무선 환경에서 802.11 MAC의 MIB 정보를 이용한 TCP 성능 개선 방법

준회원 신 광 식*, 정회원 김 기 원**, 준회원 윤 준 철*, 김 경 섭*, 장 문 석*, 정회원 최 상 방*

TCP Performance Improvement Scheme Using 802.11 MAC MIB in the Wireless Environment

Kwang-Sik Shin* ***Associate Member***, Ki-Won Kim** ***Regular Member***, Jun-Chul Yoon*, Kyung-Sub Kim*, Mun-Suck Jang* ***Associate Members***, Sang-Bang Choi* ***Regular Member***

요 약

TCP에서의 혼잡제어는 패킷 손실이 발생하면 이를 네트워크의 혼잡상황으로 판단해서 전송률을 줄인다. 무선 네트워크에서는 채널 에러로 인해 패킷 손실이 발생하는데, 기존의 유선환경에서의 TCP는 이를 혼잡으로 인한 손실로 착각하여 성능을 떨어뜨리는 결과를 초래한다. 그러므로 유·무선 통합네트워크에서의 TCP 성능 저하를 막기 위해 혼잡손실과 무선손실을 구별하는 연구가 진행되고 있다. 기존의 무선 TCP에 대한 연구는 주로 패킷이 전달되는 시간의 변화를 통해 네트워크의 혼잡상황을 유추해서 패킷 손실 시 혼잡손실과 무선손실을 예측하지만, 패킷의 전송시간은 여러 가지 다른 요인에 영향을 받기 때문에 정확한 손실구분은 불가능하다. 그러므로 본 논문에서는 IEEE 802.11 MAC에서 정의하고 있는 MIB(Management Information Base)의 무선손실 정보를 이용하여 유선손실과 무선손실을 구별하는 알고리즘을 제안한다. MAC 계층의 MIB를 수집하여 사용하는 제안된 알고리즘과 패킷의 지연 시간을 이용하는 기존의 알고리즘을 시뮬레이션을 통하여 비교하고 분석한 결과 무선 채널에서의 에러율이 10%인 경우에, Spike 알고리즘에 비해 12%, mBiaz 알고리즘에 비해 32%의 성능 향상을 보였다.

Key Words : Wireless TCP, IEEE802.11, MAC MIB, TCP congestion control, ROTT

ABSTRACT

Congestion control of the TCP reduces transmission rate when it detects packet loss because packet loss origines from congestion in the wired network. In the wireless network, packet loss comes from channel errors. Wired TCP degrades performance when there are wireless losses because it does not classify type of loss. These day, there are many researches which classify type of loss between congestion loss and wireless loss for wired-wireless hybrid network. For wireless TCP, many of existing algorithms are based on the estimated bandwidth or variations of packet arrival time. In this paper, we propose a new TCP scheme to distinguish the wireless packet losses from the congestion packet losses using MIB of the IEEE 802.11 MAC. We perform excessive simulations using the NS-2 network simulator and analyze the simulation results to compare the performance of the proposed algorithm to other well-known algorithms. From simulation results, we know that proposed algorithm improves performance about 12% and 32% compared with Spike algorithm and mBiaz algorithm, respectively.

※ 이 논문은 2008학년도 인하대학교의 지원에 의하여 연구되었음
* 인하대학교 전자공학과 컴퓨터구조 및 네트워크 연구실(kwangsik@inha.ac.kr), ** 파이오링크(주)
논문번호 : KICS2007-12-561, 접수일자 : 2007년 12월 18일, 최종논문접수일자 : 2008년 6월 9일

Ⅰ. 서 론

무선 통신 기술의 급속한 발달로 인하여 유선망으로만 구성되었던 기존의 네트워크 환경은 점차 유·무선망이 혼합된 통합 네트워크 형태로 변화하고 있다. 이러한 네트워크 환경의 변화와 달리 기존의 유선네트워크 환경에서 사용되던 네트워크 프로토콜은 아직까지 변화한 환경을 따라가지 못하고 있는 실정이다. 특히 네트워크 서비스를 위한 여러 계층의 프로토콜 중에서 전송계층을 담당하는 대표적인 프로토콜인 TCP는 환경변화의 영향을 가장 많이 받는다. 유선 네트워크 환경에서는 채널에러로 인한 패킷 손실이 거의 없기 때문에 네트워크 혼잡한 경우에만 패킷이 손실된다고 가정한다. 그러므로 기존의 TCP는 송신측에서 패킷 손실을 감지하면, 전송률을 줄임으로 써 네트워크상의 혼잡상황을 해소하는 혼잡제어 기능을 지원한다.

무선 네트워크 환경에서는 기존의 유선 네트워크 환경과 달리 무선 채널을 통한 전송과정에서 채널에러가 빈번히 발생한다. 그러므로 기존의 TCP의 혼잡제어를 그대로 사용할 경우 무선 채널 에러를 네트워크의 혼잡으로 잘못 판단하여 불필요하게 전송률을 줄이는 문제가 발생할 수 있다. 불필요한 전송률 감소는 전체적인 TCP 전송 성능저하를 초래하기 때문에 유·무선 통합 네트워크 환경에서는 혼잡제어를 적용하기 전에 혼잡으로 인해 발생하는 에러와 무선 채널 에러를 반드시 구별해야 한다.

유·무선 통합 네트워크 환경에서 TCP의 혼잡제어로 인한 불필요한 성능저하를 막기 위하여 다양한 무선 손실 구분 알고리즘이 연구되고 있다. 무선 손실 구분 알고리즘은 유선 네트워크와 무선 네트워크 부분으로 연결되는 기지국에서 무선 손실을 처리해주는 방식과 패킷의 지연시간 정보를 바탕으로 손실 유형을 결정하는 방식 두 가지로 구분된다.

무선손실을 별도로 처리하는 방법으로 Indirect TCP와 Snoop TCP가 있는데, 이러한 알고리즘은 기지국에서 무선 손실을 처리하기위해 별도의 기능을 수행해야 하는 단점이 있다. 한편, ROTT(relative one-way trip time), RTT(round trip time) 등의 정보를 통하여 패킷의 지연시간을 이용하여 손실 유형을 구분하는 알고리즘으로는 Biaz, mBiz, Spike 등이 있는데, 송·수신 단의 TCP 연결을 그대로 유지하는 장점은 있으나 패킷의 지연시간은 네트워크의 여러 가지 요인으로 인해 영향을 받으므로 정확한 패킷 손실을 검출하는 데는 한계가 있다[1-4]. 지연시간은 무선손실을 감지에 대한 확률을 높이는 것이기 때문에 보다 정확한 손실 구분을 위해서는 실제 무선 채널로의 전송을 담당하는 MAC 계층의 정보를 이용해야만 한다.

