무선 네트워크에서 재전송 손실 복구를 통한 TCP SACK 성능 향상 방안

(Performance Improvement of TCP SACK using Retransmission Fiailure Recovery in Wireless Networks)

박 건 영 † 김 범 준 †† 김 동 민 † 한 제 찬 † 이 재 용 †††

(Gunyoung Park) (Beomjoon Kim) (Dongmin Kim) (Jechan Han) (Jaiyong Lee)

요 약 무선 전송 기술이 발전함에 따라 현재 유선 네트워크에서 주로 동작하는 인터넷은 무선 환경으로 확장되어 가고 있다. 인터넷의 주요 수송 계층 프로토콜인 TCP(transmission control protocol)는 신뢰성이 높은 유선 네트워크상에서 동작한다는 가정 하에 설계되고 개발되었다. 그러나 무선 환경에서는 패킷 손실이 망의 혼잡(network congestion)에 의해서뿐만 아니라 전송 과정에서의 물리적인 현상에 의한 에러에 의해 발생할 수 있고, 이로 인해 발생하는 비 혼잡 패킷 손실(non-congestion packet loss)에 의해서 TCP의 성능은 크게 저하될 수 있다. 전반적인 TCP의 처리율(throughput)은 재전송 타임아웃(retransmission timeout)의 발생 빈도에 의해 큰 영향을 받기 때문에 이를 해결하기 위한 많은 연구가 진행되어 왔다. 그러나 재전송된 패킷 손실(lost retransmission)로 인한 재전송 타임아웃은 여전히 해결되지 못한 상태이다. 따라서 본 논문에서는 재전송 손실을 감지하고 이를 복구할 수 있는 간단한 알고리듬을 제안한다. 제안된 알고리듬의 성능을 분석하기 위해서 무선 환경에서 발생하는 두 가지 형태의 패킷 손실 모델에 대한 시뮬레이션을 수행하였다. 시뮬레이션 결과를 통해서 제안된 알고리듬이 손실 복구 차원에서 TCP의 성능을 상당히 향상시킴을 보인다.

키워드 : TCP, TCP 손실 복구, 선택승인 옵션, 비혼잡 패킷 손실, 재전송 손실

Abstract As today's networks evolve towards an IP-based integrated network, the role of transmission control protocol(TCP) has been increasing as well. As a well-known issue, the performance of TCP is affected by its loss recovery mechanism that is comprised of two algorithms; fast retransmit and fast recovery. Although retransmission timeout(RTO) caused by multiple packet losses can be avoided by using selective acknowledgement(SACK) option, RTO cannot be avoided if a retransmitted packet is lost. Therefore, we propose a simple modification to make it possible for a TCP sender using SACK option to detect a lost retransmission. In order to evaluate the proposed algorithm, simulations have been performed for two scenarios where packet losses are random and correlated. Simulation results show that the proposed algorithm can improve TCP performance significantly.

Key words : TCP, TCP loss recovery, selective acknowledgement (SACK) option, non-congestion packet loss, lost retransmission

1. 서 론

인터넷에서 널리 사용되고 있는 수송 계층 프로토콜인 TCP의 주요 기능은 망의 혼잡을 방지하기 위해서 사용자의 전송율을 직접 제어하는 혼잡제어(congestion control)라고 할 수 있다[1]. 혼잡제어는 패킷 손실을 감지하고 이를 재전송에 의해서 복구하기 위한 기능을 포함하는데 이를 흔히 손실복구(loss recovery)라고 한다. 손실 복구 과정에서 손실된 패킷을 재전송에 의해 복구하는 것이 불가능한 경우 재전송 타임아웃(retransmission

· 본 연구는 한국과학재단 목적기초연구(R01-2002-000-00531-0) 지원으로 수행되었음.

† 비 회 원 : 연세대학교·전기전자공학과
hjcyy@nasla.yonsei.ac.kr
†† 비 회 원 : LG전자 이동통신기술연구소 연구원
beom@lge.com
††† 종신회원 : 연세대학교 전자공학과 교수
jyl@yonsei.ac.kr

논문접수 : 2003년 12월 24일
심사완료 : 2005년 2월 28일

timeout)이 발생하게 된다[2]. 재전송 타임아웃이 발생하면 TCP 송신원은 이 기간동안 패킷을 전혀 전송하지 못할 뿐 아니라 윈도우가 초기 상태인 1로 초기화되어 slow start상태에서 다시 증가해야 하기 때문에 잦은 재전송 타임아웃의 발생은 처리율과 같은 TCP의 전반적인 성능에 큰 영향을 미친다.

이 문제를 극복하기 위해서 TCP Reno[2]의 fast recovery 알고리듬은 지속적으로 수정되어 왔는데 대표적인 것으로 TCP NewReno[3]와 선택승인(selective acknowledgement)옵션의 사용[4,5]을 들 수 있다. 특히 선택 옵션의 사용은 TCP NewReno에 비해서 손실 복구 과정을 보다 빠른 시간 내에 마칠 수 있는 장점이 있기 때문에[6] 가까운 미래에 널리 도입될 것으로 예상된다.[1] 그러나 이들 개선된 종류의 TCP 구현을 포함한 모든 TCP는 재전송이 손실되는 경우 발생하는 타임아웃을 방지할 수 없다[6,7]. 실제 인터넷상의 측정 결과[7]에 따르면 약 4%의 타임아웃의 발생 원인이 재전송의 손실에 의해서 발생한다.

