무선 랜 환경에서 TCP Flow의 성능향상을 위한 MAC계층과 TCP계층의 연동기법

김재훈O, 정규민, 정광수
광운대학교 전자공학부 컴퓨터통신연구실
{jhkimO,kmjeong}@adams.kw.ac.kr, kchung@daisy.kw.ac.kr

Interaction TCP and MAC-layer to Improve TCP Flow Performance over WLANs

Jaehoon Kim0, Kyumin Jeong, Kwangsue Chung
School of Electronics Engineering, Kwangwoon Univ.

요 약

Snoop 프로토콜은 유·무선 혼합망에서 무선 링크에서 발생하는 TCP 패킷 손실을 효과적으로 보상하여 TCP 전송률을 향상시킬 수 있는 효율적인 프로토콜이다. 하지만, 무선 링크에서 연집한 패킷 손실이 발생하는 경우에는 지역 재전송을 효과적으로 수행하지 못하여 전송 효율이 떨어진다는 문제점이 있다. 본 논문에서는 Snoop 프로토콜의 이러한 문제점을 개선하기위해 MAC 계층의 재전송 메커니즘인 Stop & Wait ARQ 기법을 기반으로 하는 A2Snoop (ARQ Assistance Snoop) 프로토콜을 제안한다. A2Snoop 프로토콜은 현재 유·무선 혼합망에서 가장 널리 사용되는 IEEE 802.11 MAC 프로토콜 기반의 지역 재전송 메커니즘으로서, MAC 계층의 ARQ 기법과 TCP의 혼잡제어 메커니즘의 연동을 통해 효율적인 재전송을 수행한다. ns-2 시뮬레이터를 이용한 실험을 통해 A2Snoop의 지역 재전송 기법은 무선 구간의 연집적인 패킷 손실에 대해 효율적인 보상을 수행하며, 전송률을 유지하는 것을 확인할 수 있었다.

1. 서 론

TCP는 지난 20년을 통틀어 가장 성공적인 전송 계층 프로토콜이다. 이러한 TCP의 성공 요인으로서 신뢰성 있는 데이터 전송, 효과적인 트래픽 관리, IP와의 효율적인 결합 등을 들 수 있다. 이와 같은 장점에 의해 대다수의 인터넷 응용 프로그램들은 전송 계층 프로토콜로서 TCP를 사용하고 있다. 하지만 유선망에 최적화 되도록 진화해온 TCP는 무선망이 가지는 불안정한 링크에 의한 데이터 손실을 유선망에서처럼 네트워크의 혼잡으로 인한 손실로 오해한다. 그 결과 혼잡 제어 메커니즘이 수행되어 전송율을 줄이므로 네트워크 성능이 저하되는 문제점을 초래 한다[1].

이러한 이유로 최근 몇 년간 TCP의 성능을 향상시키기 위한 연구가 활발히 이루어지고 있다. 유·무선 혼합망에서 TCP의 성능을 향상시키기 위한 방법은 크게 링크계층 메커니즘, 전송계층 메커니즘으로 분류 할 수 있다. Snoop, Delayed Duplicate Acknowledgments, TULIP등이 링크계층 메커니즘이며, I-TCP, M-TCP, ELN등이 전송계층의 메커니즘 이다.

위의 다양한 메커니즘 중에서 Snoop 프로토콜은 가장 좋은 성능을 보여준다[1]. 이는 무선구간의 Random Loss에 대해 BS(Base Station)가 빠른 응답을 보이기 때문이다. 그러나 무선구간의 손실은 Random loss가 아닌 Burst loss가 대부분이다. 따라서 각 Local-RTT(Round Trip Time)마다 하나의 손실된 패킷만을 복구하는 Snoop 프로토콜은 Burst loss의 복구에 많은 한계를 가진다. 이러한 Snoop의 Burst loss 복구 한계를 효율적으로 해결하기 위해 무선구간에 SACK 옵션 또는 SACK과 유사한 추가옵션을 사용하는 방식이 제안되었다[2].

하지만 기존의 Snoop 프로토콜과 같이 전송계층의 중복 ACKs기반의 재전송을 사용한다는 점과 ACK 패킷에 SACK 옵션의 사용하는 점은 무선구간의 Burst Loss를 효율적으로 복구하는데 있어 많은 한계를 가진다.

현재 유.무선 혼합망에서 IEEE 802.11 MAC 프로토콜은 가장 널리 사용되는 채널 접근 기법이다. 이는 Stop & Wait ARQ 기법을 이용하여 무선구간의 신뢰적인 전송을 제공하지만, 이 또한 무선구간의 Burst Loss를 효율적으로 복구하는데 있어 한계를 가진다.

