TCP를 사용하는 위성링크에서의 성능 향상을 위한 ENA 알고리즘

정회원 이 정 규*, 김 상 희*

A ENA algorithm for Performance Enhancement of Satellite Link using TCP

Jong-kyu Lee*, Sang-hee Kim* *Regular Members*

요 약

본 논문에서는 긴 전파 지연과 높은 에러율을 지닌 위성링크에 TCP를 사용하는 경우 나타나는 문제점을 도출하고 그에 대한 성능향상방안으로 ENA(Error Notification Ack) 알고리즘을 제안했다. TCP는 에러에 의한 세그먼트 손실과 Network Congestion에 의한 손실을 구분하지 못하고 두 경우 모두 Network Congestion으로 판단한 후 Slow Start나 Congestion Avoidance 알고리즘을 적용시킨다. 위성링크에서는 에러에 의한 세그먼트 손실이 자주 발생하는데 이 손실을 Network Congestion에 의한 손실로 받아들여 매번 Congestion Control 알고리즘을 적용시킨다. 그로 인해 전송률은 줄어들고 성능은 급격히 떨어진다. 본 논문에서는 에러에 의한 손실과 Congestion 의한 손실을 구분해 주는 ENA 알고리즘을 제안하고 구현 방법 또한 제시하였다. 그리고 ENA 알고리즘을 Tahoe, Reno, Sack TCP에 적용하여 성능의 변화를 비교 분석했다. 그 결과, 에러율이 높을수록 성능이 더 향상됨을 알 수 있었다.

ABSTRACT

In this paper, We report on the performance issues faced by TCP based applications on satellite link having long propagation delay and high probability of bit erros and propose ENA(Error Notification Ack) algorithm for TCP Performance Enhancement. TCP Protocol cannot distinguish errored segments(in noisy medium) from losses of genuine network congestion and react as if there is network congestion. Therefore, Slow Start and Congestion avoidance mechanism are initiated. It happen this case in satellite link. Therefore it reduce the transmission rate and drop the performance. So, in this paper We propose ENA algorithm which is distinguished errored segments from losses of network congestion. And We propose the method of algorithm's implementation. And We evaluate the performance of Tahoe, Reno, Sack TCP with ENA. As results, TCP Performance is better.

Ⅰ. 서 론

주로 문자 형태의 저 트래픽이 주를 이루었던 인터넷이 멀티미디어 서비스를 제공하기 위해 음성, 데이터, 비디오 등 여러 종류의 트래픽 형태로 변환되고 있다. 이러한 멀티미디어 서비스를 만족하기 위해 위성망을 이용한 인터넷 서비스가 관심의 대상이 되고 있다. 그러나 정지위성은 기존의 지상망

* 한양대학교 전자계산학과
논문번호 : 99504-1230, 접수일자 : 1999년 12월 30일

과는 다른 특징을 가지고 있다. 지상망보다 상대적으로 긴 전파지연시간과 위성과 같은 무선환경에서의 높은 에러율, 비대칭적 사용, 가변적인 RTT (Round Trip Time), 간헐적인 연결등이 그 특징이다. 특히 긴 전파지연시간과 높은 에러율은 TCP Congestion Control 알고리즘에 영향을 준다. TCP는 에러에 의한 세그먼트 손실과 Network Congestion에 의한 손실을 구분하지 못하고 두 경우 모두를 Network Congestion으로 판단한다. 그래서 Slow Start나 Congestion Avoidance 알고리즘을 적용시킨다. 그 결과 TCP의 성능이 감소하게 되는 것이다. 따라서 본 논문은 정지위성이 가진 특징 중 위 두 가지 관점에서 TCP성능을 연구하게 된다.

지금까지 TCP에 대한 많은 연구들이 진행되었다. 우선 TCP의 성능을 향상시키기 위한 방안에 대한 연구이다. Bikram S.Bakshi은 무선환경에서 성능을 향상시키는 방법에 대해 연구했다[17]. Aldar C.F.Chan은 무선환경에서의 Tahoe TCP 환경에서 성능을 분석하고 NACK(Nagative ACK)알고리즘을 제안했다[1]. Kevin Fall은 시뮬레이션을 통해 유선망 환경에서 SACK을 사용하였을 경우 TCP성능이 향상됨을 보였다[2]. Mark Allman은 위성에서 초기 윈도우를 1 segment가 아닌 4 segment로 설정하는 것이 더 좋은 성능을 얻을 수 있다는 것을 보였다[3]. Craig Partirdge는 위성환경에서의 TCP 성능을 저해할 수 있는 요인들을 종합정리하였다[4]. Mark Allman과 Hans Kruse는 T1 또는 T1/2의 대역폭을 갖는 위성망에서의 성능을 평가했다. 다음은 TCP의 수학적으로 분석한 연구들이다[5]. T.V.Laskshman와 Aldar C.F.Chan는 Tahoe와 Reno에서의 TCP 성능을 분석하였다[6,12].

