무선통신 환경에서 TCP의 성능개선을 위한 분할 ACKs 기법

(Split ACKs Mechanism for Improving the Performance of TCP in Wireless Communication Environments)

김 길 연 † 진 교 홍 †† 이 정 태 †††

(Kil-Lyeon Kim) (Kyo-Hong Jin) (Jung-Tae Lee)

요 약 최근 이동통신 서비스의 보급이 날로 증가됨에 따라 무선 접속 인터넷 서비스의 사용에 대한 요구가 급증하고 있다. 그러나 인터넷에서 사용되는 TCP 프로토콜은 에러 발생율이 낮은 유선망을 고려하여 설계되었기 때문에 망에서 발생되는 패킷 손실은 망내의 폭주로 인한 것으로 가정하고 폭주제어 알고리즘을 동작시켜 윈도우 크기를 줄인다. 그러나 무선통신망과 같이 에러 발생율이 높은 환경에서는 패킷 손실이 주로 에러 발생에 기인하는데, 이 경우 기존의 TCP 프로토콜을 사용하면 폭주제어 알고리즘이 동작되어 TCP의 성능을 저하시키는 문제점이 발생된다.

따라서 본 논문에서는 유무선 복합망에서 TCP 프로토콜의 성능을 개선하기 위한 Split ACKs 기법을 제안하였다. 이 기법은 기지국에서 무선링크의 패킷 손실 이후에 수신된 ACK 패킷을 여러 개로 쪼개서 TCP 송신측으로 전달한다. 따라서 여러 개의 ACK 패킷을 수신한 TCP 송신측은 폭주제어 알고리즘이 동작되어 감소시킨 윈도우 크기를 빠르게 복귀시켜 주기 때문에, 저하된 TCP 프로토콜의 성능을 신속히 향상시킬 수 있다. 아울러 제안된 기법은 기존 TCP 프로토콜을 그대로 사용할 수 있으며, TCP의 End-to-end Semantics가 유지되는 장점이 있다. 시뮬레이션을 통한 성능분석 결과 이 기법은 기존의 TCP 프로토콜에 비해 약 20%의 성능향상을 보였다.

Abstract Recently, with the advances in wireless communication, more and more users want to access the Internet with mobile terminals or computers. The TCP, popular transport protocol in the Internet, assumes a relatively reliable underlying network where most packet losses are due to congestion. However, in a wireless network, packet losses will occur more often due to vulnerable wireless environments than due to congestion. Therefore, when we use TCP protocol over lossy wireless links, each packet loss on the wireless links results in invoking congestion control mechanism at the source. This causes degradation of overall TCP performance as well as network utilization severely.

In this paper, we propose a 'Split ACKs' mechanism to improve the end-to-end performance of TCP over wireless links. With 'Split ACKs', the base station splits an ACK packet into a few ACK packets and transmits them to the TCP sender when packet losses occur over wireless links. And then TCP sender received several ACK packets recovers the window size quickly to improve the end-to-end performance of TCP. In order to measure throughput of the proposed method, we perform computer simulation. The simulation results show that 'Split ACKs' mechanism improves the performance of TCP about 20%. Other advantages of our approach are that modification of TCP source code is not required and the end-to-end TCP semantics can be maintained.

· 본 연구는 부산대학교 기성회 재원 학술연구조성비에 의한 연구임
· 본 연구는 정보통신부 정보통신분야 우수학교 지원사업비에 의한 연구임

† 비 회 원 : 부산대학교 컴퓨터공학과
kimkl@hyowon.pusan.ac.kr
†† 정 회 원 : 동의대학교 멀티미디어공학과 교수
khjin@hyomin.dongeui.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
jtlee@hyowon.pusan.ac.kr

논문접수 : 1999년 9월 17일
심사완료 : 2000년 5월 31일

1. 서 론

최근 휴대용 단말기의 보급이 증가됨에 따라 언제, 어디서나 인터넷 서비스를 이용하고자 하는 욕구가 날로 증가되고 있다. 한편, 전자우편이나 웹과 같은 일반적인 인터넷 서비스는 TCP(Transmission Control Protocol) 프로토콜을 트랜스포트 프로토콜로 사용하고 있는데, TCP 프로토콜은 패킷 손실율이 낮은 유선망과 고정단말기만을 고려하여 설계되었다. 따라서 유선망에서의 패킷 손실은 대부분 망의 중간 노드에서 버퍼 오버플로우 등으로 인한 폭주(Congestion) 현상 때문에 발생된다. 이에 따라 TCP 프로토콜에서는 신뢰성있는 서비스를 제공하기 위하여 Slow Start와 폭주회피(Congestion Avoidance) 등과 같은 폭주제어 알고리즘을 수행시켜, 망으로 유입되는 데이타의 양을 줄여 더 이상의 폭주 현상이 발생되지 않도록 하고 있다[1].

한편 휴대용 단말기를 이용해서 인터넷을 액세스할 경우, 무선통신의 사용이 필수적이다. 무선통신망은 유선통신망과 달리 에러 발생율이 높아(10^{-3}~10^{-5} 수준의 BER) 패킷 손실이 자주 발생된다[2]. 따라서 유무선 복합망에서 기존의 TCP 프로토콜을 그대로 사용할 경우, TCP 송신측에서는 무선통신망에서 발생된 패킷 손실을 망에 폭주가 발생된 것으로 판단하고 폭주제어 알고리즘을 수행시켜 TCP의 전송속도를 낮추게 된다. 이로 인하여 TCP 프로토콜의 성능은 급격히 저하되고 망의 효율도 떨어지게 된다.

