무선링크 에러에 강인한 TCP 혼잡 알고리즘

정회원 박 홍 성*, 전 선 국**, 윤 건**

Robust TCP Congestion Algorithm over Lossy Wireless Links

Hong Seong Park*, Sheon-Kook Jeon**, Gun Yoon** *Regular Members*

요 약

본 논문은 TCP-Reno등의 TCP 등에서 사용되는 다른 혼잡 제어 알고리즘보다 에러가 많은 무선 환경에 더욱 강인한 향상된 TCP 혼잡 제어 알고리즘을 제안한다. 제안하는 알고리즘은 혼잡 윈도우 크기를 패킷 에러율과 현 TCP의 상태가 fast recovery 혹은 slow start 상태에 있느냐에 따라 결정한다. 시뮬레이션을 통해 제안하는 알고리즘의 유효성을 보여 주며, 여러 TCP 혼잡 제어 알고리즘들과 혼잡 윈도우 크기와 효율과 같은 성능 지표 관점에서 비교하였다. 제안하는 알고리즘은 다른 알고리즘보다 높은 PER하에서는 높은 효율을 가지며, 낮은 PER하에서는 비슷한 효율을 가진다.

Key Words : Congestion Algorithm; lossy wireless links.

ABSTRACT

This paper suggests an improved TCP congestion algorithm, which is more robust to lossy wireless environment than other algorithms such as TCP-Reno. The suggested algorithm decides on the size of a congestion window depending on both PER (Packet Error Rate) and its state, which is one of fast recovery state and slow start state. Some simulations are given to validate the suggested algorithm and the algorithm is compared with other TCP congestion algorithm from the point of view of performance measures such as a congestion window and throughput. The suggested algorithm has better throughput than other algorithm over wireless links with high PER and similar throughput to others over wireless links with low BER.

Ⅰ. 서 론

최근 무선 통신 기술과 인터넷 기술의 발전에 따라 무선 인터넷은 인터넷 검색, E-mail, 전자상거래 같은 대부분 PC기반의 서비스들도 가능하게 되었고 앞으로 멀티미디어 서비스로 확대될 전망이다. 하지만 무선 인터넷 서비스의 가장 큰 문제점은 유선 매체와 달리 무선 채널은 사용대역폭에 제한이 많고 전송시 에러 생길 확률이 높다는 것이다.

TCP는 현재 인터넷에서 사용되는 일반적인 전송 프로토콜이며, 동적으로 윈도우의 크기를 변화시키는 혼잡제어 알고리즘을 수행하며 전송 패킷 에러율(packet error rate)이 낮은 유선 통신매체를 대상으로 설계되었다. 따라서 데이터 전송시 발생한 모든 오류는 통신망의 과부하로 인해 발생하는 것으로 생각한다. 실제적으로 무선망에서는 잡음, 페이딩, 간섭 등에 의한 높은 비트오류율(Bit Error Rate)과 단말기 이동에 의한 핸드오프 등 여러가지 요인에 의해 패킷 손실 혹은 패킷 에러가 발생할 수 있다. 즉, 무선환경에서는 과부하가 아닌 패킷 에러로 인해 혼잡제어 알고리즘이 수행되어 TCP 윈도우 크기를 조정하여 성능 저하를 초래할 수 있다. 무선 환경에서 TCP를 사용할 때 생기는 이런 문제점에 대한 효과적인 대안으로 무선환경에 적합

* 강원대학교 전기전자정보통신공학부(hspark@kangwon.ac.kr), ** 강원대학교 제어계측공학과
논문번호 : 020340-0805, 접수일자 : 2002년 8월 6일

한 트랜스포트 계층 제어 방식들이 연구되어 왔다[1,2,3].

현재까지 무선 환경에서 생기는 높은 에러률을 제어하여, 보다 성능이 우수한 TCP 혼잡 제어 알고리즘(congestion control algorithm)들이 제안되었고 관련 연구도 계속되고 있다[3-12]. 무선 환경에서 TCP의 성능 향상을 위해 알려진 기법들은 주로 링크계층[4-7]과 트랜스포트계층 프로토콜[8-12]로 나눌 수 있다.

링크계층과 관련된 방법에는 CDMA 및 TDMA를 이용하는 기존의 셀룰라 시스템의 링크계층에서 주로 사용하는 ARQ(Automatic Repeater reQuest)와 FEC(Forward Error Correction)와 같은 에러 정정 방법 및 ARQ와 FEC기술을 결합한 AIRMAIL[4]과 같은 프로토콜들이 있다. 그외에도 Snoop프로토콜[5], ELN(Explicit Loss Notification)기법[6] 등이 있다. 하지만 이런 링크계층 기법들은 에러 발생율이 적을 경우에는 송신 호스트의 TCP의 트랜스포트계층 프로토콜과 링크계층구간의 역효과가 발생하여 성능이 저하될 가능성도 있다[7]. 그리고 오버헤드가 크고 처리지연시간이 크다는 단점도 가지고 있다.

