韓國航海港灣學會 第27券 第1號, pp.1~5, 2003.

TCP Sack와 NewReno 프로토콜의 성능비교에 관한 연구

이행남*, 서경현*, 박승섭**
*부경대학교 대학원, **부경대학교 전자컴퓨터정보통신 공학부 교수

A Study on Performance Comparision in TCP Sack and NewReno Protocol

*Hang-Nam Lee**, *Kyoung-Hyun Seo**, *Seong-Seob Park***
* *PuKyong National University, Graduated School*
** *Dept. of Computer Multimedia, PKNU*

요 약 : 최근의 인터넷에서의 데이터 흐름을 보면 비대칭흐름의 경향이 있다. 비대칭흐름은 주로 하향링크의 데이터 흐름이 많은 것이 특징이며 하향링크에서 데이터흐름을 저해하는 요소인 응답(acknowledgement)을 줄여나가는 기법이 필요하다. 본 논문에서는, 시뮬레이션 결과로, 데이터의 손실이 많은 병목구간에서는 Sack의 성능이 더 높다는 것을 알 수 있었다. 패킷 손실면에서 비교해보면 NewReno와 Sack 중에서 NewReno가 Sack보다 더 낮은 것으로 나타났다. 그리고 NewReno와 Sack의 Ack packet에 대한 처리율을 비교해 볼 때 10초 동안의 실험에서 NewReno가 Sack보다 처리되는 패킷의 수도 많다는 것을 볼 수 있었다. 결론적으로 비대칭링크에서의 처리율은 NewReno가 Sack보다 앞서는 것을 알 수 있었다.

핵심용어 : Sack, NewReno, 빠른 재복귀, TCP 메커니즘

ABSTRACT : *Recently, there is asymmetrical transmission in Internet data stream. The asymmetrical transmission has much more downstream than upstream. Owing to this point, it needs to reduce the acknowledgement as element of the obsrtuction in downstream.*

In this paper, according to simulation's result, we know that Sack has good performance than NewReno in bottleneck environment. Comparing two protocols in packet loss rate, NewReno is lower than Sack. And also comparing two protocols in throughput of ack packet, not only NewReno processes ack packet more quickly than Sack, but also NewReno processes more ack packet than Sack protocol during ten seconds in simulation. As a result, NewReno is batter than Sack in throughput of asymmetrical link.

KEY WORDS : *Sack, NewReno, Fast Recovery Algorithm, TCP Mechanism*

1. 서 론

최근의 인터넷 이용 현황을 보면 TCP 연결에 있어서 비대칭적 흐름을 볼 수 있다는 것은 정보의 순방향으로의 데이터 흐름보다는 역방향으로의 데이터 흐름이 많다는 것이다.

실제로 이러한 사실을 뒷받침해 주고 있는 것으로 국내 초고속인터넷 시장에서 비대칭기술에 기반해 있는 ADSL의 성장에서 볼 수 있다.

2001년 우리 나라 초고속 인터넷 가입 현황을 보면 ADSL이 418만명으로 전년동기대비 117.8%, 케이블 모뎀이 약 247만명으로 전년동기대비 88.6% 성장했다. 이처럼 ADSL이 케이블모뎀의 약 2배 정도의 점유율을 보이면서 국내 초고속 시장을 이런 최근의 국내외적 상황으로 볼 때, 현재 TCP에서도 비대칭링크의 흐름을 적절하게 수용하면서 속도와 성능을 향상시킬 수 있는 TCP 개발이 급선무이라는 것을 알 수 있다[4].

따라서, 본 논문에서는 TCP 프로토콜 별로 일반적인 성능과 속도를 비교해 보고, 혼잡 링크에서의 프로토콜별 성능 및 속도, 하향 링크의 데이터 흐름이 상향 링크의 데이터 흐름보다 많은 현재의 네트워크 상황에서 하향 링크의 데이터 흐름의 감소를 가져올 수 있는 방안으로 Ack 패킷에 대한 처리율을 대표적인 두 프로토콜별로 분석해 보았다[2].

이를 바탕으로 두 프로토콜의 패킷손실률을 분석하여 성능을 비교하였다.

*정회원, hnam82@kr.yahoo.com, 011)9317-6894
***정회원, parkss@pknu.ac.kr 051)620 6389

본 논문 내용은 1장 서론에 이어, 2장은 TCP 메커니즘과 역사를 기술하였고, TCP Sack Protocol과 TCP NewReno Protocol의 특성을 Ack에 초점을 맞추어 설명하였으며, 그리고 3장에서는 시뮬레이션 환경, 4장에서는 시뮬레이션 결과 분석, 5장에서는 결론을 서술하였다.

