TCP 혼잡제어 알고리즘을 이용한 무선 메쉬 네트워크의 성능 개선

Performance Improvement of Wireless Mesh Networks using TCP Congestion Control Algorithm

이혜림*, 문일영*

Hye-Rim Lee*, Il-Young Moon*

요 약

무선 메쉬 네트워크는 Base Station 기반과 더불어 애드혹 네트워크나 블루투스와 같이 유연성을 가진 네트워크이다. 하지만 무선 메쉬 네트워크는 높은 패킷 손실률을 보이고 TCP(Transport Control Protocol) 알고리즘은 무선 메쉬 네트워크의 패킷손실 원인을 네트워크 내의 혼잡으로 인식하기 때문에 TCP 혼잡제어 알고리즘(Congestion Control Algorithm)을 실행하게 된다. 이러한 TCP 동작은 혼잡 손실이 아닌 패킷 손실로 발생 할 경우, 상당한 성능 저하를 초래하게 된다. 본 논문에서는 무선 메쉬 네트워크에서 TCP 혼잡제어 알고리즘의 성능을 개선시키기 위해 혼잡 윈도우를 무선망에서도 적응력 있게 조절하는 연구를 제안하였다.

Abstract

Wireless mesh network is flexible network like Ad hoc network or bluetooth together based on base station. But, wireless mesh network shows high packet loss and when TCP was created, however as it was design based on wired link, wireless link made more transmission error than wired link. It is existent problem of TCP congestion control algorithm that TCP unfairness and congestion collapse over wireless mesh network. When TCP operation occurs with the packet loss where is not the congestion loss, it brings the performance degradation which is serious. In this paper, in order to improve efficient TCP congestion control algorithm in wireless mesh network, we proposed that TCP can adaptively regulate the congestion window in wireless link.

Key Word : Wireless Mesh Network, TCP, TCP Congestion Control Algorithm, Congestion Window, Mobile Node

I. 서 론

IP는 단순히 패킷만 전달하려는 목적으로 만들어졌기 때문에 TCP는 이러한 IP에 데이터 패킷을 안전하게 전달할 수 있도록 네트워크상의 여러 가지 변수들을 고려하여 최대한 목적지까지 정확히 전송되도록 유지시켜준다. 여러 가지 변수 들 중에서 가장 TCP의 성능을 좌지우지 하는 것은 혼잡 윈도우이다[1],[2]. 혼잡 윈도우는 송신자가 데이터 패킷을 보낼 수 있는 크기를 말하며, 데이터 패킷은 혼잡 윈도우

* 한국기술교육대학교 정보미디어공학과(Information Media Eng., Korea University of Technology and Education)
· 제1저자 (First Author) : 이혜림
· 투고일자 : 2010년 2월 9일
· 심사(수정)일자 : 2010년 2월 9일 (수정일자 : 2010년 3월 18일)
· 게재일자 : 2010년 4월 30일

사이즈에 맞춰 데이터 패킷을 전송한다. 이러한 혼잡 윈도우는 TCP의 혼잡제어 알고리즘에서 조절하게 된다[3],[4]. 즉, TCP 혼잡제어(congestion control) 알고리즘은 네트워크로 들어가는 정보 소통량을 조절하여 네트워크가 혼잡해지지 않게 조절하는 것을 말한다. 예를 들어, 정보 소통량이 과다한 것을 감지하여 패킷을 적게 보내면 혼잡 붕괴 현상이 일어나는 것을 막을 수 있다[5].

초기의 혼잡제어 알고리즘은 흐름제어와 혼잡회피 단계로 이루어진 간단한 구조였으나, 인터넷의 발달로 인해 고속의 데이터와 대량의 패킷 전송 등의 이유로 끊임없이 업그레이드되어 왔다. 그리고 유선 네트워크의 발전이 주춤해지자 무선 네트워크가 급속한 성장세를 보였고, 이로 인해 무선 네트워크에 적용할 수 있는 TCP 혼잡제어 알고리즘들이 연구되고 있다.

이에 본 논문에서는, 유비쿼터스의 새로운 인프라로 주목받고 있는 무선 메쉬 네트워크에서 TCP 혼잡제어 알고리즘의 성능을 향상시키기 위해 혼잡 윈도우를 무선망에서도 적응력 있게 조절하는 연구를 하였다.