트랜스포트계층 관련된 방법들은 기지국의 트랜트포트 계층에서 연동하여 유무선망을 통합하려는 방식이며 주로 무선구간의 핸드오프나 패킷 손실에 대한 대안으로 제시되었다. 이들 방법들에는 Congestion or Corruption 방법[8], ITCP (Indirect-TCP) 프로토콜[9], SACK (Selective Acknowledgement)[10], SMART(Simple Method to Aid Retransmission)[11], 기존의 혼잡 제어 알고리즘을 수정한 방법[12] 등 이 있다.

유무선 혼합망에서 유선구간과 무선구간을 분류하여 서비스하는 I-TCP같은 Split-Connection프로토콜들은 기존의 양극단 TCP 프로토콜의 의미를 훼손한다는 단점이 있고 SACK같은 트랜스포트계층의 프로토콜들은 우선 수신측에서 보내는 ACK의 크기가 커지고 수신측에 많은 부하를 준다는 단점을 가지고 있다. 앞에서 살펴본 기존 방법들은 유무선 환경이 통합된 환경에서 일반적이고 호환성이 있으면서 동시에 성능을 향상 시키지 못하고 있다. 기존의 방법들중 부분적으로 효율적인 성능개선을 이룬 알고리즘들도 있지만 적용환경이 극히 제한되었고 특수화 되어 있기에 일반적이지 못하고 실용성이 많이 떨어진다. [8]-[11]에서 제시된 방법은 오버헤드가 많고 표준 TCP에서 사용되는 기본 규정들을 훼손하는 단점이 있다. 그러나 [12]에서 제안하는 방법은 이러한 단점들 없이 TCP의 혼잡제어 알고리즘의 slow start 및 fast recovery부분을 수정하여 성능을 향상시킨 것이다. 하지만 [12]에서는 혼잡 윈도우(Congestion window)의 크기의 전송채널의 상태를 고려하지 않고 무조건적으로 손실된 패킷량만큼 보정했다. 하지만 실제 전송채널상의 패킷손실이 큰 경우에는 오히려 보정을 이루는 것이 패킷손실률을 증가시키는 문제를 발생한다. 본 논문은 이런 문제점을 해결하는 새로운 기법을 제안하였다.

본 논문에서는 무선 환경에서 보다 잘 적응되도록 slow start 상태 혹은 fast recovery 상태에 따라 PER에 따른 보정 계수를 달리 하여 혼잡 윈도우 크기를 결정하는 방법을 제안하였다. 이 방법은 slow start 과정과fast recovery 과정은 서로 다른 원인에 의해 수행되기 때문이다.

본 논문의 구성은 다음과 같다. 2절에서는Reno 방법, [12]에서 제안하는 방법을 간략히 설명하고, 제안하는 새로운 방법(Error-Robust) 알고리즘이라 칭함)에 대해 설명한다. 3절에서는 시뮬레이션을 통해 무선 링크상에 다양한 에러를 발생시켜 Reno, SACK, [12]의 방법 등 및 본 논문에서 제안하는 방법들에 대해 윈도우 크기 변화와 효율(throughput)을 비교하고, 마지막으로 결론을 제시한다.

Ⅱ. ER (Error-Robust) 알고리즘

기존의 TCP알고리즘은 전송과정에 패킷손실이 발생하게 되면 송신측의 TCP계층은 이를 혼잡(Congestion)에 의한 에러로 간주하고 혼잡 윈도우(Congestion window)크기를 적절한 알고리즘에 따라 줄이지만 무선 링크구간이 있을 경우 이런 방법은 적합하지 않다. 그 원인은 무선환경에서는 이동성에 의하여 채널 환경이 수시로 변하고 이동성이 없더라도 주위 환경의 변화에 따라 무선 링크가 매우 불안정하여 패킷 에러 혹은 손실이 발생할 수 있기 때문이다. 무선 링크의 특성상 많은 패킷 손실이 발생할 경우에는 데이터 링크계층에서 프레임의 계속적인 재전송이 일어나게 되고 이는 TCP계층에서 재전송 타임아웃을 빈번하게 발생시킨다. 이 경우에는 혼잡이 아님에도 불구하고 TCP는 혼잡 윈도우 크기를 줄이게 된다. 또한 이런 패킷 손실이 연속적으로 반복하게 될 경우에는 혼잡 윈도우 크

기도 연쇄적으로 줄이게 된다.