그림 1은 기존 TCP 프로토콜을 유무선 복합망에 적용할 경우 성능저하 현상이 발생되는 원인을 보여주고 있다. 먼저 고정호스트(FH: Fixed Host)에서 전송된 데이타는 기지국(BS: Base Station)을 거쳐 이동호스트(MH: Mobile Host)로 전달된다. 그런데 MH에서 그에 대한 응답으로 전달되는 ACK 패킷이 손실되면 FH에서는 타임아웃이 되어 재전송을 하게 되고 폭주제어 알고리즘에 따라 윈도우 크기(Congestion Window Size: W)를 1로 줄인다. 또한, FH에서 전송된 데이타가 무선링크상에서 손실되더라도 FH에서는 타임아웃이 발생되어 재전송과 함께 윈도우의 크기를 1로 줄이게 된다. 따라서 망에 실제 폭주가 발생되지 않더라도 무선망의 높은 BER로 인하여 윈도우 크기가 감소되어 TCP 프로토콜의 성능이 급격히 저하될 수 있다.

따라서 본 논문에서는 유무선 복합망에서 TCP 프로토콜의 성능을 향상시키기 위하여 BS에서 ACK 패킷을 여러 개로 쪼개어 전달하는 Split ACKs 기법을 제안하였다. 이 기법은 무선링크에서 패킷 손실이 발생되어

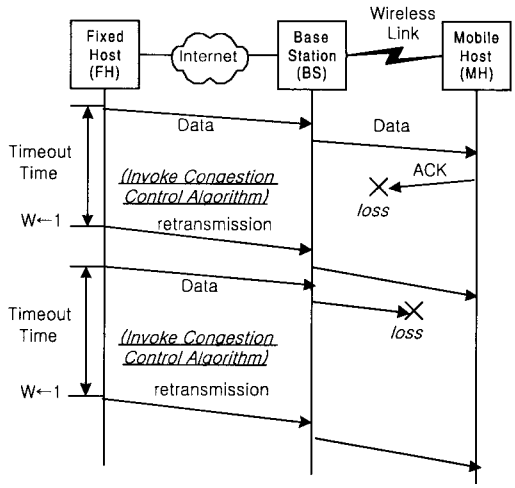

그림 1 무선환경에서의 패킷손실

송신측의 윈도우 크기가 감소되더라도 BS에서 ACK 패킷을 여러 개로 쪼개어 FH에 보냄으로써 윈도우 크기를 빠르게 복구시켜 TCP 프로토콜의 성능을 향상시켜 주는 기법이다. 제안된 기법에 대한 성능분석 결과 높은 BER 환경에서도 기존의 TCP 프로토콜에 비하여 우수한 성능을 나타내었다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 TCP 프로토콜의 폭주제어 알고리즘에 대해서 언급하였으며, 3장에서는 관련연구에 대해서 살펴보았다. 그리고 4장에서는 제안된 Split ACKs 기법을 자세히 기술하였으며, 5장에서는 시뮬레이션을 통해 제안된 기법의 성능을 분석하였다. 마지막으로 6장에서는 결론을 기술하였다.

2. TCP의 폭주제어 알고리즘

TCP 프로토콜은 종단 호스트간의 신뢰성 있는 데이타 전송을 위하여 에러제어, 흐름제어(Flow Control) 및 폭주제어(Congestion Control) 등의 기능을 제공한다. 이중 폭주제어 알고리즘은 종단 호스트를 제외한 나머지 중간의 라우터에서 버퍼 오버플로우가 발생되어 패킷이 손실되는 경우 손실된 패킷을 복구하고 더 이상의 폭주가 발생되지 않도록 흐름을 제어한다. 현재 TCP 프로토콜로 널리 사용되고 있는 4.3BSD의 TCP-Reno버전은 Slow Start, Congestion Avoidance, Fast Retransmit 및 Fast Recovery 알고리즘을 이용하여 폭주를 제어한다[1]. 그림 2는 이들 알고리즘의 동작원리를 보여주고 있다.

그림 2에서 W는 현재의 윈도우 크기로 송신측에서

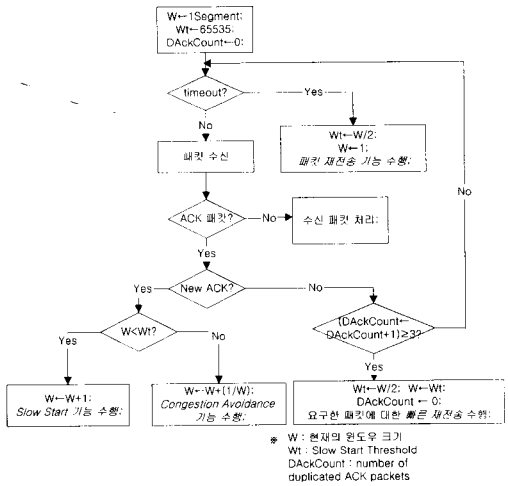

그림 2 폭주제어 알고리즘의 동작원리

ACK없이 한번에 송신할 수 있는 최대 데이타 양이며, Wt는 Slow Start Threshold 값이다. 호설정시 W는 1 세그먼트(Segment)로, Wt는 65,535 바이트로 초기화된다. 송신측에서 데이타 패킷을 전송한 후 새로운 ACK를 수신하게 되면 W와 Wt를 비교하여 W가 작으면 Slow Start 알고리즘이 동작되어 W 값을 1세그먼트씩 증가시키며, Wt가 작으면 폭주회피 알고리즘에 따라 W를 1/W 씩 증가시킨다.

한편, 송신측에서 3개의 동일한 ACK를 수신하면 수신측에서 해당 데이타 패킷만을 수신하지 못하여 대기하고 있는 것으로 간주하고, 송신측은 Fast Retransmit 알고리즘을 적용하여 해당 데이타 패킷부터 즉시 재전송하고 Wt는 W/2로 두고 W는 Wt로 감소시킨다. 그리고 재전송 이후 새로운 패킷이 수신되면 Congestion Avoidance 알고리즘이 동작된다.

또한 송신측에서 데이타 패킷을 전송 후 타임아웃이 되면 망내에 폭주가 발생된 것으로 보고, Wt는 W/2로 두고, W는 1 세그먼트로 설정하여 망으로 유입되는 데이타의 양을 현저히 줄이게 된다.