현재 가장 광범위하게 사용되고 있는 무선 네트워크 표준인 IEEE 802.11의 MAC은 링크 상에서 발생하는 무선 패킷 손실 과정을 MIB-II에 기록한다. 그러므로 본 논문은 이러한 MIB (Management Information Base) 정보를 이용한 유·무선 손실 구분 알고리즘을 제안한다. MIB 정보 중에서 dot11FailedCount는 패킷 에러로 인한 재전송 시 전송횟수 제한에 도달하여 전송이 포기된 패킷의 수로서 송신 시에 이용되며, dot11FailedCount는 FCS (frame check sequence) 에러가 발생한 패킷의 수로서 수신시에 이용된다.

TCP 연결의 수신측이 무선 노드일 때, 패킷 수신시 dot11FCSErrorCount 값을 확인하여 이 값이 재전송 제한 횟수인 dot11LongRetryLimit 또는 dot11Short- RetryLimit 보다 크거나 같아지면 전송이 포기된 것(패킷이 무선 링크에서 사라진 것)으로 판단한다. 송신측에게 무선 손실이 발생했음을 알리면, 패킷의 손실이 발생하더라도 혼잡에 의한 손실이 아니라는 것을 알기 때문에 전송률을 줄일 필요가 없다. 그러므로 불필요한 전송률 감소를 피함으로서 TCP의 전송 성능을 향상시킬 수 있다.

본 논문은 다음과 같이 구성된다. Ⅱ절에서는 패킷손실의 유형을 구분하는 방법 및 기존의 손실 구분을 위해 행해졌던 연구에 대해 소개하고, Ⅲ절에서 본 논문에서 제안된 다중 흐름에서 좀 더 정확하게 패킷 손실의 유형을 구분할 수 있는 방법과 그 동작 절차에 대해 자세히 설명한다. Ⅲ절에서는 본 논문에서 제안한 손실 구분 알고리즘을 시뮬레이션을 통해 분석하고, 같은 환경에서의 다른 알고리즘과 성능 비교를 통하여 Ⅴ절에서 본 알고리즘의 유용성에 대해 논의한 후 향후 연구 과제를 제시하며 본 논문을 마친다.

Ⅱ. 패킷의 손실 유형 구분과 관련된 연구

2.1 I-TCP(Indirect-TCP)

I-TCP는 송신자와 수신자 사이의 종단 대 종단

TCP 연결을 기지국을 중심으로 송신자에서 기지국으로, 그리고 기지국에서 수신자로 유선 네트워크와 무선 네트워크로 연결을 나눈다. TCP 분리 연결을 통한 접근 방식으로 무선망에서 발생된 패킷 손실이 유선망에 영향을 주지 않기 위해서 기지국을 기준으로 유선 네트워크와 무선 네트워크로 구분한다. 기지국은 유선 네트워크에서는 수신자로서, 무선 네트워크에서는 송신자로서 역할을 한다. 즉, 무선 노드에서의 패킷 손실 시 송신자가 재전송하는 것이 아니라 기지국의 TCP 계층에서 마치 송신자인 것처럼 재전송 해주는 것이다. 그러므로 송신측에서는 유선 네트워크에서의 혼잡으로 인한 손실에 대해서만 처리하면 된다. I-TCP의 장점은 둘로 나누어진 연결이 각자 자신의 환경에 최적화된 프로토콜을 사용하기 때문에 무선 손실로 인한 패킷 손실에 효율적으로 대처할 수 있다. 그러나 종단 대 종단 TCP 연결 구조를 유지하지 않는다는 단점이 있다. 송신자와 수신자사이의 TCP 연결이 송신자와 기지국사이의 TCP 연결과 기지국과 수신자사이의 TCP 연결로 나눠지기 때문에 두 개의 TCP 연결을 지나는 동안 프로세싱 시간이 지연되고, 기지국은 송신자로부터 수신한 모든 패킷을 수신자에게 성공적으로 전달할 때까지 가지고 있어야 하기 때문에 트래픽이 많은 경우에는 많은 양의 버퍼가 요구된다[1].

2.2 Snoop TCP

Snoop TCP는 기지국에 Snoop Agent 모듈을 구현하여 TCP 연결의 양방향으로 전달되는 모든 패킷을 모니터링 한다. Snoop Agent는 손실된 패킷이 버퍼에 저장되어 있으면 이를 재전송하고 송신자로 중복 ACK를 전송하는 것을 막는다. 그러나 기지국의 버퍼에 해당 패킷이 없을 경우에는 중복 ACK를 송신자에게 보내고 송신자는 이를 혼잡 손실로 판단하여 혼잡 제어 알고리즘을 시작하게 된다. Snoop TCP의 장점은 불필요한 혼잡 제어를 방지하는 점이다. 그리고 종단 대 종단 TCP 구조를 유지하고, 기존 응용프로그램과의 완전 호환성을 제공하기 때문에 응용 프로그램의 수정 없이도 TCP 연결의 성능을 상당히 향상시킬 수 있다. 그러나 기지국에서 이루어지는 재전송으로 인해 중복 ACK가 송신자에게 전송되지 않더라도 송신자가 ACK를 받지 못한 패킷에 대해 타임아웃을 발생시키는 현상은 막을 수 없고 TCP 헤더가 암호화 되거나 TCP 패킷과 ACK가 서로 다른 경로를 통해 전달될 경우에는 사용할 수 없다는 단점이 있다[2].

2.3 Biaz 알고리즘

Biaz 알고리즘은 TCP 구조에는 변화를 주지 않고, 송신측에서 네트워크에서의 패킷 지연을 지속적으로 감지하면서 손실 유형을 구분한다.

