달리 핸드오버를 처리하면서 혼잡 윈도우의 크기를 줄이지 않고 중복되거나 새로운 ACK를 받을 때마다 패킷을 하나씩 새로 보내게 되기 때문에 보낸 패킷의 개수 $P_{proposed}$는 (3)의 수식을 활용해서

$$P_{Proposed} = \sum_{i=0}^{cwnd} V_i \quad (6)$$

표 2. 수식 분석 결과값 (혼잡 윈도우 크기 = 50)

기법	처리량 (KB/s)
TCP New Reno	155.09
nodupack	154.44
제안한 기법	315.59

가 되는 것을 구할 수 있다.

위에서 구한 처리량을 네트워크 모델에 대입했을 때 핸드오버 직전 혼잡 윈도우 크기를 50, 데이터 패킷의 크기를 1000 바이트, ACK 패킷의 크기를 20 바이트로 가정했을 때의 얻은 처리량의 값은 다음 표 2와 같다. 수식 분석의 결과 본 논문에서 제안한 기법의 처리량이 가장 좋은 것을 볼 수 있다. 그러나 이는 간단한 모델을 통해 단순화한 성능 분석 결과이기 때문에 실제와는 차이가 있을 수 있다.

4.2 시뮬레이션 성능 평가

시뮬레이션은 C++을 이용하여 작성한 TCP 핸드오버 시뮬레이터를 사용하였다. 시뮬레이션을 시작하는 시점에서 수신자는 UMTS 망을 이용해서 송신자로부터 데이터를 전송받다가 80초 후에 무선 LAN으로 하향 수직 핸드오버를 수행하게 된다. 송신자 측에서 전송하는 데이터 세그먼트는 각 1000 bytes의 FTP 데이터이며, 본 논문에서 제안한 기법을 TCP New Reno, nodupack와 비교하였다. 시뮬레이션을 통해 패킷 손실이 발생하지 않는 경우에 하향 수직 핸드오버시 송신자 측의 혼잡 윈도우 크기의 변화, 하향 수직 핸드오버를 시작한 직후부터 송신자가 UMTS 망으로 전송되는 패킷을 모두 받는 데까지 걸리는 시간 동안의 처리량과 그 동안 재전송한 데이터 패킷의 수, 하향 수직 핸드오버 수행 후 TCP가 무선 LAN에 완전히 적응하였다고 할 수 있는 무선 LAN AP에서 처음으로 버퍼 넘침으로 인한 패킷 손실이 일어나는 데까지 걸리는 시간과 그 동안의 처리량을 측정하였다. 또한 UMTS BS와 WLAN AP의 버퍼 사용량 측정을 통해 본 논문에서 제안한 기법이 버스트를 방지하고 있는 것을 확인하였다. 또한 핸드오버 과정에서 UMTS 망에서 패킷 손실이 발생한 경우에 대해서도 버퍼 사용량 측정을 제외하고 패킷 손실이 없는 경우와 동일한 시뮬레이션을 수행하였으며, 추가적으로 핸드오버 도중 UMTS 망에서 발생하는 패킷 손실률에 따라 TCP New Reno와 본 논문에서 제안한 기법간의 성능 차이를 비교하였다.

4.2.1 패킷 손실이 없는 경우

먼저 패킷 손실이 없는 경우 cwnd 변화에 대한 결과는 그림 4와 같다. 본 논문에서 제안한 TCP는 하향 수직 핸드오버가 시작될 때 약간의 cwnd 저하를 보여주는데, 이는 송신자가 보낸 데이터뿐만 아니라 수신자가 보낸 ACK도 하향 수직 핸드오버 과정에서 재정렬이 발생하였기 때문이다.