따라서 TCP 프로토콜이 무선통신망에 적용되는 경우, 실제 폭주로 인한 패킷 손실보다는 에러 발생으로 인한 패킷 손실이 자주 발생되기 때문에, W값이 빈번히 1 세그먼트나 W/2로 감소하게 된다. 이 경우에는 윈도우 크기를 늘려야 하는데 반대로 줄이게 되므로 TCP 프로토콜의 성능은 급격히 떨어지게 된다.

다음 장에서는 무선통신환경에서 TCP 프로토콜의 성능저하 문제를 해결하기 위한 기존의 연구결과에 대하여 살펴보았다.

3. 관련연구

무선통신 환경에서 TCP 프로토콜의 성능을 개선하기 위한 연구가 활발히 수행되었으며, 이들 연구는 크게 End-to-end 기법, Split Connection 기법, Link Layer에서의 재전송 기법, 및 TCP/IP 헤더 압축 기법 등으로 분류될 수 있다. 이 장에서는 각 기법의 동작원리와 장단점을 살펴보았다.

3.1 End-to-end 기법

End-to-end 기법은 기존 TCP 프로토콜과 동일하게 종단간 연결을 설정하고, 무선통신 환경에 적합하도록 TCP 프로토콜을 수정하는 기법이다. 관련된 기법으로는 빠른 재전송[3], ATO(Auxiliary TimeOut)[4], 및 TCP-R[5] 등이 있다.

빠른 재전송 기법은 이동호스트(MH)의 핸드오프(Handoff)로 인해 발생되는 지연과 패킷 손실을 복구하기 위해 고안된 기법이다. 기존 TCP 프로토콜은 세 개의 동일 ACK 패킷을 연속적으로 수신한 경우, 패킷 손실이 일어난 것으로 간주하고 타임아웃이 되지 않더라도 바로 재전송을 수행한다[1]. 빠른 재전송 기법에서는 이러한 TCP 프로토콜의 Fast Retransmit 알고리즘을 무선통신 환경에 적용하기 위해 고정호스트(FH)와 MH의 TCP 프로토콜을 수정하였다. [1]

빠른 재전송 기법의 동작과정은 다음과 같다. 먼저 MH에 핸드오프가 발생되어 FH에서 전달된 데이타가 손실되면, MH의 TCP는 즉시 FH에게 세 개의 동일한 ACK 패킷을 보내준다. 세 개의 동일한 ACK 패킷을 수신한 FH의 TCP는 타임아웃을 기다리지 않고 Fast Retransmit 알고리즘을 수행하여 손실된 패킷부터 재전송한다. 한편, MH에서 송신한 데이타가 핸드오프로 인해 손실되면 MH는 FH의 TCP에게 핸드오프 완료신호를 보내어, FH에서 손실된 패킷에 대한 세 개의 동일한 ACK 패킷을 보내도록 한다. 이에 따라 MH에서는 Fast Retransmit 알고리즘이 수행되어 손실된 패킷부터 재전송한다.

따라서 빠른 재전송 기법은 Fast Retransmit 알고리즘을 이용하여 현재의 윈도우 크기가 1 세그먼트로 감소되는 것을 막아줌으로써 TCP 프로토콜의 성능 저하를 방지한다. 그러나 FH와 MH의 TCP 프로토콜을 수정해야 하는 단점이 있고, 핸드오프로 인해 여러 개의

1) "빠른 재전송 기법"이라는 용어는 참고문헌[3]에서 제안한 기법의 이름이고, "Fast Retransmit 알고리즘"이라는 용어는 TCP 프로토콜에서 폭주제어 알고리즘의 일종으로 사용되는데, 두 가지를 구분하기 위하여 한글 및 영어를 사용하였음.

패킷이 손실되는 경우에 대한 처리방안이 고려되지 않은 문제점이 있다.

그리고 ATO 기법에서는 기존의 유선망에서의 폭주 제어를 준수하면서 시간 만료가 발생한 후 보조 만료 시간 동안 지연된 ACK 패킷을 인식할 수 있도록 함으로써 불필요한 재전송과 오판으로 인한 폭주 제어를 탈피하도록 해준다. TCP-R은 TCP의 확장으로써 MH가 핸드오프를 감지하면 Redirection 메시지를 이용하여 FH에게 변경된 IP 주소를 전달하고 FH의 TCP 계층에서는 새로운 주소를 이용하여 패킷을 전송하므로 TCP 계층에서 종단간의 이동성을 지원하게 된다. 이 두 가지 기법 모두가 TCP 프로토콜을 수정해야 하는 단점이 있다.

3.2 Split Connection 기법

Split Connection 기법은 유무선망의 특성을 고려하여 종단간의 TCP 연결을 FH와 BS간의 연결과 BS에서 MH간의 연결로 분리하여, FH에서는 무선링크상에서 발생되는 패킷 손실을 인식하지 못하도록 하는 기법이다.

Split Connection 기법을 이용한 기존의 연구에는 I-TCP(Indirect-TCP)[6], M-TCP[7], Mobile-TCP[8] 및 METP[9] 등이 있으며, 각각은 적용된 메커니즘이 조금씩 다를 뿐 TCP 연결을 두 부분으로 나눈다는 점에서는 유사하다. 먼저 I-TCP의 동작원리를 살펴보면 그림 3과 같다.