최근 무선 전송 기술의 발전에 따라서 인터넷의 영역이 무선 환경으로 확장되고 있는데, 무선 환경에서 이 문제는 더욱 심각해질 수 있다. 무선 환경에서는 다중경로페이딩(multipath fading)을 비롯한 다양한 물리적인 현상들에 의해 비트 오류의 발생 빈도(bit error rate)가 유선 환경에 비해서 상대적으로 높기 때문에 모든 패킷 손실은 망의 혼잡으로 인해서 발생한다는 가정[2]은 성립될 수 없기 때문이다[8-10].

따라서 본 논문에서는 선택 승인을 사용하는 TCP 송신원이 손실된 재전송을 감지하고 이를 재전송에 의해서 복구할 수 있도록 하기 위한 간단한 알고리듬을 제안하고 무선 환경에서 동작하는 경우의 성능을 시뮬레이션을 통해서 분석하였다. 제안된 알고리듬은 TCP 송신원을 간략히 수정함으로써 구현될 수 있고, TCP의 고유 특성중 하나인 종단간 동작 원칙(end-to-end semantics)을 준수함으로써 기존 TCP와 완벽하게 호환되는 장점을 가지고 있다.

2. TCP 선택승인(selective acknowledgement) 옵션

2.1 등장 배경

기본적인 TCP의 명세[11]가 발표된 이래 혼잡으로 인한 TCP의 급격한 성능 저하를 방지하기 위해서 TCP 혼잡제어의 기본 개념인 보존원칙(conservative of packets)[1]이 등장하였다. 처음 고안된 TCP 혼잡제어 기능은 망이 평형 상태에 도달하고 머물러 있도록 하기 위한 두 알고리듬인 slow start와 congestion avoidance 그리고 손실된 패킷을 재전송에 의해 복구하기 위한 fast retransmit 알고리듬으로 구성되었는데, 이에 기초하여 구현된 TCP를 흔히 TCP Tahoe라고 한다[1]. 이후 네트워크의 대역폭이 증가함에 따라 손실된 패킷을 fast retransmit에 의해 재전송한 후 다시 slow start를 하지 않아도 충분한 동기 상태(clocked state)를 유지할 수 있다는 사실에 의거하여 fast recovery 알고리듬이 추가되었는데[2], 이의 구현을 흔히 TCP Reno라고 한다. TCP Reno는 최근까지 가장 널리 사용된 TCP로 이 후 개발된 대부분의 TCP들은 TCP Reno를 근간으로 하고 있다.

TCP Reno의 fast recovery 알고리듬은 동일한 윈도우에서 하나의 패킷이 손실되는 경우에 효율적으로 동작하도록 설계되었기 때문에, 동일한 윈도우에서 다수개의 패킷 손실이 발생하는 경우 성능이 크게 감소하는 단점을 가지고 있다[6]. 이 문제를 극복하기 위해서 제안된 TCP NewReno는 Reno의 fast recovery 알고리듬을 일부 수정되었으며[3], 그 결과 TCP NewReno의 fast recovery 알고리듬은 다수 개의 패킷 손실이 발생한 경우에도 매 라운드트립시간(round-trip time; RTT)마다 하나씩 손실된 패킷을 재전송할 수 있다. 이와 비슷한 배경에서 선택승인(selective acknowledgement)옵션[4,5]이 고안되었는데, 선택승인 옵션을 사용하는 TCP는 전송한 패킷들 가운데 손실된 패킷과 정상적으로 전송된 패킷에 대한 정확한 정보를 얻을 수 있다. 따라서 동일한 개수의 패킷 손실에 대해서, 이들을 복구하기까지 발생하는 지연이 TCP NewReno보다 훨씬 짧다는 장점을 가지고 있기 때문에[6] 가까운 미래에 널리 도입될 것으로 예상된다.

2.2 선택승인 옵션의 형식과 적용

그림 1에는 선택승인 옵션의 형식을 나타내었다. 선택승인 옵션을 사용하기 위해서는 송신원과 수신원이 모두 이의 사용을 지원해야 하고 선택승인 정보는 수신원이 전달하는 중복승인 패킷의 헤더의 40-byte 옵션 영역에 포함되어 송신원에게 전달된다. 16bit의 선택승인 허용(SACK-permitted) 옵션에는 현재 사용 중인 옵션의 종류와 길이에 대한 정보가 포함되고, 나머지 영역에는 현재 수신원에 정상적으로 수신된 패킷들에 대한 정보가 32bit의 TCP 순번(sequence number)에 의해서 표현된다. 정상적으로 수신된 패킷들의 집합을 블록이라고 하는데 40-byte인 TCP의 옵션 영역에는 최대 네 개의 블록에 대한 정보가 포함될 수 있다.[2]

1) 각 TCP 구현의 손실 복구 과정의 동작은 [6]에 자세히 비교되어 있다.

2) 이는 $8\times n+2\leq 40$에 의해 간단하게 계산될 수 있다.

