재전송 손실 복구를 통한 TCP SACK의 성능 향상 모델링 및 분석

정회원 김 범 준*, 김 동 민**, 이 재 용**

Improving Loss Recovery Performance of TCP SACK by Retransmission Loss Recovery

Beomjoon Kim*, Dongmin Kim**, and Jaiyong Lee** *Regular Members*

요 약

TCP(transmission control protocol)의 성능은 손실 복구 과정의 성능에 크게 좌우되는데, 특히 패킷 손실이 발생했을 때 이를 RTO(retransmission timeout)을 유발하지 않고 재전송에 의해서 복구가 가능한가의 여부는 매우 중요한 문제라고 할 수 있다. TCP SACK(selective acknowledgement)은 다수 개의 패킷 손실이 발생하더라도 재전송에 의해서 효율적으로 복구할 수 있는 장점을 가지고 있지만, 재 전송한 패킷이 다시 손실되는 경우에는 언제나 RTO를 유발시키는 문제점이 있다. 본 논문에서는 이 문제를 해결하기 위한 알고리듬을 제안한다. 제안된 알고리듬을 사용하는 TCP SACK+는 기존의 TCP와의 호환성을 완벽하게 유지하는 동시에 재전송 패킷 손실을 감지할 수 있는 장점을 가지고 있다. TCP SACK+의 성능을 평가하기 위해서 모델링을 이용한 확률적 분석과 시뮬레이션을 도입한다. 결과를 통해서 TCP SACK+는 거의 모든 재전송 손실을 복구할 수 있기 때문에 TCP SACK보다 손실 복구 성능 차원에서 상당히 성능을 향상시킬 수 있음을 알 수 있다.

키워드: TCP 손실 복구, SACK 옵션, TCP SACK+, TCP 모델 분석 및 평가, 재전송 손실 및 복구,

ABSTRACT

The performance of transmission control protocol (TCP) is largely dependent upon its loss recovery. Therefore, it is a very important issue whether the packet losses may be recovered without retransmission timeout (RTO) or not. Although TCP SACK can recover multiple packet losses in a window, it cannot avoid RTO if a retransmitted packet is lost again. In order to alleviate this problem, we propose a simple change to TCP SACK, which is called TCP SACK+ in simple. We use a stochastic model to evaluate the performance of TCP SACK+, and compare it with TCP SACK. Numerical results evaluated by simulations show that SACK+ can improve the loss recovery of TCP SACK significantly in presence of random losses.

Ⅰ. 서 론

TCP (transmission control protocol)의 초기 설계 사항 (specification)이 발표된 후[1], 성능을 향상시키기 위한 많은 노력들이 계속되어 왔다. 특히, 혼잡 제어(congestion control)측면에서 많은 발전이 이루어져 왔고, 현재는 손실된 패킷을 빠른 시간 안에 재전송하기 위한 fast retransmit 알고리듬이 구현된 TCP Tahoe[2]와 송신원과 수신원사이의 넓어진 대역폭을 효율적으로 이용하기 위한 fast

* LG전자 이동통신기술연구소 표준화그룹 (beom@lge.com),
** 연세대학교 전기전자공학과 네크워크 연구실 (jyl@nasla.yonsei.ac.kr)
※본 연구는 한국과학재단 특정기초연구(R01-2002-000-00531-0)지원으로 수행되었음.
논문번호 : 030366-0820, 접수일자 : 2003년 8월 20일

recovery 알고리듬이 추가로 구현된 TCP Reno[3]가 널리 사용되고 있다. 근래에 들어 하나의 윈도우 내에서 다수 개의 패킷 손실이 발생하는 경우 성능이 크게 저하되는 TCP Reno의 문제점을 개선하기 위한 TCP NewReno[3]와 SACK(selective acknowledgement)[4] 옵션에 대한 표준 명세가 완료 단계에 접어들고 있고, 곧 기존의 TCP Tahoe와 TCP Reno를 대신해서 널리 사용될 것으로 기대되고 있다. 같은 개수의 패킷 손실들에 대해서 한 RTT(round-trip time)동안 한 개의 패킷 손실만을 복구할 수 있는 TCP NewReno에 비해, SACK 옵션을 사용하는 경우 송신원은 손실된 패킷의 순번(sequence number)에 대한 정확한 정보를 알 수 있기 때문에 대부분의 경우 한 RTT동안 손실된 모든 패킷들을 재 전송할 수 있다는 장점을 가지고 있다.

