논문번호 08-01-02

GMAHN 환경에서의 TCP 프로토콜 성능 분석

TCP Protocol Performance Evaluation of GMAHN

Se-Duk Oh* , Jae-Ho Kim* , Jin-Seung Bae* , Chan-Hyuk Jung* , Chi-Moon Lee* ,
Jae-Seung Ha* , Choong-Yeul You* , Kwang-Bae Lee* , Hyun-Wook Kim*
오 세 덕* , 김 재 호* , 배 진 승* , 정 찬 혁* , 이 치 문* ,
하 재 승* , 유 충 렬* , 이 광 배* , 김 현 욱*

Abstract

Recently, GMAHN that provides interface between MANET and Wired Network has been focused in mobile communication. It is necessary that the technology provide reliable data transmission technology between mobile node and wired network in MANET environment that is varied by the node movement. In this paper, using the TCP protocol(Tahoe, Reno, Vegas, SACK)that increases reliability between source and destination, we applied the TCP protocol mechanism to various environment, and proposed the most efficient TCP mechanism by comparing each mechanism.

요 약

최근들어 MANET(Mobile Ad-hoc Network)망과 기존 유선망을 연결한 GMAHN(Global Mobile Ad Hoc Network)에 대한 관심이 이동 통신분야에서 증가하고 있다. 시시각각 변화하는 노드의 이동성으로 인해 토폴로지가 지속적으로 변화하는 MANET 환경에서 이동 노드와 유선망간의 신뢰성 있는 데이터 전송을 위한 기술은 필수적이라고 할 수 있다. 본 논문에서는 GMAHN 환경에서 송신단과 수신단사이의 신뢰성 있는 데이터 전송을 보장하는 TCP(Transmission Control Protocol) 프로토콜(Tahoe, Reno, Vegas, SACK) 메커니즘을 이용하여 다양한 환경에 적용, 결과 값을 비교 분석하며 가장 효율적인 TCP 메커니즘을 제시하였다.

Key words : MANET, GMAHN, TCP (Reno, Tahoe, Vegas, SACK)

Ⅰ. 서론

무선 통신 기술이 발달하고 사용자가 급격히 증가함에 따라 이동 노드간의 데이터 전송과 네트워크에 대한 중요성이 커지고 있는 가운데, 데이터 손실을 최소화 하기위한 기술의 필요성 또한 커지게 되었다[1]. 최근 들어 MANET에서 뿐만 아니라 기존의 유선망과 연결하여 광범위한 네트워크를 구축하기 위해 MANET망과 유선망 환경을 결합한 GMAHN(Global Mobile Ad Hoc Network)에 대한 관심이 증가하고 있다.

기존의 연구에서는 실시간 처리를 하는 특정 인터넷 트래픽에 관한 연구나 무선망에서 사용되는 라우팅 프로토콜 각각에 대한 성능을 분석하는 연구가 진행되어 왔다. 그러나 GMAHN 망에서 TCP 메커니즘에 대한 비교 분석과 TCP 메커니즘 중에서 어떤 메커니즘이 더 효율적인 전송을 하는지에 대한 연구는 미미한 실정이다.

따라서 본 논문에서는 Ad Hoc 라우팅 프로토콜 중 하나인 AODV(Ad-hoc On demand Distance Vector)

* 明知大學校 電子工學科
(Dept. of Electronics Engineering, MyongJi University)
接受日:2008年 1月 28日, 修正完了日: 2008年 3月 5日
※ 명지대학교 산업기술연구소 지원하에 수행된 연구임.

를 사용하여, 이동 노드의 속도와 무선망의 크기에 따른 TCP(Tahoe, Reno, Vegas, SACK) 성능을 비교 분석하였다.

본 논문에서는 기본 TCP 메커니즘을 사용한 Tahoe, Reno, Vegas, SACK을 이용하여 Ali Hamidian에 의해 제안된 GMAHN 환경에서의 이동 노드와 유선 호스트와의 패킷 처리율을 비교 분석하였다[2].

본 논문의 구성은 Ⅰ.서론에 이어, Ⅱ.본론에서는 실험에 사용된 AODV 라우팅 프로토콜과 TCP 메커니즘별 특징에 대해 설명하고, Ⅲ.시뮬레이션 환경에서는 시뮬레이션 파라미터 값과 실험 환경에 대해 설명하고, Ⅳ.시뮬레이션 결과분석에서는 시뮬레이션 파라미터 값에 따른 결과를 비교 분석한다. 마지막으로 Ⅴ.결론으로 구성되어 있다.