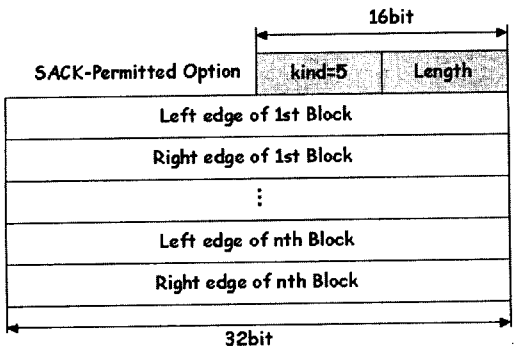

그림 1 선택승인 옵션의 형식

그림 2에는 선택승인 옵션을 사용하는 경우의 선택승인 정보가 형성되는 과정에 대한 예를 나타내었다. 송신원이 패킷 1에서 8까지 여덟 개의 패킷을 전송했을 때 패킷 1, 3, 그리고 7이 손실된 경우를 생각해보자. 손실되지 않은 다섯 개의 패킷들에 의해서 패킷 1에 대한 중복승인이 발생하게 되는데 이들 각각에는 현재 수신원이 정상적으로 수신한 패킷들에 대한 정보가 포함된다. 패킷 2가 수신되었을 때 수신원은 패킷 1을 수신하지 못한 상태이기 때문에 이를 버퍼에 저장하고, 패킷 2의 왼쪽 경계(left edge; LE)와 오른쪽 경계(right edge; RE)가 첫 번째 블록이라는 것을 알려준다. 이 후 패킷 4가 수신되면 패킷 3 역시 수신하지 못한 상태이기 때문에 두 개의 블록에 대한 정보를 전달한다. 이 때 새로 형성된 블록이 항상 첫 번째 블록이 된다. 패킷 5와 패킷 6이 수신되면 첫 번째 블록의 오른쪽 경계만 증가하게 되고 패킷 8이 수신되면 새로운 첫 번째 블록이 형성된다. 이 때 정상적으로 수신된 연속된 패킷들의 집합을 블록이라고 하고 각 블록들 사이의 손실된 패킷들을 홀(hole)이라고 한다. 따라서 패킷 8까지 모두 수신했을 때 수신원의 버퍼에는 그림 2와 같이 세 개의 블록이 형성되게 된다.

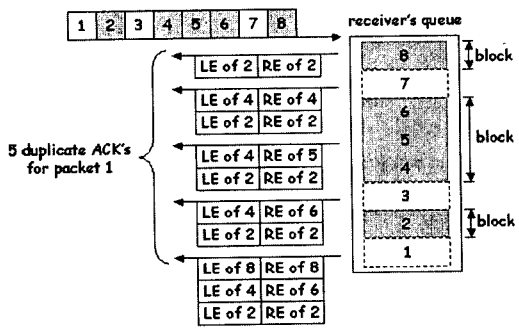

그림 2 선택승인 정보의 형성과 전달

2.3 선택 승인 옵션을 사용하는 TCP의 동작

선택승인은 TCP가 사용할 수 있는 여러 옵션들 가운데 하나이기 때문에 그 자체가 특정한 혼잡제어 알고리듬을 규명하고 있지는 않다. 따라서 본 논문에서는 선택승인 옵션을 사용하는 TCP로 가장 널리 알려진 "Sack1"[6]을 선택하였다.

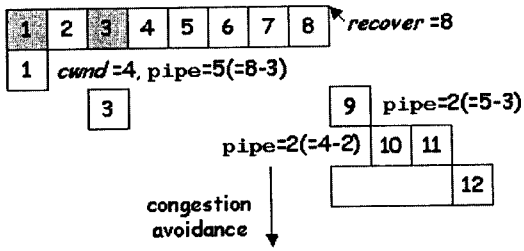

그림 3 두 개의 패킷 손실에 대한 선택 승인 옵션을 사용하는 TCP의 손실 복구 동작

Sack1은 현재 전송 진행 중인 패킷(outstanding packet)의 수를 측정하기 위해서 pipe라는 하나의 변수와 선택승인 정보에 따라서 정상적으로 전송된 패킷과 그렇지 않은 패킷에 대한 정보를 저장하기 위한 scoreboard라는 데이타 구조를 사용한다[6]. 그림 3에는 두 개의 패킷 손실에 대한 "Sack1"의 동작을 나타내었다. 패킷 5에 의해서 패킷 1에 대한 세 번째 중복승인을 수신했을 때 송신원은 이를 fast retransmit에 의해 재전송하고 pipe를 5(=8-3)로 설정하는데, 이는 세 개의 중복 승인이 수신되었다는 것은 전송 중이었던 8개의 패킷들 가운데 세 개의 패킷이 전송 완료되었다는 사실을 의미한다는 점을 반영한 것이다.[3] 이 후 fast recovery동안 중복승인이 수신될 때마다 pipe값은 1씩 감소하여 반으로 감소한 윈도우의 크기인 4(=8/2)보다 작아질 때마다 하나의 새로운 패킷을 전송할 수 있다.[4] 패킷 7에 의한 다섯 번째 중복승인을 수신했을 때 pipe값은 3으로 감소하기 때문에 하나의 패킷을 전송할 수 있는데, 이 시점에서 이미 송신원은 선택승인 정보에 의해서 패킷 3 역시 손실되었다는 사실을 알고 있기 때문에 패킷 3을 재전송하고 pipe의 값을 다시 4로 증가시킨다. 마지막 여섯 번째 중복승인을 수신했을 때 pipe는 다시 1 감소하고 더 이상 손실된 패킷이 없기 때문에 새로운 패킷

3) 이에 의해 현재 전송중인 패킷의 수를 다섯 개로 추정할 수 있다. 물론 이 경우에 패킷 3 역시 손실되어 실제 전송중인 패킷의 개수는 패킷 6, 7, 8과 재전송한 패킷 1, 네 개이지만 이는 재전송한 패킷 1에 의해서 패킷 3에 대한 부분승인(partial acknowledgement)를 수신한 후에 반영된다.

4) 이는 손실 복구 과정동안 전송중인 패킷의 수를 fast retransmit 당시 반으로 감소한 혼잡윈도우의 크기로 제한하기 위한 것이다.

9가 전송된다. 재전송한 패킷 1에 의해서 패킷 3에 대한 부분승인(partial acknowledgement)[3]을 수신함에 따라 pipe의 값은 2만큼 감소하기 때문에[5] 새로운 패킷 10과 11이 전송된다. 마지막으로 재전송한 패킷 3에 의해서 패킷 8까지 정상적으로 전송되었다는 승인이 전달되기 때문에 현재의 fast recovery과정은 종료되고 크기가 4인 혼잡윈도우로 congestion avoidance상태가 시작된다.