그러나, SACK 옵션을 사용하더라도 재 전송한 패킷이 다시 손실되는 경우에는 이를 감지하지 못하고 RTO(retransmission timeout)가 발생한다는 문제점이 있다[7]. 한 연구[8]에 따르면, 전체 RTO의 약 4%가 이 같은 재전송 패킷 손실에 의해 발생한다. 본 논문에서는 이 문제점을 개선하기 위한 간단한 알고리듬을 제안하고 이를 사용하는 TCP 구현을 간단히 TCP SACK+라고 명명한다. SACK+의 송신원은 재 전송한 패킷들 가운데 발생한 손실을 정확하게 감지할 수 있으므로 이를 RTO없이 복구할 수 있다. 제안된 알고리듬은 기존에 SACK 옵션에서 사용하는 정보만을 사용하기 때문에 구현이 매우 간단한 동시에 기존의 모든 TCP들과도 완벽하게 호환될 수 있다는 장점을 가지고 있다. TCP SACK+의 성능을 분석하고 평가하기 위해서 모델링을 통한 확률적 분석과 시뮬레이션을 수행하였고, 이 결과들을 통해 TCP SACK+는 TCP SACK의 성능을 손실 복구 차원에서 상당히 향상시킬 수 있는 것을 알 수 있다.

Ⅱ. TCP SACK+의 동작

TCP SACK 옵션을 사용하는 TCP의 구현으로서 "pipe"라는 변수와 "*scoreboard*"라는 데이터 구조를 통해서 동작하는 "Sack1"[7]을 기준으로 고려하였다. 시뮬레이션을 통해서 본 Sack1의 동작은 참고문헌[7]에 상세히 설명되어 있다.

SACK+는 손실 복구 과정 동안의 TCP 송신원의 동작을 간단히 수정함으로써 구현될 수 있는데, 이의 핵심적인 동작 원리는 다음과 같이 설명될 수 있다. 손실된 패킷 손실에 대한 재전송이 성공적이라면, 이에 대해서 발생하는 중복 승인(duplicate acknowledgement)들에 포함된 SACK 정보의 첫 번째 블록의 오른쪽 경계는 재전송을 수행하기 전에 전송한 패킷들의 가장 높은 순번(highest sequence number)보다 큰 값을 가질 수 없다. 즉, 재전송이전에 전송했던 패킷들의 최대 패킷 순번보다 큰 오른쪽 경계를 가지는 블록이 존재한다는 것은 수신원이 재전송 후에 전송한 패킷을 수신할 때까지 손실된 패킷에 대한 재전송을 수신하지 못했다는 것을 의미하기 때문이다. 따라서, SACK+의 송신원은 패킷을 재 전송할 때 이미 전송한 패킷의 최대 순번 값을 저장하고, 이 후에 재 전송한 패킷에 대한 중복 승인이 수신될 때마다 이에 포함된 SACK 정보의 첫 번째 블록의 오른쪽 경계와 저장된 값을 비교한다. 만약 저장된 값보다 오른쪽 경계의 값이 크다면 송신원은 재전송 패킷 손실이 발생했다는 사실을 감지하고 이를 즉시 재 전송한다. 결과적으로 손실된 패킷의 재전송 후에 적어도 하나의 새로운 패킷이 전송되는 경우 이 패킷에 의해서 재전송 패킷 손실 여부를 감지하는 것이 가능하다. 다음 그림에 나타난 예를 통해서 SACK+의 동작을 쉽게 이해할 수 있다.

그림 1은 윈도우에서 하나의 패킷 손실이 발생하고 그 재전송이 다시 손실된 경우의 SACK+의 동작을 보여준다. 송신원은 약 0.75초경에 여덟 개의 패킷들 7~14를 전송하는데, 이들 중 패킷 7이 손실된 경우를 생각해보자. 패킷 7에 대한 세 개의 중복 승인을 수신됨에 따라, 약 1초경에 송신원은 fast retransmit에 의해 패킷 7을 재전송하고 현재 전송 완료된 패킷들 중 가장 높은 패킷 순번인 14를 저장한다. 이 후에 수신되는 네 개의 중복 승인들에 의해 pipe 값은 1(=8-7)로 감소하므로 세 개의 새로운 패킷들 15~17이 추가로 전송된다. 패킷 7의 재전송이 다시 손실되었으므로 1.25초경에 패킷 15~17의 전송의 결과로 패킷 7에 대한 세 개의 중복 승인을 추가로 수신하게 된다. 세 개의 중복 승인들 중 첫 번째 중복 승인의 첫 번째 블록의 오른쪽 경계는 15이다. 즉, 패킷 7을 재 전송했음에도 불구하고 오른쪽 경계가 이전에 전송한 패킷들 중 가장 나중에 전송된 패킷 14보다 큰 패킷 15를 포함하는 블록이 존재한다는 것은 패킷 7의 재전송이 다시 손실되었다는 것을 의미한다. 따라서 송신원은 이를 확인한 즉시 패킷 7을 다시 재 전송한다. 이 후에 수신되는 두 개의 중복 승인은 pipe값을 두 개의

패킷만큼 감소시키므로 두 개의 새로운 패킷들 18~19가 전송된다.