Ⅱ. 본 론

본 장에서는 GMAHN과 ADOV 라우팅 프로토콜의 알고리즘을 설명하고, TCP 메커니즘별 특징에 대해 설명한다.

2.1 GMAHN

MANET(Mobile Ad-hoc Network)은 이동 노드들이 자율적으로 구성하는 네트워크로서 노드 상호간에 통신을 가능하게 해주며, 모바일의 특성상 위치가 변하기 때문에 다양한 토폴로지를 형성하게 된다. 본 논문에서는 GW(Gateway)를 이용하여 MANET과 유선망을 연결한 GMAHN 알고리즘을 사용하고 라우팅 프로토콜로는 AODV를 사용할 것이다. 그림 1은 Ad-hoc 라우팅 프로토콜을 나타낸 것이다[3].

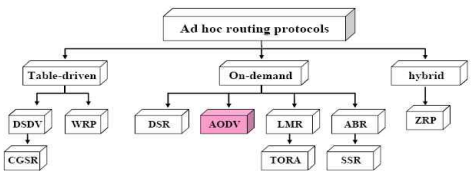

Figure 1. Classification of Ad Hoc routing protocol
그림 1. Ad Hoc 라우팅 프로토콜의 분류

그림 2는 본 논문에서 구성한 GMAHN 망의 구조를 나타낸 것이다. 이동 노드들 중 소스노드는 인접한 중간노드를 거쳐 유선 기간망에 연결된 GW(Gateway)와 라우터를 통해 목적지 노드인 호스트로 경로가 설정된다. 이와 같이 설정된 소스에서 목적지 노드까지의 경로는 2개이며, Window size는 32MSS(Maximum Segment Size)이고, packet size는 512byte이다.

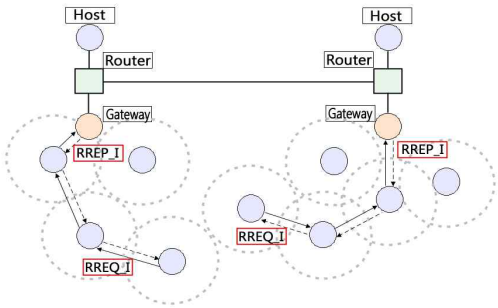

Figure 2. Architecture of GMAHN
그림 2. GMAHN의 구조

2.2 AODV

AODV 라우팅 프로토콜은 소스 노드에서 목적지 노드까지의 경로 발견 과정과 경로 유지과정으로 이루어진다. 경로 발견 및 경로 유지는 다음과 같이 이루어진다.

2.2.1 경로 발견

경로 설정을 원하는 소스노드에서 RREQ(Route Request) 패킷을 이웃 노드에게 Broadcast하면 이웃 노드에서는 RREQ에 담긴 목적지 주소와 Sequence Number를 확인한다. 목적지 주소를 가지고 있는 경우에는 소스노드에게 RREP(Route Reply) 패킷을 Unicast방식으로 보냄으로써 응답하고, 주소가 없을 경우에는 이웃노드에게 broadcast한다. 이때 이웃노드에서는 같은 방식으로 목적지주소를 찾으며, 동일한 RREQ패킷을 받았을 경우에는 Sequence Number를 확인하여 이전 패킷에 대한 정보는 폐기한다[4][5].

2.2.2 경로 유지

이동 노드들은 일정시간마다 Hello 메시지를 주고받는데 이 메시지를 수신한 노드는 Hello 메시지를 전

송한 노드에 대한 정보를 라우팅 테이블의 이웃 노드 리스트에 기록한다. 만약 일정 시간안에 이웃 노드 리스트에 기록된 노드들로부터 Hello 메시지를 받지 못한 경우에는 RERR(Route Error) 메시지를 소스 노드에게 보냄으로써 경로단절을 알리게 된다. 경로가 단절된 경우에는 새로운 경로를 찾는 과정을 거쳐서 경로 유지를 하게 된다.