cwnd 변화와 함께 그림 5의 하향 수직 핸드오버 수행 중 재전송된 패킷의 수를 보면 TCP New Reno는 재정렬로 인해 재전송 및 빠른 복구를 통한 혼잡 제어를 수행하면서 손실되지 않은 패킷들을 재전송하고 있는 것을 볼 수 있다. 반면 nodupack과 본 논문에서 제안하는 TCP는 하향 수직 핸드오버로 인해 발생하는 재정렬 처리를 성공적으로 수행하여 재전송한 패킷이 없는 것을 볼 수 있다. 또한 수신자가 하향 수직 핸드오버를 시작한 직후부터 UMTS 망을 통해 전달되던 데이터를 모두 받는데 까지 걸리는 시간 동안의 처리량을 보면, 본 논문에서 제안한 기법이 TCP New Reno에 비해 272%, nodupack에 비해 705% 향상된 것을 볼 수 있다. 이는 하향 수직 핸드오버 수행 중 재정렬로 인해 발생한 중복된 ACK들을 버리지 않고 이를 이용해서 혼잡 윈도우의 크기를 증가시키면서 많은 패킷을 전송할 수 있었기 때문이다. 그림 5의 결과 값이 수식을 통해 분석한 결과 값과 차이가 생긴 이유로는 실제보다 단순화된 모델을 이용하여 수식화 하여 계산했다는 점과 함께 수식을 통해 계산할 때 사용한 혼잡 윈도우의 크기가 시뮬레이션에서와 약간의 차이가 있었다는 것 때문으로 풀이되며, 특히 네트워크 중간에 위치한 라우터 등에서 지연되는 시간을 수식적인 모델에서는 고려하지 않고 있기 때문에 TCP New Reno와 본 논문에서 제안한 기법의 처리량이 시뮬레이션에 비해 상대적으로 낮게 계산된 원인일 것이다.

그림 6은 TCP가 무선 LAN에 적응하여 무선 LAN의 BDP를 활용한다고 할 수 있는 하향 수직 핸드오버 이후 무선 LAN AP에서 첫 번째 패킷 손실이 발생하는 데까지 걸리는 시간과 그 동안의 시간당 처리량을 비교한 그래프이다. 이 그래프에서 보면 본 논문에서 제안한 TCP가 TCP New Reno에 비해서는 적응 시간이 1.4초 빠르지만 nodupack에 비해 약 0.4초 늦다. 그러나 하향 수직 핸드오버 발생 직후부터 무선 LAN에 적응하는 시점까지 전송한 처리량은 본 논문에서 제안한 TCP가 가장 높은 것을 볼 수 있다.