본 논문에서는 IEEE 802.11 MAC 프로토콜의 ARQ 기법과 TCP의 지역 재전송 기법의 연동을 통한 효율적인 지역 재전송 기법을 제안한다.

본 논문은 다음과 같이 구성된다. 2장에서는 무선구간에 추가옵션을 사용하는 기법인 SNACK(Selective Negative Acknowledgement) 프로토콜 및 IEEE 802.11 MAC 프로토콜의 ARQ 기법의 동작방법과 문제점에 대해 기술하고, 3장에서는 본 논문이 제시하는 새로운 재전송 기법에 대해 설명한다. 4장에서는 제안한 재전송 기법의 성능을 평가하기 위한 실험과 5장에서는 결론 및 향후 과제에 대해 기술한다.

이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(R01-2005-000-10934-0)

2. 관련 연구

2.1. SNACK 프로토콜

SNACK는 Snoop 프로토콜 메커니즘에서 Burst Loss 복구의 한계를 해결하기 위해 제안된 기법이다. 기본적인 동작 방법은 Snoop프로토콜과 유사하다. 단, 무선 구간의 Burst Loss를 빠르게 복구하기 위해 ACK 패킷의 헤더에 SACK과 유사한 SNACK 옵션을 사용하여 한 Local-RTT안에 손실된 모든 패킷을 재전송 하는 방법을 사용한다. 그림 1은 SNACK 프로토콜의 MH(Mobile Host)에서 FH(Fixed Host)로 데이터 전송과 FH에서 MH로 데이터를 전송하는 과정을 그림으로 나타낸 것이다.

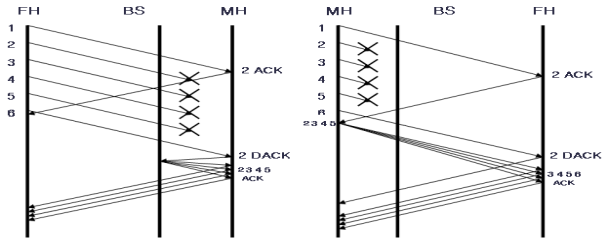

그림 1. SNACK 프로토콜의 에러 복구 과정

기존의 Snoop 프로토콜의 재전송 기법은 무선구간의 연속된 패킷 손실을 복구하는 과정에서 많은 Local-RTT를 소모한다. 이러한 재전송 과정의 많은 지연시간은 FH의 타임아웃을 발생시켜 심각한 성능 저하를 가져오는 원인이 된다. 하지만 그림 1과 같이 SANCK 프로토콜은 무선구간의 연속되는 패킷손실을 빠르게 복구하기위해 전송계층의 ACK에 추가 옵션을 사용한다. 따라서 하나의 Local-RTT에 손실된 모든 패킷을 복구하게 되고, FH의 타임아웃 발생을 방지할 수 있게 된다.

2.2 IEEE 802.11 MAC 프로토콜의 동작방법

현재 IEEE 802.11 MAC 프로토콜은 유·무선이 혼합된 패킷망에서 가장 보편적으로 사용되는 전송 프로토콜이다. IEEE 802.11 MAC 프로토콜은 데이터 전송의 신뢰성을 보장하기 위해 Stop & Wait 방식의 ARQ 기법을 이용한다. 이러한 신뢰성 보장 기법은 미리 정의된 시간동안 응답 패킷이 수신되지 않을 경우 Retry Limit이라는 Threshold만큼 재전송을 수행하는 방법이다. 하지만, 연집된 패킷 손실이 발생하는 무선 랜 환경에서 이러한 ARQ기법은 Retry Limit이라는 한정된 값 때문에 재전송에 실패하게 되며, 전송 버퍼에서 해당 패킷을 제거하게 된다. 이는 TCP의 Timeout을 유발하여 전송률을 떨어뜨리는 문제점을 발생시킨다. 이러한 과정을 그림 2에 나타내었다.