2.3 GW와 Mobile Node 연결방식

GMAHN는 이동 노드와 유선 기간망의 호스트를 연결하기 위해 GW의 설정이 무엇보다도 중요하다. 자신의 반경안에 있는 이동 노드들에게 주기적으로 자신의 존재를 알리기 위해서는 GA(Gateway Advertisement)메시지를 주기적으로 전송한다. 그림 2와 같이 GW가 관리하고 있는 네트워크 반경내에 자신의 정보를 전송한 후 유선 기반망의 호스트와 통신을 원하는 이동 노드로부터 RREQ_I(경로 설정 요구패킷)를 수신하게 된다면 GW 정보를 가지고 있는 이동 노드는 현재 이웃노드에게 바로 전송함으로써 신속하게 유선 기반망의 호스트와 연결을 원활하게 한다.

RREP_I 패킷은 GW에서 RREQ_I에 대한 응답 패킷으로 이동 노드와 유선 기반망 호스트의 연결을 확인 할 수 있다. RREQ_I 패킷은 AODV 에서 사용되는 RREQ 패킷과 구별되는 패킷으로 MANET 이동 노드가 유선 기간망 호스트를 찾기 위해 GW를 향해 전송되는 패킷을 말한다.

2.4 TCP 알고리즘

인터넷 망에서 신뢰성 있는 전송이 가능한 TCP의 알고리즘은 크게 Slow Start, Congestion Avoidance, Fast Retransmit, Fast Recovery로 나눌 수 있다. TCP 알고리즘을 사용한 대표적인 메커니즘으로는 Tahoe, Reno, Vegas, SACK가 있다[6].

2.4.1 Tahoe

Tahoe는 처음에 Slow Start로 시작하여 패킷을 전달하고 congestion이 발생하면 임계값을 1/2로 줄이고 다시 Slow Start를 실행한다. Congestion Window(cwnd)값이 임계값과 같아지면 Congestion Avoidance상태로 들어가고, 전송도중 패킷이 손실되면 Fast Recovery상태가 되어 손실된 패킷을 재전송한다. 재전송 이후엔 다시 Slow Start상태로 모든 패킷을 재전송하기 때문에 목적지 노드에서 중복된 패킷을 받게 되는 문제점이 있으나 안정성이 높다.

2.4.2 Reno

Reno는 Tahoe의 기능에 Fast Recovery를 추가시킨 메커니즘으로 Tahoe에서 재전송 후 Slow Start상태가 되는 것을 막고, 임계값의 1/2에서 패킷을 전송한다. Reno는 적은 개수의 패킷이 손실되었을 경우에는 유용하나 다수의 패킷이 손실된 경우에는 Congestion Window(cwnd)크기가 계속 절반씩 줄어들게 되어 비효율적이다. 또한 동일한 Congestion Window(cwnd)안에서 다중 패킷이 손실된 경우에는 손실된 패킷의 재전송이 완벽히 이루어지지 않는다.

2.4.3 Vegas

Vegas는 혼잡회피를 위한 예측이라는 새로운 재전송 기법을 사용하는 메커니즘이다. 패킷 손실에 의해 혼잡상태를 발견하는 Reno와는 달리 예측한 전송률과 실제의 전송률을 비교하고 Congestion Window(cwnd)를 조절하여 네트워크 혼잡을 방지한다. Vegas는 동일한 Congestion Window(cwnd)에서 다중 패킷의 손실문제를 부분적으로 해결하였으며, Reno보다 37~71% 높은 효율성을 가진다.

2.4.4 SACK

Reno의 문제점을 보완하기 위한 메커니즘으로 세개의 ACK를 수신하여 패킷의 손실 여부를 아는 것과는 달리, ACK안에 SACK option을 두고 수신된 패킷들의 정보를 담아서 소스 노드에 보낸다. 소스 노드는 SACK option을 확인하여 목적지 노드에서 받은 패킷들과 손실된 패킷들을 알 수 있게 된다. 이를 이용하면 한 Congestion Window(cwnd)안에서 다중 패킷이 손실된 경우에도 이를 확인하고 재전송할 수 있어서 매우 효과적이지만, 송•수신 노드가 모두 SACK를 구현할 수 있어야 한다.

Ⅲ. 시뮬레이션 환경

본 장에서는 시뮬레이션 환경에서 사용된 파라미터

와 망 모델에 대하여 설명한다. 시뮬레이션 툴은 버클리 대학에서 개발한 분산 객체 네트워크 시뮬레이터인 NS-2(Network Simulator)를 사용하였고, GMAHN환경에서 on-demand방식인 AODV 프로토콜을 사용하였다.[7]

본 논문에서 라우팅 프로토콜을 분석하기 위한 TCP 메커니즘으로 Tahoe, Reno, Vegas, SACK을 이용하여 패킷 처리율을 측정하였다. 표 1은 시뮬레이션 실험 환경으로 사용한 하드웨어와 소프트웨어의 정보를 보여주는 것이고, 표 2는 NS2(Network Simulator)에 설정된 파라미터 값을 나타낸 것이다.