혼잡 손실은 네트워크에 트래픽이 몰려 라우터의 큐에 패킷이 가득차고, 더 이상 저장할 곳이 없을 때 발생하기 때문에 큐잉 지연이 길어진다. 반면 무선 손실은 무선 채널 상에서 사라지는 것이므로 네트워크상의 지연에 가장 큰 영향을 미치는 큐잉 지연을 수반하지 않는다. 이러한 두 손실 유형의 특성에 착안 하여 Biaz 알고리즘은 패킷이 손실됐을 때, 네트워크의 지연시간을 확인하여 지연시간이 작으면, 무선손실로 판단한다[3]. 간혹 Biaz 알고리즘은 무선 손실을 구분 짓는 무선 손실 윈도우의 크기가 너무 커서 혼잡에 의해 발생되는 패킷 손실을 무선 손실로 오인하는 문제가 발생한다. 그러므로 윈도우의 상한 경계 값을 Biaz 알고리즘보다 향상시켜 제안한 알고리즘이 mBiaz이다. 하지만 mBiaz 알고리즘 역시 Biaz 알고리즘과 같이 송신자가 패킷을 일정한 시간 간격을 두고 보낸다는 가정 하에 만들어졌기 때문에 패킷을 보내는 시간이 일정하지 않은 실제 네트워크에서는 그리 좋은 성능을 보여주지 않는다.

2.4 Spike 알고리즘

Spike 알고리즘은 패킷의 지연시간을 기준으로 손실 유형을 구분한다는 점에서는 Biaz와 비슷하지만, 절대적인 시간이 아니라 이전 상황과 현재 상황을 비교하는 상대적인 패킷 이동시간을 이용하여 패킷 손실의 유형을 추측한다는 점에서 차이가 있다. 상대적인 이동시간을 측정하기 위해 송신자는 패킷을 보낼 때 패킷의 헤더에 타임스탬프를 같이 보내고 수신자는 이 패킷을 받은 시간을 기록하여 두 시간의 차를 이용하게 된다. 그리고 이 시간차를 ROTT(Relative One-way Trip Time)라 정의한다. 이렇게 구한 ROTT의 변화는 실제 네트워크에서의 전송시간 지연을 좀 더 정확하게 반영한다[4].

Ⅲ. 제안된 손실 구분 알고리즘

3.1 손실 구분 알고리즘

본 논문에서 제안하는 손실 구분 알고리즘의 주된 목표는 TCP 연결을 나누지 않고, 하드웨어의 변경 없이 종단 대 종단 TCP 접근 방식으로 정확하게 패킷의 손실 유형을 구분하여 TCP가 손실 유형에 맞게 적절히 동작하게 하는 것이다. 지금까지의 end-to-end TCP 수정을 통한 접근 방식들은 ROTT나 RTT를 이용하여 네트워크의 상태를 추측하고, 그것을 기초로 패킷의 손실 유형을 판단했는데, 앞 장에서 언급한 것과 같이 ROTT나 RTT로 네트워크를 예측하는 방법에는 정확도에 있어서 한계가 있을 수밖에 없다[5-6]. 따라서 그 한계 이상의 안정적인 정확도를 확보하기 위해 실제 링크 계층에서 전송 중 발생하는 패킷의 손실에 대한 정보가 필요하다.

현재 무선 링크에서 가장 많이 사용되고 있는 802.11 표준에는 패킷이 무선 링크에서 손실되는 과정 중에 나타나는 이벤트를 MIB-II 에 기록한다. 본 논문에서 제안하는 손실 구분 알고리즘은 이러한 기록을 이용해 패킷의 무선 손실을 판단하게 된다. TCP가 패킷 손실의 원인을 구분하기 위해서는 실제 패킷이 어떻게 손실되는지를 알아야 한다. 첫째 유선망에서의 패킷 손실은 경로상의 라우터에서 대기 큐가 꽉 차게 되어 라우터의 폐기 정책에 따라 일부러 패킷을 버려버리게 되는 것이 원인이고, 둘째로 무선망에서의 패킷 손실은 첫 번째와 같이 경로에서 라우터의 대기큐가 꽉차는 경우와 전송 중 채널 상태가 나빠져 전송이 실패하는 것이 원인이다.

본 논문에서 다루고 있는 802.11 표준에서 무선 링크단에서의 패킷 손실 과정은 다음과 같다. 무선링크의 송신 노드는 패킷을 보낸 후 ACK가 돌아오는 동안 타이머를 진행시킨다. 패킷을 받은 수신노드는 FCS 에러 체크를 하고, 패킷에 문제가 있다면 ACK를 보내지 않는다. 송신노드는 ACK를 받지 못하기 때문에 타이머를 완료하고, 해당 패킷을 재전송한다. 무선 링크에서 이와 같은 재전송 과정이 반복되어 제한 횟수에 다다르면, 송신노드는 해당 패킷의 전송을 포기하여 무선 링크에서 패킷의 손실이 발생한다. 이러한 패킷 손실 과정은 MIB-II에 항목별로 기록되는데, 본 논문에서는 이렇게 802.11 표준에서 기록하고 있는 무선 링크단에서의 패킷 손실 정보들을 이용하여 네트워크 혼잡으로 인해 발생하는 손실과 구별한다.

3.1.1 제안된 알고리즘에 사용되는 MIB-II 속성

다음은 제안된 패킷 손실 구분 알고리즘에서 사용되는 802.11 표준의 MIB-II 속성들이다[7].