2. TCP 메커니즘과 역사

TCP/IP 인터넷 프로토콜에 의해 제공되는 신뢰성이 보장되는 데이터 전송 서비스를 TCP라고 한다.

TCP 프로토콜은 신뢰성 있는 전송을 수행하기 위하여 TCP헤더에 송신지 포트와 수신지 포트를 가지고 두 종단 간에 데이터를 실어 나르고 종단간의 대화를 통하여 데이터가 정확히 수신되었는지를 체크하여 전달상의 이상유무를 판단하고 에러가 생겼을 경우나 혼잡이 생겼을 경우 이를 적절히 제어하는 역할을 수행한다.

이러한 TCP프로토콜은 초기에는 신뢰성 없는 네트워크상에서 바이어 스트림의 효율적이고 신뢰성 있는 전송을 위해 타임아웃에 기반을 둔 재전송과 윈도우 흐름제어가 도입되었고 RFC 793에서 정의되었다[9].

두 번째 버전인 TCP Tahoe에서는 혼잡제어와 빠른 재전송이 추가되었다. 세 번째 버전은 TCP Reno로 빠른 회복알고리즘을 포함하는 혼잡제어알고리즘이 확장되었다.

Reno는 RFC2001에서 표준화되었고 RFC2581에서 일반화되었다[11][12].

TCP Reno는 매우 대중적인 인기를 모았으나 한 윈도우에서 여러 개의 패킷이 유실되었을 때 성능이 크게 떨어지는 문제점을 보였다.

그래서 NewReno, Sack 버전에서는 이 문제를 두 가지 면에서 다르게 접근했는데 NewReno에서는 빠른 회복 구간에서 빠른 재전송을 시도함으로써 한 윈도우내에서 여러 개의 패킷이 유실되었을 때에 패킷을 안정적으로 회복하려는 알고리즘을 구사하였다. 반면에 Sack에서는 수신자가 수신된 또는 Queue에 저장된 패킷에 대하여 송신자에게 알려줌으로써 송신자가 유실된 패킷에 대한 정보를 알고 이에 대한 패킷을 재전송하는 기법을 사용한다.

NewReno와 비교할 때 한 윈도우 내에서 여러 개의 패킷이 유실될 때 보다 나은 성능을 발휘한다. 그리고 Fack는 Duplicate Ack를 기다리기까지는 오랜 시간을 걸려야 한다는 점에 착안하여 좀 더 빨리 Selective Ack를 사용할 수 있다는 점에 착안하여 만든 Sack를 개선하여 만든 알고리즘이다.

Brakmo와 Peterson에 의해서 새롭게 제안된 손실회피알고리즘이 Vagas로 Vagas는 패킷의 손실을 피하기 위하여 송신자가 송신할 수 있는 패킷의 양을 정확하게 계산하는 방법으로 측정된 RTT(Round Trip Time)를 사용하는 알고리즘이다. 이와 같이 TCP 혼잡제어 알고리즘의 변화는 **Table 1**에서 정리하여 비교하였다[13].

Table 1 Comparison of TCP version algorithm

TCP algorithms	loss-recovery/avoidance	Modification sender/receiver	Features Enhanced
oldTahoe	Recovery	-	· Window based flow control · Timer Granularity
Tahoe	Recovery	-	· Slow Start · Congestion avoidance · Fast retransmision
Reno	Recovery	-	· Fast recovery
TCP congestion control	Recovery	No	· General congestion control Schme of Reno
SACK	Recovery	Both	· Better estimation of outstanding packet during fast recovery · Earlier fast retransmission
Rate-Halving	Recovery	Both	· Smoothing window reduction during fast recovery · Additional updating the TCP state variables after fast recovery
SSDR	Recovery	Sender	· Smooth start · Dynamic recovery
Vagas	Recovery	Sender	· New congestion avoidance · Modified slow start mechanism · Earlier packet loss detection
Pseudo-Rate	Recovery	Sender	· New RTT estimation · Direct drop decrease · Exporiential increase
FACK	Recovery	Both	· Better estimation of outstanding packet during fast recovery · Earlier fast retransmission
NewReno	Recovery	Sender	· Response to partial Acks

다음은 한 윈도우 내에서 다른 패킷이 유실될 때 특히 특별한 Ack를 사용함으로써 패킷의 손실을 회복하고 있는 최근의 알고리즘인 NewReno와 Sack의 특징들에 대해서 설명한다.