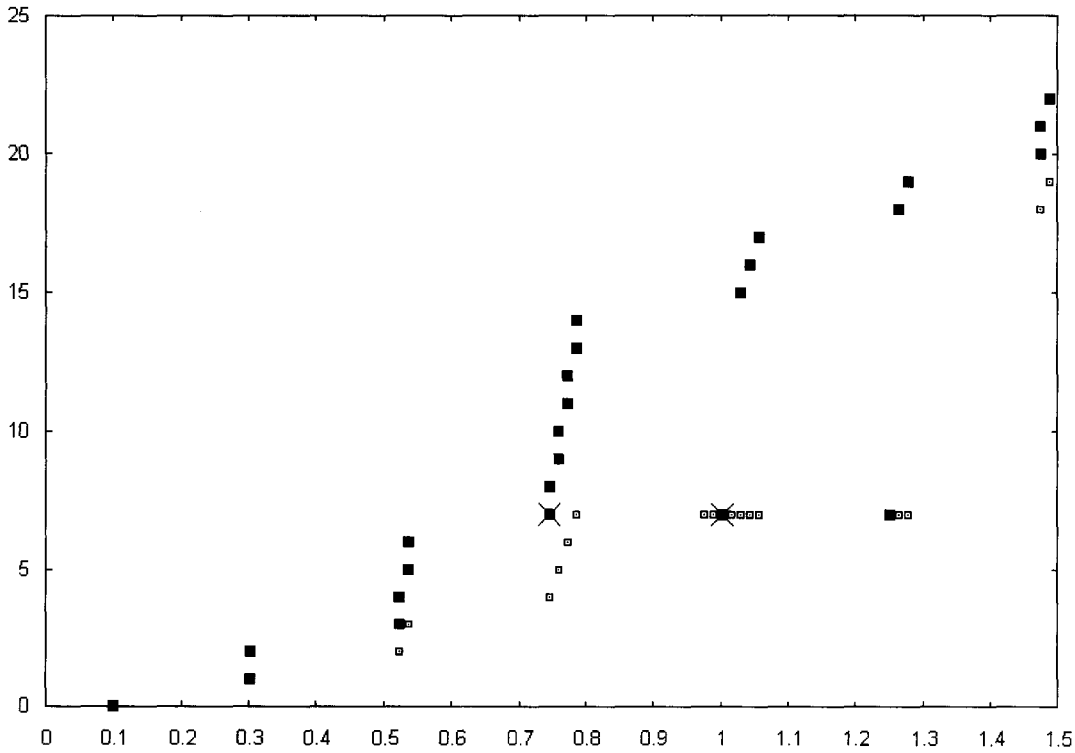


그림 1. 하나의 재전송 손실에 대한 TCP SACK+의 동작

Ⅲ. TCP SACK+의 손실 복구 과정 모델링

TCP SACK+의 손실 복구 과정의 모델링은 '라운드' 단위로 이루어지고[10], 윈도우의 정상 상태 분포(steady-state distribution)를 얻기 위해서 Markov 프로세스를 이용한 과정[9]을 도입한다. 이를 통해 TCP의 손실 복구 과정을 유도하고 윈도우의 크기에 대해서 일반화하는 것이 가능하다.

1. 가정 및 정의

TCP의 손실 복구 과정을 모델링하기 위해서 몇 가지 가정과 정의가 필요하다. 우선 각 패킷은 임의의 (random) 확률 p로 손실되고 패킷 손실들은 서로 독립적으로 발생하는 것으로 가정한다. 송신원은 전송해야 할 패킷들을 충분히 가지고 있어서 항상 윈도우가 허용하는 최대 개수의 패킷들을 전송하고 수신원은 지연 승인(delayed acknowledgement)[11]을 사용하지 않으므로 하나의 패킷을 수신할 때마다 하나의 승인 패킷을 송신원에 전달하는 것으로 가정한다.