Ⅱ. 배경 및 관련연구

2-1 무선 메쉬 네트워크

무선 메쉬 네트워크는 전통적인 네트워크 기술인 Base Station 기반과 더불어 애드혹 네트워크나 블루투스와 같이 유연성을 가진 네트워크로, 각각의 구성 디바이스들 간에 데이터 통신을 하는 주체가 되고 같은 네트워크안의 다른 디바이스들로 부터 받은 데이터 패킷을 다른 디바이스로 릴레이해주고 라우팅해주는 기능을 갖는다.

무선 메쉬 네트워크는 모든 단말기들이 라우터의 역할을 하며 유저 단말기 쪽만 보면 애드혹 네트워크와 흡사하다는 것을 알 수 있다. 하지만 무선 메쉬 네트워크의 특징은 인터넷 망으로 연결해주는 고정된 라우터들이 존재하고 이러한 라우터들은 애드혹 통신을 하는 단말기들을 서로 다른 망으로 빠르게 연결할 수 있다. 즉, 그림1과 같이 무선 인프라 환경에서 고정이나 이동 중에 모든 노드가 통신을 할 수 있는 다중경로를 가지는 메쉬형 토폴로지를 구성한다.

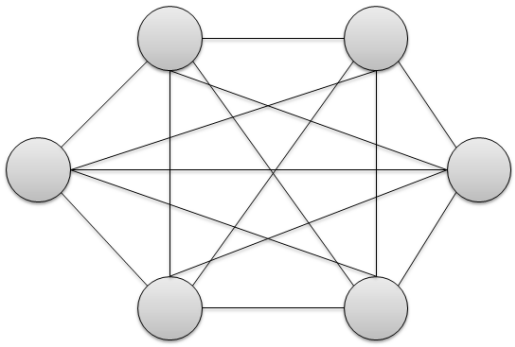

그림 1. 메쉬형 토폴로지(Mesh Topology)
Fig. 1. Mesh Topology

무선 엑세스 포인트를 통해 각종 단말기를 연결하는 기존의 무선 네트워크는 수신 범위를 벗어나면 연결이 끊어지는 데 비해 무선 메쉬 네트워크는 그림1과 같이 각 단말기들이 그물망처럼 연결되므로 네트워크를 혁신적으로 확장할 수 있다. 따라서 무선 메쉬 네트워크는 액세스 포인트가 커버해야 할 커버리지가 비교적 많은 광범위한 지역에서 인터넷 망을 사용할 때 효율적으로 커버리지를 확장하고 엑세스 포인트의 설치비용을 절감하는 등의 효과가 있다.

2-2 TCP 혼잡제어 알고리즘

TCP는 흐름제어(Flow Control)와 혼잡제어 메커니즘을 이용하여 End-to-End 간의 신뢰성 있는 전송을 보장한다. 흐름제어는 송신자가 수신자로부터 슬라이딩윈도우 크기를 받은 후 그것보다는 적게 보냄으로써 네트워크상의 흐름을 시간이나 크기단위로 조절하는 방법이고 혼잡제어는 송신자가 네트워크 상황을 보고 스스로 패킷을 조절하는 방법이다. 라우터가 부하를 감당할 수 없게 되어 심각한 지연이나 패킷이 손실 되는 경우를 혼잡이라고 한다. 빠른 LAN에서 느린 WAN으로 전송하거나 다수의 입력 스트림이 출력 수용량이 입력보다 적은 라우터에 도착했을 때 혼잡이 발생하는데 이런 문제를 해결하기 위한 방

법을 혼잡제어라고 한다[6].

TCP의 혼잡제어는 1980년대 반 제이콥슨이 도입한 것으로, 초기에 컴퓨터들은 정보를 빨리 보내기 위하여 정해진 시간 내에 보낼 수 있는 최대의 패킷을 보냈다[7]. 이 때문에 일부 라우터에서는 혼잡 현상이 발생하여 정해진 시간 내에 받은 패킷들을 모두 처리하지 않아 결과적으로 네트워크의 혼잡 현상이 더 심해졌다. 이러한 혼잡현상을 막기 위해 TCP 혼잡제어 기법이 탄생하였으며 TCP의 혼잡제어는 송신단에서 보내는 패킷의 양을 조절한다.