3. 제안하는 알고리듬

3.1 재전송 손실로 인한 문제점

선택승인 옵션을 사용함으로써 동일한 윈도우에서 발생한 다수 개의 패킷손실들을 타임아웃없이 재전송에 의해서 복구할 수 있는 장점이 있지만, 재전송한 패킷들 가운데 다시 손실이 발생할 경우 항상 타임아웃이 발생하는 문제점이 있다. 그림 4에는 윈도우의 크기가 8인 상태에서 재전송한 패킷이 다시 손실된 경우 선택승인 옵션을 사용하는 TCP의 손실 복구 과정의 동작을 나타내었다.

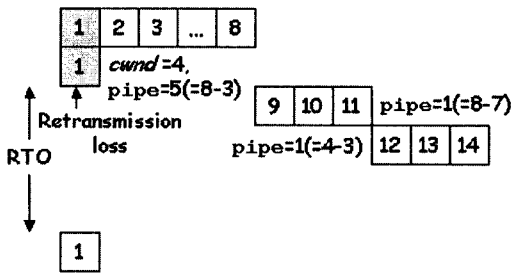

그림 4 재전송 손실에 의한 타임아웃의 발생

패킷 1에 대한 세 번째 중복승인을 수신했을 때, 송신원은 패킷 1을 fast retransmit에 의해 재전송하고 혼잡윈도우의 크기와 pipe의 값을 각각 4, 5로 설정한다. 이후 송신원은 네 개의 추가적인 중복승인을 수신하게 되므로 세 개의 새로운 패킷들인 9, 10, 그리고 11을 전송한다. 재전송한 패킷 1이 다시 손실되었으므로 송신원은 재전송한 패킷 1 이후에 전송한 패킷들에 대해서도 여전히 패킷 1에 대한 중복승인만을 수신하게 된다. 따라서 패킷 9, 10, 11에 의해서 발생하는 중복승인들에 의해서 pipe값은 3만큼 줄어들기 때문에 다시 새로운 패킷 12, 13, 그리고 14가 전송된다. 마찬가지로 이들에 의해서도 패킷 1에 대한 중복승인이 발생할 뿐, 패킷 1에 대한 타이머가 만기될 때까지 패킷 1이 전송되지 않는 한 정상적인 승인을 수신할 수 없기 때문에 결국 패킷 1은 재전송 타임아웃 후 다시 전송되게 된다. 이러한 현상은 윈도우의 크기, 혹은 손실된 패킷의 개수에 관계없이 재전송된 패킷이 손실되는 모든 경우에 발생한다.

3.2 제안하는 재전송 손실 감지 알고리듬

본 논문에서 제안하는 재전송 손실 감지 알고리듬[6]이 동작하기 위해서는 패킷 손실 당 하나의 변수가 필요하다. 이 변수에는 해당 패킷 손실을 재전송하는 시점에 이미 전송한 패킷들 가운데 가장 높은 순번을 가지는 패킷의 순번이 저장된다. 따라서 재전송한 패킷에 대해서 저장된 순번보다 높은 순번의 오른쪽 경계를 가지는 첫 번째 블록이 존재하는 경우 해당 재전송은 손실된 것으로 감지될 수 있다. 이해를 돕기 위해서 그림 4의 예에 대한 제안된 알고리듬을 사용하는 TCP의 동작을 그림 5에 나타내었다.

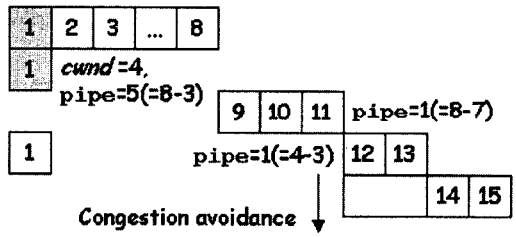

그림 5 재전송 손실의 감지 및 복구

패킷 1에 대한 세 번째 중복승인을 수신했을 때 송신원은 혼잡윈도우의 크기와 pipe외에 현재 전송중인 가장 높은 순번을 가지는 패킷의 순번인 8을 저장한다. 이후 fast recovery과정에서 중복승인이 수신됨에 따라서 새로운 패킷 9, 10, 11이 전송된다. 재전송한 패킷 1이 다시 손실되었으므로 패킷 9에 의해서도 패킷 1에 대한 중복승인이 발생하는데, 이 때 중복승인에 포함된 첫 번째 블록은 패킷 2부터 패킷 9까지라는 것을 알려주게 된다. 따라서 이는 저장된 값인 8보다 오른쪽 경계의 순번이 높다는 것을 의미하므로 재전송한 패킷 1은 다시 전송된다. 이 후 패킷 10과 11에 의해서 두 개의 중복승인이 수신되면, 원래의 동작과 마찬가지로 pipe의 감소로 두 개의 추가적인 패킷 12와 13이 전송된다. 두 번째 재전송한 패킷 1이 정상적으로 전송되면 패킷 11까지 정상적으로 전송되었다는 승인을 수신하므로 현재의 fast recovery과정은 종료되고 congestion avoidance 상태가 이어진다.

5) 이는 재전송한 하나의 패킷이 정상적으로 전송되었다는 것과 하나의 추가적인 패킷 손실이 발생했다는 것을 반영한다.

6) 선택승인 옵션을 사용하지 않는 TCP의 재전송 손실을 감지하고 복구하기 위한 중복승인 개수에 기반한 알고리듬이 [12]에 제안된 바 있다. 그러나 선택승인 옵션을 사용하는 경우 이 보다 훨씬 간단한 방법에 의해서 재전송 손실이 감지될 수 있다.