- dot11FailedCount (OID: 1.2.840.10036.2.2.1.3): MSDU (MAC service data unit)가 성공적으로 전송되지 않았을 때 증가하는 카운터이다. 한 MSDU에 대한 전송 시도가 dot11ShortRetryLimit 또는 dot11LongRetryLimit을 초과하게 되면 MSDU의 전송이 포기 되고 dot11FailedCount가 증가 한다.
- dot11FCSErrorCount (OID: 1.2.840.10036.2.2.1.12): MPDU 수신 후 FCS 검사에서 에러가 발생하면 1씩 증가한다.
- dot11RTSThreshold (OID: 1.2.840.10036.2.1.1.2): 데이터 또는 제어 프레임 전에 전송되는 RTS 프레임의 전송 여부를 결정한다. 이 속성의 값은 RTS 프레임의 전송이 필요한 가장 작은 프레임의 크기를 결정한다. 이 속성의 값보다 크기가 작은 프레임은 RTS의 전송이 선행되지 않는다. 기본값은 2347이고, 실제로 모든 프레임에 대해 RTS 전송을 하지 않게 한다. 그러나 프레임 에러나 재전송 같은 채널의 상태를 나타내는 MAC 카운터의 값이 증가하면 이 값을 조정하여 RTS를 전송하게 할 수 있다.
- dot11ShortRetryLimit (OID: 1.2.840.10036.2.1.1.3): dot11RTSThreshold 보다 크기가 작은 프레임의 ACK가 돌아오지 않는 경우, 프레임의 전송이 포기되기 전까지의 재전송 횟수를 나타낸다. 기본값은 7이다.
- dot11LongRetryLimit (OID: 1.2.840.10036.2.1.1.4): dot11RTSThreshold 보다 크기가 크거나 같은 프레임의 ACK가 돌아오지 않는 경우, 프레임이의 전송이 포기되기 전까지의 재전송 횟수를 나타내고, 기본값은 4이다.

3.1.2 패킷 손실 구분

본 논문에서 제안하는 패킷 손실 구분 알고리즘은 추측성 계산을 따로 하지 않는다. 즉 다른 기존의 다른 알고리즘들처럼 네트워크의 상황을 예측하지 않고 바로 자신의 링크에서 일어나는 손실 관련 기록들만으로 손실의 유형을 구분한다.

```
When the receiver receives an out of order DATA(seq)
 if (size_of_received packet > dot11RTSThreshold)
        ACK(seq).wireless_loss = Δdot11FCSErrorCount / dot11LongRetryLimit
 else
        ACK(seq).wireless_loss = Δdot11FCSErrorCount / dot11ShortRetryLimit
 send ACK(seq)
```

그림 1. 수신측에서 무선채널을 사용하는 경우: 수신측 손실구별 알고리즘

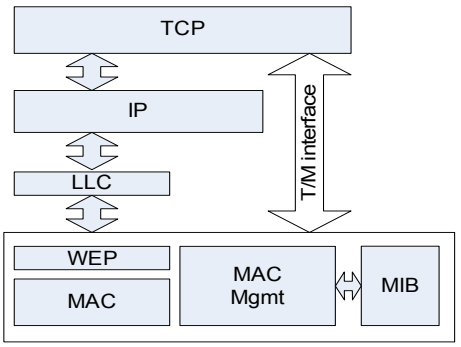

그림 2. NS-2를 사용한 T/M 인터페이스 구현

그러나 TCP에서 하위 계층의 정보인 MIB를 접근하기 위해서는 TCP계층과 MIB 간에 별도의 인터페이스가 필요하다.

본 논문에서는 NS-2 시뮬레이터에 이와 같은 T/M(TCP/MAC) 인터페이스를 구현하여 시뮬레이션 하였다[8]. 그림 2는 본 논문에서 시뮬레이션 시 구현된 T/M 인터페이스이다.

제안된 손실구분 알고리즘은 송신측이 무선 채널을 사용할 경우와 수신측이 무선 채널을 사용할 경우에 대해 각각 다음과 같은 동작을 수행한다. 먼저 송신측이 무선 노드일 경우를 살펴본다. 그림 1은 송신측이 무선 노드로써 패킷을 전송할 때 무선 링크에서 발생한 패킷 손실을 감지한 후 TCP에서 패킷을 재전송하는 과정을 보여준다. 그림에서 ACK(seq)는 "seq"의 시퀀스 번호를 갖는 ACK 패킷, DATA(seq)는 "seq"의 시퀀스 번호를 갖는 데이터 패킷을 나타낸다. number_of_ACK(seq)는 수신된 ACK(seq)의 개수이고, CongWnd는 혼잡윈도우의 크기이다. 송신측에서는 패킷을 전송하기 전에 T/M 인터페이스를 통해 재전송 횟수 제한에 도달하여 전송이 포기된 패킷의 수를 의미하는 dot11FailedCount 속성을 확인하여 채널에서의 손실을 파악한다. 이 값이 이전 값에 비해 증가했을 경우, 결국 바로 전에 전송한 패킷이 무선 채널에서 손실 된 것을 의미한다. 송신측의 TCP는 패킷을 전송한 후 중복 ACK를 받으면, dot11FailedCount 속성을 확인하여 그 값이 0이면 손실 패킷을 재전송한 후 혼잡회피모드로 진입하여 혼잡윈도우의 크기를 반으로 줄이고, 0보다 크면 송신측 채널에러임을 인식하여 혼잡윈도우의 크기를 그대로 유지한 채 손실된 패킷을 재전송한다.

다음은 수신측이 무선노드일 경우 무선 채널에 의한 손실을 구분하는 과정을 설명한다. 그림 3은 수신측에서 채널에서의 패킷 손실을 감지하여, ACK에 채널에러를 표시하는 비트를 설정하고 채널에러를 송신측으로 알려주는 과정을 보여준다. 그림 4는 송신측에서 중복 ACK를 받은 후 해당 패킷의 손실 원인을 파악하는 과정과 이에 따른 동작을 보여준다. 수신측에서는 패킷을 받은 후 패킷의 크기(size of received packet)를 dot11RTSThreshold 속성과 비교한다. 이 속성의 기본

```
When the sender receives an ACK(seq)
 number_of_ACK(seq)++
 if (number_of_ACK(seq) = 3)
        resend DATA(seq)
        if (Δdot11FailedCount = 0)
                CongWnd /= 2
```

그림 3. 송신측이 무선채널을 사용하는 경우: 송신측 손실구별 알고리즘

```
When the sender receives an ACK(seq)
 number_of_ACK(seq)++
 wireless_loss_count += ACK(seq).wireless_loss
 if (number_of_ACK(seq) = 3)
        resend DATA(seq)
        if (wireless_loss_count > 0)
                wireless_loss_count--
        else
                CongWnd /= 2
```

그림 4. 수신측에서 무선채널을 사용하는 경우: 송신측 손실구별 알고리즘

값은 2347 바이트이고, 전송하는 패킷의 크기가 이 값보다 크거나 같으면 dot11LongRetryLimit, 작으면 dot11ShortRetryLimit이 적용되어 재전송 횟수를 제한한다. 그 후 이전 패킷이 도착했을 때의 dot11FCSErrorCounti-1 값과 현재 패킷의 dot11FCSErrorCounti의 차를 확인한다. 단일 플로우의 경우, 이전 dot11FCSErrorCounti-1와 현재 dot11FCSErrorCounti와의 차이 △dot11-

FCSErrorCount가 패킷의 크기에 따른 재전송 횟수 제한 보다 크거나 같으면 현재 받은 패킷 이전의 패킷이 무선 링크에서 전송이 포기되었다는 것을 알 수 있다. 즉, 이전 패킷과 현재 패킷 사이의 패킷이 무선 링크에서 손실되었음을 알 수 있다.