2. TCP SACK의 손실 복구 과정 모델링

$\Omega = u$를 만족하는 손실윈도우에서 n개의 패킷 손실이 발생한 경우, l_1에 대해서 Φ_0개의 중복 승인을 수신하게 된다. 세 개의 중복 승인을 수신하여 l_1을 fast retransmit에 의해 다시 전송한 후, 모든 중복 승인을 수신했을 때의 pipe값은 $n(=u-\Phi_0)$과 같다. 만약 n이 반으로 줄어든 $cwnd\,(=\lfloor u/2 \rfloor)$값보다 크거나 같다면 ($\lfloor a \rfloor$는 a의 정수 부분을 의미한다.), 새로운 패킷은 전송될 수 없으므로 $\Phi_1 = 0$이다. 첫 번째 손실된 패킷의 재전송에 의해서 두 번째 손실된 패킷에 대한 부분 승인 (partial acknowledgement)을 수신하면 pipe 값은 두 개의 패킷만큼 감소하므로 $n-2$가 된다. 만약 $n-2 \geq \lfloor u/2 \rfloor$라면 송신원은 새로운 패킷이나 재전송 패킷에 관계없이 더 이상의 패킷 전송을 진행할 수 없기 때문에 이 후의 손실된 패킷들

은 RTO후에 전송되게 된다. 결과적으로 하나의 손실 윈도우 내에서 발생한 패킷 손실의 수가

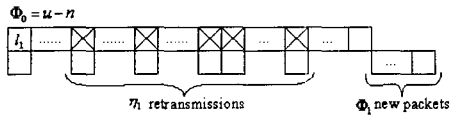


그림 2. 손실 복구 과정의 첫 번째 라운드에서 n 개의 재전송이 모두 전송되는 경우의 TCP SACK의 동작

$[u/2]+2$ 이상인 경우 TCP SACK은 RTO를 유발하게 되므로, 이를 고려한 TCP SACK의 손실 복구 확률 R_{SACK} 은 l_1을 fast retransmit에 의해 재전송하기 위해 송신원이 수신해야 할 중복 승인의 개수를 K, $m = min(u-K, [u/2]+1)$이라고 했을 때 다음과 같이 주어진다.

$$R_{SACK} = \sum_{n=1}^{m} \binom{u-1}{n-1} p^{n-1}(1-p)^{u-n}(1-p)$$
$$= \sum_{n=1}^{m} \binom{u-1}{n-1} p^{n-1}(1-p)^{u} \quad (1)$$

식 (1)의 $(1-p)^n$은 n개의 재전송 패킷은 반드시 손실되지 않아야 한다는 것을 반영한 것이다.

3. TCP SACK+

TCP SACK+의 손실 복구 과정의 모델링에 앞서, n 개의 재전송들 가운데 단지 하나의 재전송 패킷 손실만이 TCP SACK+에 의해 복구될 수 있고 같은 재전송에 대한 복구 기회는 단 한번만 주어지는 것으로 가정한다. 따라서 TCP SACK+의 손실 복구 확률 R_{SACK+} 는 다음의 식으로 주어진다.

$$R_{SACK+} = R_{SACK} + p \cdot \Delta \quad (2)$$

식 (2)에서 Δ는 손실된 하나의 재전송이 SACK+에 의해서 복구될 확률을 의미하므로 $p \cdot \Delta$은 재전송들 가운데 하나가 손실되고 다시 재전송에 의해 복구될 확률을 의미한다. TCP SACK의 손실 복구 과정은 손실된 패킷의 개수뿐만 아니라 손실된 패킷의 손실 윈도우 내에서의 위치에 따라 다양한 동작으로 이루어질 수 있다. 본 논문에서는 모델링의 단순화를 위해서 손실된 n개의 패킷에 대한 재전송이 첫 번째 라운드에서 완료되는 경우와 두 번째 라운드에서 완료되는 두 가지 경우만을 고려한다. (실제로 이는 거의 모든 경우를 포함한다.)

그림 2는 손실 복구 과정의 첫 번째 라운드에서, 즉 송신원이 fast retransmit에 의해 재 전송한 l_1에 의한 부분 승인을 수신하기 전에, n 개의 패킷 손실이 전부 재 전송되는 경우의 TCP SACK의 동작을 보여준다. 손실 복구 과정의 k 번째 라운드에서 pipe값의 감소에 의해서 전송되는 재전송 패킷의 수를 η_k 라고 했을 때 다음과 같은 관계가 성립한다.

$$n = \eta_1 + 1 \quad (3)$$

l_1 에 대한 Φ_0 개의 중복 승인을 모두 수신했을 때 $cwnd$와 pipe의 값은 각각 $[u/2]$, n과 같으므로 첫 번째 라운드에서 전송되는 총 패킷의 수는 재전송 패킷과 새롭게 전송되는 패킷을 포함해서 다음과 같이 주어진다.