하지만 에러에 의한 손실과 Network Congestion으로 인한 손실을 구분해 줌으로써 성능을 향상시키는 부분에 대해서는 아직 연구된 바가 없다. 따라서 본 논문에서는 에러에 의한 손실과 Congestion 의한 손실을 구분해 주는 ENA(Error Notification Ack) 알고리즘을 제안했다. 그리고 Tahoe, Reno, Sack TCP에 적용하여 성능을 평가했다. 성능평가결과 제안된 알고리즘을 사용하는 경우 성능이 우수한 것으로 나타났다.

본 논문의 구성은 다음과 같다. II장에서는 Slow Start, Congestion Avoidance, Fast Retransmit, Fast Recovery등의 TCP Congestion Control과 기존의 알고리즘을 적용한 위성망에서 TCP성능문제점에 대해 알아보았고 3장에서는 위성환경에서 TCP 성능향상을 위해 제안된 표준기법들을 살펴보았다. 4장에서는 ENA(Error Notification Ack)라는 새로운 알고리즘 및 구현방법을 제시하였으며 5장에서는 시뮬레이션을 위한 시스템 모델을 설정하고 ENA알고리즘을 적용한 경우와 적용하지 않은 경우의 성능을 비교 평가했다. 마지막으로 6장에서는 본 논문의 결론을 맺었다.

II. TCP의 Congestion Control

TCP는 인터넷 서비스(SMTP, HTTP, FTP등)를 전송하기 위해서 사용하는 프로토콜로써 신뢰성과 연결성을 갖는 전송 프로토콜이다. 송신측이 수신측으로부터 바로 이전 세그먼트의 Ack을 받아야 새로운 세그먼트를 보낼 수 있도록 Flow Control를 한다. 그리고 TCP는 효율적인 전송을 위해 Congestion Control 알고리즘을 사용한다. 만일 TCP가 빠른 속도로 세그먼트를 전송하면 네트워크상에 존재하는 라우터에 과부하를 발생시키고 이는 Congestion을 일으킨다. 이를 방지하기 위해서 TCP는 세그먼트의 전송 속도를 Congestion Control 알고리즘을 사용하여 조절한다[10,14].

1. Slow Start and Congestion Avoidance

과거의 TCP에서는 송신측이 다수개의 세그먼트를 수신측의 Advertised Window Size만큼 보내는 것을 기본으로 했다. 하지만 저속의 링크가 연결되어 있을 경우, 세그먼트들이 라우터에 대기되는 현상이 발생할 수 있다. 따라서 새로운 세그먼트를 보내고자 할 때는 수신측에서 보내는 Ack의 비율에 맞추어서 보내고자하는 것이다.

slow start와 congestion avoidance 알고리즘은 네트워크의 중간에 있는 라우터들이 자신의 buffer size를 overflow하지 않는 한도내에서 송신측이 데이터 전송량을 증가할 수 있도록 보장하는 알고리즘이다. 이런 메카니즘을 이루기 위해서는 "≅"*estion*""*Window*(*cwnd*) 라는 variable가 필요하다. TCP의 "≅"*estion*""*Window*는 송신측에서 사용하는 Sliding Window의 Size를 말한다. 그리고 *cwnd*는 Receiver's Advertised Window의 Size를 넘길 수 없다. slow start는 TCP connection이 설정이 된 직후와 congestion이 발견된 직후에 적용된다. 이 알고리즘이 처음 시작할때는 *cwnd*=1, *ssthresh*=*Receiver*""*Window*""*Size*로 설정된다. 예를 들어 첫 번째 세그먼트의 ACK이 도착하면 *cwnd*는 2 세그

먼트로 증가하고 TCP는 두 개의 세그먼트를 전송하게 된다. 이 알고리즘은 *cwnd*값이 *slow*""*sta*""*rf*"*threshold*(*ssthresh*)에 값에 이르거나 세그먼트의 손실이 발견될때까지 지수적으로 증가한다. 만약 RTO Timeout이 걸리면 TCP는 그에 대한 반응으로 *ssthresh* 의 값을 *cwnd*의 1/2값으로, *cwnd*의 값을 1로 설정함으로써 전송량을 줄이게 된다. 그리고 다시 slow start 알고리즘을 다시 시작하게 되는 것이다.

slow start phase 다음에 따라오는 알고리즘은 Congestion avoidance이다. 이 phase에서는 *cwnd*값이 *ssthresh*값보다 같거나 크게 된다. 이 알고리즘은 slow start phase에서보다 *cwnd*값을 천천히 증가시킨다. 즉, 이 Phase에서는 Ack에 대해서 *cwnd*는 $\frac{1}{cwnd}$ 만큼씩 증가하게 된다. 따라서 매 RTT마다 *cwnd*의 값이 대략적으로 1 세그먼트씩 증가한다. Congestion Avoidance 알고리즘에서는 TCP's sliding window의 size을 선형적으로 증가시키며 네크워크로 전송되는 세그먼트의 양 또한 선형증가를 나타나게 된다.