Table 1. Simulation Environment
표 1. 시뮬레이션 실험환경

O/S		WOW Linux7.3
Hardware		Pentium4 (2.4GHz CPU)
Software	Simulator	NS-2.1b9a
	Language	C++, Tcl
	Scenario file	10

Table 2. Simulation Parameter
표 2. 시뮬레이션 파라미터

Parameter	Value
Tansmission range	250m
Simulation time	100sec
Topology size	800m×500m
Number of mobile nodes	20
Number of sources	2
Number of gateways	2
TCP version	Tahoe, Reno, Vegas, SACK
TCP windows size	32MSS
Packet size	512 bytes
Pause time	5.0 sec
Maximum speed	10, 20, 30 m/s

그림 3은 NS simulator에서 시뮬레이션 과정을 그래픽으로 보여주는 툴인 NAM Editor를 사용하여 시뮬레이션 한 후 발생한 이벤트의 일부분을 캡처한 모습이다. X, Y 2차원 평면으로 구성된 전체 네트워크 토폴로지 반경은 800[m]X500[m]이고, 노드의 이동속도는 20 [m/s], 통신에 참여하는 이동노드는 20개, 유선망에서의 GW와 라우터, 호스트의 수는 각 2개씩 설정되어 있다.

통신에 실질적으로 참여하는 Flow는 2개이며, MANET망에 위치한 소스노드에서 패킷을 전송하고 유선망에 있는 호스트가 목적지 노드가 되는 시나리오로 구성되어져 있다.

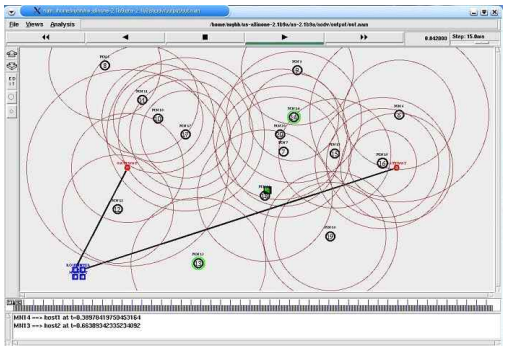

Fig 3. NAM Editor
그림 3. NAM 에디터

Ⅳ. 시뮬레이션 결과분석

본 장에서는 GMAHN에서의 TCP 메커니즘인 Tahoe, Reno, Vegas, SACK을 사용하여 이동 노드중 하나인 소스 노드(Source Node)에서 유선망의 호스트 중 하나인 목적지 노드(Destination Node)로 데이터를 전송하는데 이동 노드의 속도를 변화를 시켰을 때 패킷 처리율(Throughput[bps])과 Ack Sequence Number를 비교, 분석 평가 하였다. 패킷 처리율은 각 시간대에 GMAHN 내에서 데이터 처리율을 나타내는 것이다. Ack Sequence Number는 한 패킷의 전송이 완료시에 발생되는 응답 메시지이다.

4.1 노드이동속도 변화에 따른 패킷 처리율 변화

그림 4, 그림 5, 그림 6, 그림 7은 GMAHN 망에서 이동노드의 수가 20개, Pause time은 5초, Packet Size는 512byte일 때, 이동 노드의 이동 속도를 10, 20, 30m/s로 증가시킬 경우 Tahoe, Reno, Vegas,

Sack 메커니즘에서의 패킷 처리율을 나타낸 것이다.