혼잡 윈도우는 송신자가 데이터 패킷을 보낼 수 있는 크기를 말하고, TCP는 혼잡 윈도우 값을 사용하여 송신단에서 전송하려는 패킷의 크기를 조절하여 송신측은 혼잡 윈도우 사이즈에 맞춰 데이터 패킷을 전송한다. 이러한 혼잡 윈도우는 TCP의 혼잡제어 알고리즘에서 조절하게 된다.

이러한 TCP 혼잡제어 알고리즘은 슬로우 스타트(Slow Start), 혼잡회피(Congestion Avoidance), 빠른 재전송(Fast Retransmission), 빠른 복구(Fast Recovery)인 4가지의 메커니즘으로 이루어져 있다.

Ⅲ. 무선 메쉬 네트워크에서 개선된 TCP 혼잡제어 알고리즘

3-1 개선된 TCP 혼잡제어 알고리즘

본 논문에서 제안된 TCP 혼잡제어 알고리즘은 패킷을 전송하다가 연결이 끊기면 TCP 패킷에 경로가 변경되어 전송이 중지되었다는 것을 송신 노드에게 알려 중복 ACK를 받아도 Cwnd(Congestion Window)를 1로 줄여 슬로우 스타트 단계로 들어가는 것을 막는다. 또한 Cwnd는 전송이 끊기기 전의 Cwnd값과 현재의 ssthresh(slow-start threshold)값 중에서 작은 값을 비교하여 고른다. 이렇게 Cwnd값과 현재의 ssthresh 값 중에서 작은 값을 비교하여 고르게 되면 안정적이고 적당한 값을 뽑아낼 수 있게 된다. 또한 기존의 TCP는 경로 변경에 따른 전송 에러를 혼잡상황으로 인식하여 불필요한 혼잡제어 메커니즘을 실행하지만, 제안된 TCP는 전송에러에 대한 원인을 정확히 감지하여 혼잡제어 메커니즘을 실행하지 않고 Cwnd 값을 극단적으로 낮추지 않으며 적절한 대처를 하게 된다.

반면에 기존의 TCP는 패킷을 전송하다가 연결이 끊긴다면 송신 노드는 마지막 데이터에 대한 ACK를 받지 못해서 계속 패킷을 보내고 결국엔 중복 ACK가 3개 이상 되어 Cwnd/2로 변경하거나 패킷을 받지 못해 타임아웃을 실행하여 슬로우 스타트 단계로 들어가 Cwnd를 1로 변경하게 된다. Cwnd를 1이나 Cwnd/2로 변경함에 따라 보낼 수 있는 데이터의 량이 적어지고 이것은 TCP의 패킷 처리량을 저하시키는 원인이 된다.

3-2 개선된 TCP의 혼잡제어 알고리즘의 진행방식

그림 2를 보면, 노드들의 이동으로 경로가 변경되었을 때 패킷 손실이 발생되어(Move route) 송신노드는 ssthresh값을 1/2로 줄이고 중복 ACK를 3개 이상 받아 빠른 재전송과 빠른 복구를 실행한다.

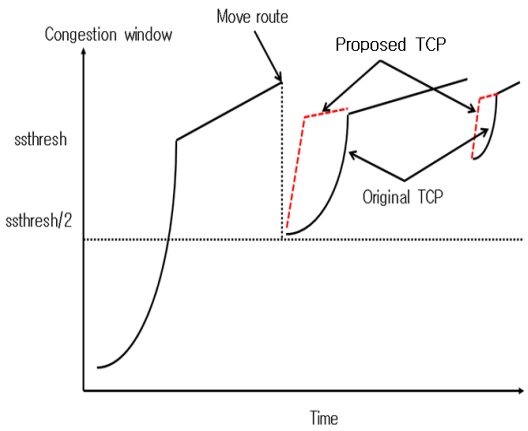

그림 2. 기존의 TCP와 제안된 TCP의 비교
Fig. 2. Comparison of Original TCP and Proposed TCP

이때 반으로 줄어든 ssthresh값에서부터 Original TCP는 천천히 지수적인 증가를 하는 것에 비해 proposed TCP는 경로변경에 의한 중복 ACK임을 알고 네트워크 혼잡이라 판단하지 않아 Cwnd값을 패킷이 손실되기 전의 크기로 보낸다. 이로 인해 사용할

수 있는 윈도우 크기를 효과적으로 쓰기 때문에 결과적으로 패킷의 처리량이 증가하게 된다.