2. Fast Retransmit and Fast Recovery

Fast Retransmit의 목적은 한 개의 손실된 세그먼트를 RTO(Retransmit Timeout)전에 발견하고 Retransmit함으로써 성능을 향상시키고자 하는 것이다. 즉, 송신측이 중복되는 Ack을 3번 받으면 그 세그먼트는 손실된 된 것으로 판단하고 재전송을 한다. Fast Retransmit 알고리즘을 사용하면 Fast Recovery 알고리즘을 같이 사용하게 된다. Fast Recovery 알고리즘은 3개의 중복 Ack을 받으면 *cwnd*를 1로 줄이지 않고 *ssthresh*의 반값으로 setting한다.

3. TCP 성능을 저하하는 위성링크의 특징

위성은 수십 Mbps에서 수 Gbps의 링크속도를 제공할 수 있으나 동시에 위성구간은 긴 전파지연시간을 갖는다[7]. 이는 두 가지 문제점을 가지고 있다. 첫 번째 slow start time이 길어진다는 점이다. slow start time은 다음과 같은 식(1)으로 표현할 수 있다.

$$slow""sta"rf"time = R\log_2 W \tag{1}$$

따라서 Round Trip Time이 긴 위성링크에서는 slow start time이 길어지면서 성능이 저하된다. 두 번째는 최대전송률 측면의 문제점이다. 원래의 TCP 표준은 TCP 수신 윈도우를 64 Kbyte로 제한한다. TCP의 수신윈도우 크기는 TCP연결의 최대 전송률이 왕복시간에 의해서 제한된다. 즉, 최대 전송률은 식(2)과 같다.

$$throughput_{\max} = \frac{receive""buffer""size}{round"trip""time} \tag{2}$$

정지위성의 경우 식(3)와 같이 성능이 제한된다.

$$throughput_{\max}(satellite) = \frac{64Kbytes}{590ms} \approx 112,000\frac{bytes}{"\sec ond"} \approx 896,000\frac{bits}{"\sec ond"} \tag{3}$$

즉, T1(1.544Mbps)의 대역폭을 갖는 위성링크를 사용해도 기존의 TCP에서는 896Kbps의 전송률밖에 얻을 수 없다. 그리고 위성망에서 제공하는 무선 링크는 지상망에 비해 높은 BER(Bit Error Rate)을 나타낸다. 이런 높은 에러율은 TCP성능을 제한한다. TCP는 에러에 의한 손실을 Congestion으로 인한 손실로 해석하여 전송률을 낮추고 성능을 저하한다.

Ⅲ. TCP성능향상을 위해 제안된 표준 기법

TCPSAT WG(Working Group)에서는 기존의 표준들을 이용하여 위성망에서의 TCP 성능향상이 가능한 기법들을 정리하여 문서화하였다[7]. 그리고 PILC(Pefromance Implications of Links characteristics) WG에서도 위성링크에서의 TCP성능향상을 위한 논의가 진행중이다[8]. 이들 중 TCP 계층에서는 Window Scale Option(RFC 1323), SACK(RFC 2018)이 있다[9,15,16]. Window Scale Option은 높은 DB(Delay × Bandwidth)를 갖는 환경에서 윈도우 크기를 증가시켜 TCP가 고대역폭 링크를 모두 활용할 수 있도록 한다. 그리고 SACK은 한 Window 내에서 여러개의 세그먼트 손실이 발생했을 때 블록화 된 Ack정보를 송신측이 수신할 수 있으므로 재전송이 용이하도록 한다.

그리고 Slow start Time을 줄이기 위한 방법이 제시되었다. 손실이 없는 상황에서 connection이 full connection speed에 다다르는 시간 즉, *cwnd*가 advertised window size에 도달하는 시간을 Slow start time이라 한다. 이 시간을 줄일수록 성능이 향상될 수 있는 것이다. 그 방안으로 초기 윈도우 사이즈를 1 세그먼트보다 크게 설정하는 방법이 다음

(4)과 같이 제안되었다.

$$IW = \min(4*MSS, \max(2*MSS, 4380)) \quad (4)$$

따라서 Slow start time은 다음(5)과 같고 시간이 짧아짐으로써 성능이 향상된다.

$$Slow\text{-}start\ time = R(\log_2 W_A - \log_2 W_I) \quad (5)$$

(W_A: advertised window, W_I: initial window)

느린 Acks의 장점은 수신 호스트가 두 개의 세그먼트가 도착할 때마다 하나의 Ack을 보냄으로써 수신 호스트와 라우터의 처리량을 줄이는 것이다. 그리고 이 알고리즘은 bulk 전송에서는 좋은 성능을 나타낸다. 하지만 Slow start time이 두 배로 증가한다는 단점이 있다. 그리고 초기 윈도우 사이즈를 1로 잡을 경우 하나의 세그먼트에 대해서는 Ack을 발생하지 않으므로 RTO까지 기다린 다음 Ack을 발생시키기 때문에 시간이 많이 소비된다.