그러므로 수신측은 이전 패킷이 손실된 원인이 채널 에러임을 송신측에 알려, 송신측에서의 불필요한 혼잡 제어를 방지해야 한다. 이를 위하여 본 논문에서는 TCP 헤더의 사용하지 않는 비트 중 하나를 무선 손실을 나타내는 비트로 사용한다. 그러므로 수신측은 해당 패킷 손실을 알리는 중복 ACK의 TCP 헤더에 미리 지정된 무선 손실을 나타내는 값(ACK(seq).wireless_loss)을 체크하여 송신측으로 전송하게 된다.

한편, 연속적인 무선 패킷 손실의 경우, 즉 △dot11FCSErrorCount가 재전송 횟수 제한의 배수인 경우, 3개의 중복된 ACK가 송신측에 도착하여 손실된 패킷이 재전송 된 이후 다시 무선 손실된 다른 시퀀스 넘버의 중복된 ACK가 도착하게 된다. 그러나 이후 들어온 중복된 ACK의 경우 무선 손실을 나타내는 wireless_loss 값이 설정되지 않는다.

그러므로, 수신측에서 무선 손실을 표시하는 wireless_loss와 송신측의 무선 패킷 손실 카운트(wireless_loss_count)는 현재 무선 채널에서 손실된 패킷의 개수를 나타낸다. wireless_loss_count는 무선 손실 값이 설정된 ACK가 도착할 때마다 카운트를 증가시킨다. 무선 손실 카운트가 0보다 큰 값인 상태에서 3번째 중복된 ACK가 도착하면 CongWnd의 크기를 줄이지 않고 무선 손실 카운트만 1만큼 감소시킨다. 이로써 송신측은 수신측에서 발생된 연속된 또는 잦은 무선 손실에 대해서도 정확하게 무선 손실로 구분한다.

그림 5는 송·수신측이 모두 무선 채널일 경우, 송신측에서 중복 ACK를 받은 후 패킷 손실의 원인을 송신측 채널손실, 혼잡손실, 수신측 채널손실임을 구별하고 이에 따라 재전송시 CongWnd의 크기를 조절하는 과정을 보여준다.

```
When the sender receives an ACK(seq)
 number_of_ACK(seq)++
 wireless_loss_count += ACK(seq).wireless_loss
 if (number_of_ACK(seq) = 3)
        resend DATA(seq)
        if (Δdot11FailedCount = 0)
                if (wireless_loss_count > 0)
                        wireless_loss_count--
                else
                        CongWnd /= 2
        else
                        CongWnd /= 2
```

그림 5. 송신측과 수신측이 모두 무선채널을 사용하는 경우: 송신측 손실구별 알고리즘

3.2 패킷 손실의 처리

제안된 손실 구분 알고리즘에 의해 패킷 손실의 유형이 구분되고, 수신측에서 보내는 ACK의 TCP 헤더에 손실의 유형에 대한 정보가 기록되어 송신측으로 보내진다. 그리고 해당 ACK를 받은 송신자는 헤더의 손실 유형 정보를 확인하고 손실의 유형에 따라 서로 다른 작업을 수행하게 된다.

제안된 손실 구분 알고리즘에 의해 패킷의 무선 손실이 발생했다고 결정되면 이 정보를 송신측에 전달해야 된다. 이것은 TCP 헤더에 사용하지 않는 6비트를 이용하는 것이 가능하다. 송신측에 도착한 ACK에 패킷의 무선 손실 개수를 나타내는 값이 0이 아니면 송신측에서는 수신측에서 무선 손실이 일어났다고 판단한다. 반대로 0으로 설정된 ACK가 도착하면 무선 손실이 잃어나지 않은 것으로 판단한다. 따라서 3번째 중복된 ACK가 도착할 때까지 무선 손실을 나타내는 비트가 0으로 설정되어 있으면 네트워크 혼잡에 의한 손실로 판단한다.

3개의 중복된 ACK에 무선 손실 구분 비트가 모두 0일 경우 혼잡 손실로 판단하게 된다. 발생된 손실이 혼잡 손실로 판단이 되면, 일단 손실된 패킷을 다시 재전송한다. 그리고 혼잡 윈도우를 현재의 반으로 줄여 전송률을 지금의 반으로 줄이게 된다. 이것은 기존의 TCP(Reno)에 적용된 AIMD(Additive Increase Multiple Decrease)와 같다.

3개의 중복된 ACK에 무선 손실 구분 비트가 한번이라도 설정이 되어있으면 발생된 손실을 무선 손실로 판단한다. 무선 패킷 손실이 발생하면 일단 손실된 패킷을 재전송하고, 기존의 TCP(Reno)와 같은 혼잡 제어를 하지 않는다. 네

트워크가 혼잡 하지 않은 상황에서의 불필요한 전송을 감소를 방지함으로써 TCP 전송 성능을 증가시킨다.

한편, 송신측에서는 3번째 중복된 ACK가 도착할 때 까지 무선 손실 구분 비트가 설정된 패킷의 개수를 카운트하게 된다. 무선 패킷 손실에 의한 재전송마다 카운트를 하나씩 감소시키고, 무선 손실 카운트가 0이 될 때까지 무선 패킷 손실 발생 시 혼잡 제어를 하지 않는다. 연속된 무선 패킷 손실의 경우, 무선 패킷 손실에 의한 재전송 바로 이후의 무선 패킷 손실을 혼잡 손실로 판단하는 것을 방지한다.