2.1 TCP Sack Protocol

Fig. 1은 수신자가 제공하는 Selective Ack를 가지고 송신자는 수신자에게 성공적으로 도착한 패킷에 정보를 가지고 실제로 손실된 세그먼트만을 재전송하는 TCP Sack 동작 원리를 나타낸다.

이와 같은 기능을 수행하기 위한 수신자의 행위는 ① 데이터 송신자가 SYN내에 있는 Sack-permitted Option을 수신자에게 보내면 수신자는 Sack Option을 사용할 수 있게 된다.

② 수신자는 수신된 세그먼트들에 대한 대응으로 송신자에게 Ack를 보내야 하고 duplicate ack도 Sack Option에서 배출하도록 해야 한다.

③ Sack Option을 가지는 Ack는 데이터 수신자의 버퍼 Q에서의 가장 최근의 상황을 반영할 수 있도록 Ack를 유발하는 세그

먼트를 갖고 있는 인접 데이터 블록을 명시한다.

④ Sack Option에서 가능한한 많은 정확한 Sack Block를 포함해야하지만 최대로 사용할 수 있는 Sack Option은 적어도 3개의 성공적인 Sack Option이 유지되도록 해야 한다.

Sack Option을 해석하고 재전송하는 데이터 송신자의 행위는 다음과 같이 명시할 수 있다

① Sack Option을 갖고 있는 Ack를 받았을 때 데이터의 송신자는 미래에 참조하기 위하여 Selective ack를 기록해둔다.

② 데이터 송신자는 Sack Option을 받기 전에 재패킷화를 수행하려고 할 때 Sack Option과 중복되지 않도록 회피한다.

③ 재전송 큐에 있는 Sack Option 블록에 있는 sacked flags bits는 송신자가 좀 더 나중에 세그먼트들이 전송되어야 할 것임을 알고 전송을 할 것이다.

④ 데이터의 송신자는 재전송타임아웃이 일어나면 모든 sacked bits를 turn off하고 윈도우 왼쪽 가장자리에 있는 세그먼트들을 재전송할 것이다[7][8].

TCP Sack Protocol은 이와 같은 수신자와 송신자의 행위로 데이터의 신뢰성을 높인다.

특히 무선 구간에서는 패킷의 유실이 많기 때문에 효율적이나 유선 구간에서의 그 성능은 복잡성에 비하여 효용성이 크지 않은 것으로 보이는데 이것은 시뮬레이션에서 검증해 갈 것이다[1].

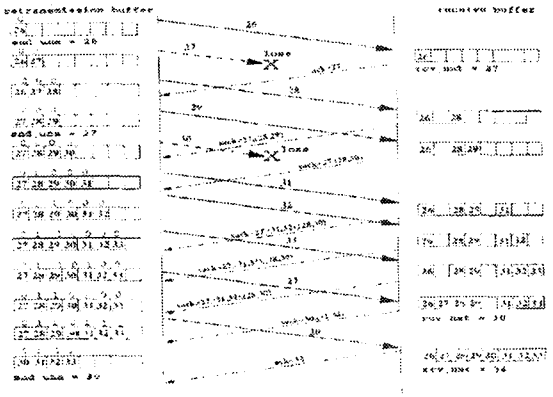

Fig. 1 An action principle of TCP Sack

2.2 The NewReno TCP's Fast Recovery Algorithm

NewReno Algorithm은 Sack가 없을 때 Partial acknowledgement로 대응할 수 있게 만들었다.

Fig. 2와 같이 Partial Ack는 하나의 윈도우에서 다수의 패킷이 유실될 때 송신자는 첫 번째로 빠른 재전송에 들어가고 이 때 이미 전송되었던 패킷에 대한 Ack를 받는데 이를 Partial Ack라고 한다.

만일 거기에 하나의 패킷이 손실되었다면 이 Ack는 빠른 전송에 들어가기 전에 전송되었던 모든 패킷에 대한 Ack가 될 것이고 만일 거기에 다수의 패킷이 손실되었다면 이 Ack는 빠른 전송이전에 전송되었던 모든 패킷이 아니고 약간의 패킷에 대한 Ack가 될 것이다.