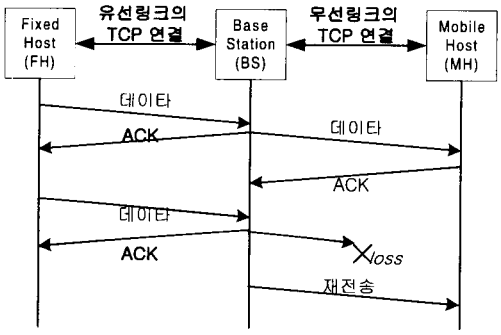

그림 3 I-TCP의 동작원리

그림 3에서 보는 바와같이 FH에서 MH로 전송된 데이타는 BS에서 MH로 중계해 주며 이에 대한 ACK가 MH로부터 도착되기 전에 BS에서 FH로 전달하여 준다. 그리고 무선링크상에서 패킷손실이 발생되면 이를 FH에게 알리지 않고 BS에서 자체적으로 MH에 재전송하여 FH의 TCP 프로토콜 성능에는 영향이 없도록 한다. 특히 BS와 MH간에는 무선링크의 특성에 적합한 새로운 프로토콜을 사용할 수 있으므로 MH의 성능도 향상시킬 수 있는 장점을 가지고 있다.

그러나 I-TCP의 가장 큰 문제점은 종단호스트간의 End-to-end Semantics가 파괴된다는 점이다. 즉 트랜스포트 프로토콜인 TCP의 종단간 연결이 I-TCP로 인해 두 개의 연결설정으로 분리되므로, 송신측에서 수신한 ACK가 수신측에서 데이타를 제대로 수신하였음을 의미하지 않는다. 또한 BS에서도 재전송을 담당해야 하므로 BS의 복잡도가 높아지는 단점도 있다.

그리고 M-TCP와 Mobile-TCP 기법은 유무선 링크를 구분하는 점에서는 I-TCP와 같다. M-TCP는 I-TCP에서 TCP의 End-to-end Semantics가 파괴되는 문제점을 개선하기 위한 메커니즘을 제공하였으며, 이동환경에서 잦은 disconnection시의 성능 향상을 목표로 하였는데 SH(Supervisor Host) 등의 도입으로 구현이 복잡하다. Mobile-TCP 기법은 MH의 오버헤드를 줄이기 위해서 MH와 BS간의 트랜스포트 계층 프로토콜을 비대칭적으로 설계하여 대부분의 작업은 Mobile Gateway에서 수행하도록 했다는 것이 특징이다.

3.3 Link Layer 기법

Link Layer 기법은 무선링크상에 패킷손실이 발생되면 이를 송신측 TCP에서 감지하기 전에 BS의 데이타 링크 계층에서 재전송하는 기법이다. 대표적인 연구로는 Snoop 모듈[10]과 AIRMAIL[11] 등을 들 수 있으며, 그림 4에서는 Snoop 모듈 방법에 대해서 보여주고 있다.

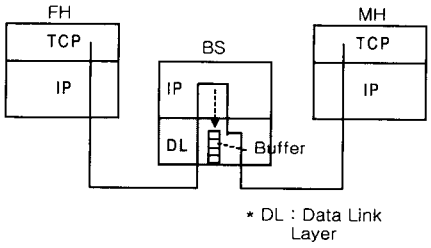

그림 4 Snoop 모듈

그림 4에서 보는 바와같이 Snoop 모듈은 BS의 데이타 링크 계층에 데이타 패킷을 저장하기 위한 버퍼를 마련한다. 그리고 FH에서 MH로 데이타를 전송하는 경우, BS의 버퍼에 데이타를 저장해 두고 무선링크상에서 패킷손실이 발생되면 BS에서 MH로 자체적으로 재전송을 수행한다. 한편, MH에서 FH로 데이타를 전송하는

경우에 패킷이 손실되면 BS에서 MH로 NACK를 송신하여 MH에서 해당 데이타를 재전송하도록 조치한다.

그리고 AIRMAIL 기법은 MH의 낮은 처리 능력 및 파워 부족 등 문제점을 고려하여, MH와 BS간의 링크 계층 프로토콜을 비대칭적으로 설계하여 BS에서 대부분 작업을 처리하도록 하였으며, 링크계층에서 FEC (Forward Error Correction) 및 재전송 기능을 제공하도록 하였다.

이러한 Link Layer 기법은 Split Connection 기법과는 달리 데이타 링크 계층에서 재전송하기 때문에 TCP의 End-to-end Semantics를 유지할 수 있는 장점이 있다. 그러나 데이타 링크 계층의 재전송과 TCP 계층의 재전송이 중복될 수 있는 문제점이 있다. 즉, FH에서는 BS의 재전송에도 불구하고 타임아웃이 발생되면 패킷을 재전송하고 폭주제어 알고리즘을 작동시킬 수 있다. 그리고 BS에서 패킷을 저장하고 재전송하므로 BS의 복잡도가 높아진다는 단점을 가지고 있다.

3.4 TCP/IP 헤더 압축 기법

Van Jacobson은 무선 링크와 같은 저속의 링크를 위해 TCP/IP 헤더 압축 기법[12]을 제안하였고, Degermark은 무선 통신망을 위한 TCP/IP 및 UDP/IP 헤더 압축 기법[13]을 제안하였다. 헤더 압축 기법은 연속적인 TCP/IP 패킷 흐름에서 헤더의 대부분 내용이 바뀌지 않는다는 점에 착안하여, 송신측에서 보내는 첫번째 TCP/IP 패킷은 완전한 헤더 정보를 포함해서 전송하고, 두번째 패킷부터의 헤더에는 최소한의 정보만 포함해서 전송한다. 수신측에서는 첫번째 패킷의 완전한 헤더 내용을 저장하고 있다가 헤더가 압축된 패킷을 받으면 저장해 둔 헤더정보와 받은 패킷의 헤더로부터 패킷의 전체 헤더를 복구한다. 이 기법에서는 헤더 정보를 전송하는데 사용되는 대역폭이 줄어들기 때문에 무선링크의 대역폭을 절약할 수 있는 장점이 있다.

그러나 송신측에서 전송한 완전한 헤더 정보가 포함된 패킷이 전송 중에 손실되면 수신측에서는 그 이후에 수신된 모든 패킷을 복원할 수 없는 문제점이 발생될 수 있을 뿐만 아니라 무선통신망에서 패킷 손실로 인한 TCP 프로토콜의 성능저하에 대한 해결 방안은 제시하지 못하였다.