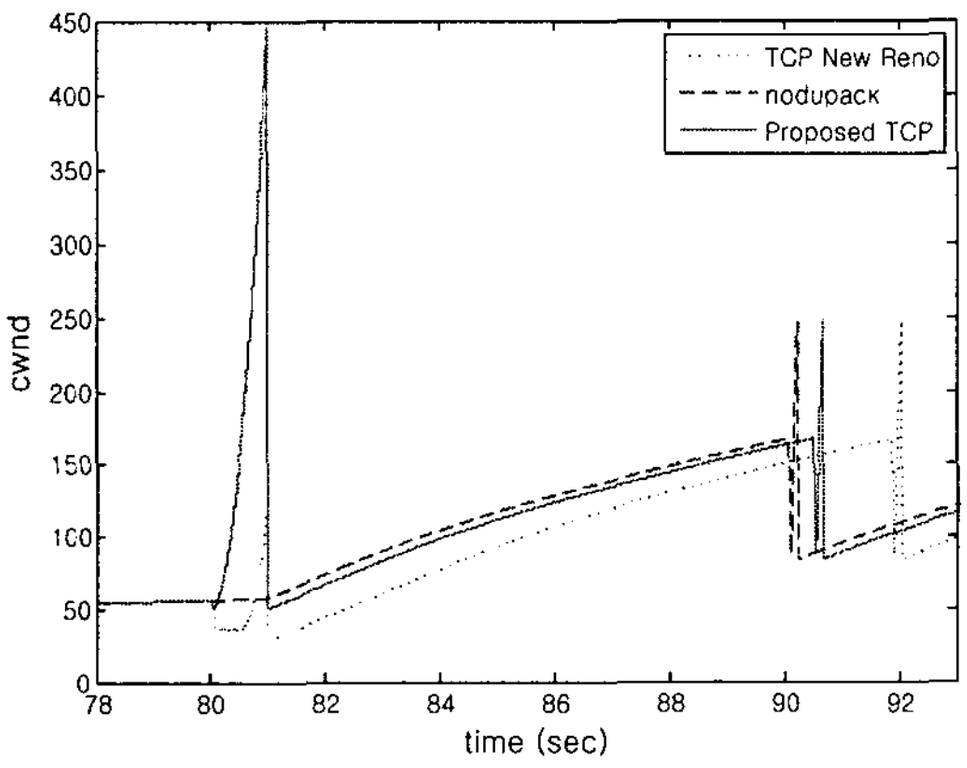


그림 4. 패킷 손실이 없는 경우 혼잡 윈도우 크기 변화
Fig. 4. cwnd time histories during DVHO, no loss

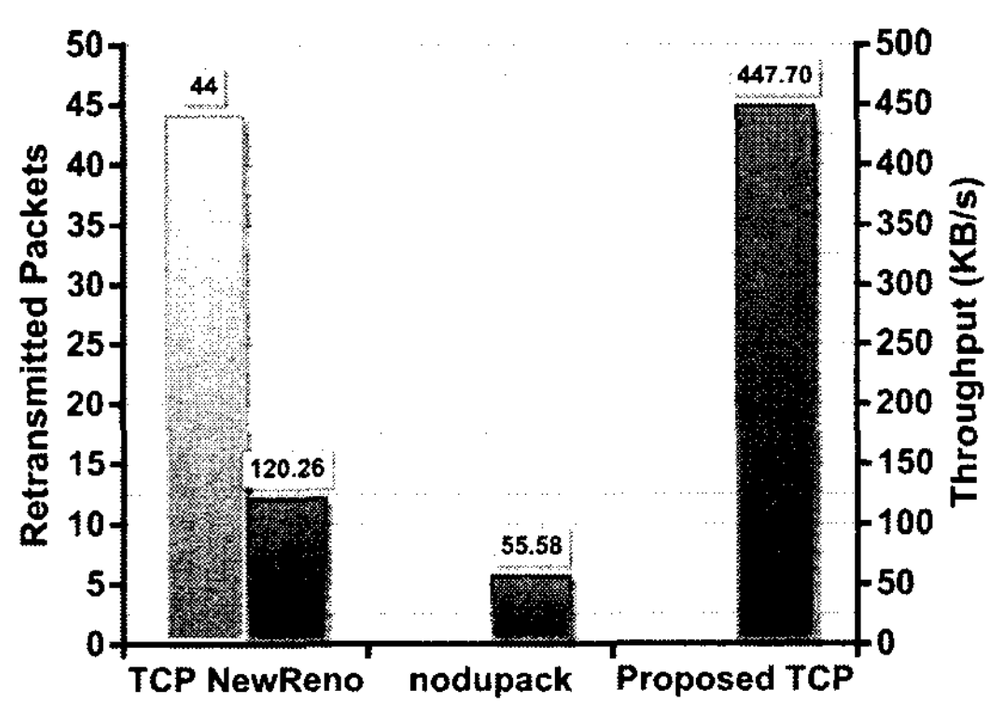


그림 5. 패킷 손실이 없을 때 하향 수직 핸드오버 수행 중 재 전송한 패킷 개수 및 처리량
Fig. 5. Retransmitted packets and throughput during DVHO, no loss