$$\eta_1 + \Phi_1 = [u/2] - n \quad (4)$$

pipe가 허용하는 한 항상 손실된 패킷에 대한 재전송이 새로운 패킷들의 전송에 대해 우선권을 가지기 때문에, 거의 모든 경우 η_1개의 재전송이 전송된 후에 Φ_1개의 패킷들이 전송되게 된다. (이 역시 패킷 손실의 손실 윈도우내의 위치와 관련된 블록의 형성에 따라서 달라질 수 있지만, 언급한 바와 같이 거의 모든 경우에 η_1개의 재전송들이 먼저 전송된다.) 따라서 $\Phi_1 \geq 1$인 경우에 η_1개의 재전송들 가운데 하나의 재전송 손실은 감지될 수 있다. 이를 위한 조건은 식 (3)과 (4)에 의해 다음의 식으로 주어진다.

$$1 \leq n \leq [u/4] \quad (5)$$

식 (5)로 주어진 조건을 만족하더라도 손실 윈도우가 포함하는 마지막 패킷이 손실된 경우에는 n개의 재전송이 첫 번째 라운드에서 종료될 수 없기 때문에 이 경우는 고려의 대상에서 제외한다. 따라서 첫 번째 라운드에서 전송한 n개의 재전송들 가운데 발생한 하나의 손실이 TCP SACK+에 의해 복구될 확률 Δ_1은 다음의 식으로 주어진다.

$$p \cdot \Delta_1 = p \sum_{n=1}^{[u/4]} \binom{u-2}{n-1} n \cdot p^{n-1}(1-p)^{u+\Phi_1} \quad (6)$$

식 (6)은 다음의 두 가지 사실을 반영한다.

i) 손실 윈도우의 마지막 패킷을 제외한 $(u-2)$개의 패킷들 가운데 $(n-1)$개의 패킷 손실이 발생한다.

ii) n개의 재전송들 가운데 한 개만 손실되고 이의 SACK+에 의한 두 번째 재전송과 Φ_1개의 새

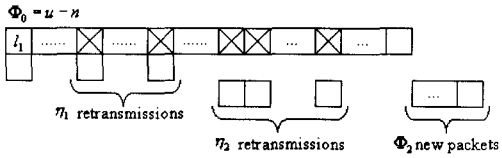


그림 3. 손실 복구 과정의 두 번째 라운드에서 n 개의 재전송이 모두 전송되는 경우의 TCP SACK의 동작

로운 패킷들은 손실되지 않아야 한다.

만약 $n \geq [u/4]+1$이라면 그림 3에서 볼 수 있는 바와 같이 손실 복구 과정은 두 번째 라운드까지 계속된다. 두 번째 라운드에서 n개의 패킷 손실에 대한 재전송이 완료된다면 다음의 관계가 성립한다.

$$n = \eta_1 + \eta_2 + 1 \tag{7}$$

재전송은 항상 새로운 패킷에 앞서 전송된다는 사실에 미루어 첫 번째 라운드에서 전송된 모든 패킷은 재전송 패킷이라 할 수 있으므로, 식 (4)에서 $\Phi_1 = 0$가 항상 성립하기 때문에, 이 경우 $\eta_1 = [u/2]-n$이다. 따라서, 두 번째 라운드에서 송신원은 모두 (η_1+1)개의 부분 승인을 수신하게 되는데, 부분 승인은 pipe값을 두 개의 패킷 크기만큼 감소시키는 것을 고려하면 두 번째 라운드에서 전송할 수 있는 총 패킷의 수는 다음과 같이 주어진다.

$$\eta_2 + \Phi_2 = 2([u/2]-n+1) \tag{8}$$

이 경우 역시 $\Phi_2 \geq 1$라면 η_2개의 재전송 가운데 발생한 하나의 손실을 복구하는 것이 가능하므로, 이를 위한 조건은 식 (7)과 (8)에 의해 다음과 같이 주어진다.

$$n \leq 0.75[u/2]+0.5 \tag{9}$$

이를 고려한 TCP SACK+의 두 번째 라운드에서의 재전송 손실 복구 확률 Δ_2는 n이 $[u/4]+1 \leq n \leq 0.75[u/2]+0.5$를 만족할 때 다음과 같이 주어진다.

$$p \cdot \Delta_2 = p\sum_n \binom{u-2}{n-1}\eta_2 \cdot p^{n-1}(1-p)^{u+\Phi_2} \tag{10}$$

식 (10)은 다음과 같은 세 가지 사실을 반영한다.