4. 제안한 Split ACKs 기법

앞장에서 살펴본 바와 같이 기존의 연구는 TCP의 End-to-end Semantics 파괴, TCP 프로토콜의 수정으로 인한 호환성 상실, BS의 높은 복잡도, 및 중복된 재전송 등의 문제점을 가지고 있다. 따라서 본 논문에서는 이러한 문제점을 해결하면서 TCP 프로토콜의 성능을 향상시키기 위한 Split ACKs 기법을 제안하였다. 제안된 기법은 FH와 MH에서는 TCP 프로토콜을 수정없이 그대로 사용하고, 대신 BS에 Split ACKs 모듈을 첨가하여 TCP 프로토콜의 성능향상을 도모하였다. 먼저 Split ACKs 기법에서 사용되는 프로토콜 구조는 그림 5와 같다.

그림 5에 제시된 바와같이 FH와 MH는 기존의 TCP/IP 프로토콜 구조를 그대로 사용하지만 BS에는 고유의 IP라우팅 기능과 더불어 Split ACKs 모듈이 첨가된다. 이러한 프로토콜 구조에서 FH와 MH간의 데이타 송수신을 위해 종단간 호설정이 이루어지고, 평상시 BS는 FH나 MH에서 보내 온 데이타를 받아 중계하는 고유의 라우팅 기능만을 수행한다.

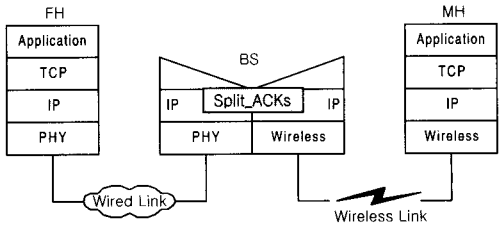

그림 5 Split ACKs 기법의 프로토콜 구조

한편, BS에서는 폭주가 발생하지 않는다고 가정하였을 경우, BS와 MH간의 무선링크상에 링크 에러 혹은 핸드오프로 인해 패킷 손실이 발생되면 FH에서는 마치 폭주현상이 발생된 것으로 오인하게 되며 윈도우 크기(W)를 1 세그먼트나 W/2로 줄이고 TCP 프로토콜의 폭주제어 알고리즘을 동작시켜 FH의 패킷 전송 속도를 낮추게 된다. 그 이후 줄어든 윈도우 크기는 새로운

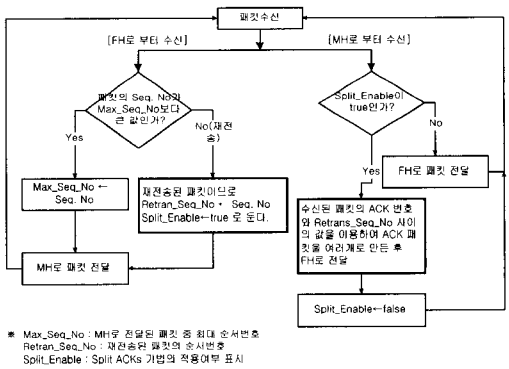

그림 6 Split ACKs 기법의 동작원리

ACK가 FH에 도착되면 Slow Start와 Congestion Avoidance 알고리즘에 따라 증가된다. 본 논문에서는 TCP 프로토콜의 폭주제어 알고리즘이 "수신된 ACK 패킷의 개수에 따라 윈도우 크기를 증가"시킨다는 점에 착안하여, BS가 MH로 부터 재전송된 패킷에 대한 ACK를 수신하면 이를 여러 개로 나누어 FH에 전달하여 FH의 윈도우 크기를 빠르게 복귀시키는 Split ACKs 기법을 고안하였다. Split ACKs 기법의 동작원리는 그림 6에 제시된 순서도에서 자세히 설명하고 있다.

그림 6에서 보는 바와 같이 BS에서는 FH로부터 패킷을 수신하면 기존의 전송된 패킷의 순서번호와 비교하여 재전송되는 패킷인지 아닌지를 판단한다. 만약 재전송되는 패킷이 아닌 경우에는 MH로 그대로 전달하지만, 재전송되는 패킷인 경우에는 패킷의 순서번호를 변수 Retrans_Seq_No에 저장하고 Split ACKs 기법을 활성화시킨다. 이후에 MH로 부터 패킷이 수신되면 수신된 패킷내의 ACK 번호와 Retrans_Seq_No 사이의 값을 ACK 번호로 하는 ACK 패킷을 여러개 만들어 FH로 전달하면, FH에서는 도착된 패킷의 개수에 따라 윈도우의 크기를 증가시키게 된다. Split ACKs 기법의 동작과정을 예를 들어 설명하면 그림 7과 같다.

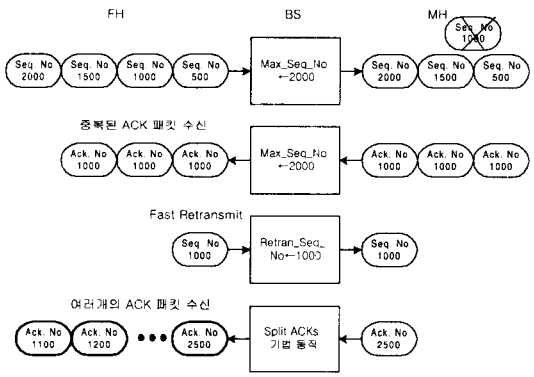

그림 7 Split ACKs 기법의 동작 예

그림 7에서 FH는 BS를 통하여 MH로 4개의 패킷을 전송하였지만, BS와 MH간의 무선링크상에서 두번째 패킷이 손실되었다. 이로 인하여 MH에서는 중복된 ACK를 FH로 보내어 손실된 패킷의 재전송을 요구하게 되며, BS에서는 Retrans_Seq_No에 1000을 저장하고 Split ACKs기법을 동작시킨다. 그리고 FH에서는 재전송 요청된 Seq_No가 1000인 패킷을 보내고 Fast Retransmit 알고리즘이 동작된다. 이후에 BS가 MH로부터 ACK 번호가 2500인 패킷을 수신하면 Split ACKs 알고리즘에 따라 [Retrans_Seq_No=1000, Ack_No=2500] 사이의 값을 이용하여 ACK 번호가 1100, 1200, , 2500인 패킷을 FH로 전달하여 준다. 따라서 FH에서는 줄어든 윈도우 크기를 빠르게 회복시켜 패킷 전송속도를 높일 수 있게 된다.