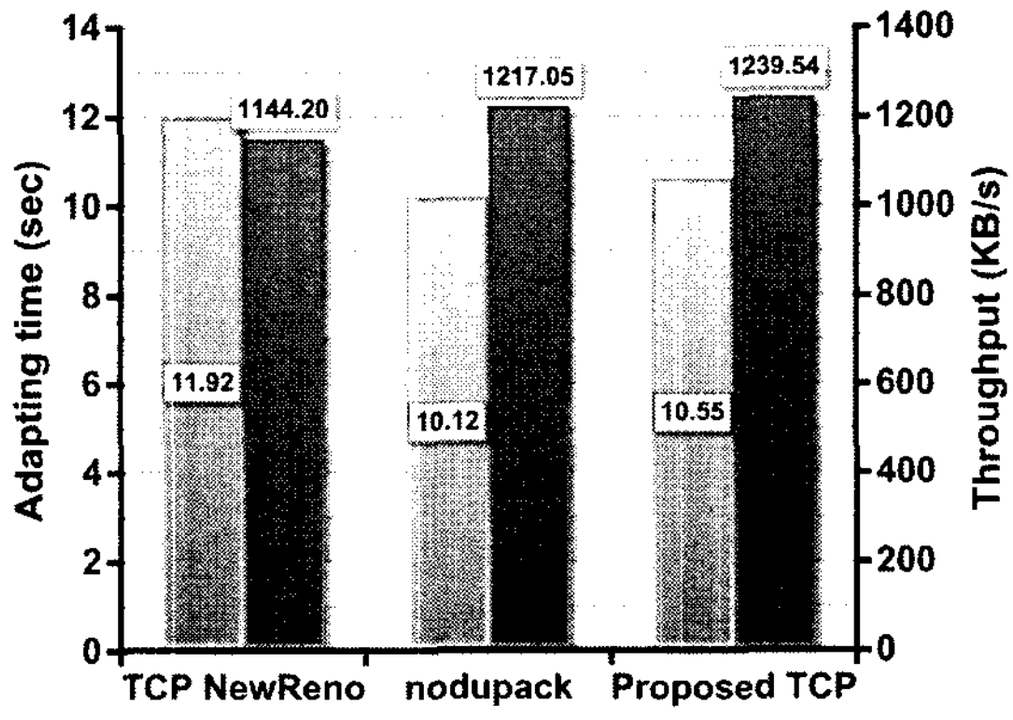


그림 6. 패킷 손실이 없을 때 하향 수직 핸드오버가 일어난 시점부터 무선 LAN에 적응하는데 걸리는 시간 및 처리량
Fig. 6. Wireless LAN adapting time and throughput after DVHO, no loss

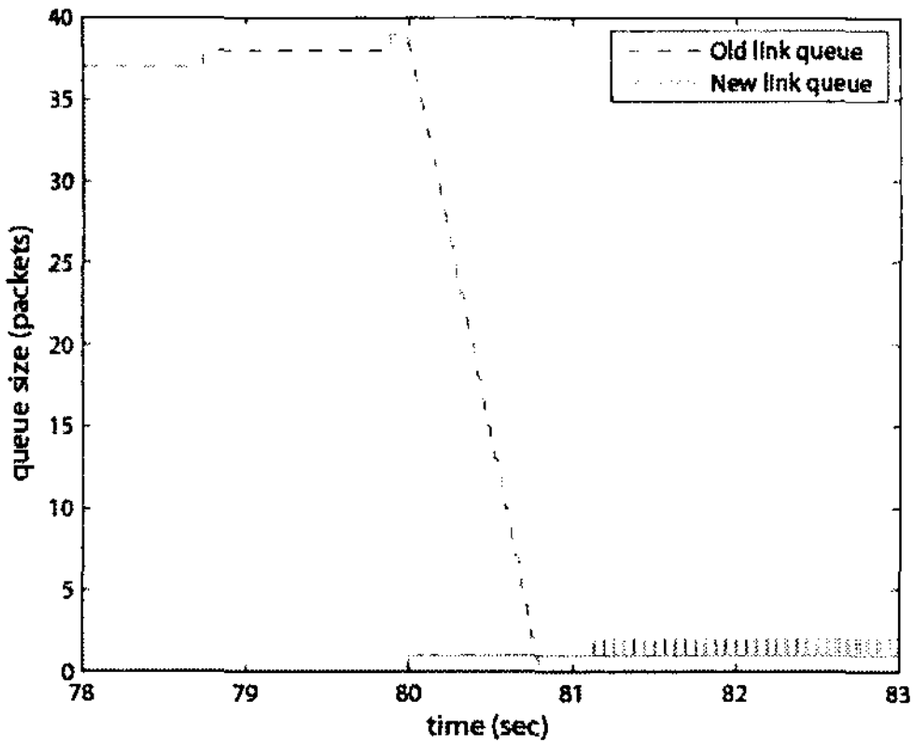


그림 7. 패킷 손실이 없을 때 본 논문에서 제안한 기법의 버퍼 사용량.
Fig. 7. Transmission queue utilization during DVHO in Proposed TCP, no loss

그림 7은 본 논문에서 제안한 기법이 하향 수직 핸드오버를 수행하는 동안 사용한 버퍼의 양을 측정한 그래프이다. 핸드오버가 진행 중인 상태에서는 UMTS 링크의 버퍼 사용량이 줄어들기 시작하여 핸드오버가 완료되는 시점에서 비워지게 되는 것을 볼 수 있다. 본 논문에서 제안한 기법은 버스트를 발생시키지 않기 때문에 핸드오버가 완료된 후에 새로 옮겨간 무선 LAN AP의 버퍼 사용량이 순간적으로 많아지게 되는 문제가 발생하지 않는다.