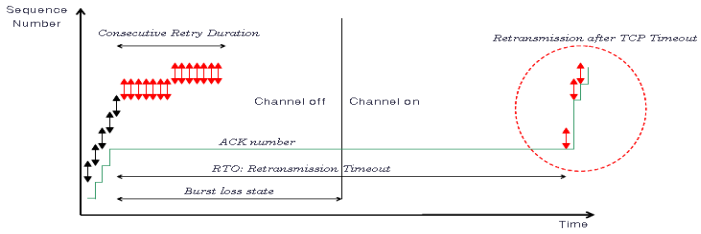

그림 2. TCP의 Timeout으로인한 재전송 과정

이러한 MAC 계층의 ARQ 기법의 문제점을 개선하기 위해서 Retry Limit의 수를 Burst Loss 구간이상 증가시켜 TCP의 전송률을 보장하는 기법인 MAC-layer LDA(Loss Differentiation Algorithm)이 제안되었다[3]. 그림 3은 MAC-layer LDA기법의 동작과정을 나타낸 것이다. 그림에 나타낸 것과 같이 Retry Limit 값을 최대 18까지 Burst Loss 구간이상 증가시켜 TCP의 Timeout을 방지하고, 전송률을 유지해 나간다.

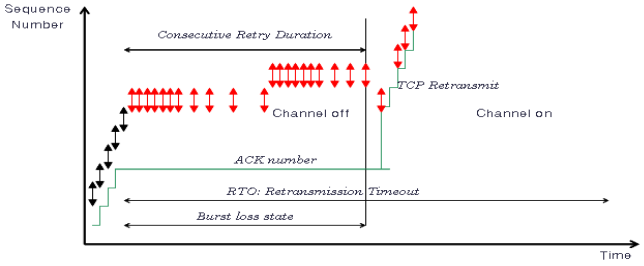

그림 3. MAC-layer LDA 재전송 기법

2.3 기존 재전송 메커니즘의 문제점

SNACK 프로토콜과 MAC-layer LDA 기법은 2.1절과 2.2절에서 언급한 장점들로 인해 기존의 재전송 메커니즘 보다 우수한 성능을 보인다. 그러나 Snoop 프로토콜과 같이 SNACK 프로토콜 또한 무선구간의 Burst Loss를 완벽히 복구하는데 있어서 몇 가지 문제점을 가지고 있다. 이러한 문제점은 크게 두 가지로 나눌 수 있다.

첫 번째는 기존의 Snoop 프로토콜과 같이 전송계층의 중복 ACK에 의한 재전송 기법이라는 점이다. 전송 손실율이 높은 무선구간에서 ACK 패킷의 손실로 인한 재전송 시간의 지연은 TCP의 타임아웃을 유발하여 성능 저하를 가져올 수 있다. 이러한 문제점은 상대적으로 TCP의 전송 윈도우의 크기가 작거나 전송방향이 MH에서 FH일 경우 더 큰 성능저하를 발생시킨다.

두 번째는 무선구간에서 SACK 옵션과 같은 추가 데이터 전송이다. 이러한 추가 옵션의 사용은 무선망의 대역폭과 무선 단말의 한정된 에너지 자원을 불필요하게 낭비하는 문제를 초래하게 된다.

또한, MAC-layer LDA 기법과 같이 Retry Limit을 증가시키는 것은 BS 또는 MH의 전송 버퍼의 패킷 처리율을 떨어뜨려 다른 Flow의 전송시간을 지연시킨다. 이러한 전송 시간의 증가는 전송률 저하로 나타나게 된다.

$$\sum^{cwnd} (Retry\ Limit \times MAC\ \ frame's\ transmission\ time) \quad (1)$$

식 (1)은 현재 MAC 계층에 수신된 TCP 패킷의 수와 Retry Limit의 증가에 따른 전송지연의 크기를 식으로 나타낸 것이다. 그림 4는 이러한 식 (1)을 바탕으로 Retry Limit의 증가에 따른 지연 시간의 증가량을 그래프로 나타낸 것이다. 그림 4에서 나타낸 것과 같이 Retry Limit의 증가에 따라 최대 2.2s의 전송 지연시간이 발생한다. 이러한 SNACK 프로토콜의 문제점

과 MAC 계층의 ARQ 기법의 문제점은 무선구간의 Burst Loss를 효율적으로 복구하는데 있어서 한계를 가진다. 따라서, MAC 계층과 전송 계층의 연동을 통해 현재의 무선 채널 상태를 고려하여 Retry Limit을 효율적으로 설정하고, TCP의 Timeout을 방지하기 위한 효율적인 지역 재전송 메커니즘이 필요하다.