그리고 ECN(Explicit Congestion Notification)을 사용하여 라우터가 송신측에서 Congestion이 발생했음을 알리고 더 이상의 세그먼트를 드롭시키지는 않는 방법 또한 연구되었다. 전달방법에 따라 두 가지로 구분된다. BECN(Backward ECN)은 라우터가 송신단에게 직접 ICMP(Internet Control Message protocol) Source Quench메세지를 전달하여 전송률을 감소시키는 방법이며 FECN(Forward ECN)은 라우터가 해당 패킷에 혼잡을 나타내는 비트를 설정하여 포워딩하면 수신단은 해당표시를 ACK 패킷에 나타내어 전달한다. ECN은 RED(Randim Early Detection)에 적용하여 능동적으로 라우터에서TCP 전송률을 조절할 수 있다.

Ⅳ. ENA 알고리즘 및 구현방법

위의 II, III을 통해서 위성망에서의 TCP성능의 문제점 및 그에 대한 해결방안으로 제시된 기법들에 대해 자세히 알아보았다. 지상망의 경우 에러에 의한 세그먼트 손실은 전체 손실의 1% 정도밖에 차지하지 않기 때문에 에러에 의한 세그먼트 손실을 Congestion에 의한 손실이라고 가정하고 알고리즘을 적용시켰다. 하지만 위성의 경우는 에러에 의한 손실들이 많이 발생한다. 이러한 손실들을 Network Congestion으로 판단하고 Congestion Control 알고리즘을 적용하는 것은 TCP의 성능을 저하시킨다. 물론 FEC(Forward Error Correction)을 통해 BER을 10^{-7}이라고 줄일수 있다. 하지만 하드웨어적으로 비용이 많이 들고 여전히 유선링크에 비해 높은 에러율을 갖는다. 이 논문에서는 이러한 문제점을 해결하여 성능 향상을 시킬 수 있는 알고리즘을 제안한다.

지금까지 TCP의 성능을 향상시키기 위한 연구가 많이 진행되어왔다. TCP는 크게 3가지 Version으로 나뉜다. Slow start phase와 Congestion avoidance phase로 구성된 Tahoe TCP, Slow start phase, Congestion avoidance phase, Fast retransmit, Fast Recovery로 구성된 Reno TCP, 다수의 세그먼트 손실이 발생하는 경우에 좋은 성능을 나타내는 SACK TCP로 나눌 수 있다. Tahoe, Reno와 SACK은 다른 ACK메카니즘을 사용한다. 따라서 각 Version마다 적용하는 알고리즘의 형태가 다르게 나타난다. Cumulative ACK을 사용하는 Tahoe와 Reno, 그리고 Selective Ack을 사용하는 SACK의 경우 각각에 대해 성능향상 알고리즘을 제안한다.

첫 번째 Cumulative Ack을 사용하는 경우에 Expected 세그먼트의 Sequence number의 정보를 Ack 실어 송신측으로 보낸다. 이 경우 에러에 의한 세그먼트 손실과 Congestion에 의한 세그먼트손실을 구분할 수 없게 된다. 따라서 송신측은 Expected Sequence Number의 세그먼트를 Congestion Control알고리즘에 적용시켜 보내게 된다. 여기서 제안하는 ENA알고리즘에서는 ACK에 bit를 추가하여 구성해봤다. 즉 이 알고리즘에서 ACK은 Expected Sequence Number의 세그먼트와 바로 전에 수신측에서 받은 세그먼트가 에러인지 아닌지 확인하는 Bit의 정보, 에러인 세그먼트의 Sequence Number를 가지고 있다. 그리고 이 ACK을 받은 송신측은 에러 Bit가 Setting되어 있다면 에러 세그먼트 먼저 전송하게 된다. 따라서 Congestion이 발생하였을 때처럼 전송률을 줄이지 않기 때문에 성능이 향상되는 것이다. 단, 세그먼트의 에러는 Bulk로 발생하지 않는다는 가정과 재전송 되는 에러 세그먼트는 다시 에러가 나지 않는다는 가정하에서이다.

두 번째 Selective ACK을 사용하는 SACK에서의 성능향상알고리즘이다. SACK Option Format은 제대로 받은 세그먼트 정보와 손실된 정보를 포함하고 있기 때문에 하나의 RTT동안 여러 개의 손실된 세그먼트를 재전송함으로써 성능을 향상시킬 수 있는 것이다. 만약 그런 정보를 표현할 수 있는 크기가 40 Bytes면 N blocks의 길이 = 8×N +2 Bytes의 식에 의해 최고 4개의 block까지 포함할