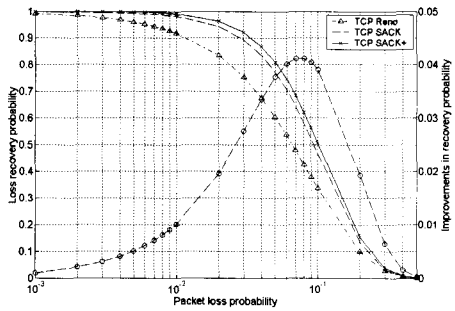


그림 4. TCP SACK과 TCP SACK+의 손실 복구 확률의 비교 ($W_{\max} = 8$)

i) 손실 윈도우의 마지막 패킷을 제외한 $(u-2)$개의 패킷들 가운데 $(n-1)$개의 패킷 손실이 발생한다.

ii) 첫 번째 라운드에서 전송되는 (η_1+1)개의 재전송들은 손실되지 않아야 한다.

iii) η_2개의 재전송들 가운데 한 개만 손실되고 이의 SACK+에 의한 두 번째 재전송과 Φ_2개의 새로운 패킷들은 손실되지 않아야 한다.

Ⅳ 수학적 분석 결과

이번 장에서는 TCP SACK과 TCP SACK+의 윈도우의 정상 상태 분포에 대해서 일반화된 (normalized) 손실 복구 확률을 계산한다. 각 그래프의 x축은 패킷 손실 확률 p를 의미하고, K의 값은 언제나 3으로 설정한다.

그림 4는 TCP SACK과 TCP SACK+의 손실 복구 확률을 TCP Reno의 손실 복구 확률과 함께 비교한 것이다. 모델링을 통한 TCP Reno의 손실 복구 확률은 이미 자세히 분석되어 있으므로[12] 이를

그대로 도입하였다. 세 개의 그래프 모두 패킷 손실 확률이 10^{-2}보다 커짐에 따라 급격하게 감소하기 시작하는 것을 볼 수 있다. TCP Reno를 제외한 TCP의 손실 복구 과정은 첫 번째 손실된 패킷이 fast retransmit에 의해 재전송이 가능한 경우 재전송 손실이 발생하지 않는 한 RTO를 유발하지 않는다. 따라서 10^{-2}보다 큰 패킷 손실 확률에 대한 손실 복구 확률의 감소는 윈도우의 크기 감소로 인해 첫 번째 손실된 패킷에 대한 fast retransmit의

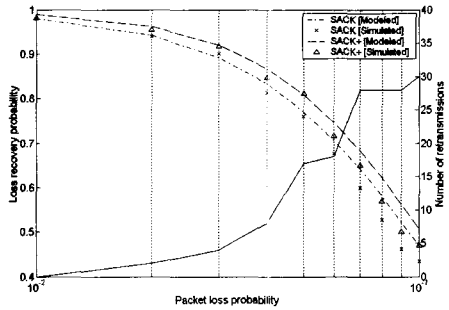


그림 5. 시뮬레이션을 통한 TCP SACK과 TCP SACK+의 손실 복구 확률의 검증 ($W_{\max} = 8$)

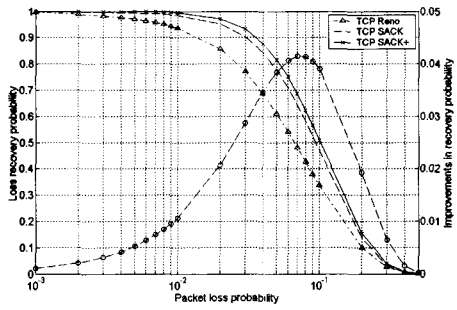


그림 6. TCP SACK과 TCP SACK+의 손실 복구 확률의 비교 ($W_{\max} = 32$)

동작 확률이 감소한다는 사실로 설명될 수 있다. 패킷 손실 확률이 10^{-1}보다 커지면 모든 TCP의 손실 복구 성능은 크게 저하된다.