Ⅳ. 시뮬레이션 및 성능 분석

4.1 시뮬레이션 환경

본 논문에서는 제안된 손실 구분 알고리즘의 성능을 측정하기 위해서 NS2(Network Simulator2)를 이용하여 시뮬레이션을 수행하였다. 제안된 손실 구분 알고리즘은 802.11 표준을 사용하는 무선링크에서의 채널 손실 정보를 이용한다. 현재 버전의 NS2 시뮬레이터가 제공하는 에러 모듈 중 ErrorModel/TwoStateMarcov 모듈을 802.11 표준의 손실 과정에 맞게 적용하여 시뮬레이션 하였다. 다음은 시뮬레이션에 사용한 네트워크 구조와 환경 변수에 대하 설명한다.

4.1.1 시뮬레이션에 사용된 네트워크 구조

본 논문에서는 제안된 패킷 손실 구분 알고리즘의 성능을 측정하기 위해 그림 6과 같이 수신노드를 제외한 네트워크는 유선으로 하고, 수신는 기지국을 통해 무선 네트워크로 연결되도록 구성하였다.

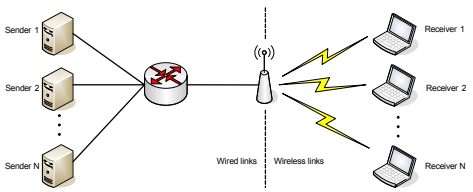

그림 6. 네트워크 구조

4.1.2 네트워크 파라미터

본 논문에서 제안하는 알고리즘은 802.11 표준에 기반을 둔 알고리즘이기 때문에 무선 링크는 802.11 인프라 모드를 사용하고, 패킷의 크기는 1024 byte로 고정하였다. 그리고 노드의 수 N을 2, 4, 6, 8로 변화시키며 다양한 시나리오의 시뮬레이션을 하였다.

한편, 사용된 어플리케이션, 손실 구분 알고리즘, 시뮬레이션 시간, 링크 대역폭, 패킷 전송 지연, 전송 시나리오 등은 각각의 시나리오 마다 다르게 설정하였다. 송신자와 수신자간의 패킷 이동을 서로 다른 난수를 사용하여 총 10번 수행하여 기록을 분석하였다. 표 1은 각 시뮬레이션에서 공통적으로 사용한 네트워크 파라미터를 보여준다.

표 1. 네트워크 파라미터

Parameter		Value
Packet Size	Wired	1024
	Wireless	1024
Bandwidth		1M
Node		2, 4, 6, 8
Wireless LAN		802.11 wireless LAN
Queueing Policy		DropTail
Type		FTP(TCP), CBR(UDP)
Period(sec)		200
Trial		10

4.2 시뮬레이션 결과 및 분석

4.2.1 시뮬레이션 1 – 혼잡 윈도우 크기 변화 비교

그림 7은 제안된 알고리즘의 전송률과 혼잡 윈도우 크기의 변화를 알아보기 위한 네트워크 구조이다.

그림 7의 시뮬레이션에서 연결 1은 송신노드 1과 수신노드 1의 연결로 200초 동안 FTP를 사용하여 데이터를 전송하고, 연결 2는 송신노드 2와 수신노드 2의 연결로 200초 동안 UDP를 사용하여 512byte 데이터 패킷을 전송한다. 이때 각 유

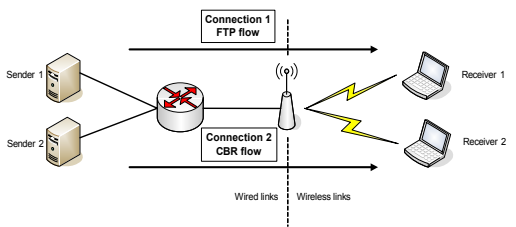

그림 7. 혼잡 윈도우 크기의 변화를 비교하기 위한 네트워크 구조

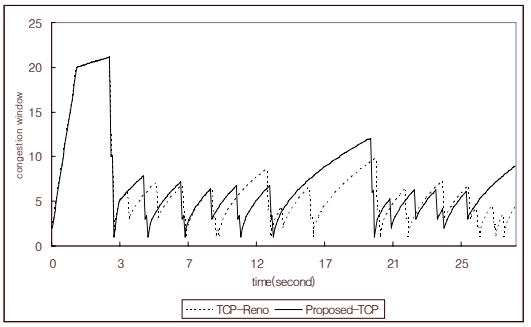

그림 8. 혼잡 윈도우의 크기 변화 비교(에러율 0.1%)

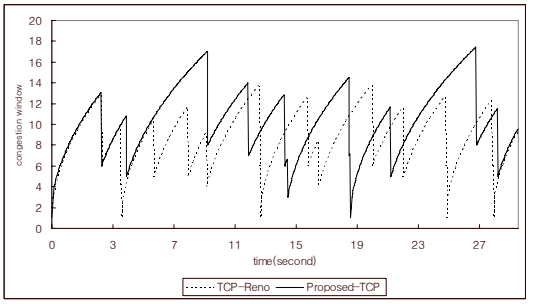

그림 9. 혼잡 윈도우의 크기 변화 비교(에러율 0.01%)

선 링크의 대역폭은 1Mb, 패킷 전송 지연은 5ms로 설정하였다. 패킷의 혼잡 손실과 무선 손실이 함께 발생하는 상황을 설정하기 위하여 라우터와 기지국 사이의 링크를 병목링크가 되게 하였다.

제안된 알고리즘의 성능을 비교하기 위하여 에러율이 0.01%, 0.1%일 때 기존 TCP Reno와 제안된 알고리즘을 이용한 경우를 각각 시뮬레이션 하여 비교하였다. 그림 8, 9는 시뮬레이션 1에서 에러율이 각각 0.01%, 0.1%일 때, FTP가 기존의 TCP(Reno)를 사용했을 때의 혼잡 윈도우와 제안된 알고리즘의 TCP를 사용했을 때의 혼잡 윈도우 크기 변화를 나타낸다.

그림 8과 9에서 볼 수 있듯이 제안된 알고리즘은 무선 손실을 구분함으로써 혼잡 윈도우가 감소되는 횟수가 감소하는 것을 볼 수 있다. 특히 그림 9에서 제안된 알고리즘이 혼잡 윈도우를 줄이는 횟수는 30초 동안 9회, TCP-Reno는 12회로 제안된 알고리즘이 전체적으로 혼잡 윈도우를 크게 유지하면서 TCP-Reno 보다 높은 전송률을 보인다.