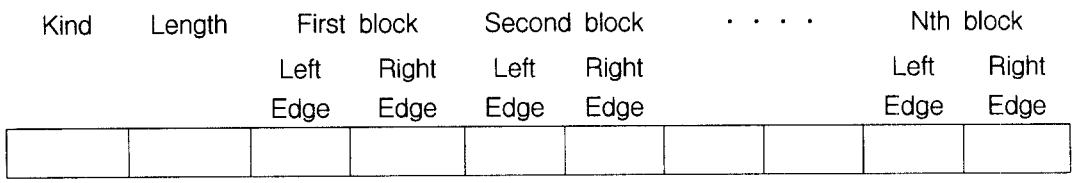

표 1. SACK Option Format

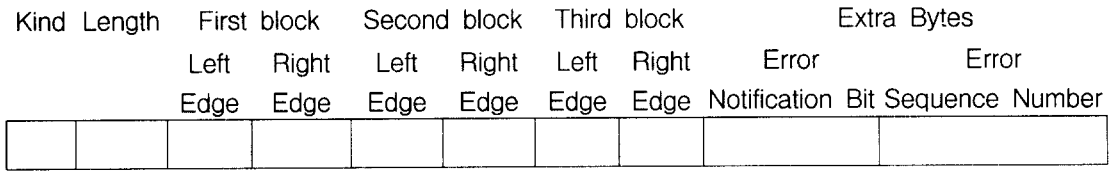

표 2. ENA를 적용한 SACK Option Format

수 있다. Kind는 5이고 Length는 가변적이다. 따라서 SACK option Format은 표1과 같다[9].

따라서 Ack은 Expected Number와 N block의 정보를 갖고 있다. 따라서 새로운 알고리즘에서는 Ack의 Format에 다음과 같은 정보를 포함한다. Expected Number와 손실된 세그먼트와 제대로 수신된 세그먼트의 정보를 포함하고 있는 3개의 Block과 세그먼트가 도착했을 때 에러인가 아닌가를 표현해 주는 Bit와 에러인 해당 세그먼트번호를 갖고 있도록 구성했다.

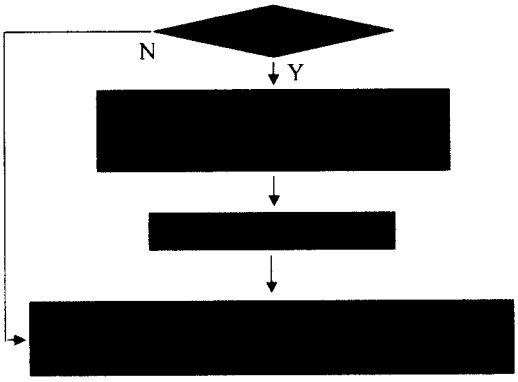

그림 1. 수신측 알고리즘

수신측은 *Errorhappen*와 *Esequence*라는 variable을 사용한다. 수신측은 우선 Error Detection방법으로 세그먼트가 에러인지 아닌지를 검사한다. 그리고 세그먼트가 에러이면 *Errorhappen*을 1로 설정한다.

그리고 에러가 난 해당 세그먼트의 Sequence Number를 *Esequence*에 입력한다.

따라서 이 Ack의 정보를 받은 송신측은 에러를 표현해 주는 Error Notification Bit를 우선 조사한다. 수신측이 에러 세그먼트를 받았다는 정보를 송신측이 확인하면 Error Sequence Number의 정보에 따라 에러상태의 세그먼트 먼저 재전송하고 그리고 *cwnd*가 available한 경우 Congestion Control 알고리즘에 의해 나머지 세그먼트들이 전송하게 된다. 대신 Congestion이 발생했을 때처럼 전송률을 줄이지 않는다. 따라서 그에 따른 성능저하를 막을 수가 있는 것이다. 그림 1과 2는 Sack에서 송신측과 수신측이 수행하는 알고리즘을 순서도로 표현한 것이다.

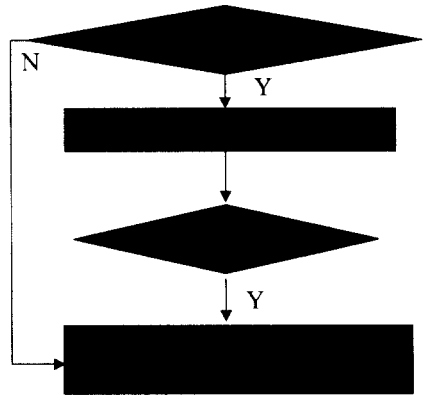

그림 2. 송신측 알고리즘

따라서 이 Ack의 정보를 받은 송신측은 에러를

표현해 주는 Bit를 우선 조사한다. 수신측이 에러 세그먼트를 받았다는 정보를 송신측이 확인하면 Ack정보에 있는 에러상태의 세그먼트 먼저 재전송하고 그리고 *cwnd*가 available한 경우 Congestion Control알고리즘에 의해 나머지 세그먼트들이 전송하게 된다. 대신 Congestion이 발생했을 때처럼 전송률을 줄이지 않는다. 따라서 그에 따른 성능저하를 막을 수가 있는 것이다. SACK도 Tahoe, Reno에서의 가정사항을 따른다.

V. 시뮬레이션을 통한 성능평가

위성망에서의 TCP 성능을 분석하기 위해 시뮬레이션 전용언어인 Simscript II.5를 사용했다. 그림3은 성능평가를 위한 Model이다.