한편, BS에서 FH로 전달되는 ACK 패킷의 개수는 다음의 식(1)을 통해서 구할 수 있으며 이러한 조건하에서 각 ACK 패킷의 ACK 번호가 결정된다. 식(1)에서는 수신측의 수신버퍼크기를 고려해서 쪼개는 ACK 패킷 수를 결정하였으며 사용한 파라미터는 두개가 있는데, 그중 awnd는 데이타 송수신시 MH에서 알려주는 Advertised Window Size이고, MSS(Maximum Segment Size)는 호설정시에 정해지는 패킷의 최대 크기다. 예를 들어, 만약 MSS가 256바이트이고, awnd가 4096바이트이면 식(1)에 의해 16개의 ACK 패킷이 생성됨을 알 수 있다.

ACK 패킷의 개수 = awnd / MSS ..... (1)
(awnd : advertised window size,
MSS : Maximum Segment Size)

위에서 설명한 바와같이 Split ACKs 기법은 FH와 MH의 TCP 프로토콜을 수정없이 그대로 사용할 수 있을 뿐만 아니라, BS의 복잡도도 기존의 다른 기법에 비하여 낮은 편이다. 또한 TCP의 End-to-end Semantics를 유지하며 재전송이 중복되는 경우도 발생되지 않는다.

5. 성능분석

관련연구에서 살펴보았듯이, 기존의 기법들은 TCP 소스 코드의 수정을 필요로 하거나, TCP의 End-to-end Semantics 파괴, 트랜스포트 계층과 링크계층에서의 중복된 재전송 등 문제점을 안고 있다. 이러한 문제점이 해결되지 못한 기존 연구와 문제점을 해결한 Split ACKs 기법의 성능을 비교하는 것 보다는 기존 TCP와의 성능비교가 더욱 관심 대상이 될 것으로 사료된다. 따라서 5.1절에서는 Split ACKs 기법과 기존 기법의 장단점을 비교분석하였고, 5.2절에서는 시뮬레이션을 통하여 Split ACKs 기법과 기존 TCP의 성능을 비교분석하였다.

5.1 기존연구와의 비교분석

표 1은 제안한 Split ACKs 기법과 기존 연구의 장단점을 비교분석한 결과이다.

표1에서 보여주듯이, Split ACKs 기법은 I-TCP와는

표 1 기존연구와의 비교분석

비교항목 / 기법	End-to-end Semantics	TCP source modification	BS의 복잡도	중복된 재전송
빠른 재전송	유지	필요	낮음	없음
I-TCP	파괴	불필요	높음	있음
Snoop 모듈	유지	불필요	높음	있음
헤더 압축 기법	유지	필요	낮음	없음
Split ACKs 기법	유지	불필요	낮음	없음

달리 TCP의 End-to-end Semantics를 유지할 수 있으며, 또한 빠른 재전송 기법과는 달리 FH와 MH의 TCP 소스를 수정할 필요가 없으므로 기존의 응용프로그램을 그대로 사용할 수 있는 장점이 있다. 그리고 BS에서 재전송을 하거나 버퍼링을 많이 할 필요가 없으므로 I-TCP나 Snoop 모듈에 비해 BS의 복잡도도 낮은 편이다. 뿐만아니라 Snoop 모듈과는 달리 하위계층의 재전송과 중복되는 경우도 피할 수 있다.

5.2 시뮬레이션을 통한 성능분석

이 절에서는 제안한 Split ACKs 기법의 성능을 컴퓨터 시뮬레이션을 통하여 기존의 TCP와 비교분석하였다. 시뮬레이션은 REAL simulator[14]를 사용하였으며 시뮬레이션 모델은 그림 8과 같다.

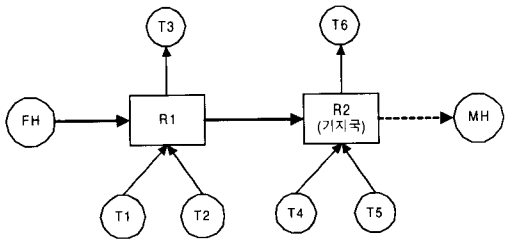

그림 8 시뮬레이션 모델

그림 8에서 보는 바와 같이 유무선 복합망의 특성을 고려하여 설계한 시뮬레이션 모델은 총 10개의 노드로 구성하였으며, FH는 라우터 R1과 기지국 R2를 거쳐 MH로 데이타를 전송한다. 그리고 (T1, T2) 및 (T4, T5)는 Poisson 분포 특성을 가진 트래픽을 발생시켜 각각 T3와 T6에 전달한다. 각 노드와 링크에는 유무선 환경에서 데이타 전송 시의 지연, 무선 링크의 패킷 전송능력 등을 고려하여 다음과 같이 파라미터를 할당하였다. 먼저 R2에서 MH로 가는 링크는 대역폭을 100Kbps로 하고, 나머지 링크는 2Mbps의 대역폭을 가진 것으로 가정하였다. 그리고 R2에서 MH간의 전파지연은 10ms로 가정하고 나머지 링크는 30ms의 전파지연을 갖는 것으로 가정하였다. 전달되는 데이타 패킷의 크기는 500바이트, ACK 패킷의 크기는 40바이트로 하였다. 초기에 송신측 윈도우 크기는 20 세그먼트로 설정하였으며, FH 및 MH에 적용된 TCP 프로토콜은 4.3BSD Reno 버전이다. 한편, 시뮬레이션에서 사용한 모델에서는 망의 폭주로 인한 패킷 손실이 발생하지 않는 것으로 가정하였다. 시뮬레이션은 FH에서 10,000개의 데이타 패킷을 MH로 전달하고 R2와 MH간의 BER을 10^{-3}~10^{-5}까지 조정해가면서 Split ACKs 기법을 적용한 경우와 적용하지 않은 경우로 나누어서 성능을 측정하였다.