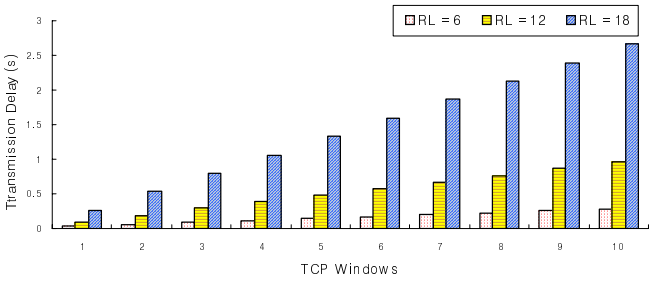

그림 4. Retry Limit의 증가에 따른 전송시간의 변화

3. A²Snoop(ARQ Assistance Snoop)

A²Snoop은 기존의 MAC 계층 및 전송 계층의 재전송 문제점을 해결하기 위해서 무선구간의 TCP 패킷의 처리시간을 기반으로 재전송 타이머를 설정하고, MAC 프레임의 처리시간을 기반으로 Retry Limit 값을 효율적으로 조절하여, 다른 Flow의 전송률 저하를 방지한다.

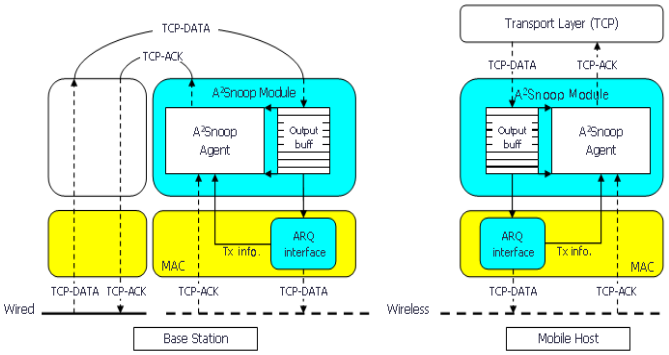

그림 5. A²Snoop 프로토콜의 시스템 아키텍처

그림 5는 Cross layer 기법이 적용된 BS와 MH의 시스템 아키텍처를 나타낸 것이다. 기존의 재전송 문제점을 개선하기 위해 그림 2에서 보는 바와 같이 각 시스템에 재전송 기능을 담당하는 A²Snoop Agent(IP 계층)와 패킷의 전송 정보를 담당하는 ARQ interface를 추가하였다. 그림 5와 같은 시스템 아키텍처가 가지는 장점은 기존의 지역 재전송 기법과 달리 표준화된 전송 메커니즘의 수정이 불필요하며, 손실된 패킷을 복구하는 과정에서 BS와 MH가 자신의 ARQ 정보를 이용하여 채널 상태를 판단하고, 독립적으로 지역 재전송을 수행함으로써 전송 방향에 관계없이 효율적인 전송 성능향상을 기대 할 수 있다[5]. 또한, 어느 한쪽이 A²Snoop 프로토콜을 지원하지 않아도 A²Snoop을 지원하는 방향에서 전송이 이루어질 경우 높은 전송 성능 향상을 기대할 수 있다.

A²Snoop의 기본적인 동작 방법은 기존의 Snoop 프로토콜, SNACK 메커니즘 등과 같이 지역 재전송을 사용하여 성능 향상을 한다는 점에서 기본적인 동작 방법은 유사하다. 따라서 본 논문에서는 새롭게 A²Snoop에서 제안하는 기법인 계층 간의 상호작용과 재전송 타이머 산출방법 및 Retry Limit의 조절 방법에 대해서만 언급하고자 한다.

3.1 MAC 계층과의 상호 작용

IEEE 802.11 MAC 프로토콜은 기본적으로 무선 구간에서 신뢰할 수 있는 데이터 전송을 보장하기 위해서 MAC 계층의 ARQ 기법을 사용한다. MAC 계층은 무선구간의 손실을 가장 빠르게 판단할 수 있는 계층이다. 그러므로 A²Snoop은 패킷 전송관련 정보를 A²Snoop Agent에 빠르게 통보하고 현재의 채널 상태와 다른 Flow의 유/무에 따라 Retry Limit값을 재설정하기위해서 MAC 계층의 ARQ 기법에 ARQ-interface를 추가하였다. 패킷 전송관련 정보의 통보를 위해서 ARQ-interface에 다음과 같은 2가지 이벤트를 아래와 같이 정의한다.

1) Channel-On 이벤트: 이 이벤트는 목적지 노드로 패킷의 전송을 성공하였을 경우 발생된다. 목적지 노드에 의해서 MAC 계층의 ACK 프레임을 정확히 수신하였을 경우에 ARQ-interface에 의해 발생된다.