Ⅳ. 시뮬레이션 모델 및 결과

4-1 시뮬레이션 환경

본 논문에서는 QualNet 4.5를 이용하여 무선 메쉬 네트워크에서 개선된 TCP 혼잡제어 알고리즘의 성능을 평가하도록 한다.

토폴로지는 그림 3과 같은 구조로 이루어져 있으며, 메쉬 라우터와 메쉬 클라이언트로 이루어진다. 그림 3을 보면 메쉬 클라이언트들은 메쉬 라우터들을 통해 서버와 연결하게 된다. 또한 메쉬 라우터들은 서로간의 통신을 할 수 있으며 AODV(Ad hoc On-Demand Distance Vector Routing) 라우팅 프로토콜을 사용하기 때문에 목적지까지 여러 홉을 거쳐 패킷 전달을 할 수 있다.

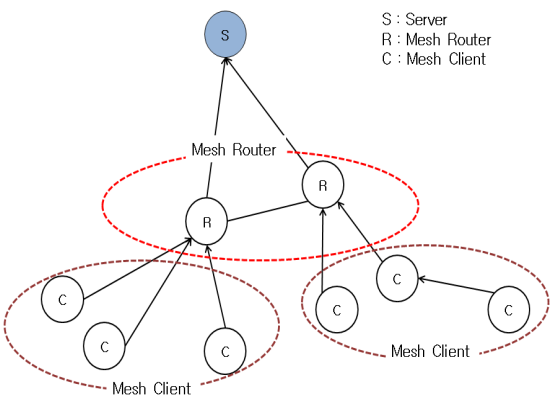

그림 3. 기본 토폴로지의 구성도
Fig. 3. Proposed TCP Topology Structure

기본적으로 링크계층은 IEEE 802.11 MAC 프로토콜 표준을 사용하였으며[8], 대역폭은 15Mbps이고 전송 지연 시간은 10ms, 패킷 크기는 512 byte로 설정하고, 응용프로그램은 FTP(File Transfer Protocol)를 사용하였다. 그림 3과 같이 토폴로지를 구성하고, 이 토폴로지는 1000M*1000M이내의 평탄한 경우라고 지정한다. 비교 대상이 되는 TCP 알고리즘은 현재 가장 널리 쓰이고 있는 TCP Reno와 버스트한 패킷 손실에 강한 TCP New-Reno를 사용하여 비교하였다.

시뮬레이션 시간은 총 120초로, 노드들은 40초 이후가 되면 서서히 움직이기 시작한다. 그리고 30~50초 후 다시 경로를 변경하고 노드들이 서버 쪽으로 이동한다. 즉, 70~90초 사이에 서버와 거리가 멀어진 클라이언트들은 연결이 완전히 단절되어 패킷을 전송하지 못하고, 90초 이후엔 거의 모든 노드들이 다시 제자리를 찾아 서버와 연결되어 패킷을 재전송하게 된다.

노드의 개수가 8개이며 이 중 서버 1개에 2개의 메쉬 라우터를 통해 5개의 메쉬 클라이언트가 그림 3과 같은 구성으로 데이터를 전송한다. 라우터 역할을 하는 노드들은 메쉬 라우터의 특성으로 움직이지 못하지만, 클라이언트 노드들은 자유로이 이동하게 된다. 클라이언트의 움직임은 일정한 이동성을 가지고 있으며, 클라이언트 노드가 이동할 땐, 대부분의 패킷이 전송되지 않는다. 이 과정을 통해 경로 변경으로 인한 패킷 유실을 기존의 TCP는 혼잡상황으로 인식하게 되지만 제안된 TCP는 패킷이 경로변경으로 유실되었음을 인지하여 경로이동에 따른 효과적인 TCP 알고리즘의 성능을 분석할 수 있다.