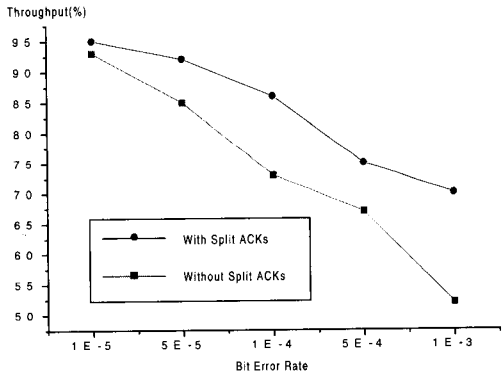

그림 9 Split ACKs 기법의 성능분석

그림 9는 Split ACKs 기법을 적용하지 않은 TCP 프로토콜의 성능과 Split ACKs 기법을 적용한 TCP 프로토콜의 성능을 비교분석한 그래프다. 그림 9에서 성능은(Throughput) BER에 따라 10,000개의 패킷을 모두 전송하는데 소요되는 시간을 측정하여 전송속도를 계산한다음, 이를 에러가 없을 때의 전송속도에 대한 백분율로 표시한 것이다. 즉, 에러가 없을 경우, 10,000개의 패킷을 성공적으로 전송하는데 소요되는 시간 S를 측정하고 이를 이용하여 식(2)와 같이 전송속도 $Speed_0$를 계산하였다.

$$Speed_0 = \frac{(PacketSize \times 10,000)}{S} Bps \quad \cdots\cdots\cdots (2)$$

그리고 10^{-5}~10^{-3}의 BER 환경에서, 10,000개의 패킷을 전송하는데 소요되는 시간을 측정하고 식(2)와 유사하게 전송속도 $Speed_{BER}$를 계산하였다. 이와 같이 계산

된 $Speed_0$와 $Speed_{BER}$를 이용하여 성능을 $\frac{Speed_{BER}}{Speed_0} \times 100\%$로 계산하였으며, 이를 그림 9의 Y 축으로 표시하고 X 축은 BER를 표시했다. 그림 9에서 보여주는 바와 같이, BER이 10^{-3}일 경우, Split ACKs 기법을 적용하면 약 25%의 성능향상이 있음을 알 수 있으며, BER이 $5*10^{-5}$일 경우에는 약 10%의 성능 향상이 있음을 알 수 있다. 따라서 무선링크의 에러율이 10^{-3}~10^{-5}까지 변할 때, Split ACKs 기법을 적용한 경우에 기존의 TCP 보다 약 10~25%의 성능향상이 있음을 알 수 있다. 이것은 높은 에러율로 인하여 윈도우 크기가 감소되더라도 Split ACKs 알고리즘에 따라 윈도우 크기를 빠르게 증가시키기 때문이다.

그림 10은 시간에 따른 송신측 윈도우 크기의 변화를 보여주는 그래프로써 Split ACKs 기법을 적용하지 않은 것에 비하여 Split ACKs 기법을 적용한 경우에 윈도우 크기가 계속해서 큰 값을 유지하는 것을 알 수 있다. 그림 10에서, 윈도우 값이 증가되다가 급격히 기존 값의 반으로 줄거나 1로 줄어드는 시점은 재전송이 발생한 시점이다. 예를 들면, time=3.0, 9.0, 16.0 등 시점에서 재전송 혹은 빠른 재전송이 발생하였으므로 윈도우 값이 급격히 떨어졌음을 알 수 있다. 그리고 그림 10의 특정 시점에서 기존의 윈도우 값이 더 크게 나타나는 경우가 있는데, 이것은 제안한 기법과 기존 TCP의 성능 차이 때문에 발생한 것으로, 제안한 기법의 윈도우 크기가 기존 TCP보다 좀더 빠르게 변하므로써, 어떤 특정한 시점에서 기존의 TCP가 좀더 큰 윈도우 값을 나타낼 수도 있다. 하지만 전체적으로 볼 때, 제안한 Split ACKs 기법의 윈도우가 더 큰 값을 나타낸다.

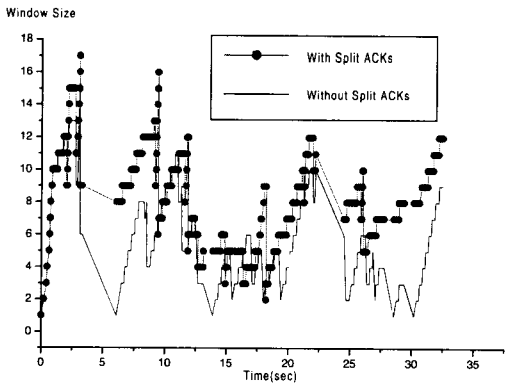

그림 10 시간에 따른 윈도우 크기의 변화

한편, 위에서 기술한 시뮬레이션의 단편성을 고려하여, 그림 8의 시뮬레이션 모델에서 MH의 개수를 증가시키면서 시뮬레이션을 재수행하였다. 그리고 시뮬레이션 모델에서 각 링크에 주어지는 파라미터 값을 조정해 가면서 시뮬레이션을 수행했다. 그 결과 제안한 기법이 기존의 TCP 보다 평균 10~25%의 성능향상이 있음을 알 수 있었다.