2) Channel-Off 이벤트: ARQ 기법에 미리 정의된 수만큼 재전송을 수행한 후 더 이상 패킷을 전송할 수 없을 경우에 발생된다. 즉, MAC 계층의 Timeout이 발생된 경우에 ARQ-interface에 의해 발생되는 이벤트다.

3.2 상위 계층 또는 유선구간과의 상호 작용

A²Snoop Agent는 전송계층 또는 유선구간으로부터 수신되는 패킷을 조사하여 큐잉 지연시간을 포함한 필요한 모든 정보를 저장한다. 또한 하위계층으로 전달되기 직전에 전송할 패킷을 저장한다. 패킷이 하위 계층으로 전달된 이후에 A²Snoop Agent는 ARQ-interface로부터 전송관련 이벤트가 발생되기를 기다린다. 만약, Channel-On 이벤트가 ARQ-interface에 의해 발생된 경우 A²Snoop Agent는 해당 패킷을 버퍼에서 삭제하고, Retry Limit의 조절에 사용되는 MAC 프레임의 처리시간을 갱신한다. 또한, 이전의 전송 실패로 인해 저장한 패킷이 같은 목적지를 가지고 있을 경우, 가능한 빨리 손실된 패킷을 재전송 한다. 반대로 Channel-Off 이벤트가 발생한 경우 재전송을 위해 손실된 패킷을 저장하고, Retry Limit을 현재 전송 버퍼 내에 다른 Flow의 존재 여부에 따라 MAC 프레임의 처리시간을 기반으로 조절한다. 또한, TCP Timeout이전에 재전송을 수행하기 위해 TCP 패킷의 처리시간을 기반으로 산출된 재전송 타이머로 손실된 패킷을 관리한다. 마지막으로, TCP-ACK 패킷이 수신된 경우 해당 Flow의 TCP 패킷의 처리시간을 갱신한다. 이러한 TCP 패킷 및 MAC 프레임의 처리시간의 산출 방법은 3.3절에서 정의한다.

3.3 Retry Limit 조절 기법 및 재전송 타이머 산출 방법

IEEE 802.11은 CSMA/CA(Carrier Sense Multiple Access

with Collision Avoidance)를 기본적인 매체 접근 메커니즘으로 사용한다. 또한, 하나의 추가 옵션으로써 MAC 계층의 데이터 패킷과 응답 패킷을 주고받기 전에 RTS/CTS(Request-To-Send/Clear-To-Send) 패킷을 교환한다. 이러한 매체 접근 방법은 무선구간에서 패킷 충돌을 최소화 하는 효율적인 방법으로 알려져 있다. 그림 6은 이러한 매체 접근 방법을 이용한 하나의 MAC 프레임과 TCP 패킷의 전송과정을 나타낸 것이다.

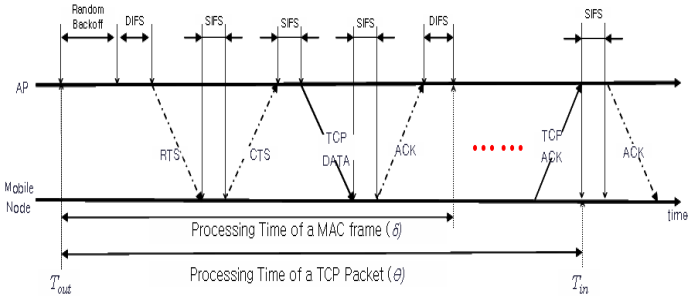

그림 6. TCP 패킷 및 MAC 프레임의 전송과정

T_{out}에서 소스 노드는 데이터를 전송하기 위해서 매체 접근 메커니즘을 수행하며, 매체가 전송 범위 내의 다른 노드에 의해 점유되었음을 감지하고 Backoff 윈도우의 초기 크기와 함께 지수급수 백오프(Exponential backoff) 알고리즘을 수행한다. 그 후에 소스 노드는 매체가 다른 노드에 의해 점유되지 않음을 감지하고, 매체를 점유하기 위해 RTS 프레임을 목적지 노드로 전송한다. RTS 프레임을 수신한 노드는 SIFS(Short Inter-Frame Space)의 시간 후에 자신의 전송범위 내에 있는 모든 노드의 NAV(Network Allocation Vector) 갱신을 위해 CTS 프레임으로 응답한다. 이러한 과정 이후에 소스 노드는 데이터 패킷을 전송하며, 그 패킷을 정확히 수신한 목적지 노드는 ACK 패킷을 전송한다. T_{in}에서 소스 노드는 목적지 노드로부터 TCP-ACK 패킷을 수신하게 된다.