NewReno의 Fast Recovery produre는 three Duplicate Ack를 받았을 때 시작하고 재전송타임아웃이나 빠른 회복과정이 시작되었을 때 나타나는 데이터를 포함하여 그 때까지 모든 데이터를 응답하는 Ack가 도달했을 때 끝난다.

three Duplicate Ack를 받고 송신자가 아직 빠른 회복과정에 들어가지 않았을 때 임계치의 값은 SSthresh = max(Flightsize /2, 2*MSS)이고 손실된 세그먼트를 재전송하고 혼잡윈도우를 ssthresh + 3*MSS로 설정한다.

그리고 각각의 부가적인 Duplicate Ack를 받고 있는 동안 MSS에 의해 혼잡윈도우가 증가한다.

만일 혼잡윈도우의 새로운 값이 허락되고 수신자에게 윈도우를 광고했다면 세그먼트를 전송한다.

재전송타이머는 첫 번째 Partial Ack 후에 갱신되고 만일 많은 수의 패킷이 하나의 윈도우 데이터로부터 손실될 때 TCP 송신자의 재전송타이머의 크기는 일반적으로 RTT보다 크지 않기 때문에 작은 수의 패킷이 손실될 때에도 재전송타임아웃이 일어날 수가 있으며 빠른 회복기간에 재전송타임아웃이 일어나면 또 불필요하게 빠른 재전송이 일어날 수 있다. 다시 말하면 데이터 송신자는 모든 데이터가 다 수신되었다는 응답이 있을 때까지 빠른 회복기간에 머무를 수 있다는 것이다.

그러므로 빠른 재전송을 회피하기 위한 전략이 필요한데 이것은 세 개의 Duplicate Ack가 왔을 때 Duplicate Ack가 send_high로 들어갈 때 send_ high보다 번호가 높은 시퀀스 번호를 가지면 빠른 재전송에 들어가지 않으며 그보다 낮은 시퀀스 번호가 왔다면 빠른 재전송에 들어가게 하는 방법을 구사하고 있다[10].

그러나 여전히 NewReno는 Sack Protocol이 없을 때만 사용할 수 있는 유용한 방법이라는 사실에는 변함이 없다.

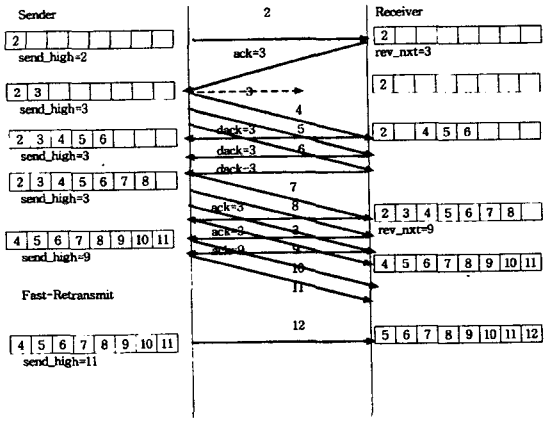

Fig. 2 An action principle of NewReno

3. 시뮬레이션 환경

본 연구에서는 네트워크의 대표적인 시뮬레이션 도구인 ns-2를 가지고 실험하였다[5].

Fig. 3과 같이 구현된 망에 사용된 파라미터는 **Table 2**와 같다.

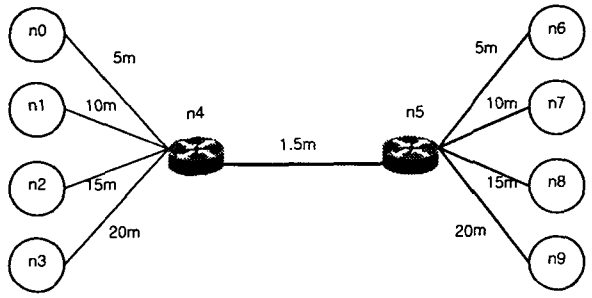

Fig. 3 Simulation topology

Table 2 Simulation parameter

window size	8
packet size	512byte
application	FTP
ms	2ms
end time	10 seconds
queue	drop tail
선의 종류	duplex-link(양방향)
망의 길이	5M/10M/15M/20M
TCP Sender	sack1/NewReno
TCP Reciver	TCPSink

4. 결과 및 분석

ns-2에 있는 n0와 n1을 지나는 하나의 노드를 가진 환경인 simple.tcl을 가지고 NewReno와 Sack에 대한 기본적인 성능을 평가하였다.