현재 널리 쓰이고 있는 TCP-Reno의 경우에는 혼잡 윈도우 크기는 slow start, fast recovery의 알고리즘을 통해 변화한다. 노드는 새로운 TCP 세그먼트 1개 크기로 초기화된다. 그리고 Ack가 도착할 때마다 윈도우 크기를 1개의 세그먼트 크기만큼 늘린다. 이를 slow start라고 한다. 혼잡이 일어날 경우, 윈도우 크기를 반으로 줄이고, 매 라운드 트립 시간마다 1개의 세그먼트 만큼 늘리는 혼잡 회피(Congestion avoidance) 알고리즘을 수행한다. 송신부가 혼잡을 판단하는 기준은 재전송 기준시간 초과와 중복 Ack (duplicate Ack)의 수신의 두 가지 방법이 있다. 중복 Ack가 수신될 경우는 slow start 알고리즘을 실행하지 않고, 혼잡 회피 알고리즘을 수행하게 되는데 이를 fast recovery라고 한다[13]. TCP-Reno에서 fast recovery과정 혹은 slow start과정이 실행될 때 혼잡 윈도우의 크기 변화는 그림 1과 같다.

```
if { Fast  Recovery } then {
  Cwnd = halfwin
}
if { Slow  Start} then {
  Cwnd=int(wnd_restart)
}
```

그림 1. TCP-Reno 혼잡 제어 알고리즘
Fig. 1 TCP-Reno Congestion Control Algorithm

그림 1에서 Cwnd는 채널 환경을 반영한 혼잡 윈도우 크기 값이고, halfwin는 기존의 혼잡제어 알고리즘에 의해 산출되는 윈도우의 크기 값, 즉 패킷 손실이 반영되었을 때의 윈도우 크기의 절반값이고 int(wnd_restart)값은 slow start 과정이 발생할때의 혼잡 윈도우 초기화 값이다.

[12]에서 제시한 혼잡 제어 알고리즘은 무선 링크상의 패킷 손실이 BER에 의한 패킷 에러에 기인하여 발생하기 때문에 BER혹은 PER를 고려하여 혼잡 윈도우 크기를 계산하였다. 본 논문에서는 이 알고리즘을 단순 BER 혼잡제어 알고리즘이라 한다.

```
if { Fast  Recovery } then {
  Cwnd = halfwin + f(SNR_AV)*win
}
if { Slow Start } then {
  Cwnd = int(wnd_restart) + f(SNR_AV)*win
}
```

그림 2. 단순 BER 혼잡 제어 알고리즘
Fig. 2 Simple BER Congstion Control Algorithm

$$f(SNR_{AV}) = 1 - S\left(\frac{1}{2}\sqrt{1-\left(\frac{SNR_{AV}}{1+SNR_{AV}}\right)}\right) \quad (1)$$

여기서, S( )는 평균 BER에 따른 효율 값을 의미하고 S( )의 인자는 평균 SNR에 따른 평균적인 BER값이다.

지금부터는 ER(Error-Robust) 혼잡 제어 알고리즘을 제시한다. 제시하는 ER 알고리즘은 무선 채널 환경의 악화에 따른 TCP성능저하를 최대한 줄이기 위해 혼잡 윈도우 크기에 현재의 채널 환경을 반영한 알고리즘이다. 제안하는 방법은 fast recovery 과정과 slow start과정의 두 경우에 대해 서로 다르게 혼잡 윈도우 크기를 보정하는 방법을 사용한다. 즉 fast recovery경우에는 상대적으로 패킷손실을 적게 발생하고 손실된 패킷을 재전송 한 다음 차츰 정상 상태를 복귀할 가능성이 크기때문에 PER값에 대한 윈도우의 보정을 많이 하기 위해 fast recovery 과정에서는 $(1-PER)$의 가중치를 사용하여 보정한다. 또한 채널 상태가 나쁠 경우는 전송데이타를 줄이기 위해 보정 윈도우 값도 함께 적당히 줄인다. 반면에 slow start 단계는 채널 상태가 좋지 않아 PER값이 크다는 점을 고려하여서 PER값에따라 보정 윈도우의 값을 적절하게 조절한다. 즉 TCP의 기능상전송과정에 패킷이 손실되면 송신측 TCP는 수신측으로부터 중복 Ack를 받게 되고 다시 손실된 패킷들을 재전송하게 되는데 재전송이 실패하게 되면 타임아웃(timeout)이 발생한다. 처음 타임아웃이 되면서 slow start과정에서 전송에 성공하는 경우도 있지만 어떤 경우에는 여러 번 타임아웃이 발생한 후에 성공하거나 실패하게 된다. 따라서, 제안하는 ER 알고리즘에서는 패킷전송 실패를 하면 다시 수신측으로부터 중복 Ack를 받고 다시 손실된 패킷부터 재전송을 하게 되는데 다시 재전송을 실패하게 되면 손실된 패킷은 Slow start과정에서 전송된다. 즉, slow start 과정은 환경이 보다 나쁜 상황으로 갈 수 있기 때문에 PER의 크기에 따라 보정하는 가중치를 달리하여야 한다. 이러한 가중치를 달리하는 경계값을 결정하기 위해 가중치가 PER^2인 경우와 PER인 경우에서 효율(throughput)을 비교하기 위해 시뮬레이션 결과를 그림 3과 그림 4에 표시하였다.