4.2.2 하향 수직 핸드오버 수행 중 UMTS 망에서 패킷 손실이 발생한 경우

하향 수직 핸드오버를 수행하는 동안 UMTS 망에서 한 개의 데이터 패킷이 손실되는 경우 cwnd 변화에 대한 결과는 그림 8과 같다. TCP New Reno 및 본 논문에서 제안한 기법은 그림 4와 비교할 때 cwnd 변화 면에서 큰 차이를 보이지 않지만 nodupack은 손실이 없는 상황에서와 달리 수신자 측에서 보낸 중복된 ACK로 인해 재전송 및 혼잡 제어를 과정을 거치기 때문에 많은 차이를 보이다. 이 결과는 그림 9에서 나타낸 핸드오버가 완료되는 시간까지의 처리량에서도 볼 수 있는데 손실이 없을 때와 비교하면 TCP New Reno는 처리량이 2.52%, 본 논문에서 제안하는 기법은 2.94% 감소하였으나 nodupack은 18.17%가 저하되어서 본 논문에서 제안한 TCP가 TCP New Reno에 비해 270%, nodupack에 비해서는 855% 향상된 처리량을 보여준다. 또한 재전송한 패킷의 수를 보면 본 논문에서 제안한 TCP는 1개의 패킷만을 재전송하였는데, 이는 UMTS에서 손실된 패킷을 재전송한 것으로 제안된 기법이 성공적으로 실제 손실된 패킷을 찾아내고 있다는 것을 알 수 있다.

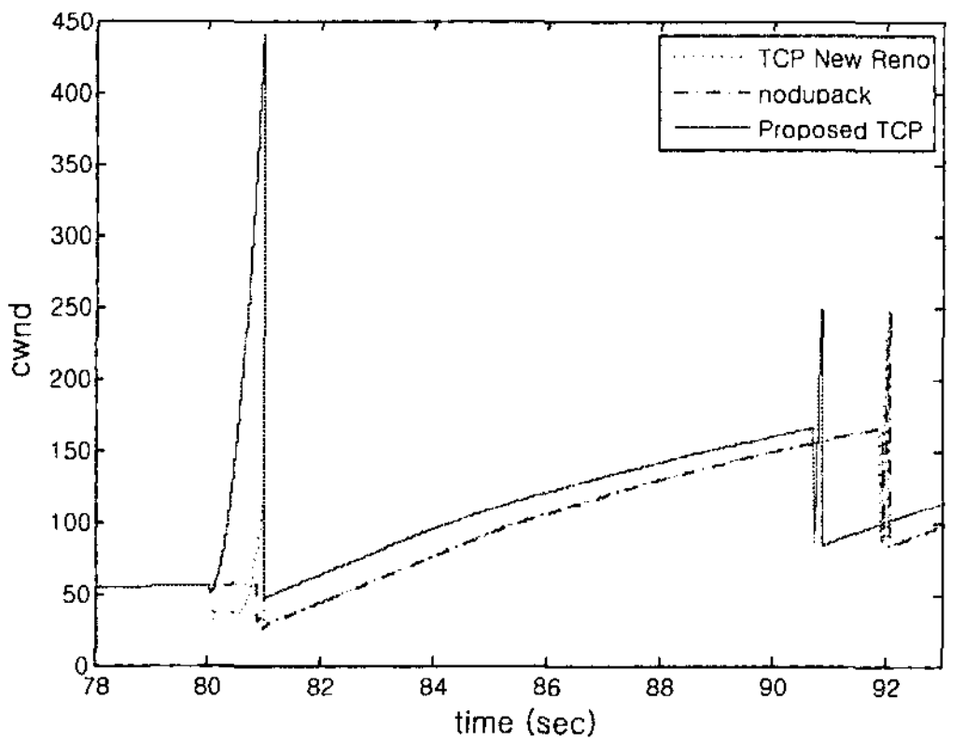


그림 8. UMTS 망에서 패킷 손실이 발생한 경우 cwnd 변화
Fig. 8. cwnd time histories during DVHO, packet loss in UMTS link