A^{2}Snoop은 손실된 패킷을 전송계층의 Timeout 발생 이전에 손실된 패킷을 복구하기위해서 지역 재전송 타이머를 사용한다. 타이머를 산출하기위해 앞서 언급한 기본적인 매체 접근 및 데이터 패킷 전송 과정을 고려하여 무선 구간의 RTT(W_{RTT})를 식(2)와 같이 산출한다.

$$W_{RTT} = T_{in} - T_{out} \quad (2)$$

$T_{in} - T_{out}$은 노드가 매체에 접근하기 위해서 지연되는 모든 시간을 포함한다. 이와 유사한 산출 방법은 Giuseppe Bianchi에 의해 처음으로 표현되었다[6].

손실된 패킷을 효율적으로 복구하기 위해서 A^{2}Snoop은 전송계층에서 사용되는 RTO(Retransmission Timeout) 계산 방법과 유사하게 W_{RTT}를 기반으로 무선구간의 RTO(W_{RTO})를 산출한다. W_{RTO}는 A^{2}Snoop agent가 ARQ-interface로부터 Channel-On 이벤트를 수신한 경우 계속해서 새롭게 갱신되며, 산출 과정은 식 (3)와 같다.

$$\begin{aligned} SW_{RTT} &= (1-a)\times SW_{RTT} + a\times W_{RTT} \\ RW_{RTT} &= (1-\beta)\times RW_{RTT} + \beta\times |SW_{RTT} - W_{RTT}| \\ W_{RTO} &= SW_{RTT} + 4RW_{RTT} \end{aligned} \quad (3)$$

A^{2}Snoop agent는 ARQ-interface로부터 Channel-Off 이벤트를 수신한 경우에, 재전송을 위해 저장된 패킷에 타이머를 적용하여 관리한다. A^{2}Snoop의 재전송 타이머는 FH의 Timeout의 발생 이전에 손실된 패킷을 복구하기 위해서 식 (3)의 W_{RTO}값을 사용한다.

또한, A^{2}Snoop은 무선 채널의 상태에 따라 Retry Limit값을 조절하기 위해서 MAC 프레임의 처리시간을 기준으로 한다. 이는 프레임의 전송이 원활이 이루어지고 있을 경우에 소모되는 패킷의 처리 시간은 다른 Flow의 전송 시간 지연에 영향을 주지 않는 최소 단위이기 때문이다. 이러한 MAC 프레임의 계산 방법은 식 (4)와 같이 계산한다.

$$\delta_i = \sum_{i=0}^{RL} T_{Backoff}(i) + (RL+1)T_{tx} \quad (4)$$

$T_{backoff}$는 MAC 계층의 전송 과정에서 발생하는 Backoff 시간을 나타낸 것이고 T_{tx}는 하나의 MAC 프레임을 전송하기 위해 소모되는 시간을 나타낸 것이다. 이 두 값의 계산 방법은 식 (5), (6)과 같다.

$$\begin{aligned} T_{Backoff}(i) &= Random(0, CW_i)\times timeslot \\ &= CW_i/2 \times timeslot = (2^{k+i}-1)/2 \times timeslot \end{aligned} \quad (5)$$

$$\begin{aligned} T_{tx} &= t_{DIFS} + t_{PHY} + t_{MPDU} + t_{SIFS} + t_{PHY} + t_{ACK} \\ &= t_{DIFS} + t_{SIFS} + 2D_{PHY}/Basic\_rate + (D_{MPDU}+D_{ACK})/Data\_rate \end{aligned} \quad (6)$$

A^{2}Snoop은 Channel-On 이벤트가 발생한 경우 전송을 성공하기위해 수행한 재전송 과정까지 고려한 시간을 기반으로 Channel-Off 이벤트가 발생 했을 때 식 (7)과 같이 계산된 값을 기분으로 재전송 횟수를 결정한다.

$$\overline{\delta_i} = \alpha \times \overline{\delta_{i-1}} + (1-\alpha) \times \overline{\delta_i} \quad (7)$$

이러한 MAC 프레임의 처리시간을 기반으로 A^{2}Snoop은 Channel-Off 상태가 되었을 때 현재 전송 버퍼에 다른 Flow가 존재하지 않을 경우 Retry Limit의 증가가 다른 Flow의 전송 시간에 영향을 주지 않으므로, 그림 7과 같이 재전송 타이머보다 Retry Limit값이 크지 않는 한 Retry Limit의 값을 $\overline{\delta}$만큼씩 증가시켜 손실된 패킷을 효율적으로 복구한다.