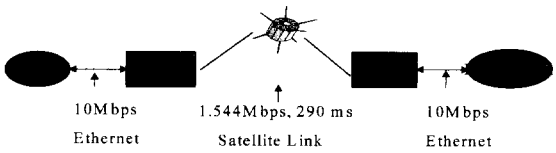

그림 3. 시뮬레이션 모델

지상망으로는 10Mbps의 Ethernet으로, 위성링크는 T1의 대역폭으로 설정하였다. 그리고 위성링크의 전파지연시간은 290 ms로 잡고 Ethernet의 전파지연시간은 위성전파지연시간에 비해 이주 작기 때문에 없는 걸로 가정했다.

표 3. 시뮬레이션 파라미터

파라미터	값
TCP Segment	512 Bytes
Satellite Router Buffer Size	100
Maximum window Size	64 KBytes, 110 KBytes
Minimum Slow-Start threshold	1

표3은 기본적인 TCP의 파라미터들이다. 위성의 링크손실은 $10^{-5}, 10^{-6}, 10^{-7}$의 세 가지 경우를 시뮬레이션 했다. 그리고 ACK의 Size는 SACK TCP의 경우 30 Bytes로 잡고 3개의 Block정보를 가지면서 ENA알고리즘을 갖는 경우에 대해 시뮬레이션 했다. Test File Size는 100K에서 10MB까지 log 스타이로 증가한다. 그리고 Maximum Window Size은 표준 TCP에서 사용되는 64Kbytes와 Window Scale Option을 사용한 110Kbytes의 두 가지 경우에 대해 실험했다. 송신측에서 위성으로 가는 라우터를 기준으로 받는 대역폭과 보내는 대역폭을 다르게 설정하고 유한개의 Buffer를 설정함으로써 Congestion이 발생하도록 Modeling하였다.

그리고 이 논문에서의 성능평가시 가정사항은 다음과 같다. 위성의 전파지연시간은 290 ms로 잡고 Ethernet의 전파지연시간은 위성전파지연시간에 비해 무시할 정도로 작기 때문에 없는걸로 가정한다. 그리고 에러발생은 Bulk로 발생하지 않으며 에러인 세그먼트는 반드시 수신측에 도달한다는 가정한다. 또 재전송한 세그먼트는 다시 손실되지 않는다고 가정한다. 그리고 송신측에는 Ack을 받았을 때 전송해야 될 세그먼트가 항상 존재한다고 가정한다.

T1의 위성링크속도, 110 KBytes의 Large Window환경에서 에러가 없는 경우와 10^{-7}의 에러율을 갖는 환경에서 Tahoe, Reno와 Sack의 성능을 비교해 봤다. 그림 4, 5의 결과처럼 Sack의 성능이 Tahoe, Reno보다 훨씬 좋게 나타남을 알 수 있다. Sack은 다수의 손실된 세그먼트의 정보를 가지고 있기 때문에 전파지연이 긴 위성환경에서 좋은 성능을 갖게 되는 것이다.

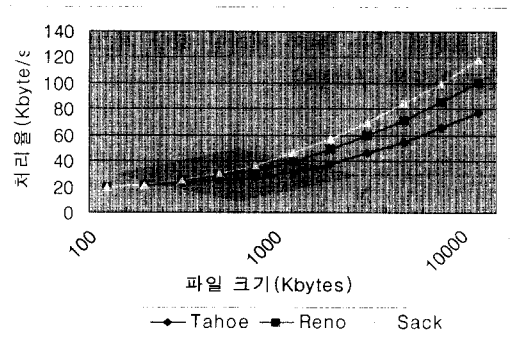

그림 4. Tahoe, Reno, Sack의 처리율비교(Without Error)

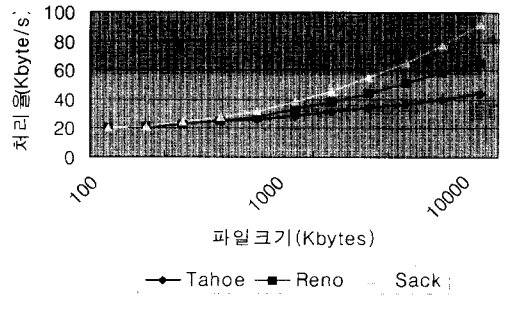

그림 5. Tahoe, Reno, Sack의 처리율비교(with Error)

본 논문에서는 Tahoe, Reno의 경우와 Sack의 경우 각각의 성능향상 알고리즘을 제안했다. 따라서 시뮬레이션도 두 가지 경우로 나누어서 수행했다. 그림6는 Tahoe와 Reno에서 ENA 알고리즘을 적용한 경우와 하지 않은 경우의 처리율을 비교한 것이다. 각각의 Version에 대해 ENA 알고리즘을 적용했을 때 성능이 향상됨을 알 수 있다.