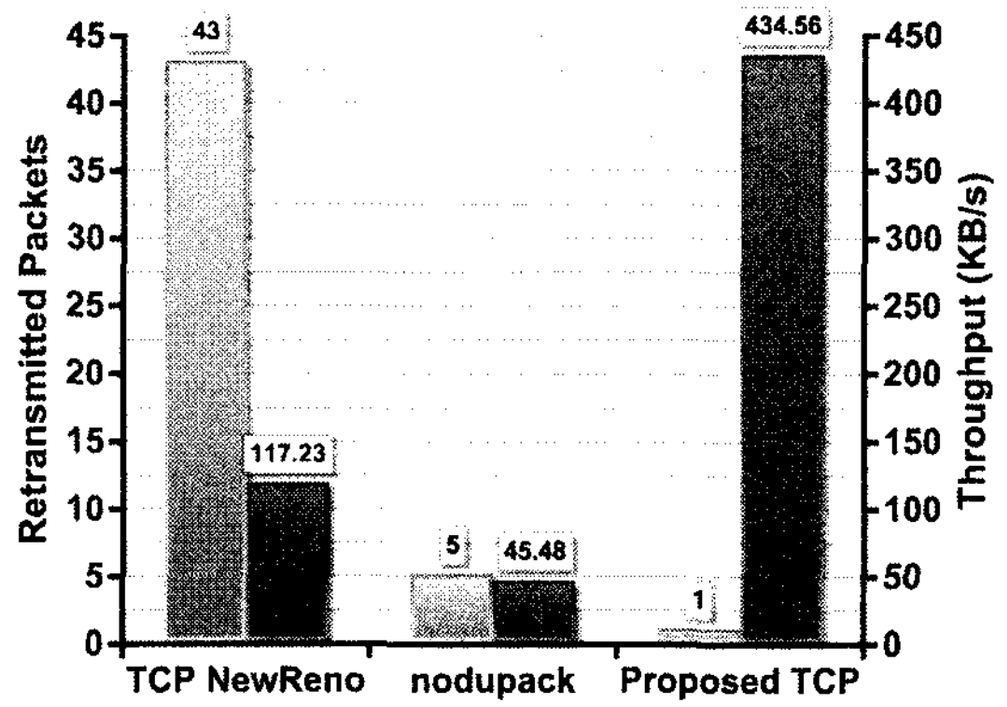


그림 9. UMTS 망에서 패킷 손실이 발생한 경우 하향 수직 핸드오버 수행 중 재 전송한 패킷 개수 및 처리량
Fig. 9. Retransmitted packets and throughput during DVHO, a packet loss in UMTS link

그림 10에서는 무선 LAN에 적응하는데 걸리는 시간 및 그 동안의 처리량이 나타나 있는데 이 또한 본 논문에서 제안한 기법이 TCP New Reno 및 nodupack에 비해 1.2초 빠르며, 하향 수직 핸드오버 발생 직후부터 무선 LAN에 적응하는데 걸리는 시점까지의 처리량 또한 가장 높은 것을 볼 수 있다. 이는 본 논문에서 제안한 TCP가 패킷 손실에 대해서도 성공적으로 대응하고 있음을 의미한다.

마지막으로 그림 11은 하향 수직 핸드오버를 수행중인 상태에서 UMTS 망에서 발생하는 패킷 손실률을 변화시키면서 핸드오버가 수행되는 동안의 처리량을 비교한 그래프이다. nodupack은 UMTS 망에서 패킷 손실이 발생한 경우 TCP New Reno

에 비해 처리량이 떨어지기 때문에 이 비교에서는 제외하였다. 그림 11을 보면 TCP New Reno와 본 논문에서 제안한 기법 모두 패킷 손실률이 증가함에 따라 처리량이 줄어들지만 본 논문에서 제안한 기법이 TCP New Reno에 비해 3.5에서 4배 정도 높은 처리량을 보여주는 것을 볼 수 있으며, 이는 핸드오버 도중 발생하는 패킷 손실에도 성공적으로 대응하고 있다는 것을 의미한다.