1) Tahoe의 Throughput(bps)

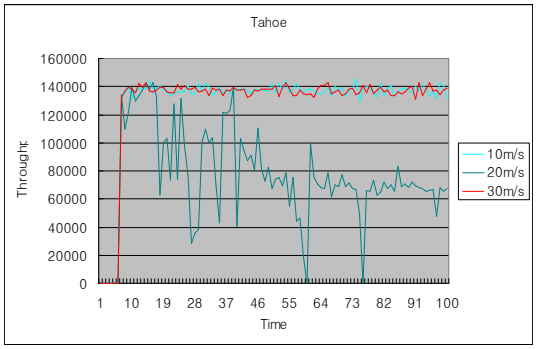

Fig 4. Throughput(bps) of Tahoe
그림 4. Tahoe의 처리율(bps)

그림 4를 살펴보면, Tahoe는 10m/s와 30m/s일 때 7초부터 점점 증가하다가 약15초 부분에서부터 안정적인 처리율을 보이는 반면에, 20m/s에서는 약 15초 후부터 혼잡이 발생되어 데이터 처리율의 변화가 심해지는 것으로 미루어 보아 네트워크 내에서의 노드들 간의 데이터 전송에 많은 문제가 발생함을 알 수 있다. 이동 노드의 속도가 20m/s일 때 Tahoe는 매우 불안정한 처리율을 보여주고 있으며, 본 논문의 환경에서 시뮬레이션을 실행한 결과, 적합하지 않음을 알 수 있다. 반면에 10m/s와 30m/s는 Tahoe의 메커니즘으로 데이터 전송을 안정화 하려는 것을 살펴 볼 수 있다.

2) Reno의 Throughput(bps)

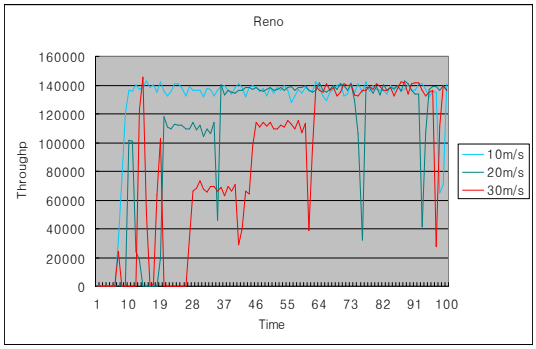

Fig 5. Throughput(bps) of Reno
그림 5. Reno의 처리율(bps)

그림 5는 Reno에서 처리되는 패킷의 처리율을 나타낸다. 전체적인 처리율의 흐름을 살펴보면 현재 환경에서는 10m/s 일 때 가장 안정적으로 140kbps 처리율을 유지하는 반면, 노드의 이동속도가 증가 할수록 네트워크의 혼잡이 발생하여 데이터의 처리율이 떨어짐을 확인 할 수 있다. 가장 빠른 30m/s에서의 데이터 전송에서 안정화가 가장 늦음을 확인 할 수 있다. 20m/s에서는 19초까지는 불안정한 상태를 보여주다가 안정화 상태로 가지만 36초에서는 처리율의 변화가 발생한다. 이는 액티브 경로상태에서 중간노드가 이동함으로 인한 경로가 끊어진 현상으로 경로 재설정 과정을 의미한다. 이때 처리율이 0으로 떨어지지 않는 이유는 중간노드가 라우팅 캐시에 패킷을 저장하였기 때문에 그 만큼 빠른 경로 복구로 인한 데이터의 손실을 막았기 때문이다. 30m/s에서도 이와 같은 현상이 발생함을 그림 5를 통해서 알 수 있다.

3) Vegas의 Throughput(bps)

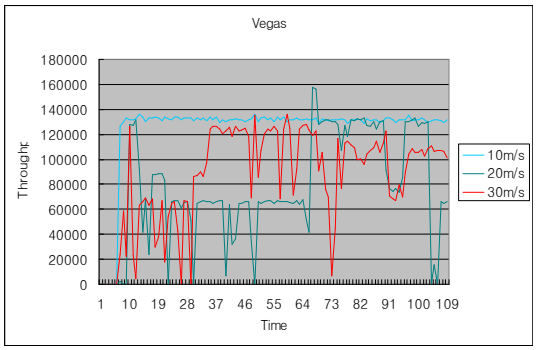

Fig 6. Throughput(bps) of Vegas
그림 6. Vegas의 처리율(bps)

그림 6은 Vegas에서 처리되는 패킷의 처리율을 나타낸다. 10m/s일 때가 가장 안정적인 상태인 140kbps의 처리율을 유지하는 반면, 20m/s와 30m/s일 때는 Reno 보다 심한 데이터의 혼잡이 발생함을 알 수 있다.