4-2 시뮬레이션 결과 분석

그림 4는 노드가 8개 일 때 서버의 ssthresh값을 시간대 별로 나타낸 것이다. sshresh값은 슬로우 스타트 단계에서 사용하는 윈도우 임계값이라고 불리며 혼잡 윈도우는 ssthresh값을 기점으로 이 값에 도달하면 지수적인 증가를 멈추고 혼잡회피단계로 들어가 1씩 증가하게 된다. 제안된 TCP 혼잡제어 알고리즘은 기존의 TCP 혼잡제어 알고리즘보다 상대적으로 높은 ssthresh값을 유지하고 있다. 이렇게 가용한 윈도우를 활용함으로써 더 많은 데이터를 전송하게 됨으로 결과적으로는 앞서 본 시뮬레이션 결과그래프처럼 기존의 TCP 혼잡제어 알고리즘에 비해 높은 성능을 보이게 되는 것이다.

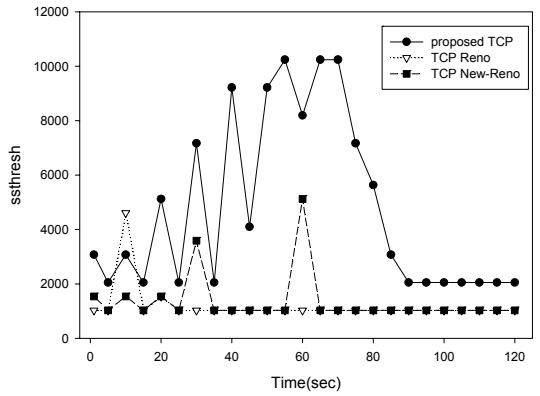

그림 4. 송신자의 ssthresh 변화추이
Fig. 4. ssthresh Flow of Sender

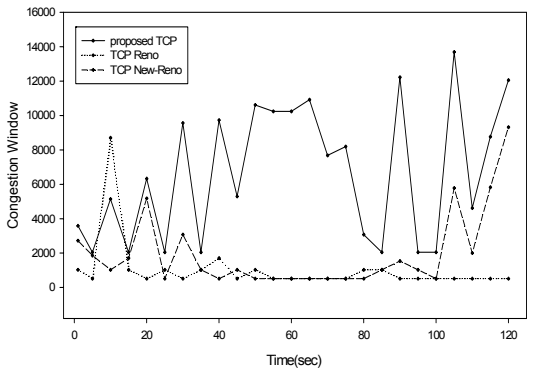

그림 5. 혼잡 윈도우 크기
Fig. 5. Congestion Window Size

그림 5는 노드가 8개 일 때 혼잡 윈도우의 크기를 나타낸 것이다. 시뮬레이션 시간은 총 120초 중에서 40초부터 90초까지는 클라이언트 노드들이 경로를 유니폼하게 이동하고 움직이기 때문에 패킷이 유실될 가능성이 많다. 기존의 TCP 혼잡제어 알고리즘은 이러한 패킷 유실로 인해 혼잡 윈도우가 줄어들게 되고 네트워크가 받을 수 있는 양은 충분하지만 그 가용한 대역을 사용하지 않아 성능 저하가 발생한다. 반면에 제안된 TCP 혼잡제어 알고리즘은 노드가 이동하고 움직여도 혼잡상황이 아니라 단순한 경로변경임을 알고 혼잡 윈도우의 크기를 그전과 비슷하게 유지한다. 이는 경로가 연결되어도 언제든지 신속하게 많은 양의 데이터를 보낼 수 있으며, 사용할 수 있는 공간을 최대한 활용함으로써 대역의 낭비를 줄이게 된다. 그림 6을 보면 연결이 끊기고 다시 재설정되는 80초부터 120초 까지 시뮬레이션된 혼잡 윈도우의 크기를 나타내는 그래프이다. 그림에서 보는 것과 같이 80초에서 TCP New-Reno는 혼잡 윈도우가 천천히 올라가는 반면에 제안된 TCP 혼잡제어 알고리즘은 거의 수직방향으로 높게 올라간 것을 볼 수 있다. 즉, 혼잡 윈도우에는 제안된 TCP 혼잡제어 알고리즘은 올라간 혼잡 윈도우만큼의 쓸 수 있는 공간이 남아 있음에도 불구하고, TCP New-Reno는 천천히 그리고 조금씩 올라가서 효율적으로 데이터를 보내지 못하는 단점이 있다. 따라서 제안된 TCP 혼잡제어 알고리즘은 이러한 단점을 극복해주기 때문에 전체적인 성능이 높게 나오고 또한 기존의 TCP 보다 성능이 향상되었음을 알 수 있다.