6. 결 론

TCP 프로토콜은 패킷손실율이 낮은 유선망을 고려하여 설계되었기 때문에 에러가 발생되면 주로 버퍼 오버플로우가 원인인 것으로 생각하고 폭주제어 알고리즘을 동작시킨다. 반면, 유무선망이 복합된 통신망 구조에서는 무선망의 높은 에러 발생으로 인하여 재전송이 발생되는데 이 경우에는 폭주제어 알고리즘이 동작되면 TCP 프로토콜의 성능은 더욱 저하되는 문제점을 안고 있다.

따라서 본 논문에서는 이러한 문제점을 해결하기 위하여 Split ACKs 기법을 제안하였다. 제안된 기법은 BS에서 무선링크의 패킷손실 이후에 수신된 ACK 패킷을 여러 개로 나누어 TCP 송신측에 전달한다. 이에 따라 TCP 프로토콜은 기존의 폭주제어 알고리즘에 따라 줄어든 윈도우 크기를 빠르게 복원하여 TCP 프로토콜의 성능을 향상시킬 수 있다. Split ACKs 기법은 기존의 연구결과들과는 달리 TCP 프로토콜을 수정할 필요가 없으며, End-to-end TCP Semantics도 유지된다. 또한 BS의 복잡도도 낮을 뿐만 아니라 재전송이 중복되는 현상도 발생되지 않는 장점을 가지고 있다.

제안된 기법의 성능을 검증하기 위하여 시뮬레이션을 수행하였으며, 그 결과 기존의 TCP 프로토콜에 비하여 에러가 많이 발생하는 환경에서도 약 20%의 성능향상을 나타내었다.

참 고 문 헌

[1] W. Richard Stevens, TCP/IP Illustrated, Vol. 1, Addison Wesley, 1994.

[2] M. G. Gouda and S. Paul, "A Wireless Link Protocol: Design by Refinement," Proceedings of International Conference on Network Protocols, pp.192-200, Nov., 1995.

[3] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE Journal on Selected Areas in Communications, 13(5), June, 1995.

[4] Gon Chun and Byeong Gi Lee, "Auxiliary Timeout and Selective Packet Discard Schemes to Improve TCP Performance in PCN Environment," Proceedings of ICC'97, Vol. 1, pp.381-385, June, 1997.
[5] Daichi Funato, Kinuko Yasuda and Hideyuki Tokuda, "TCP-R: TCP Mobility Support for Continuous Operation," Proceedings of International Conference on Network Protocols, pp.229-236, Oct., 1997.
[6] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," Proceedings of 15th International Conference on Distributed Computing Systems, Vancouver, Canada, pp.136-143, May, 1995.
[7] K. Brown and Suresh Singh, "M-TCP: TCP for Mobile Cellular Networks," ACM SIGCOMM, Computer Communication Review, pp.19-43, July, 1997.
[8] Zygmunt J. Haas and Prathima Agrawal, "Mobile-TCP: An Asymmetric Transport Protocol Design for Mobile Systems," Proceedings of ICC'97 in Montreal Canada, Vol. 2, pp.1054-1058, June, 1997.
[9] Kuang-Yeh Wang and Satish K. Tripathi, "Mobile-End Transport Protocol an Alternative to TCP/IP over Wireless Links," Proceedings of INFOCOM'98, Vol. 3, pp.1046-1053, April, 1998.
[10] Elan Amir, Hari Balakrishnan, Srinivasan and Randy H. Katz, "Efficient TCP over Networks with Wireless Links," Proceedings of 5th Workshop on Hot Topics in Operating Systems, pp.35-40, May, 1995.
[11] E. Ayanoglu, S. Paul, T.F. Laporta, K. K. Sabnani, and R.D. Gitlin, "AIRMAIL : A Link-Layer Protocol for Wireless Networks," ACM/Baltzer Wireless Networks Journal, pp.47-60, Feb., 1995.
[12] V. Jacobson, "Compressing TCP/IP Header for Low-Speed Serial Links," Network Working Group, Request for Comments 1144, Feb., 1990.
[13] M. Degermark, M. Engan, B. Nordgen, and S. Pink, "Low-loss TCP/IP Header Compression for Wireless Networks," MOBICOM'96, pp.1-14, Rye, New York, Nov., 1996.
[14] S. Keshav, "REAL: A network simulator," Report 88/472, Computer Science Department, University of California at Berkeley, CA, 1988.

김 길 연
1993년 중국 청도해양대학교 전자계산학과(학사). 1996년 동아대학교 컴퓨터공학과(석사). 1996년 ~ 현재 부산대학교 컴퓨터공학과 박사과정 재학중. 관심분야는 TCP/IP 프로토콜, 이동 인터넷 프로토콜, 고속 네트워크.

진 교 홍
1991년 부산대학교 컴퓨터공학과(학사). 1993년 부산대학교 컴퓨터공학과(석사). 1997년 부산대학교 컴퓨터공학과(공학박사). 1997년 7월 ~ 2000년 2월 국방과학연구소 선임연구원. 2000년 3월 ~ 현재 동의대학교 멀티미디어공학과 전임강사. 관심분야는 TCP/IP 프로토콜, 인터넷 서비스, 광통신.

이 정 태
1976년 부산대학교 공과대학 전자공학과(학사). 1983년 서울대학교 공과대학 컴퓨터공학과(석사). 1989년 서울대학교 공과대학 컴퓨터공학과(공학박사). 1977년 3월 ~ 1977년 12월 한국과학기술원 연구원. 1977년 12월 ~ 1985년 2월 한국전자통신연구소 선임연구원. 1985년 3월 ~ 1988년 2월 동아대학교 공과대학 조교수. 1992년 8월 ~ 1993년 7월 일본 NTT 연구소 초빙연구원. 1988년 3월 ~ 현재 부산대학교 컴퓨터공학과 교수. 관심분야는 고속 트랜스포트 프로토콜, 인터넷 서비스, ATM 프로토콜, 개인이동통신, VoIP.