4. 시뮬레이션 및 결과 분석

4.1 시뮬레이션 환경

제안된 알고리듬의 성능을 평가하기 위해서 그림 6과 같은 구성을 가지는 두 개의 가상 네트워크에서 ns를 이용한 시뮬레이션을 수행하였다.

그림 6-(a)에 나타낸 네트워크는 각각 하나의 송신원과 수신원이 연결된 간단한 형태의 네트워크를 가정하고 있으며, 송신원과 수신원은 무선 링크를 통해서 연결된 것으로 설정하였다. 송신원과 수신원을 연결하는 무선 링크는 3세대 무선 네크워크 명세에서 명시하고 있는 600kbps의 대역폭과 100msec의 전달 지연을 가지는 것으로 설정하였고[13], 이 무선 링크 상에서 전송되는 패킷들이 4.2에서 설명할 두 가지 형태로 손실되는 것으로 가정한다.

제안된 알고리듬을 사용하는 TCP와 그렇지 않은 TCP간의 공평성을 비교하기 위해서 그림 6-(b)에 나타난 형태의 네트워크상에서 시뮬레이션을 수행하였다. 두 개의 송신원이 각각 두 개의 유선 단말(Fixed Host)에 구현되며, 두 개의 수신원이 각 이동 단말(Mobile Host)에 구현되는 것으로 설정하였다. 두 송신원은 하나의 라우터를 공유하며 이들은 70msec, 20Mbps의 유선 링크로 연결되어 있다. 라우터는 유한한 크기의 버퍼를 가지고 있으므로 라우터에서는 혼잡으로 인한 패킷 손실이 발생할 수 있다. 라우터와 기지국은 50msec, 10Mbps의 상대적으로 큰 대역폭과 짧은 전달 지연을 가지는 유선 링크로 연결되어 있고, 기지국과 두 개의 이동 단말은 6-(a)에서와 같은 100msec, 600kbps의 무선 링크로 연결되어 있다. 이 네트워크 모델에서 두 개의 송신원이 모두 기존 TCP를 사용하는 경우와 두 개의 송신원이 각각 기존 TCP와 제안된 TCP를 사용하는 경우에 대해서 각 송신원의 평균 전송율을 측정한 후, 세 개의 기존 TCP의 평균 전송율을 비교함으로써 제안된 TCP가 기존 TCP의 전송에 영향을 주는지 아닌지를 평가하는 것이 가능하다.

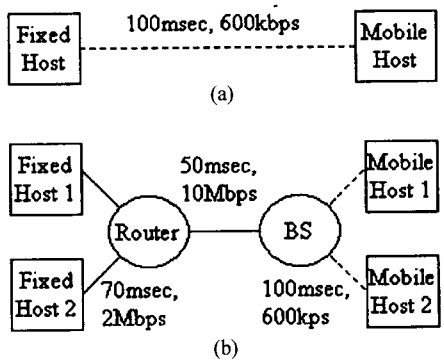

그림 6 시뮬레이션을 위한 네트워크의 구성

각 송신원은 크기가 1 kbyte인 패킷을 FTP(File Transfer Protocol)을 통해서 지속적으로 전송한다. 시뮬레이션동안 윈도우는 최대 32만큼 증가할 수 있는 것으로 설정하였으며, 보다 객관적인 성능 비교를 위해서 제한전송(Limited Transmit)[14]을 사용하도록 설정하였다. 수신원은 지연승인(delayed acknowledgement)방식을 사용하지 않으므로 정상적으로 패킷을 수신할 때마다 이에 적절한 승인을 전달한다.

4.2 패킷 손실 모델

무선 환경에서 발생하는 패킷 손실의 형태로 임의 독립적으로 발생하는 경우와 상호 연관성을 가지고 발생하는 경우를 고려하였다. 상호 연관성을 가지고 발생하는 패킷 손실을 모델링하기 위해서 2-state Markov chain을 도입하였다[15-17]. 채널은 좋은 상태(good state)와 나쁜 상태(bad state)를 가질 수 있고, 좋은 상태에서 나쁜 상태로 천이할 확률을 p, 나쁜 상태에서 좋은 상태로 천이할 확률을 q라고 했을 때 평균 패킷 손실 확률(π_E)은 $p/(p+q)$로 나타낼 수 있으며 나쁜 상태의 평균 지속시간은 $1/q$로 나타낼 수 있다.

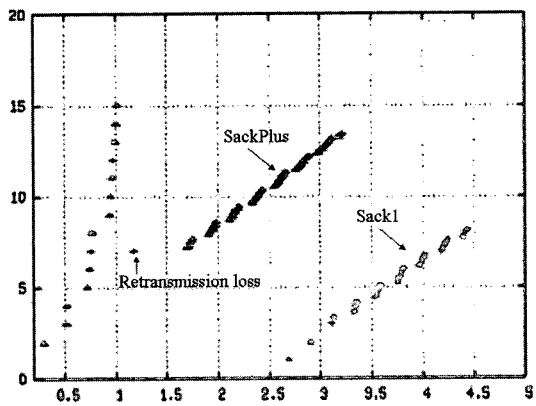

그림 7 하나의 재전송 손실에 대한 SackPlus와 Sack1의 윈도우 증가 비교(x축: 시간 y축: 혼잡윈도우의 크기)

4.3 시뮬레이션 결과

그림 7과 8에는 100개의 패킷을 전송하는 과정에서 하나의 재전송 손실이 발생한 경우에 대한 "SackPlus"와 "Sack1"의 윈도우의 크기 분포와 전송한 패킷들의 시간에 대한 숫자를 비교한 것이다. 재전송된 패킷을 손실시키기 위해서 특정한 순서의 패킷을 강제로 손실되도록 설정하였다[6].