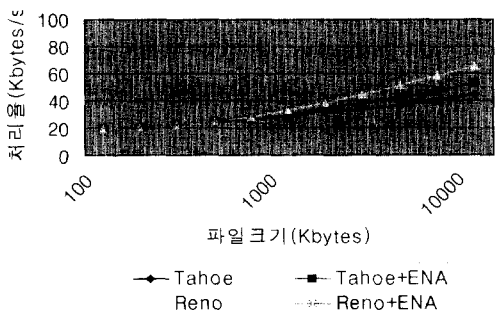

그림 6. ENA를 적용한 Tahoe, Reno의 처리율

그림 7은 위성환경에서 좋은 성능을 나타내는 Sack에 ENA알고리즘을 적용한 경우와 그렇지 않은 경우에 대해 성능을 평가했다. Sack의 경우도 Tahoe, Reno와 마찬가지로 ENA 알고리즘을 적용하는게 좀 더 나은 성능을 얻을 수 있다.

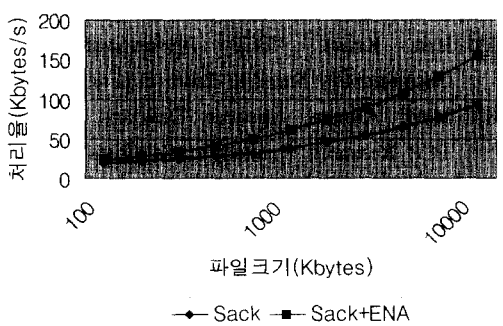

그림 7. ENA를 적용한 Sack의 처리율

그림8은 TCP성능을 향상시키는 방법으로 제안된 Window Scaling의 성능을 평가한 것이다. T1의 위성링크속도, Window Scaling Option의 적용유무에 따른 성능평가이다. BER은 10^{-7}이다. 위성과 같이 RTT가 큰 경우에는 Window Size를 크게 해줌으로써 성능을 향상시킬 수 있다. TCP 표준에 규정된 64KBytes일때와 T1의 최대 전송률을 사용할 수 있는 110KByte의 두 가지 경우에 대해 실험했다. 그 결과 110KBytes의 경우 좋은 성능을 나타내고 있다.

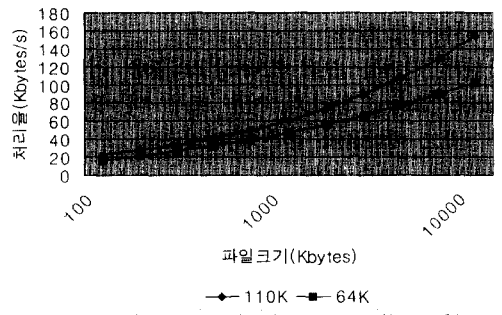

그림 8. Window Scale Option 적용유무에 따른 처리율

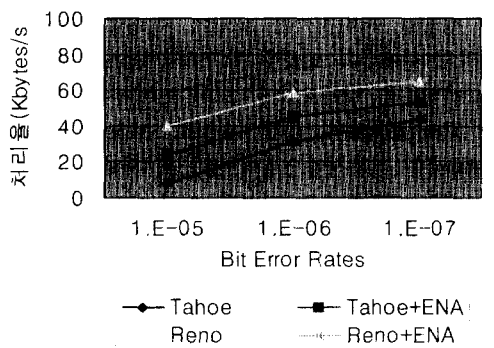

그림 9. 에러율에 따른 Tahoe, Reno의 처리율

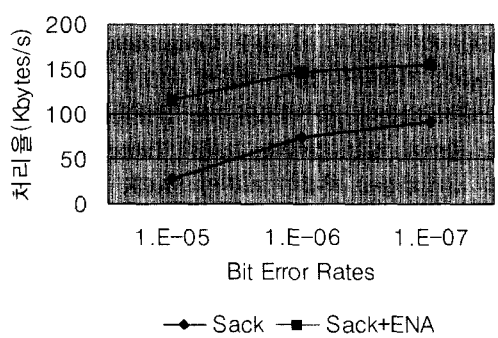

그림 10. 에러율에 따른 Sack의 처리율

그림 9는 Tahoe와 Reno에서 에러율을 변화시켜 가면서 ENA를 적용한 경우와 적용하지 않은 경우의 처리율을 비교한 것이다. 에러율이 높을수록 ENA를 적용했을 때와 그렇지 않았을 때의 처리율이 크게 차이가 남을 볼 수 있다. 즉, 에러율이 높은 경우일수록 ENA알고리즘을 적용시키는게 좀 더

나은 성능을 얻을 수 있다.

그림10은 에러율의 변화에 따라 ENA를 적용한 경우와 그렇지 않은 경우의 Sack을 시뮬레이션 한 결과이다. 마찬가지로 에러율이 높을수록 좋은 성능을 나타내고 있다.

위의 결과를 통해 ENA알고리즘을 적용했을 때 Tahoe, Reno, Sack의 경우 모두 다 좋은 성능을 나타내고 에러율이 높은 환경일수록 그 성능은 더 향상됨을 알 수 있다.