본 논문의 시뮬레이션 환경에서 Vegas가 Reno보다 20m/s, 30m/s에서 안정화 상태의 처리율로 이동하는데 보다 많은 시간이 소요되고, 처리율의 변화가 커짐으로 인한 패킷 손실도 많아짐을 알 수 있다. 이동

노드의 속도에 따른 경로 유지 시간도 짧아지고 있음을 그림 6을 통해서 확인할 수 있으며, 혼잡 발생시 Tahoe의 메커니즘으로 회복하려는 것을 확인 할 수 있다.

4) SACK의 Throughput(bps)

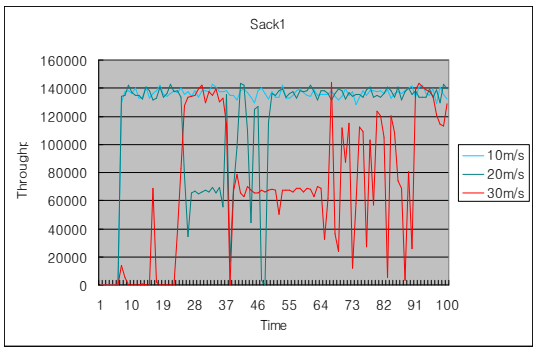

Fig 7. Throughput(bps) of SACK
그림 7. SACK의 처리율(bps)

그림 7은 Sack에서 처리되는 패킷의 처리율을 나타낸다. 10m/s에서 가장 안정적인 상태인 140kbps를 유지하는 반면, 20m/s에서는 22초에서 48초 사이에 노드 및 네트워크 환경으로 인한 처리율의 심한 변화를 보이지만 48초 이후에는 다시 안정되는 모습을 보인다. 30m/s일 때는 노드의 빠른 이동속도로 인해 데이터 전송이 원활하지 못해 불안정한 모습을 보임을 알 수 있다.

노드의 이동 속도에 따른 Tahoe, Reno, Vegas, Sack 메커니즘에서 처리되는 패킷의 처리율을 살펴본 결과 가장 안정된 흐름을 보이는 것은 10m/s임을 알 수 있고, 또한 혼잡이 발생하였을 경우 각 메커니즘별로 회복 하는 것을 확인해 볼 수 있었다.

4.2 Throughput(bps)의 변화

1) Node Speed(10m/s)에 따른 Throughput의 변화

그림 8은 10m/s 일 때 Tahoe, Reno, Vegas, Sack의 처리율의 변화로 데이터 전송 시작 후 약 2~3초 후 처리율이 140kbps로 안정적으로 데이터가 전송됨을 보여주고 있다. 90초 이후에 Reno에서 약간의 혼잡이 보여지는데 Reno의 빠른 회복 알고리즘으로 안정화를 되찾는 것을 그래프로 알 수 있다. 전체적으로 안정한 상태를 확인 할 수 있다.

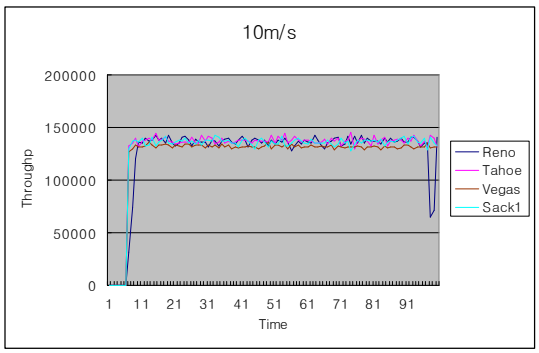

Fig. 8. Throughput(bps) of 10m/s
그림 8. 10m/s의 Throughput(bps)

2) Node Speed(20m/s)에 따른 Throughput의 변화

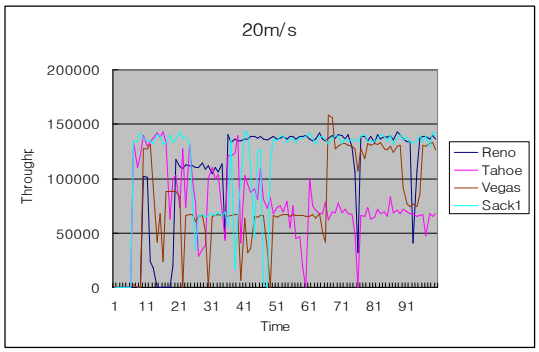

Fig. 9. Throughput(bps) of 20m/s
그림 9. 20m/s의 Throughput(bps)

그림 9는 20m/s에서 Tahoe, Reno, Vegas, Sack의 처리율 변화이다. Reno는 불안정한 처리율을 보이지만 30초 후반부터는 안정된 모습을 보이는 반면, Tahoe는 전반적으로 불안정한 상태를 나타낸다. Vegas는 안정한 상태로 시작되다가 25초 부분부터 경로 복구 및 데이터 손실이 나타나지만 45초부터 점차 안정적인 상태를 보인다. 전체 적으로 Vegas와 Tahoe는 노드의 이동 속도가 증가하면서 처리율의 변화가 심하고 데이터 손실도 많아짐을 알 수 있다.