약 1초경에 전송한 여덟 개의 패킷들 가운데 첫 번째

패킷이 손실되고 이에 대한 fast retransmit이 약 1.1초 경에 이루어졌다. 이의 재전송을 다시 손실되도록 하였으므로 Sack1은 이를 감지하지 못하고 약 2.6초경에 타임아웃이 발생하여 윈도우의 크기가 1인 상태에서 다시 slow start를 시작하는 것을 볼 수 있다. 반면, SackPlus는 재전송 손실을 감지할 수 있으므로 이의 두 번째 재전송 후에 타임아웃을 유발시키지 않고 congestion avoidance상태에서 전송을 계속한다. 100개의 패킷을 모두 전송하는 기간동안 평균 윈도우의 크기는 SackPlus의 경우 약 7.5로 4.7인 Sack1보다 상당히 큰 값을 갖는다.

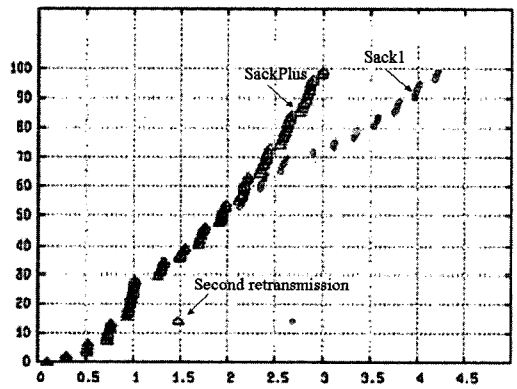

그림 8 하나의 재전송 손실에 대한 SackPlus와 Sack1의 전송된 패킷 순번 비교(x축: 시간 y축: 패킷 순번)

그림 8은 100개의 패킷을 전송하기까지 패킷 순번의 증가에 있어서 SackPlus와 Sack1을 비교한 것이다. 약 1.5초경 SackPlus는 재전송 패킷의 손실을 감지하고 이에 대한 재전송을 수행하는 것을 볼 수 있다. 100개의 패킷을 모두 전송하는 시점에 있어서 SackPlus의 경우 3.2초인데 반해 Sack1의 경우 4.3초로 약 1.1초 정도의 차이가 있고 이는 전송율 차원에서 약 38%정도 SackPlus가 빠른 결과로 이어지게 된다.

두 번째 실험으로 무선 링크에서 임의 독립적으로 발생하는 패킷 손실에 대한 두 TCP의 성능을 비교하였다. 10^6개의 패킷을 전송하는 동안 시뮬레이션이 진행되고 이 과정에서 타임아웃의 발생 횟수(N_{TO})와 재전송 횟수(N_R) 그리고 총 손실된 패킷(N_D)의 개수를 측정하였다. 패킷 손실 확률(p)이 5×10^{-3}에서 8×10^{-2}까지 변화할 때 각각의 결과를 표 1에 정리하였다. 패킷 손실 확률이 증가함에 따라서 N_D, N_R, 그리고 N_{TO} 모두 증가하는 것을 볼 수 있다. 모든 경우에 있어서 SackPlus

표 1 임의 패킷 손실에 대한 Sack1과 SackPlus의 타임아웃 발생 수, 패킷 손실 개수, 재전송 횟수의 비교

p	Sack1			SackPlus		
	N_{TO}	N_D	N_R	N_{TO}	N_D	N_R
0.5%	30	5050	5027	3	5050	5050
0.6%	46	6071	6023	1	6071	6071
0.7%	51	7069	7009	3	7069	7069
0.8%	68	8102	8023	5	8102	8026
0.9%	86	9113	9006	6	9113	9113
1.0%	95	10112	10004	5	10112	10112
2.0%	397	20321	19707	94	20321	20141
3.0%	1039	30660	28883	478	30660	29715
4.0%	2007	41299	37713	1251	41297	38920
5.0%	3470	52454	46366	2595	52451	47770
6.0%	5381	63745	54377	4449	63740	55902
7.0%	7912	75287	61829	6943	75281	63388
8.0%	10850	87132	68886	9863	87120	70442

표 2 상호연관성을 가지는 패킷 손실에 대한 Sack1과 SackPlus의 타임아웃 발생 수, 패킷 손실 개수, 재전송 횟수의 비교

π_E	Sack1			SackPlus		
	N_{TO}	N_D	N_R	N_{TO}	N_D	N_R
0.5%	15	2463	2306	13	2463	2312
0.6%	22	2929	.2714	18	2929	2726
0.7%	31	3448	3161	23	3448	3184
0.8%	51	3944	3533	41	3944	3561
0.9%	62	4472	3996	50	4472	4032
1.0%	88	4942	4454	73	4942	4497
2.0%	411	10066	7998	350	10066	8207
3.0%	1015	14797	10725	907	14797	10978
4.0%	2139	20544	13441	1953	20540	13818
5.0%	3481	26006	15425	3251	26006	15853
6.0%	4477	29846	17002	4218	29845	17469
7.0%	6960	37626	19417	6628	37618	19989
8.0%	9183	43384	20763	8814	43383	21360
9.0%	11699	49703	22188	11313	49703	22844

는 Sack1보다 낮은 N_{TO}와 높은 N_R을 갖는 것을 볼 수 있는데 이는 재전송 손실의 복구 여부를 반영하는 것이 b다. 같은 패킷 손실 확률에 대해서 SackPlus는 항상 적은 횟수의 타임아웃이 발생하기 때문에 그만큼 효율적인 전송을 할 수 있다고 할 수 있다.

상호 연관성을 가지는 패킷 손실에 대해서도 같은 실험을 반복하였다. 표 2에는 패킷 손실 확률(π_E)에 대한 두 TCP의 손실된 패킷의 수, 타임아웃 발생 수, 그리고 재전송에 의해 복구된 패킷의 수를 나타내었다. 앞의 표 1의 임의 패킷 손실에 대한 결과와 거의 비슷한 형태의 결과를 보임을 알 수 있다.