그림 3과 그림 4로부터 PER이 0.5근방을 경계로 하여 성능에서 차이가 생김을 알수 있다. 즉, PER

이 0.5보다 작을 경우, 가중치에 PER^2 값을 적용한 것이 PER 값만을 적용한 것보다 효율이 향상됨을 보여주었다. 그러나, PER이 0.5보다 클 경우에는 그와 반대되는 효과가 나타남을 알 수 있다. 경계점은 0.45-0.5사이에 있지만 이들 값들이 가지는 효율들은 매우 비슷한 값을 가진다는 것을 알 수 있다. 따라서, 본 논문에서는 알고리즘의 단순성을 위해서 경계치를 0.5로 가정한다. 즉, 제안하는 방법의 slow start과정에서는 PER=0.5를 경계로 하여 서로 다른 보정용 가중치를 적용시킨다.

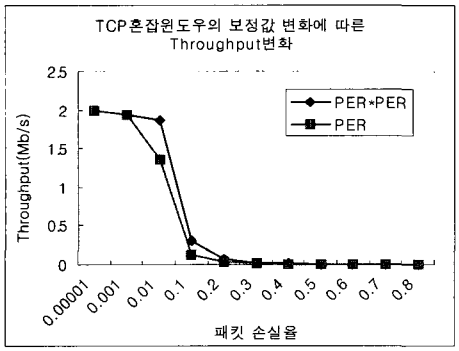


그림 3. 보정 가중치 변화에 따른 효율 변화
Fig. 3 Throughput as a function of PER for two compensation weighting factors

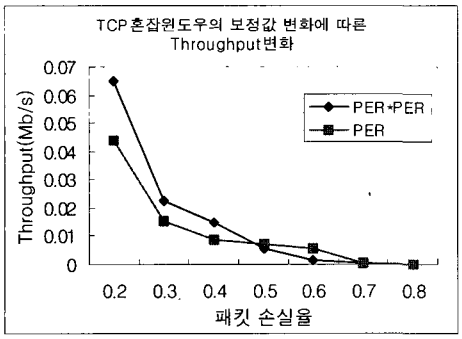


그림 4. 그림3의 패킷손실율 0.2 ~ 0.8 확대 구간
Fig. 4 enlarged region for PER 0.2 ~ 0.8 in Fig. 3

일반 유선 환경에서와 같이 패킷 에러가 아주 적은 경우($PER \le 10^{-3}$)에 혼잡이 발생하더라도 TCP-Reno에 적용한 혼잡 제어 알고리즘을 따르도록 하여 패킷손실에 유연하게 대처하도록 하는 것이 유무선상에 공통적으로 사용할수 있게 하는 것이 중요하다. 이를 위해 slow start 과정에서 사용되는 보정용 가중치를 PER^2 혹은 PER를 사용하여 유선 환경과 같이 PER이 적은 환경에서는 기존 혼잡 제어 알고리즘과 같이 동작하게 하였다. 참고로 [12]에서 제안한 단순 BER 혼잡 제어 알고리즘은 PER값이 작을경우에는 거의 보정이이루지 않는 단점도 있다. 그러나, 제안하는 그림5의 ER 알고리즘은 이런 문제점도 해결하였다.

```
if { Fast  Recovery } then {
  Cwnd = halfwin + PER * win
}
if { Slow  Start } then {
  if( PER < 0.5) {
    Cwnd=int(wnd_restart) + PER^2 * win
  }
    Cwnd=int(wnd_restart) + PER * win
}
```

그림 5. ER 혼잡 제어 알고리즘
Fig. 5 ER Congestion Control Algorithm

제안하는 방법의 특징은 PER이 클 때는 fast recovery과정에 윈도우보정을 적게 하고 채널상태가 좋아질 경우에는 윈도우의 크기를 상대적으로 많은 보정을 이룰 수 있다. slow start 단계에서 손실된 패킷값만큼 보내지 않고 채널 상태가 나쁜 상황을 고려하여서 에러에 적절한 값으로 조절하여 보정을 적용한다.

Ⅲ. 시뮬레이션 및 성능분석

본 논문에서 시뮬레이션은 아래와 같은 가정에서 이루어졌다. 순수한 무선구간에 채널상의 링크에러만 고려하기 위하여 유무선 혼합망에서 같이 무선 연결구간은 1 홉(hop)으로 구성되고 혼잡이 발생되지 않는 무선 네트워크를 가정한다. 그리고 TCP/IP 프로토콜 스택내에는 패킷 에러/손실이 없고 패킷 에러/손실은 오직 무선 채널 에러만 의해 발생한다고 가정한다.

시뮬레이션은 LBNL(Lawrence Berkeley National Laboratory) ns-2.1b8a버전을 사용하였다. 전송 대역폭은 2Mbps, 무선 전파지연은 10ms로 설정하였고 송신측에서 FTP프로그램을 사용한다. 전송 패킷의 크기는 1 Kbyte로 설정한다. 적용한 에러모델은 패킷에러가 시간에 따라 burst하게 분포되는 경우를 고려하지 않고 균일하게(Uniform) 분포되어 있다고 가정하였다. 또한 채널 모니터(Channel monitor)는 송신측에서 사용한다. 채널 모니터는 자신의 서비스 영역 내에서 패킷 손실율을 감지한다고 가정한다.