3) Node Speed(30m/s)에 따른 Throughput의 변화

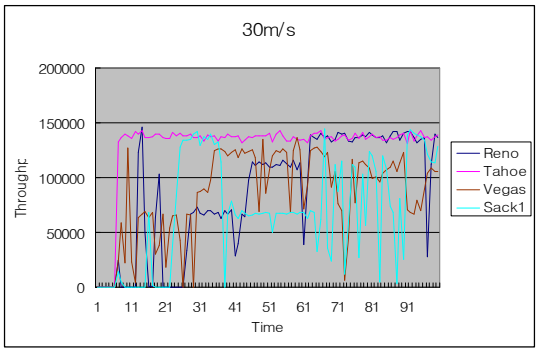

Fig. 10. Throughput(bps) of 30m/s
그림 10. 30m/s의 Throughput(bps)

그림 10은 가장 빠른 속도인 30m/s에서 각 메커니즘의 데이터처리율을 보여주고 있다. 10초일 때 Tahoe가 안정된 처리율을 보여주고 있으며, Reno, Vegas, SACK는 불안정한 처리율에서 시작하여 일시적인 안정상태와 불안정상태를 반복적으로 보여주고 있다. 전체적으로 가장 불안한 상태를 확인 할 수 있다.

4.3 Ack Sequence Number의 변화

1) Tahoe에서의 Ack 변화

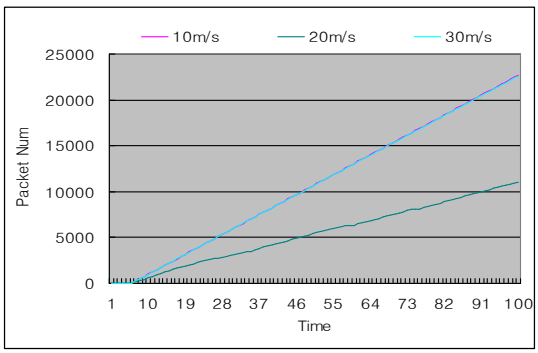

Fig. 11. Ack of Tahoe
그림 11. Tahoe의 Ack

2) Reno에서의 Ack 변화

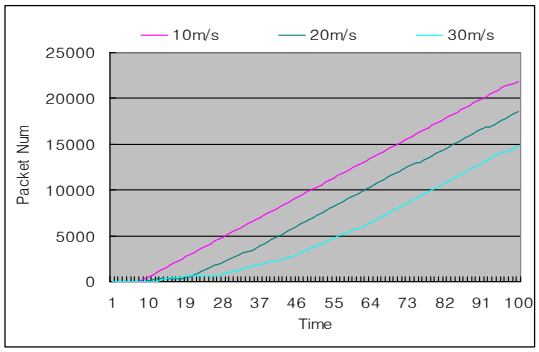

Fig. 12. Ack of Reno
그림 12. Reno의 Ack

3) Vegas에서의 Ack 변화

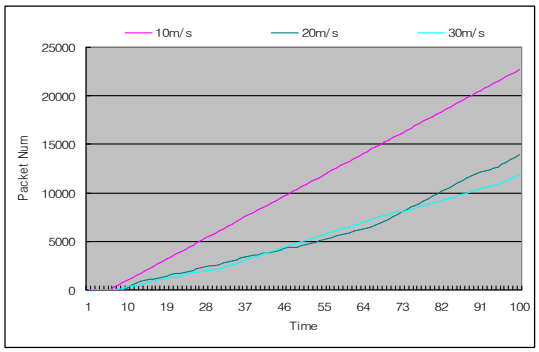

Fig. 13. Ack of Vegas
그림 13. Vegas의 Ack

4) SACK에서의 Ack 변화

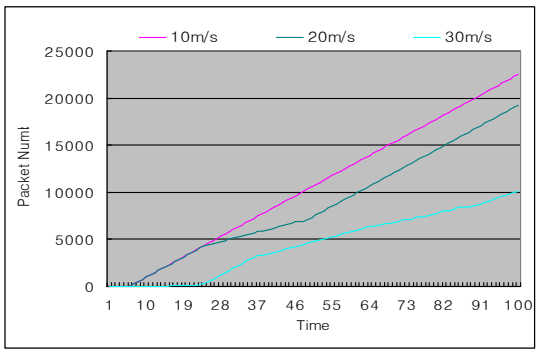

Fig. 14. Ack of SACK
그림 14. SACK의 Ack

그림 11, 그림 12, 그림 13, 그림 14는 각 TCP 메커니즘별로 노드의 이동속도 변화에 따른 Ack Sequence Number를 나타낸다. 소스 노드에서 패킷을 전송한 후 목적지 노드에서 패킷을 수신하여 다시 소스 노드로 Ack(응답신호)를 보내게 되는데 이때 Ack가 생성될 때 각 Ack에 붙여지는 Sequence Number의 증가를 볼 수 있다. 앞에서 살펴본 이동 노드에 따른 데이터 처리율(Throughput)의 변화에 따라 Ack Sequence Number의 수신으로 지연 및 중복 수신을 확인 할 수 있다. 이것은 네트워크내의 혼잡으로 인해 패킷의 손실이 발생하여 목적지에서 Ack packet의 지연 및 손실로 인한 결과이다.

결과를 분석해 보면 10m/s일 때 가장 빠른 Ack Sequence Number의 증가를 보이고 있다. 10m/s일 때 Reno, Vegas, SACK에서 가장 빠른 Ack Sequence Number의 증가를 보이고 있고, 노드의 이동속도가 증가할수록 Ack의 Sequence Number가 더디게 증가함을 알 수 있다.

또한 각 메커니즘에서 혼잡이 발생했을 때 회복 단계의 알고리즘과 이에 따른 Ack Sequence Number의 변화를 확인 할 수 있다.

V. 결 론