4.2.2 시뮬레이션 2 – 제안된 알고리즘의 손실 구분 오류율

혼잡 손실이 발생하였으나 이를 무선 손실로 구분하는 오류가 발생한 비율을 시뮬레이션을 통해 측정하였다. 이러한 오류는 혼잡 제어 알고리즘이 실행되지 않고 혼잡 상황임에도 불구하고 지속적으로 패킷을 보내기 때문에 패킷 손실률이 높아지고, 수신자가 받는 데이터의 양도 줄어들게 된다. 또한 무선 손실로 구분하는 오류가 많이 발생한 알고리즘의 경우 그렇지 못한 알고리즘에 비해 전송률을 줄이지 않음으로 해서 대역폭을 더 많이 사용하고 결국 대역폭을 독식 할 수 있다.

본 논문에서 제안하는 알고리즘은 802.11 표준의 MIB 기록을 읽어 패킷 손실의 원인을 구분하는데 사용한다. 그러므로 MIB 정보의 실시간성이 알고리즘의 정확도에 큰 영향을 준다고 할 수 있다. 예를 들어 MIB 정보를 읽어오고 다음으로 읽어오기 이전에 패킷 손실이 발생 되었다면 제안된 알고리즘은 무선 패킷 손실 이전에 MIB 정보를 확인했으므로 현재의 손실을 혼잡 손실로 판단할 것이다. 따라서 제안된 알고리즘에서 MIB 정보를 읽어오는 주기가 알고리즘의 정확도를 결정할 만큼 중요하다.

시뮬레이션 1에서 사용된 네트워크 환경을 그대로 적용하고, MIB를 확인하는 주기만 다르게 하여 시뮬레이션 하였다. 그림 10은 제안된 알고리즘이 MIB 속성을 확인하는 주기에 따른 손실 구분 오류율이다. 주기가 1ms에서 2ms인 경우, 손실 구분 오류율이 1%에서 2%사이를 기록하지만 3ms인 경우, 5%에서 6%까지 범위로 증가한다. 그리고 4ms 이상부터는 10% 이상의 높은 오류율을 보인다.

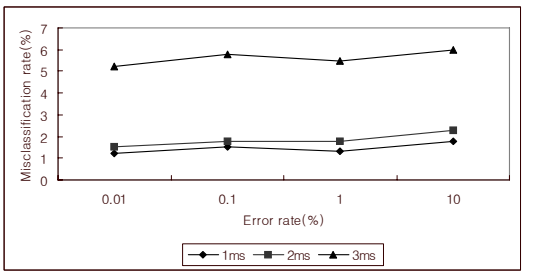

그림 10. MIB 속성을 확인하는 주기에 따른 손실 구분 오류율

4.2.3 시뮬레이션 3 - 손실 구분의 오류율 비교

제안된 알고리즘의 손실 구분 오류율을 측정하여 다른 알고리즘과 성능 비교를 하기 위해 그림 7의 네트워크 구조에 무선 노드와 유선 노드를 각각 3개씩 설정하여 시뮬레이션 하였다.

각 노드들은 TCP Reno, mBiaz, Spike, 제안된 알고리즘을 사용하여 손실 구분을 하게 된다. 제안된 알고리즘의 MIB 정보 확인 주기는 1ms로 설정하였다. 시뮬레이션 결과를 나타내는 그림 11에서와 같이, 제안된 알고리즘의 혼잡 손실의 구분 오류가 1%에서 2% 사이로 낮다는 것을 알 수 있다. 한편, 기존에 제안된 mBiz와 Spike는 상당히 높은 손실 구분 오류율을 나타내는데 이것은 802.11 표준의 패킷 손실 과정 때문이라고 사료된다. 즉, 패킷이 무선 링크에서 손상되어 링크 상에서 재전송되는데 패킷이 손실되기까지의 재전송 횟수가 기본적으로 최소 4회 또는 7회이다.

시뮬레이션에서 사용된 네트워크 구조는 3홉으로 이루어져 있다. 짧은 홉의 연결에서 전송을 하기 때문에 한 링크에서의 재전송 시간은 전체 ROTT시간에 많은 영향을 주게 된다. 또한, Spike의 경우 무선 링크에서 재전송에 걸리는 시간이 증가함에 따라 전체 ROTT가 증가하게 되고 증가된 ROTT가 상한 임계값을 초과할 경우 무선 손실을 혼잡 손실로 구분하게 된다.

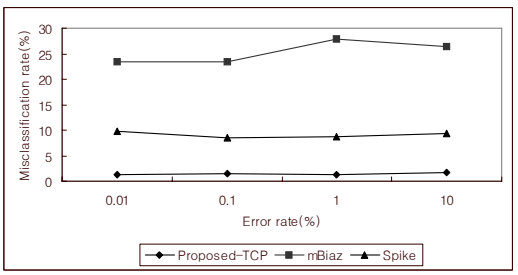

그림 11. 손실 구분 오류율

4.2.4 시뮬레이션 4 - 데이터 전송률 비교

시뮬레이션 1에서 사용한 네트워크 환경을 동일하게 적용하여 FTP의 전송 계층 프로토콜로 각각 TCP Reno, mBiaz, Spike와 제안된 알고리즘을 사용한 시뮬레이션을 하였다. 에러율(무선채널에서의 손실률)이 0%, 0.01%, 0.1%, 1%, 10%인 5가지의 경우에 제안된 알고리즘과 기존 알고리즘들의 전송률을 알아보기 위한 시뮬레이션이다.