1. TCP 혼잡 윈도우의 크기 비교

Reno, SACK, 단순 BER 알고리즘, ER 알고리즘, new Reno, Tahoe에서 사용하는 혼잡제어알고리즘들을 시뮬레이션에 적용하였을 때 다양한 PER에 따른 혼잡 윈도우 크기의 변화를 살펴본다. 그림 6 - 그림 9에는 그림의 복잡성 때문에 Reno, SACK, 단순 BER 알고리즘, ER 알고리즘들 사용하여 혼잡 윈도 크기의 변화를 살펴보며, 표 1에서는 다양한 PER에 대해 Reno, SACK, 단순 BER 알고리즘, ER 알고리즘, new Reno, Tahoe들의 평균 혼잡 윈도우 크기를 비교하였다.

그림6과 그림7에서는, 즉 $PER \le 10^{-3}$일 때는 ER 기법이 성능이 조금 더 좋은 것을 볼 수 있다. 즉 PER값이 적을 경우에는 혼잡윈도우에 대한 보정이 별로 효과가 없는 것을 알수 있다. $PER > 10^{-3}$인 경우에는 ER알고리즘은 혼잡 윈도우 크기 보정에 효과가 있지만 단순 BER 알고리즘은 Reno와 거의 비슷한 결과를 얻는다.

그림6 - 그림9들로 부터 무선 링크 에러로 인한 패킷손실 값이 커짐에 따라 어떤 종류의 TCP를 막론하고 혼잡 윈도우 크기변화는 아주 불안정적으로 작아지는 변화를 관찰할 수 있다. 즉 무선 링크구간에 패킷손실 값이 커짐에 따라 송신측의 TCP 성능이 심각하게 저하된다는 것을 의미한다. 반면에 기존의 TCP들 중 SACK 알고리즘이 상대적으로 좋은 성능을 가지는 것을 볼 수 있었다. 특히 ER알고리즘의 성능이 기존의 어떤 프로토콜보다 성능이 좋은 것을 관찰할 수 있다.

여러 TCP들이 비교된 표 1을 살펴보면 PER이 낮은 환경이나 높은 환경 하에서도 제안하는 ER알고리즘의 성능이 우수함을 알 수 있다.

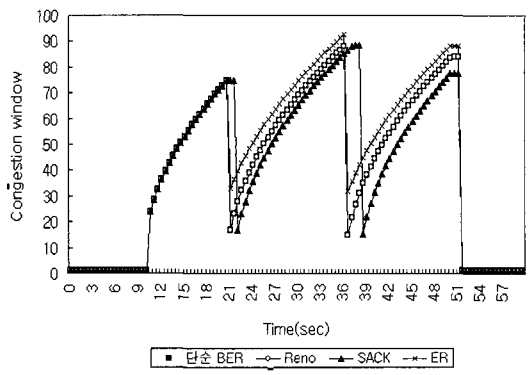


그림 6. $PER = 10^{-5}$일때 혼잡 윈도우 크기 변화
Fig. 6 Variation of Congestion Window size for

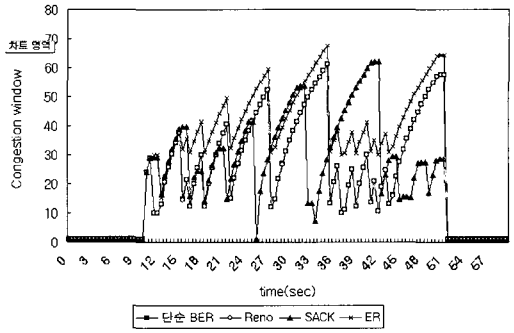


그림 7. $PER = 10^{-3}$ 일때 혼잡 윈도우 크기 변화
Fig. 7 Variation of Congestion Window size for $PER = 10^{-3}$

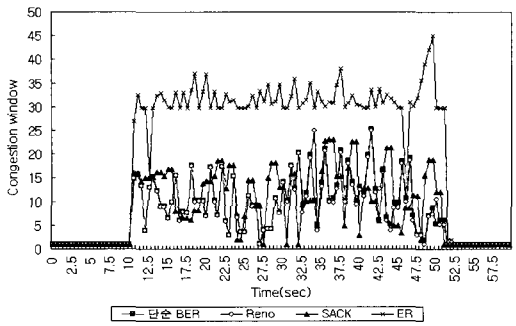


그림 8. $PER = 10^{-2}$ 일때 혼잡 윈도우 크기 변화
Fig. 8 Variation of Congestion Window size for $PER = 10^{-2}$