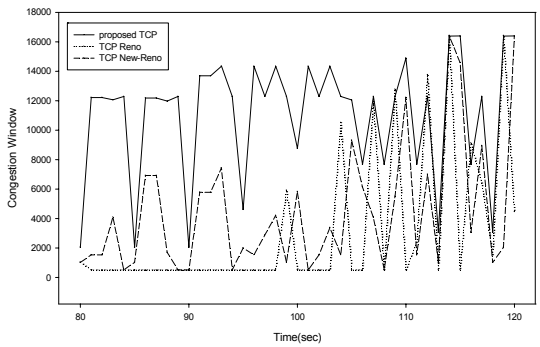

그림 6. 송신자의 혼잡 윈도우 크기(80초 이후부터)
Fig. 6. Congestion Window Size of Sender

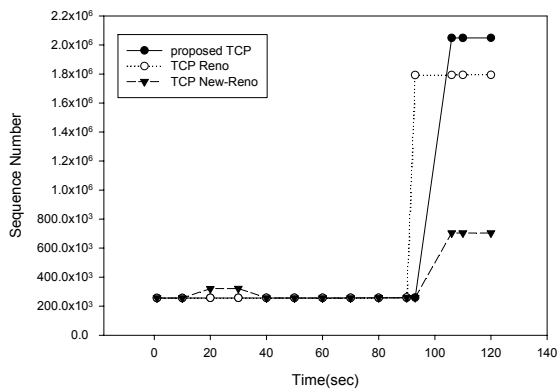

그림 7. 수신자에게 받은 순차번호
Fig. 7. Sequence Number Received from Sender

그림 7은 노드가 8개 일 때, 서버에서 받은 TCP의 순차번호이다. TCP의 순차번호는 데이터를 전송하

기 전 맨 처음 임의의 숫자를 정한 뒤, 그 번호를 ACK 메시지와 함께 이어나간다. TCP의 순차번호를 확인함으로써 TCP를 통해 데이터가 순서가 바뀌지 않고 제대로 들어왔음을 나타내준다. 특히나 무선 네트워크 환경에서는 TCP의 데이터가 순서가 뒤바껴서 오는 경우가 많고 이러한 경우 TCP는 패킷이 순서대로 오지 않아 패킷 재전송을 요구하는 ACK를 보내고 이후에 순서가 뒤바뀐 패킷이 오게 되었을 때 중복 ACK를 3회 이상 보내면 불필요한 혼잡제어를 실행하게 되어 상당한 성능 저하로 이어진다. 따라서 TCP의 순차번호를 확인하는 것은 중요하고 또한 순차번호와 그림 8과 같이 누적 ACK메시지를 확인하는 것도 필요하다. 그림 8은 노드가 8개 일 때 서버에서 받은 누적 ACK 번호이며 기존의 TCP 혼잡제어 알고리즘보다 제안된 TCP 혼잡제어 알고리즘이 더 좋은 성능을 보이는 것을 알 수 있다.