만약 다른 Flow가 전송 버퍼 안에 존재한다면, 그림 8과 같이 불필요한 Retry Limit증가로 인한 다른 Flow의 전송률 저하를 방지하기 위해서 Retry Limit 값을 $\overline{\delta}$으로 줄이고 재전송 타이머를 이용하여 TCP의 Timeout 이전에 손실된 패킷을 복구한다.

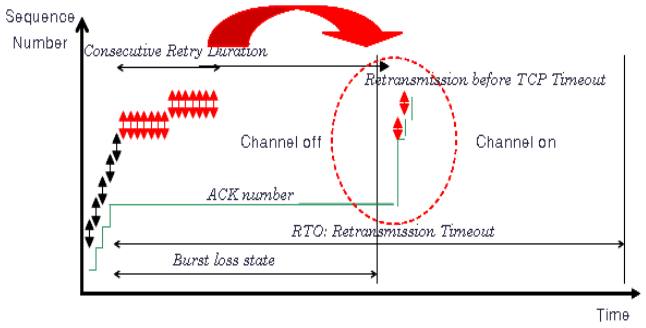

그림 7. 증가된 Retry Limit으로 손실된 패킷 복구

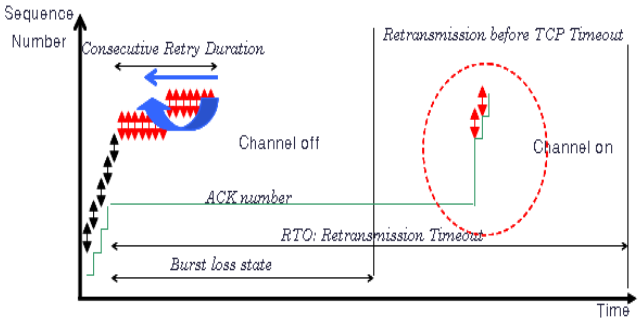

그림 8. 재전송 타이머를 이용한 손실된 패킷 복구

4. 실험 및 성능 평가

본 장에서는 새로이 제안한 A^{2}Snoop 프로토콜의 성능 평가를 위해 무선 구간의 연집적인 패킷 손실이 발생하는 유·무선 혼합망에서 전송 성능 및 이동 단말의 에너지 효율성 실험을 LBNL(Lawrence Berkely National Laboratory)의 ns-2(network simulator)를 사용하여 수행하였고, 기존의 지역 재전송 메커니즘인 SNACK 메커니즘, MAC-layer LDA와 비교, 평가 하였다[7].

유·무선 혼합망은 신뢰성이 상대적으로 낮은 전송 매체인 무선 채널을 포함하고 있어 전송 에러에 의한 패킷 손실이 빈번히 발생한다. 따라서 본 절에서는 전송 에러에 의해 패킷 손실이 발생하는 무선 채널을 가진 유·무선 혼합망에서 A^{2}Snoop 프로토콜의 성능 평가를 위해 SNACK 메커니즘, MAC-layer LDA와 전송 성능 및 이동 단말의 에너지 효율을 비교하였으며, 결과를 통해 A^{2}Snoop 프로토콜이 기존의 연구에 비해 유·무선 혼합망에서 더 효율적으로 동작한다는 것을 확인 하였다.

4.1. 실험 환경

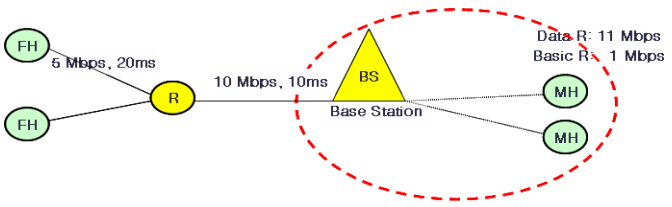

그림 9. 실험 환경

유·무선 혼합망에서 제안한 A^{2}Snoop 프로토콜의 성능을 평가하기 위해 그림 4과 같은 실험 환경을 구성하였다. 실험을 위한 파라미터 값은 IEEE 802.11b의 기본 값을 사용하였다.

성능 평가를 위해 채널 오류에 의한 연집적인 패킷 손실률을 0%~10%로 각각 나누어 실험 하며, 실험환경에서 2개의 Flow는 총 60초 동안 1Kbytes 크기의 패킷을 계속적으로 전송한다.