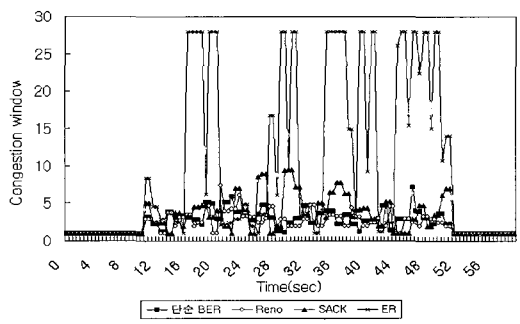


그림 9. $PER = 10^{-1}$ 일때 혼잡 윈도우 크기 변화
Fig. 9 Variation of Congestion Window size for $PER = 10^{-1}$

표 1. PER 변화에 따른 평균 혼잡윈도우 크기의 비교
Table 1. Comparison of Mean Congestion Window Sizes according to various PERs

패킷손실률	newreno	dd	reno	sack	ER	tahoe
0.00001	59.90184	59.89478	59.8947	59.8072	65.32461	59.61624
0.001	34.10799	32.45573	32.4492	33.08823	43.74345	33.12618
0.01	14.17116	11.4901	11.43289	13.21385	32.75611	10.91446
0.1	3.492003	3.764369	3.509857	4.659453	15.10569	3.166419
0.2	3.012549	3.092141	2.688702	2.945775	4.471242	2.604216
0.3	2.968866	3.02413	2.968866	3.013543	3.488173	2.961162

2. 효율의 비교

우선 시뮬레이션을 통해 링크 채널에러로 인한

패킷손실값이 TCP의 효율에 미치는 영향이 그림 7 그림 15에 표시되어 있다. 여기서 언급된 효율은 초당 전송 데이터 량을 의미한다. 이들 그림에서는 단순 BER 알고리즘과 본 논문에서 제안하는 ER 알고리즘을 적용하였을 때 나온 효율들을 비교한다.

그림 10 - 그림15로부터 패킷에러율이 커짐에 따라 패킷 손실이 증가하여 TCP 성능이 심각하게 감소되는 것을 관찰할 수 있었다. 그리고 그림7로부터 $PER \le 10^{-3}$인 경우에는 PER 혹은 BER이 매우 작기 때문에 비교되는 2 개의 알고리즘에 의한 보정 효과가 거의 나타나지 않아 비슷한 효율을 얻을 수 있음을 알 수 있다.

그림11 - 그림13으로부터, 즉 $10^{-2} \le PER \le 0.5$인 경우에는 제안하는 ER 알고리즘이 단순 BER 알고리즘보다 효율이 약 27.8 ~ 60.79% 정도 향상됨을 알 수 있었다. 이 경우에는 패킷 에러/손실로 인해 송신부의 전송 과정에는 fast recovery 과정 혹은 slow start과정이 다 포함되는데, ER 알고리즘이 보다 효과적이고 성능을 개선시키는 것을 알 수 있다.

그림 11로부터 $0.5 < PER < 0.8$에는 ER알고리즘과 단순 BER 알고리즘이 거의 비슷한 효율을 내는 것을 알 수 있다. 이 경우에는 송신부 전송과정에 slow start과정만 동작할 가능성만 있기 때문에 알고리즘을 보면 같은 혼잡 윈도우 크기를 가지기 때문에 같은 효율을 가지게 된다. 많은 패킷 손실이 일어나는 경우에는 기존의 어떤 다른 프로토콜를 사용하여도 똑같이 심각한 성능저하가 되는 것을 실제 시뮬레이션은 통해 알 수 있었다. 이런 경우에는 패킷 에러율를 줄이기 위한 방법들을 사용하여야 한다. 이러한 방법에 ARQ, FEC혹은 Hybrid ARQ같은 기법들이 전형적인 예이다.

그림 15로 부터 $PER \ge 0.8$일 경우에는 아무런 패킷전송도 이루어지지 않는 것을 알 수 있다.

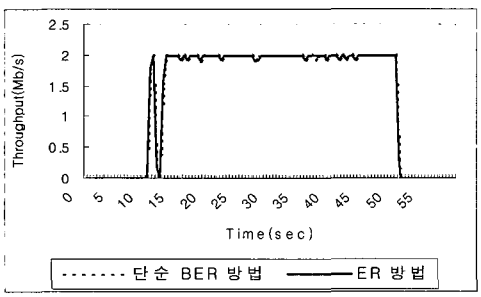


그림 10. ___ ___인 경우 효율 비교
Fig. 10 Comparison of Throughput for ___ ___

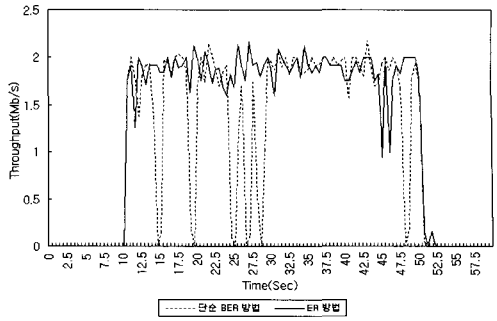


그림 11. $PER = 10^{-2}$ 인 경우 효율 비교
Fig. 11 Comparison of Throughput for $PER = 10^{-2}$