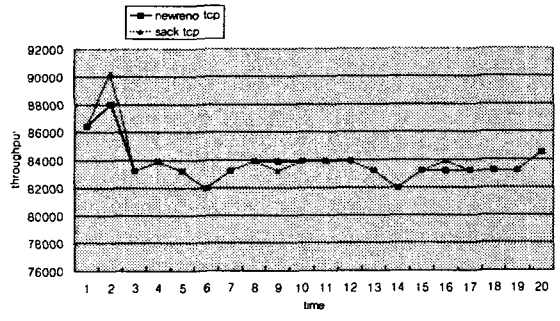

Fig. 4 General throughput of NewReno and Sack

Fig. 4와 같이 초기에는 Sack protocol의 packet에 대한 처리율이 NewReno보다 앞섰으나 시간이 지날수록 큰 차이가 없음을 볼 수 있다. 이것은 패킷의 손실이 많이 일어나는 병목 구간이 아닌 일반적인 데이터흐름을 가지는 구간의 경우에는 두 프로토콜의 성능에는 별 차이가 없다는 것을 나타내었다.

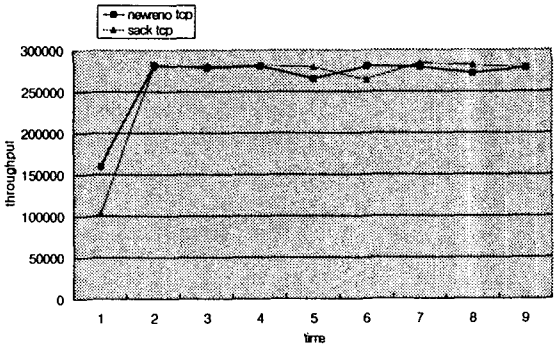

Fig. 5 Throughput of NewReno and Sack that pass a bottleneck environment

Fig. 5는 각각의 4개의 노드가 붙어있는 n4와 n5의 병목구간을 지나가는 NewReno와 Sack TCP의 처리율을 비교한 결과이다. 결과를 보면 초기의 TCP packet에 대한 처리율에 대해서는 NewReno가 성능이 높고 3초이후에는 Sack가 높은 처리율을 보이면서 안정적으로 패킷을 처리하고 있다는 것을 볼 수 있다. 즉 초기에는 NewReno의 성능이 그리고 시간이 지날수록 Sack의 성능이 높다는 것을 보여준다. 이것은 패킷의 수가 적은 초기에 패킷이 손실될 때 대응하는 정도가 NewReno가 빠르다는 것이며 Sack는 많은 패킷이 손실될 때 패킷을 안정적으로 처리하고 있음을 보여주는 것이라 하겠다[14].

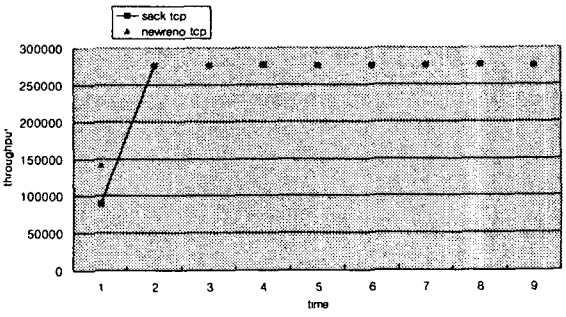

Fig. 6 Throughput of Ack packet of NewReno and Sack

Fig. 6과 같이 병목 구간을 지나는 Ack packet에 대한 처리율을 각각 비교해 보면 NewReno가 sack protocol보다 시간적으로나 처리되는 패킷의 수를 보나 기능이 앞선 것으로 볼 수 있다. 두 번째 실험과 세 번 째 실험에서 순방향에서의 손실율 P와 역방향에서의 손실율 Q를 구하면 다음과 같다[3].

$P = 1 - (1 - E1)^{s1}$ $Q = 1 - (1 - E2)^{s2}$

E1 = 순방향에서의 bit error rate
s1 = packet size
E2 = 역방향에서의 bit error rate
s2 = 역방향에서의 ack size

1-E1은 bit error rate를 뺀 TCP throughput라고 볼 수 있고 1-E2는 Ack throughput라고 할 수 있다. 위의 식으로 프로토콜별로 손실율을 비교해 보면 순방향에서의 손실율은 거의 비슷하고 역방향에서의 손실율은 Sack가 NewReno보다 약간 앞선다. 따라서 손실율 부분에서 프로토콜별 성능비교해 보면 NewReno가 Sack보다 약간 앞선다고 할 수 있다.