TCP SACK의 손실 복구 확률은 TCP Reno에 비해서 상당히 높은 것을 볼 수 있다. 이는 다수 개의 패킷 손실에 대한 복구 능력이 손실 복구 확률에 미치는 영향을 반영한 것이다. 그러나, TCP SACK의 손실 복구 확률은 TCP SACK+의 손실 복구 확률에 비해서는 다소 낮다. 이는 TCP SACK+가 재전송 손실을 복구할 수 있음으로 인해서 얻을 수 있는 손실 복구 성능의 향상을 의미한다. 두 손실 복구 확률의 차이는 원으로 표시된 점선으로 표시되는데, TCP SACK+의 손실 복구 확률은 TCP SACK에 비해서 최대 약 4%정도 향상되는 것을 볼 수 있다. 패킷 손실이 증가함에 따라 패킷 손실의 수가 증가하기 시작하고 재전송 패킷 손실이 발생할 가능성도 높아진다. 그러나 패킷 손실이 증가하면 잦은 손실 복구 과정으로 인해서 윈도우의 크

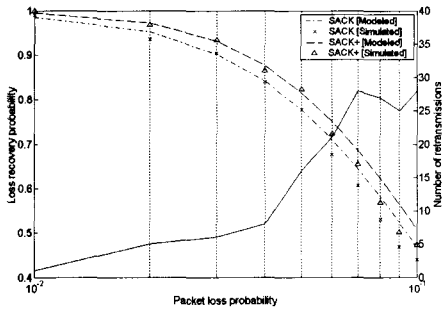


그림 7. 시뮬레이션을 통한 TCP SACK과 TCP SACK+의 손실 복구 확률의 검증 ($W_{\max} = 32$)

기는 작은 값에 머무르게 되기 때문에 실제로 하나의 윈도우 내에 포함되는 패킷 손실의 수는 오히려 줄어든다. 즉, 정상 상태에서의 임의 패킷 손실에 대한 재전송 손실이 발생할 확률 자체가 매우 낮기 때문에, TCP SACK+의 성능 향상이 상당히 낮은 값으로 계산되는 동시에 패킷 손실 확률이 일정한 값보다 커지게 되면 감소하는 형태로 나타나게 된다.

TCP SACK과 TCP SACK+의 손실 복구 확률의 모델링을 통한 계산 결과를 검증하기 위해서 *ns*를 이용한 시뮬레이션 결과를 계산 결과와 비교해서 그림 5에 나타내었다. 하나의 송신원과 수신원은 패킷 손실이 임의로 발생하는 10Mbps, 10msec의 대역폭이 큰 (long-fat) 링크로 연결되어 있다. 송신원은 FTP(file transfer protocol)를 통해서 10^4개의 크기가 1kbyte인 패킷을 전송하는 동안 시뮬레이션은 계속되고, 이 동안 혼잡 윈도우의 최대 값은 $W_{\max}$로 제한된다. 손실 복구 확률은 전송한 패킷들 중에서 손실된 패킷의 개수의 합과 재전송에 의해서만 복구된 패킷 개수의 비로 정의된다. 그림 5에서 볼 수 있듯 모델링을 통한 결과와 시뮬레이션

결과가 서로 일치하는 것을 볼 수 있다. 그림의 실선은 TCP SACK+에 의해 복구되는 재전송 손실의 개수를 의미한다.

그림 6과 7은 $W_{\max} = 32$ 인 경우에 대한 손실 복구 확률의 계산 결과와 시뮬레이션 결과를 보여준다. 두 그림의 형태가 각각 그림 4, 5와 거의 일치하는 것을 볼 수 있는데, 이를 통해서 손실 복구 확률에는 $W_{\max}$ 의 값이 거의 영향을 미치지 않음을 알 수 있다. 패킷 손실 확률이 낮을 때에는 하나의 윈도우에서 다수 개의 패킷 손실이 발생할 확률은 역시 매우 낮기 때문에 대부분의 경우 하나의 패킷 손실이 발생한다. 하나의 패킷 손실은 이에 대해서 세 개의 중복 승인을 수신하는 경우 항상 복구되므로 윈도우가 적어도 손실된 패킷을 포함해서 네 개의 패킷을 포함하기만 한다면 손실 복구 확률에는 영향을 미치지 않는다. 패킷 손실 확률이 증가함에 따라서 다수 개의 패킷 손실이 발생할 수 있지만 이 경우에도 첫 번째 손실된 패킷이 fast retransmit에 의해 재전송이 가능하다면 나머지 손실된 패킷들 모두 복구되기 때문에 역시 손실 복구 확률에는 아무런 영향을 미치지 않는다. 즉, 윈도우의 크기가 첫 번째 손실된 패킷에 대한 fast retransmit이 가능할 정도로만 크다면 그 이상의 값을 갖더라도 손실 복구 확률에는 영향을 미치지 않는 것으로 분석할 수 있다.