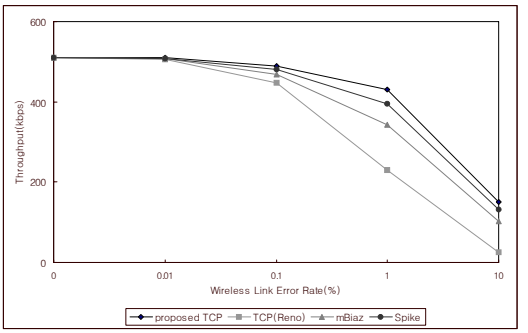

그림 12. 에러율에 따른 전송률

그림 12는 각각의 에러율에 따른 TCP Reno, mBiaz, Spike와 제안된 알고리즘의 전송률을 나타낸다. 에러율이 0.1%인 경우, Spike 알고리즘에 비해 2%, mBiaz 알고리즘에 비해 4.5%, TCP Reno에 비해 8.8%의 성능 향상을 보여 준다. 에러율이 1%인 경우, Spike 알고리즘에 비해 8.1%, mBiaz 알고리즘에 비해 20.2%, TCP Reno에 비해 46.5%의 성능 향상을 보여 준다. 에러율이 10%인 경우, Spike 알고리즘에 비해 12%, mBiaz 알고리즘에 비해 32%, TCP Reno에 비해 83.3%의 성능 향상을 보여 준다. 에러율이 10%일 때의 TCP Reno의 전송률은 0%일 때의 전송률과 비교했을 때 4.9% 수준이지만, 제안된 알고리즘은 에러율이 0%일 때와 비교했을 때 약 30% 수준의 성능을 보이고 있다.

V. 결 론

본 논문은 유·무선 통합망을 갖는 복합적인 네트워크 환경에서 패킷 손실이 발생했을 때 이를 무선 손실과 혼잡 손실로 구분하기 위한 알고리즘을 제안하였다. 제안된 알고리즘은 현재 가장 많이 사용되고 있는 802.11 표준에서 일어나는 패킷 손실 메커니즘과 MIB 정보를 이용하여 정확하게 패킷 손실을 구분한다.

제안된 알고리즘의 성능을 평가하기 위해 NS2를 사용하여 기존의 알고리즘들과 비교·평가 하였다. 무선 링크 상에서의 손실 과정을 알고 또한 이 과정에서 기록되는 정보를 MAC 계층에서 직접 수집하여 손실을 구분하기 때문에 시뮬레이션을 통하여 제안된 알고리즘이 기존의 알고리즘에 비해 정확하다는 것을 확인하였다.

제안된 알고리즘의 혼잡 손실의 구분 오류는

1%에서 2% 사이로 기존에 제안된 mBiz와 Spike와 비교했을 때 상당히 낮다는 것을 알 수 있다. 그러므로 에러율이 0.1%인 경우, Spike 알고리즘에 비해 2%, mBiaz 알고리즘에 비해 4.5%, TCP Reno에 비해 8.8%의 성능 향상을 보였다. 에러율이 1%인 경우, Spike 알고리즘에 비해 8.1%, mBiaz 알고리즘에 비해 20.2%, TCP Reno에 비해 46.5%의 성능 향상이 있었고, 에러율이 10%인 경우, Spike 알고리즘에 비해 12%, mBiaz 알고리즘에 비해 32%, TCP Reno에 비해 83.3%의 성능 향상을 보였다.

참 고 문 헌

〔1〕 A. Bakre and B. Badrinath, "I-TCP: indirect TCP for mobile hosts," *in Proc. 15th Int. Conf. Distributed Computing Systems (ICDCS), Vancouver*, BC, Canada, pp 136-143, May 1995.

〔2〕 H. Balakrishman, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "Improving TCP/IP performance over wireless networks," *in Proc. 1st ACM Int'l Conf. on Mobile Computing and Networking*, pp 2-11, Nov. 1995.

〔3〕 S. Biaz and N. Vaidya, "Discriminating congestion losses from wireless losses using interarrival times at the receiver," *in Proc. IEEE Symp. Application-Specific Systems and Software Engineering and Technology*, Richardson, TX, pp.10-17, Mar. 1999.

〔4〕 Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, "Achieving moderate fairness for UDP flows by path-status classification," *in Proc. 25th Annu. IEEE Conf. Local Computer Networks (LCN 2000)*, Tampa, FL, pp.252-261, Nov. 2000.

〔5〕 S. Cen, P. C. Cosman, G. M. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Transactions on Networking*, Vol.11, No.5, pp.703-717, Oct. 2003.

〔6〕 K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *IEEE JSAC*, Vol.22, No.4, pp.747-56, May 2004.

〔7〕 B. O'hara and A. Petrick IEEE 802.11 handbook - A designer's companion, Standards Information Network, *IEEE press*, Mar. 2005.

〔8〕 ns-2 Network Simulator, http;//www isi.edu/nsnam/, 2007.

신 광 식 (Kwang-Sik Shin) 준회원

2001년 2월 인하대학교 전자공학과 졸업
2003년 2월 인하대학교 전자공학과 석사
2003년 3월~현재 인하대학교 전자공학과 박사 과정
<관심분야> 멀티미디어 데이터 통신, 컴퓨터 네트워크, 무선 통신

김 기 원 (Ki-Won Kim) 정회원

2004년 2월 인하대학교 전자공학과 졸업
2006년 8월 인하대학교 전자공학과 석사
2006년~현재 파이오링크(주), 연구원
<관심분야> 컴퓨터 네트워크, 무선 통신, 네트워크 프로토콜

윤 준 철 (Jun-Chul Yoon) 준회원

2006년 2월 한국산업기술대 전자공학과 졸업
2008년 2월 인하대학교 전자공학과 석사
2008년~현재 (주)아이엔텍 연구원
<관심분야> 컴퓨터 구조, 컴퓨터 네트워크, 무선 통신

김 경 섭 (Kyung-Sub Kim) 준회원

2002년 2월 한남대 정보통신공학과 졸업

2006년 2월~현재 인하대학교 전자공학과 석사과정

<관심분야> 컴퓨터 구조, 컴퓨터 네트워크, 무선 통신

장 문 석 (Mun-Suck Jang) 준회원

1997년 8월 건양대학교 컴퓨터공학과 졸업

2000년 2월 인하대학교 전자공학과 석사

2005년 3월~ 현재 인하대학교 전자공학과 박사 과정

<관심분야> 센서네트워크, 컴퓨터구조, 병렬 및 분산 처리 시스템, Fault-tolerant computing

최 상 방 (Sang-Bang Choi) 정회원

1981년 2월 한양 대학교 전자공학과 졸업

1981년~1986년 LG 정보통신(주)

1988년 3월 Univ. of Washington 석사

1990년 8월 Univ. of Washington 박사

1991년~현재 인하대학교 전자공학과 교수

<관심분야> 컴퓨터 구조, 컴퓨터 네트워크, 무선 통신, 병렬 및 분산 처리 시스템, Fault-tolerant computing