5. 결론

일반적인 구간에서 NewReno와 Sack protocol을 비교해 보면 Sack의 성능이 약간 앞서는 것으로 나타났으나 시간이 지나감에 따라 큰 차이가 없음을 볼 수 있었다는 사실에서 볼 때 NewReno와 Sack protocol 성능의 차이는 데이터흐름이 많은 병목구간에서 확연히 나타난다는 것을 알 수 있다.

병목구간에서 TCP 패킷에 대한 처리율은 초기에는 NewReno가 높고 시간이 지날수록 Sack가 높은 처리율을 보여주고 있음을 볼 수 있었다.

따라서 데이터 흐름이 많기 때문에 패킷의 손실이 많은 수 있는 병목 구간에서 Sack가 NewReno보다 높은 성능을 나타낸다고 볼 수 있었다. 이것은 기존에 발표된 사실과 크게 다르지 않음을 볼 수 있었다. 그리고 본 논문에서 살펴본 역방향의 데이터 흐름을 되먹임하는 Ack packet에 대한 처리율을 보면서 역방향에서의 Sack와 NewReno의 성능을 비교하는 부분에서는 10초를 기준으로 1초에서 6초까지는 NewReno가 시간적으로나 Ack packet에 대한 처리율에서도 앞서고 있다는 것을 볼 수 있었다. 그러나 7초 이후에 가면 Ack packet에 대한 처리하는 시간이 Sack가 앞서고 처리율은 비슷하게 나타나는 것을 볼 수 있었다. 손실율 부분에서 보면 순방향에서의 packet 손실율은 비슷하였으나 역방향에서 Ack packet 손실율은 Sack가 더 높은 것으로 나타났다.

위의 사실을 볼 때 NewReno는 TCP packet에 대한 처리율보다 Ack 패킷에 대한 처리율이 빠르고 적은 손실율을 보인 점에서 Sack보다 성능이 앞선다는 것을 볼 수 있는데 이것은 비대칭링크 특히 하향 링크에서 NewReno의 성능이 Sack 성능보다 비교적 앞서고 있다는 것으로 볼 수 있으며 따라서 비대칭링크에서의 처리율과 손실율 부분에서는 NewReno가 Sack보다 더 적합한 프로토콜임을 볼 수 있었다. 향후 다른 기준을 가지고 두 프로토콜에 대해서 비교해 본다면 비대칭링크에서의 NewReno와 Sack의 성능을 보다 더 제대로 파악할 수 있는 실험이 되리라고 본다.

참고문헌

[1] 이상연, 정충교, "유무선 통합망에서의 SACK TCP 프로토콜 성능 개선 방안", Journal of Telecommunication and Information, Vol.2, 1998.

[2] 이은상, "TCP ACK 패킷의 차등 처리에 의한 인터넷 트래픽 성능에 관한 연구", 한국통신학회논문지, 2000-7 vol.25. No.7B

[3] Farooq Anjum and Ravi Jain, "Performance of TCP over Lossy Upstream and Downstream Links with Link-level Retransmission", IEEE,2000]

[4] http://isis.nic.or.kr

[5] http://www.isi. edu/naman/ns

[6] K. Fall, S. Floyd, "Simulation based comparisons of Tahoe, Reno, and SACK TCP", Computer Communication Review, July, 1996.

[7] M. Mathis, J. Mahdavi, S. Floyd, A. Ronanow, "TCP select acknowledgment option", RFC 2018 1996.

[8] Mark A. Smith and Ramakrishnan, Member, "Formal Specification and Verification of Safety and Performance of TCP Selective Acknowledgment", IEEE,2002

[9] Postel. J, "Transmission Control Protocol", RFC793, September,1981

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

[11] Stevens. W, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC2001, January,1997

[12] Stevens. W, Allman. M and V. Paxson, "TCP Congestion Control", RFC 2581, April, 1999

[13] Yuan-Cheng Lai and Chang-Li Yao, "TCP Congestion control algorithms and a performance comparison", Computer Communication and Networks, 2001

[14] Zhu Jing, Li Zhengbin, Niu Zhisheng, "A Modified TCP-NewReno Retransmission Scheme for Lossy Network", APCC/OECC'99, p204-208, Oct, 1999.