V. 결 론

본 논문에서는 TCP SACK의 재전송 손실에 의한 성능 저하를 피하기 위한 알고리듬을 제안하고 확률적인 방법과 시뮬레이션을 통해서 분석하였다. 제안된 알고리듬을 사용하는 TCP SACK+는 기존의 SACK 정보를 바탕으로 동작하기 때문에 최소한의 변화를 통해서 성능 향상을 얻을 수 있다는 장점이 있다. 정상 상태에서의 임의 패킷 손실에 대한 TCP SACK+의 손실 복구 성능의 정량적인 향상은 미미하지만, 실제로 재전송 손실에 의해서 발생하는 RTO가 TCP의 성능에 미치는 영향을 고려했을 때 이는 오히려 상당한 성능 향상을 가져오는 것으로 평가할 수 있다.

참 고 문 헌

[1] J. Postel, "Transmission Control Protocol," RFC 793, 1981.

[2] V. Jacobson, "Congestion Avoidance and Control," in *Proc. ACM SIGCOMM'88,* Aug. 1988.

[3] V. Jacobson, "Modified TCP congestion avoidance algorithm," note sent to end2end-interest mailing list, 1990.

[4] J. Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP,". in *Proc. ACM SIGCOMM'96,* Aug. 1996.

[5] M. Mathis, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," RFC 2001, Oct. 1996.

[6] J. Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP,". in ACM SIGCOMM'96, Aug. 1996.

[7] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," ACM Computer Communication Review, pp. 5-21, vol. 26, no. 3, Jul. 1996.

[8] Dong Lin and H. T. Kung, "TCP Fast Retransmit Strategies: Analysis and Improvements," IEEE INFOCOM'98, pp. 263-271, 1998.

[9] Anurag Kumar, "Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 485-498, Aug. 1998.

[10] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Trans. Networking*, vol.8, no.2, pp. 133-145, Apr. 2000.

[11] R. Branden, "Requirements for Internet Hosts," RFC 1122, 1989.

[12] 김범준, 김동연, 이재용, "임의 패킷 손실에 대한 TCP의 손실 복구 과정 모델링 및 분석," 한국통신학회 논문지, 28권, 4호, 2003.

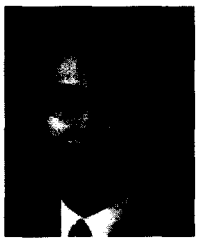
[13] Beomjoon Kim and Jaiyong Lee, "Analytic Models of Loss Recovery of TCP Reno with Packet Losses," *Lecture Notes in Computer Science (LNCS),* no. 2662, 938-947, Oct. 2003.

[14] Beomjoon Kim and Jaiyong Lee, "A Simple Model for TCP Loss Recovery Performance over Wireless Networks," accepted for publication in *Journal of Communications and Networks (JCN)*, vol. 6, 2004.

[15] Beomjoon Kim *et al.*, "Lost Retransmission Detection for TCP Part 2: for TCP using SACK option," in *Proc.* IFIP TC-6 *Networking'2004 (LNCS)*, no. 3042, 2004.

[16] Beomjoon Kim, Dongmin Kim, and Jaiyong Lee, "Lost Retransmission Detection for TCP SACK," accepted for publication in *IEEE Communications Letters*, 2004.

김 범 준(Beomjoon Kim) 정회원

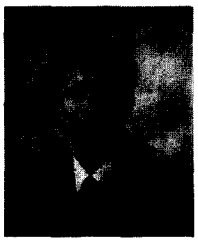


1996년 2월 : 연세대학교 전자 공학과 졸업
1998년 8월 : 연세대학교 전자 공학과 석사
2003년 8월 : 연세대학교 전자 공학과 박사
2003년 6월~2004년 1월: 연세대학교 전기전자공학과 전자정보통신 연구소 박사후 과정 연구원
2004년 2월~현재 : LG전자 이동통신기술연구소 표준화그룹
<주관심분야> IEEE 802.16, IEEE 802.21, B3G Convergence Network, TCP 성능 분석, 무선 링크상의 TCP 성능 향상 방안, IP기반 유무선 통합 네트워크, IP 트래픽 엔지니어링.

김 동 민(Dongmin Kim) 정회원

1999년 2월 : 건국대학교 전자공학과 졸업
2001년 2월 : 연세대학교 전자 공학과 석사
2001년 3월~현재 : 연세대학교 전자 공학과 박사과정
<주관심분야> 무선 TCP, 무선 센서 네트워크

이 재 용(Jaiyong Lee) 정회원



1977년 2월 : 연세대학교 전자 공학과 졸업
1984년5월 : IOWA State University 공학 석사
1987년 5월 : IOWA State University 공학 박사
1987년 6월~1994년 8월 : 포항공과대학 교수
1994년 9월~현재 : 연세대학교 전자공학과 교수
<주관심분야> Protocol Design for QoS Management, Network Management, High Speed Networks