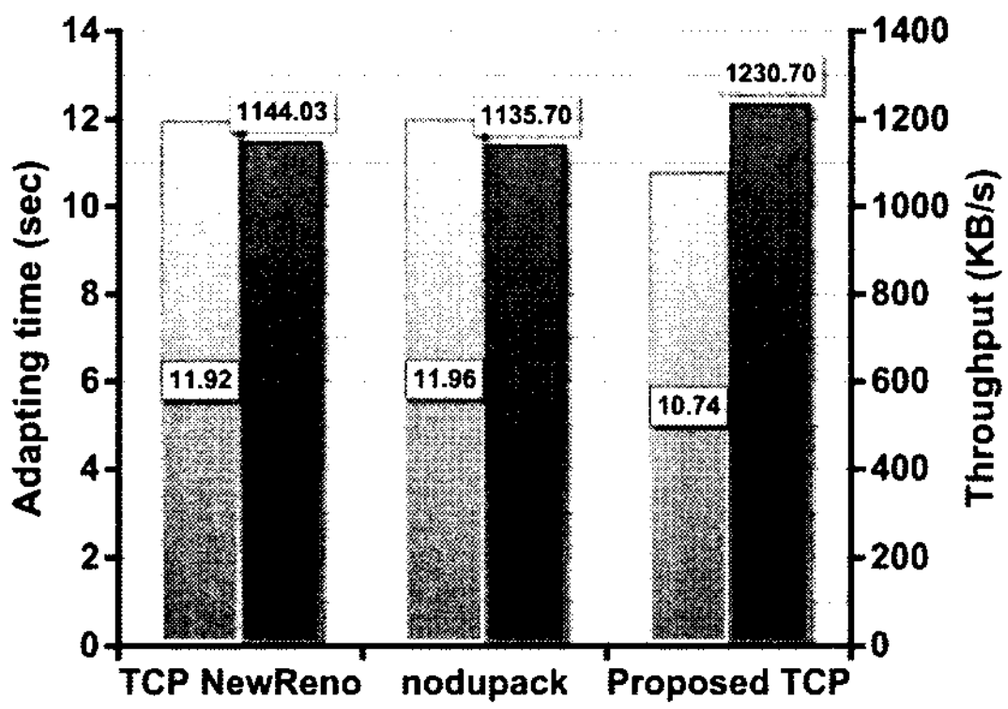


그림 10. UMTS 망에서 패킷 손실이 발생한 경우 무선 LAN에 적응하는데 걸리는 시간 및 처리량
Fig. 10. Wireless LAN adapting time and throughput after DVHO, a packet loss in UMTS link

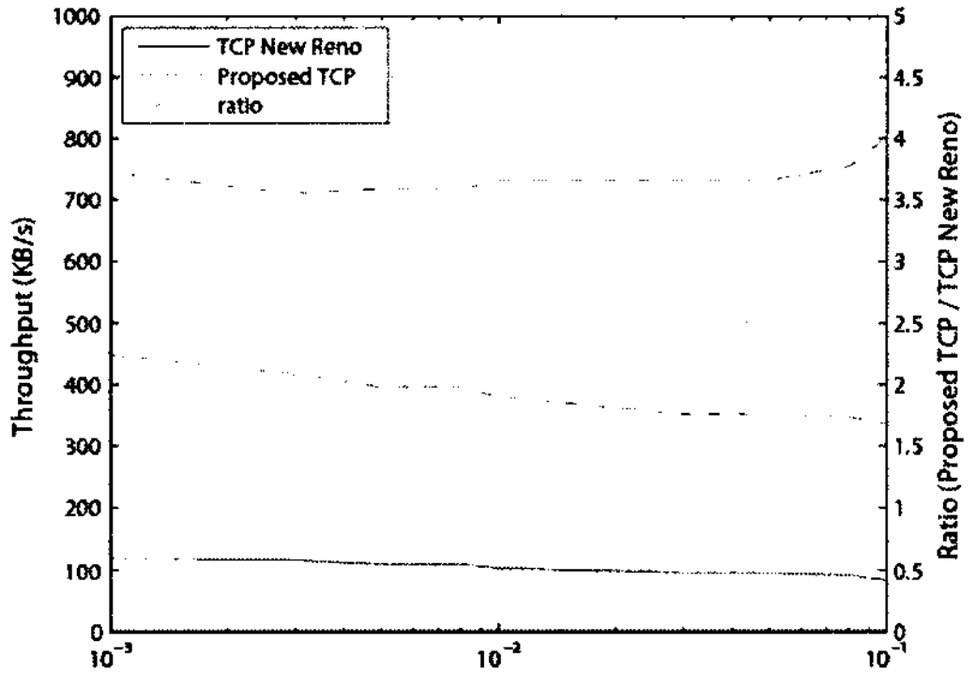


그림 11. UMTS 망의 패킷 손실률 변화에 따른 처리량 변화
Fig. 11. Throughputs during DVHO. Changing loss rate in UMTS link.