Ⅵ. 결 론

위성링크를 사용한 인터넷의 확장으로 위성망에서의 TCP 성능향상을 위한 방안들이 중요한 문제로 떠오르기 시작했다. 우선 위성망의 특징 및 TCP의 성능을 저하시키는 요인에 대해서 알아보았다. 그리고 현재까지 연구된 TCP성능향상을 위해 제시된 기법들을 알아보았다. 그러나 에러에 의한 손실을 Congestion에 의한 손실로 보고 Congestion Control을 적용시킴으로서 성능을 저하시키는 방법에 대한 해결책으로 제시된 바가 없다. 이 논문은 이 부분에 대한 해결책으로 SACK Option Format에 수정했다. 즉 3개의 Block 정보를 포함하고 나머지 Bytes에는 에러를 알리는 bit와 해당 세그먼트의 Sequence Number를 나타내는 Bytes로 구성했다. ACK을 받은 송신측은 에러에 의한 손실인지 아닌지를 판단하여 에러에 의한 세그먼트 먼저 재전송하는 ENA 알고리즘을 제안하고 그 구현방법 또한 제시하였다. 이 경우 Congestion Control알고리즘이 적용되지 않기 때문에 윈도우 사이즈는 감소하지 않는다. 즉 네트워크로 전송되는 데이터 양을 줄이지 않는 것이다. 그래서 Tahoe, Reno TCP와 위성환경에 적절한 기존의 SACK TCP에 ENA 알고리즘을 적용시켜 시뮬레이션했다. 우선 Tahoe와 Reno의 경우 ENA 알고리즘을 적용하여 File Size의 변화에 따라, BER의 변화에 따라 성능을 평가했다. 그리고 위성환경에 좋은 성능을 나타내는 SACK의 경우도 마찬가지로 수행했다.

시뮬레이션 결과 Tahoe, Reno, Sack 모두의 경우 ENA 알고리즘을 적용시켰을 때 성능이 보다 향상됨을 알 수 있었다. 그리고 에러율이 높은 경우일수록 성능이 더 향상되어 나타났다.

참 고 문 헌

[1] Aldar C.F.Chan, Danny.H.K.Tsang, Sanjay Gupta, "TCP Over Wireless Links", 97 VTC.

[2] K.Fall, S.Flyod, "simulation-based Comparisions of Tahoe, Reno and SACK TCP", Computer Communications Review, July 1996.

[3] Mark Allman, Chris Hayes, Shawn Ostermann, "An Evaluation of TCP With Larger Initial Windows", ACM Computer Communication Review, July 1998.

[4] Craig Partridge and Tim Shepard: "TCP Performance over Satellite Links." IEEE Network, 11(5), pp44-49, September/October 1997.

[5] Allman, Mark, Chris Hayes, Hans Kruse, and Shawn Ostermann: "TCP Performance Over Satellite Links", In Proceedings of the 5th International Conference on Telecommunication Systems, March 1997.

[6] T.V.Lakshman, Upamanyu Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", IEEE/ACM TRANSACTIONS ON Networking. Vol.5, No.3, June 1997.

[7] Han Kruse, "TCP Performance in a Geostationary Satellite Environment.".

[8] Mark Allman, Spencer Dawkins, "Ongoing TCP Research Related to Satellites", INTERNET DRAFT, March 1999.

[9] Mathis, J. Mahdavi, Sally Floyd, and Allyn Romanow: "TCP Selective Acknowledgement Options," RFC 2018, October 1996.

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

[11] Vern Paxson, "Empirically-Derived Analytic Model of Wide-Area TCP Connections", IEEE/ACM Transaction on Networking, pp 316-336, August 1994.

[12] Aldar C.F.Chan, Danny.H.K.Tsang, Sanjay Gupta, "Performance Analysis of TCP in the Presence of Random Losses/Errors", Globecom

98.

[13] Mark Allman, Dan Flover, "Ehancing TCP Over Satellite Channels using Standard Mechanisim", RFC 2488, January 1999.

[14] Van Jacobson, "Congestion avoidance and control", ACM SIGCOMM 88, 1998.

[15] S.Floyd, "Issues of TCP with SACK", Technical report, Mar 1996.

[16] Jacobson, V.R.Fraden and D.Borman: "TCP Extensions for high Performance, RFC 1323, February 1991.

[17] Bikram S.Bakshi, P.Krishna, "Improving Performance of TCP over Wireless Networks", Technical Report, May 1996.

이 정 규(Jong-kyu Lee)

1979년 2월 : 한양대학교 전자공학과 학사

1986년 : UCLA 전자공학과 석사

1989년 : UCLA 전자공학과 박사(컴퓨터 네트워크 전공)

1979년 3월 ~ 1984년 5월 : 국방과학연구소 연구원

1989년 3월 ~ 1990년 2월 : 삼성전자 종합기술원 정보통신부문 수석 연구원

1990년 3월 ~ 현재 : 한양대학교 전자계산학과 부교수

<주관심 분야> 무선데이타통신, 이동통신, 보안 프로토콜

김 상 희(Sang-hee Kim)

1997년 2월 : 덕성여자대학교 전자계산학과 학사

2000년 2월 : 한양대학교 전자계산학과 석사

<주관심 분야> 위성통신, TCP/IP, 네트워크 성능분석