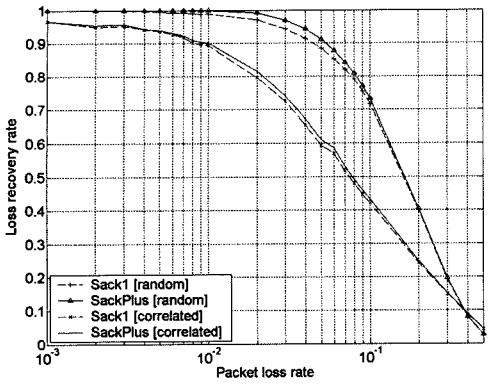

그림 9 패킷 손실 확률에 대한 Sack1과 SackPlus의 손실 복구 확률 비교

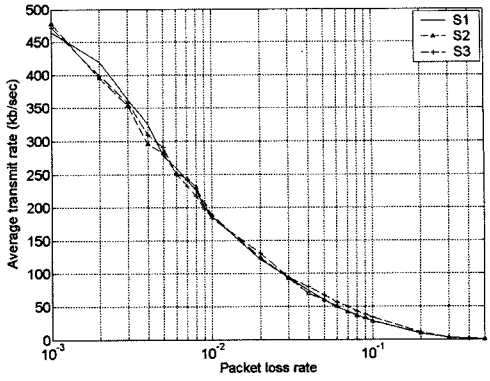

그림 10 SackPlus와 경합하는 경우와 그렇지 않은 경우의 Sack1의 평균 전송율 비교

이미 TCP의 성능은 손실 복구 확률에 비례한다는 것이 밝혀졌기 때문에[19,20], 그림 9에는 두 가지 형태의 패킷 손실에 대한 SackPlus와 Sack1의 성능을 손실 복구 확률측면에서 비교하였다. 손실 복구 확률은 손실된 패킷이 타임아웃이 아닌 재전송에 의해서만 복구될 확률을 의미하며, 총 손실된 개수에 대해서 재전송에 의해 복구한 패킷의 수의 비, 즉 N_R/N_D로 정의된다.

임의 패킷 손실인 경우 10^{-2}보다 작은 낮은 패킷 손실 확률에 대해서는 두 TCP의 손실 복구 확률이 매우 높은 값을 유지하며 거의 차이가 나지 않는 것을 볼 수 있다. 패킷 손실 확률이 증가함에 따라서 손실 복구 확률이 점차 감소하기 시작하는데, 이는 잦은 fast retransmit으로 인해서 윈도우의 크기가 감소하기 때문이다. 이 과정에서 재전송된 패킷들 가운데 다시 손실되는 경우도 증가하기 시작하므로 두 TCP의 손실 복구 확률의 차이도 증가하는 것을 볼 수 있다. 제안된 알고리듬은 거의 모든 손실된 재전송을 복구할 수 있음에도 불구하고 Sack1에 비해서 손실 복구 확률의 증가가 미미한 것을 볼 수 있는데 이는 재전송 손실의 발생 확률이 높지 않기 때문이다. 패킷 손실 확률이 매우 큰 값을 가지게 되면 대부분의 패킷 손실들은 fast retransmit에 의해 복구되기보다는 타임아웃이 발생하기 때문에 재전송 손실의 기회는 더욱 낮아지고 이로 인한 두 TCP의 똑같이 매우 낮은 손실 복구 확률을 보여준다.

패킷 손실이 상호 연관성을 가지고 연속적으로 발생하는 경우, 임의 패킷 손실인 경우에 비해 TCP의 손실 복구 확률은 크게 저하되는 것을 볼 수 있다. 패킷 손실이 연속적으로 발생하는 경우 한 윈도우의 모든 패킷이 손실되는 경우가 발생할 수 있는데, 특히 윈도우의 크기가 작을 때에는 이런 경우의 발생 빈도가 매우 높다. 따라서 제한 전송을 사용한다고 하더라도 세 번째 중복승인을 수신하기 전에 이미 타임아웃이 만료되어 손실된 패킷들이 재전송에 의해 복구될 확률이 매우 낮아지게 된다. 결과적으로 재전송 확률이 낮다는 것은 재전송되는 패킷들의 수 역시 작다는 것을 의미하기 때문에 SackPlus의 성능 향상 역시 임의 패킷 손실인 경우보다 낮아지는 것을 볼 수 있다.

마지막으로, SackPlus가 기존의 TCP와 대역폭을 경쟁할 때 서로 공평하게 동작하는지의 여부(fairness)를 알아보기 위한 시뮬레이션을 수행하였다. 이를 위해서 그림 6-(b)의 네트워크에서 두 개의 연결이 모두 Sack1을 사용하는 경우의 두 송신원(S_1, S_2)의 평균 전송율과 각각의 연결이 Sack1과 Sackplus를 사용하는 경우의 Sack1을 사용하는 송신원(S_3)의 평균 전송율을 측정하였다. 그림 10에 나타난 세 TCP 연결의 평균 전송율을 통해서 SackPlus가 기존의 Sack1을 사용하는 연결의 전송율에 아무런 영향을 미치지 않음을 알 수 있다. 이는 SackPlus가 TCP 윈도우 제어 방식인 부가적 증가 지수적 감소(Additive Increase Multiplecative Decrease; AIMD)를 그대로 준수한다는 점을 생각했을 때 당연한 결과라고 할 수 있다. 그리고 SackPlus의 재전송 손실의 감지는 재전송된 패킷보다 나중에 전송된 새로운 패킷들에 의해서 감지되므로 만약 망의 혼잡 상태가 극심하여 혹은 무선 링크의 전송 환경이 계속 나빠서 더 이상 패킷 전송을 하지 않아야 하는 경우에 손실된 재전송뿐만 아니라 이 후에 전송된 패킷들 역시 모두 손실될 확률이 매우 높다. 따라서 이 같은 경우에는 재전송 손실 감지가 동작하지 않으므로 추가적인 패킷 손실이나 망의 혼잡을 유발하지 않는다는 점에서 기존 TCP의 보존 원칙을 명백하게 준수한다고 할 수 있다.