V. 결론 및 추후 연구

본 논문에서는 하향 수직 핸드오버 상황에서 발생하는 패킷 재정렬로 인한 TCP의 성능 저하와 함께 이를 해결하면서 발생할 수 있는 버스트 문제를 분석하였으며 송신자 측의 TCP를 기반으로 한 하향 수직 핸드오버 상황에서 패킷 재정렬을 해결하기 위한 기법을 제안 하였다. 본 논문에서 제안한 기법은 송신자 측의 변경만을 요구하기 때문에 수신자 측에서는 변경 없이 적용할 수 있는 장점을 가진다. 본 기법은 셀룰러 망에서 측정된 왕복 시간을 이용하여 하향 수직 핸드오버 수행 과정에서 받게 되는 중복된 ACK들 중 패킷 재정렬로 인한 것들을 성공적으로 판별할 뿐만 아니라 이를 이용하여 송신자 측에서 혼잡 윈도우의 크기 조절을 통해 버스트를 방지하는 것은 물론 하향 수직 핸드오버 수행 과정에서의 처리량 또한 향상시키고 있다. 또한 전송 과정에서 손실된 패킷도 성공적으로 판별하여 손실된 패킷만을 재전송한다. 또한 본 기법은 UMTS 망과 무선 LAN의 지연 시간 차이에 기반하고 있기 때문에 모바일 노드가 송신자가 되어서 데이터를 보내는 업링크 환경에서도 잘 작동할 것으로 기대된다.

그러나 본 논문에서 제안한 기법은 하향 수직 핸드오버를 수행하는 동안의 패킷 재정렬 문제를 처리하는 것에 초점을 맞추고 있을 뿐 새로 옮겨간 무선 LAN의 BDP를 빠르게 활용하는 것은 고려하지 않고 있다는 것은 단점이라고 할 수 있다. 앞으로 패킷 재정렬에 대한 처리와 함께 무선 LAN의 BDP를 빠르게 활용하기 위한 연구를 계속 진행해야 할 것이다.

참 고 문 헌

[1] W. Hansmann, and M. Frank. "On Things to Happen During a TCP Handover" In Proc. *28th IEEE International Conference on Local Computer Networks*, (LCN'03), Oct, 2003.

[2] L. Daniel, and M. Kojo. "Adapting TCP for Vertical Handoffs in Wireless Networks" In Proc. *31st IEEE Conference on Local Computer Networks*, pp. 151-158, Nov, 2006.

[3] Y. Gou, D. Pearce, and P. Mitchell "A Receiver-based Vertical Handover Mechanism for TCP Congestion Control" *IEEE Transactions on Wireless Communications*, vol 5, no. 10, pp. 2824-2833, Oct, 2006.

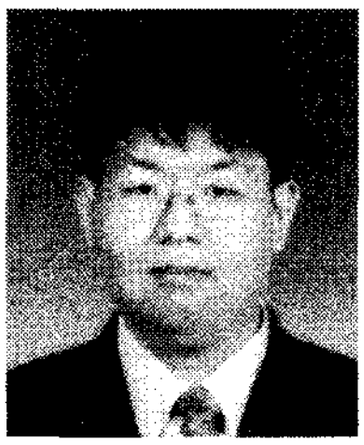
[4] P. Sarolahti, J. Korhonen, L. Daniel, and M. Kojo. "Using Quick-Start to Improve TCP Performance with Vertical Hand-offs" In Proc. *31st IEEE Conference on Local Computer*

Networks, pp. 897-904, Nov, 2006.

[5] Y. Matsushita, T. Matsuda, and M. Yamamoto "TCP Congestion Control with ACK-Pacing for Vertical Handover" *2005 IEEE Wireless Communications and Networking Conference (WCNC 2005)*, vol 3, pp. 1497-1502, Mar, 2005.

[6] S. Floyd, M. Allman, A. Jain, and P. Sarolathi "Quick-Start for TCP and IP" *RFC 4782*, Jan, 2007.

최 여 민 (Yeomin Choi) 준회원



2006년 2월 연세대학교 컴퓨터과학과 졸업
2008년 2월 연세대학교 컴퓨터과학과 석사
2008년 2월~현재 (주)제이투엠소프트

<관심분야> 무선 통신, 수직 핸드오버

송 주 석 (JooSeok Song) 정회원

1976년 2월 서울대학교 전기공학과 졸업
1979년 2월 한국과학기술원 전기·전자공학과 석사
1988년 2월 Univ. of Cali-fornia at Berkeley 박사
1997년 3월~현재 연세대학교 컴퓨터과학과 교수

<관심분야> 유·무선통신, 정보보호