4.2. 전송 방향에 따른 전송률 실험

그림 10은 무선 구간의 연집적인 패킷 손실률을 5%으로 두어 100s동안 두 가지 전송 방향(FH->MH)에 대해 실험한 결과이다. 그림 10에서 알 수 있듯이 SNACK 메커니즘에 비해 빠른 지역 재전송을 수행하고 무선 구간의 연속된 패킷 손실 구간에서 ACK 패킷의 손실에 영향을 받지 않는 A^{2}Snoop 프로토콜의 성능이 전반적으로 높은 전송률을 보이는 것을 확인할 수 있다. 또한, MAC-layer LDA는 단순히 Retry Limit 값을 기본 값에서 18개 까지 증가시키므로, 연집 손실률이 높음 환경에서는 효율적인 복구가 이루어지지 않는것을 확인할 수 있었다.

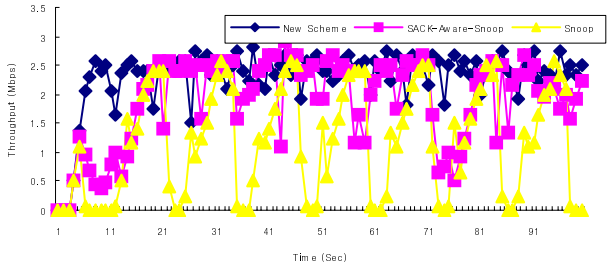

그림 10. 5. 5% 연집 손실률에 따른 전송률 (FH→MH)

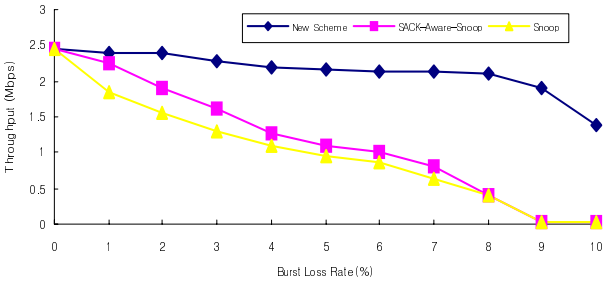

그림 11. 패킷 손실률에 따른 평균 전송률 (FH→MH)

그림 11은 무선 구간의 연집적인 패킷 손실 발생률을 0%~10%까지 변화 시켜가며 100초 동안 SNACK, MAC-layer LDA와 A^{2}Snoop 프로토콜의 평균 전송률에 대한 실험 결과이다. 기존의 MAC-layer LDA와 SNACK과 같은 지역 재전송 메커니즘은 연집적인 패킷손실률이 높아짐에 따라 한 전송윈도우 내의 대부분의 패킷이 손실되거나, 다수의 전송 계층의 ACK 패킷이 손실될 경우 심각한 성능 저하를 가져온다. 반면에, A^{2}Snoop 프로토콜은 연집적인 패킷 손실율의 증가에 심각한 전송 성능 저하가 발생하지 않는 것을 확인할 수 있다.

5. 결론 및 향후 과제

본 논문에서는 MAC-layer LDA와 SNACK이 가지는 문제점을 분석하고, 이를 해결하기위해 MAC 계층과 전송 계층의 연동을 통해 재전송을 수행하는 기법을 제안하였다. 제안된 기법은 빠른 재전송을 수행하고, 기존 프로토콜의 문제점을 해결하였다.

향후에는 기존 재전송 프로토콜의 또 다른 취약점인 BS의 버퍼관리 및 WPAN(Wireless Personal Area Network)과 같은 다른 네트워크 환경에 대해서도 고려할 것이다.

6. 참고문헌

[1] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," IEEE/ACM Transactions on Networking, December 1997.

[2] S. Lohier, Y. Doudane and G. Pujolle "MAC-layer Adaptation to Improve TCP Flow Performance in 802.11 Wireless Networks," Wireless and Mobile Computing, Networking and Communications, 2006.

[3] Fanglei Sun, Li, V.O.K., Liew, S.C.," Design of SNACK mechanism for wireless TCP with new snoop," Wireless Communications and Networking Conference, 2004.

[4] V. Raisinghani and S. Iyer, "Cross-layer Design Optimization in Wireless Protocol Stack," Computer Communications(Elsevier), May 2004.

[5] G. Xylomenos and G. Polyzos, "Quality of Service Issues in Multi-service Wireless Internet Links," Proceedings of The International Workshop on QoS in Multi-service IP Networks, pp. 347-364, 2001.

[6] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function,"IEEE Journal on Selected Areas in Communications, March 2000.

[7] The network simulator ns-2, http://www.isi.edu/nanam/ns/