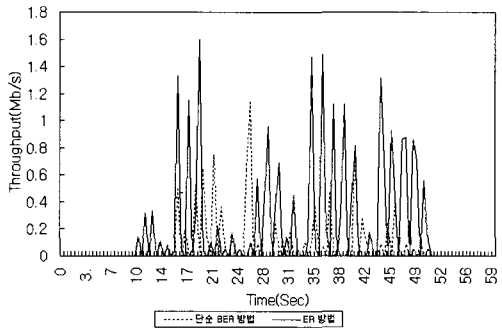


그림 12. $PER = 10^{-1}$ 인 경우 효율 비교
Fig. 12 Comparison of Throughput for $PER = 10^{-1}$

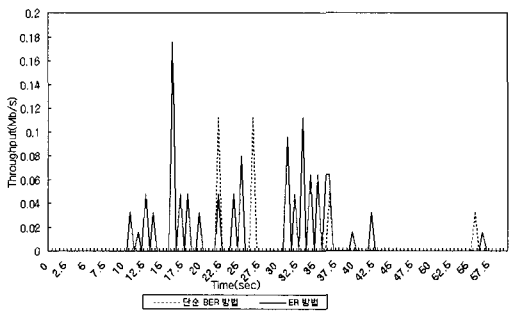


그림 13. PER= 0.4 인 경우 효율 비교
Fig. 13 Comparison of Throughput for PER= 0.4

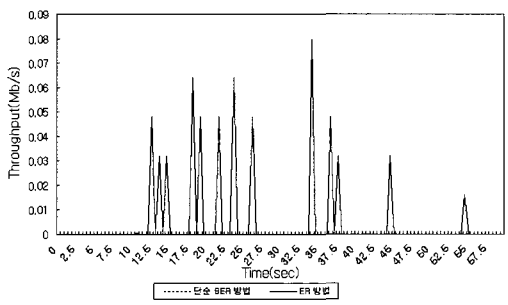


그림 14. PER= 0.5 인 경우 효율 비교

Fig. 14 Comparison of Throughput for PER= 0.5

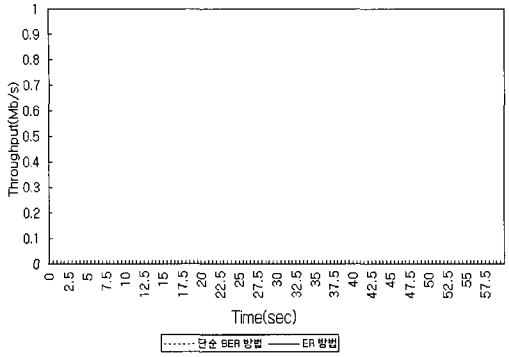


그림 15. PER= 0.8 인 경우 효율 비교
Fig. 15 Comparison of Throughput for PER= 0.8

다양한 PER에 대한 ER알고리즘과 단순 BER 알고리즘간의 효율들을 비교한 결과가 표2에 정리되어 있다.

표 2. ER알고리즘과 단순 BER 알고리즘과의 효율 비교
Table 2. Comparison of Throughput between ER algorithm and Simple BER algorithm

패 킷 손실율	ER(Mb/s)	단순 BER(Mb/s)	증가율 (%)
10^{-5}	1.998613	1.998613	-
10^{-3}	1.936463	1.936013	0.02324
10^{-2}	1.870925	1.350275	27.8104
10^{-1}	0.306008	0.119995	60.79
0.2	0.064808	0.043796	32.423
0.3	0.022408	0.015188	32.22
0.4	0.014808	0.008808	40.5186
0.5	0.007408	0.007408	-
0.6	0.005608	0.005608	-
0.7	0.000408	0.000408	-
0.8	0	0	-

실제로 $PER > 0.5$ 이면 패킷 에러율이 매우 큰 경우이기 때문에 신뢰성이 없는 망으로 볼수 있다. 실제적으로 패킷손실이 이 정도 크면 어떤 프로토콜로서도 신뢰성 있는 품질서비스는 보장할 수 없다. 따라서 패킷 에러율이 큰 환경인 경우에는 패킷 에러율를 줄이기 위해서는 패킷에 FEC을 사용하거나 전송 패킷의 크기를 줄여야 한다.