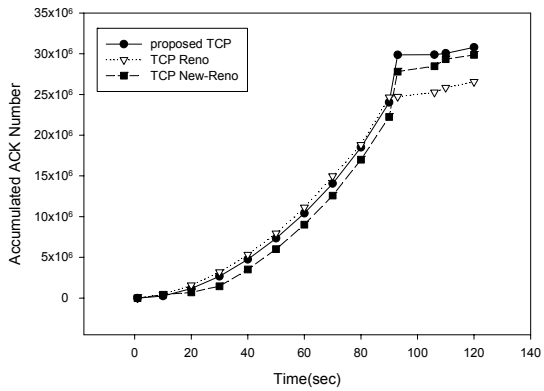

그림 8 서버에서 받은 누적 ACK번호
Fig. 8. Accumulated ACK Number Received from Sender

V. 결 론

본 논문에서는 무선 메쉬 네트워크에서 개선된 TCP 혼잡제어 알고리즘을 제안하였고 QualNet 4.5의 시뮬레이션을 통해 제안된 TCP 혼잡제어 알고리즘이 기존의 TCP 혼잡제어 알고리즘보다 성능이 향상되었음을 확인하였다. 또한 TCP 혼잡제어 알고리즘이 이동성을 가진 무선 메쉬 네트워크 환경에서 얼마만큼 영향을 미치는지 알아보았고, 대표적인 방식인 TCP Reno, TCP New-Reno를 본 논문에서 제안한 TCP 혼잡제어 알고리즘과 동일한 상황에서 모의 실험하였으며, 제안된 TCP 혼잡제어 알고리즘이 무선 메쉬 네트워크 환경에서 더 효율적 성능을 발휘하는 것을 볼 수 있었다.

제안된 TCP 혼잡제어 알고리즘은 노드가 이동함에 따라 생기는 패킷 손실에 대해 기존의 혼잡제어를 실행하지 않고 적응력 있게 혼잡 윈도우의 크기를 조절하여 노드가 움직여서 일시적인 패킷 손실이 발생해도 윈도우 사이즈를 그전과 비슷하게 유지하여 평균적인 패킷 처리량을 높였다. 현재 TCP 혼잡제어 알고리즘의 가장 큰 문제점은 패킷 손실의 원인을 정확히 파악하지 못한 채 대부분의 상황에서 혼잡제어를 실행한다는 점이다. 따라서 본 논문에서 제안한 TCP 혼잡제어 알고리즘은 무선 메쉬 네트워크에서 상대적으로 많이 발생하는 패킷 손실 원인인 이동에 따른 경로 변경을 정확히 감지할 수 있는 TCP 혼잡제어 알고리즘을 제안하였다.

참 고 문 헌

[1] G. Xylomenos and G. C. Polyzos, "TCP Performance Issues over Wireless Link," *IEEE Communications Magazine, vol. 39*, pp.52-58, April 2001.

[2] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *RFC-2001*, Jan. 1997.

[3] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP," *Comput. Commun. Rev.*, 1996.

[4] C. Barakat, E. Altman, and W. Dabbous, "On TCP Performance in a Heterogeneous Network: A Survey," *IEEE Communications Magazine*, Jan. 2000.

[5] L. Patterson, Larry and S. Bruce, Davie, Computer Networks 3rd edition, *Morgan Kaufmann,* pp. 468-478, 2003.

[6] W. Richard Stevens, TCP/IP Illustrated, Reading MA, volume 1. Addison-Wesley, 1994.

[7] V. Jacobson, "Congestion avoidance and control," *Symposium proceedings on Communications architectures and protocols*, p.314-329, August 16-18, 1988.

[8] IEEE 802.11 working group, “IEEE Std 802.11, 1999 Edition,” available from http://standards.ieee.org/catalog/olis/laman.html, ISO/IEC 8802-11, 1999.

이 혜 림 (李憓琳)

2008년 8월 : 한국기술교육대학교 인터넷미디어공학부 졸업 (공학사)
2008년 9월 ~현재 : 한국기술교육대학교 대학원 정보미디어공학과 재학 (공학석사)
관심분야 : 무선 TCP, 무선 메쉬 네트워크, 라우팅 프로토콜

문 일 영 (文日永)

2000년 2월 : 한국항공대학교 항공통신정보공학과 (공학사)
2002년 2월 : 한국항공대학교 대학원 항공통신정보공학과 (공학석사)
2005년 2월 : 한국항공대학교 대학원 정보통신공학과 졸업(공학박사)
2004년 ~2005년 : 한국정보문화진흥원 선임연구원
2005년 3월~현재 : 한국기술교육대학교 인터넷미디어 공학부 조교수
관심분야 :무선 인터넷 응용, 무선 인터넷, 모바일 IP