5. 결 론

본 논문에서는 재전송 손실로 인한 TCP의 성능 저하를 방지하기 위한 알고리듬을 제안하고 이의 성능 향상을 시뮬레이션을 통해서 입증하였다. 제안된 알고리듬은 간단하게 구현될 수 있고 기존의 TCP의 동작 원칙을 그대로 준수한다는 장점을 가지고 있다. 제안된 알고리듬은 거의 모든 재전송 손실을 감지하고 복구할 수 있음에도 불구하고 평균화된 시뮬레이션 결과로 나타난 제안된 알고리듬의 성능 향상 정도는 그리 크지 않다. 이는 재전송 손실의 발생 확률 자체가 그리 크지 않기 때문이다. 그러나 제안된 방법을 통해 TCP 혼잡제어 알고리듬 자체의 결함으로 인한 불필요한 TCP의 발생 원인들 가운데 지금까지 유일하게 해결되지 못한 원인을 매우 간단한 구현을 통해서 해결할 수 있다는데 그 의의가 있다.

참 고 문 헌

[1] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM'88*, 1988.

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

[3] Floyd, S. and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 2582, Apr. 1999.

[4] S. Floyd "TCP Selective Acknowledgment options," RFC 2018, Oct. 1996.

[5] S. Floyd, "An Extension to the Selective Acknowledgement (SACK) option for TCP," RFC 2883, Jul. 2000.

[6] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP," *ACM Comp. Comm. Rev.*, 1996.

[7] Dong Lin and H. T. Kung, "TCP fast Recovery Strategies : Analysis and Improvement," *IEEE INFOCOM'98*, pp. 263-271, 1998.

[8] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms of Improing TCP Performance over Wireless Links," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, 1997.

[9] C. Barakat, E. Altman, and W. Dabbous, "On TCP Performance in a Heterogeneous Network: A Survey," *IEEE Comm. Magazine*, 2000.

[10] 박원서, 김범준, 김내수, 최동준, 이재용, "위성링크에서 혼잡한 전송에러에 의한 패킷 손실 구분을 통한 TCP 성능개선 방안", *Telecomm. Rev.*, 제10권 6호, 2000.

[11] J. Postel, "Transmission Control Protocol," RFC 793, Apr. 1999.

[12] Beomjoon Kim and Jaiyong Lee, "Retransmission Loss Recovery by Duplicate Acknowledgement Counting," *IEEE Comm. Let.*, vol 8. no. 1, pp. 69-71, 2004.

[13] H. Inamura "TCP over 2.5/3G wireless networks," RFC 3481, Feb. 2003.

[14] M. Allman, H. Balakrishnan, and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit," RFC 3042, 2001.

[15] A. Lahanas, V. Tsaoussidis, "Improving TCP performance over networks with wireless components using "probing devices"," *IEEE WCNC'2002*, Mar. 2002.

[16] 김동민, 김범준, 박건영, 이재용 외, "Wireless link에 적합한 TCP 개발", 삼성-연세 산학 협동 과제 최종보고서, 2002

[17] A. Chockalingam, M. Zorzi, and R. R. Rao, "Performance of TCP on wireless fading links with memory," *IEEE ICC'98*, 1998.

[18] Anurag Kumar, "Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Links," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, 1998.

[19] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Trans.* Networking, vol. 8, no. 2, 2000.

[20] W. R. Stevens, TCP/IP Illustrated, vol. 1. Addison-Wesley, Nov. 1994.

박 건 영

1991년 3월~1995년 2월 해군사관학교 전자공학과 학사. 2001년 3월~2003년 2월 연세대학교 전기전자공학과 석사 2003년 3월~현재 해군 작전사령부

김 범 준

1992년 3월~1996년 2월 연세대학교 전자공학과 학사. 1996년 9월~1998년 8월 연세대학교 전기전자공학과 석사. 1998년 9월~2003년 8월 연세대학교 전기전자공학과 박사. 2003년 9월~2004년 1월 연세대학교 IT사업단 내 Center for IT of Yonsei University(CITY) 박사후과정 연구원. 2004년 1월~현재 LG전자 이동통신기술연구소 표준화그룹 선임연구원

김 동 민

1999년 2월 건국대학교 전자공학과 학사 2001년 8월 연세대학교 전자 공학과 석사. 2001년 9월~현재 연세대학교 전자공학과 박사과정. 관심분야는 TCP 모델링 및 성능 분석, 무선 TCP, 무선 랜, IPv6, Mobile IP

한 제 찬

1998년 3월~2002년 8월 연세대학교 전기전자공학과 학사. 2002년 9월~2004년 8월 연세대학교 전기전자공학과 석사 2004년 9월~현재 연세대학교 전기전자공학과 박사과정

이 재 용

1997년 2월 연세대학교 전자공학과 학사 1984년 5월 IOWA State University 공학 석사. 1987년 5월 IOWA State University 공학 박사. 1987년 6월~1994년 8월 포항 공과대학 교수. 1994년 9월~현재 연세대학교 전자공학과 교수 관심분야는 Protocol Design for QoS Management, Network Management, High Speed Networks