Ⅳ. 결 론

본 논문에서 혼잡 윈도우 크기에 대한 무선 환경에서도 효율적으로 동작하는 새로운 기법을 제안하였고 여러 종류의 TCP에서 사용되는 기법과 비교분석을 하였다. 제안하는 알고리즘은 혼잡 윈도우 크기를 패킷 에러율과 현 TCP의 상태가 fast recovery 혹은 slow start 상태에 있느냐에 따라 알맞게 변화시킨다. 즉 각 상태에 따라 PER을 기반한 알맞은 보정용 가중치를 사용하여 혼잡 윈도우 크기를 무선 환경에 맞게 보정함으로써 높은 PER의 환경하에서 일어나는 TCP의 계속적인 윈도우의 감소로 인한 성능 저하를 감소시킨다. 전반적으로 제안하는 기법은 무선 에러가 있는 환경에서 기존의 여러 TCP에서 사용되는 알고리즘들보다 에러에 강인한 특성을 보여주면서 27.8 ~60.8% 정도 효율을 향상시킨다는 것을 시뮬레이션을 통해 알 수 있었다. 또한 제안하는 기법은 시뮬레이션을 통해 단순 BER 알고리즘보다 효율이 좋다는 것을 보여주었다.

본 논문에서 제안하는 ER 알고리즘은 표준 TCP의 트랜스포트계층에서 양극간 연결을 유지함과 동시에 다른 방법들과 달리 네트워크의 오버헤드를 줄일 수 있다. 또한 기존 TCP알고리즘에 대한약간의 수정으로 쉽게이루어졌고 모든 네트워크에 적용이 가능하다는 장점을 가지고 있다.

패킷 에러율를 줄이는 방법중에 데이터 계층에서 Hybrid ARQ적용 및 전송 패킷의 길이를 줄이는 방안과 함께 고려하여 보다 효과적인 성능개선을 이루는 연구를 할 필요가 있다.

감사의 글

※ 이 연구는 강원대학교 BK21 프로그램의 일부 지원을 받아 수행한 것이다.

참 고 문 헌

[1] Ole J. Jacobsen, "The Future for TCP," *The internet protocol Journal*, Vol.3, Issue3, Sep. 2000.

[2] Prasun Sinha, Thyagarajan Nandagopal, Narayanan Venkitaraman,"WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," *MOBICOM'99*, pp 231-241, August 1999

[3] Sally Flody, "A report on recent developments in TCP Congestion Control," *IEEE*

Communication Magazine, Vol.139, No.4, pp.84-90, April 2001

[4] E. Ayanoglu, S. Paul, T. F. LaPorta, K.K. Sabnani, and R.D Gitlin, "AIRMAIL: A Link Layer Protocol for wireless networks," *ACM Wireless Network*, Vol.1, pp.47-60, February 1995

[5] Hari Balakrishnam,Srinivasan Seshan and Randy H.Katz, "Improving reliable transport and handoff performance in cellular wireless networks ," *ACM Wireless Network*, Vol .1, Issue 4, pp.469-481, December 1995

[6] H.Balakrishnan, "A Comparision of mechanisms for Improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking* , Vol. 5, No. 6 ,pp.765-769, December 1997

[7] A DeSimone, M,C Chuah, and O.C.Yue "Throughput performance of Transport-layer protocol over wireless LANs," *In proc of Globecom'93*, pp.542-549, December 1993

[8] Jorge A, Cobb and Prathima Agrawal, "Congestion or Corruption? A strategy for efficient wireless TCP sessions," *IEEE Symposium on 1995 Proceedings*, pp.262 -268, June 1995

[9] Ajoy Bakre,B.R.Badrinath "I-TCP:Indirect TCP for mobile Hosts," *15th Int'l Conference On Distributed Computing Systems(ICDCS'95)*, pp.136-143, May 1995

[10] M.Mathis , J.Mahdavi, S.Floyd, A Romanow, TCP SelectiveAcknowledgement options, *RFC2018*, April 1996.

[11] S.Keshav and S.P Morgan, "SMART: Performance with Overload and Random losses," *Proceedings of IEEE Inform97*, Vol. 3, pp.1131-1138 ,April 1997

[12] 윤지훈, 손주형, 서정환, 서승우," 무선 인터넷 환경에서 Congestion window크기의 보정을 통한 TCP성능개선," *SK Telecommu-nication Review*, 제10권 6호, pp.1233-1243, Nov./Dec. 2000

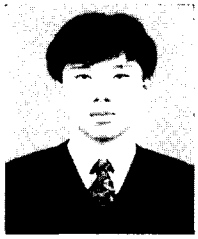
[13] Gary R. Wright, W. Stevens, *TCP/IP illustrated*, Vol. 2, Addsion Wesley, 1999.

박 홍 성(Hong Seong Park) 정회원

1983년 2월 : 서울대학교 제어계측공학과 졸업
1986년 2월 : 서울대학교 제어계측공학과 석사
1992년 2월 : 서울대학교 제어계측공학과 박사
1983년 ~ 1990년 : 삼성전자
1992년~현재: 강원대학교 전기전자정보통신공학부

<주관심분야> 무선 네트워크 시스템, 블루투스, 데이터 통신, 실시간 네트워크, 성능분석

전 선 국(Sheon-Kook Jeon) 정회원



1998년 : 연변과학기술대학 전자전산과 졸업
2002년 8월 : 강원대학교 제어계측공학과 석사
2002.9~현재: (주)링크웨어

<주관심분야> 무선 네트워크 시스템, 데이터 통신