본 논문에서는 NS-2를 사용하여 MANET(Mobile Ad-hoc Network)망과 기존 유선망을 연결한 GMAHN (Global Mobile Ad Hoc Network) 환경에서 데이터 전송시 발생하는 처리율에 대해 TCP 메커니즘별 노드의 이동 속도 변화에 따라 비교 분석하였다.

AODV를 이용하여 처리율을 모의실험 한 결과 노드의 이동 속도가 10m/s 일 때 모든 처리율이 안정적인 결과를 보였다. 20m/s일 때는 TCP 메커니즘 중 SACK가 좋았고, 30m/s일 때는 Tahoe가 가장 좋은 결과를 보였다.

노드의 이동속도에 따른 각 TCP 메커니즘의 Ack Sequence Number의 변화도 10m/s일 때 안정적인 환경이기 때문에 빠른 변화율을 보이다가 속도가 증가할수록 더뎌짐을 알 수 있었다.

결론적으로 AODV 라우팅 프로토콜을 사용하는 GMAHN망에서 TCP를 사용하는 경우, 10m/s를 이용하는 것이 안정적인 데이터 전송이 된다는 것을 알 수 있다.

향후, GMAHN환경에서의 네트워크 토폴로지의 크기와 노드정지시간, 이기종망간의 연결을 지원하는 GW와 이동노드, 유선호스트의 수에 따른 TCP 메커니즘의 영향을 분석하여 다양한 환경속에서 보다 신뢰성 있는 데이터 전송을 위한 TCP 메커니즘 연구가 진행되어야 한다.

참고문헌

[1] E. M. Royer and C.-K. Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", IEEE Personal Communications, April 1998, pp. 46-55.

[2] Ali Hamidian, A Study of Internet Connectivity for Mobile Ad Hoc Networks in NS 2, Master's thesis, Lund Institute of Technology, Sweden, January 2003.

[3] C.-K. Toh, "Ad Hoc Mobile Wireless Networks Protocols and Systems", Prentice Hall PTR, 2002, pp.13-25.

[4] C. E. Perkins, "Ad Hoc On-Demand Distance Vector (AODV) Routing", Internet Draft, IETF MANET Working Group, draft-ietf-manet-aodv-12.txt, November 2002.

[5] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing", proceedings of the 2nd IEEE workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90-100.

[6] K. Fall and S. Floyd, "Comparisons of Tahoe, Reno, and SACK TCP", December 1995.

[7] The Network Simulator-NS2, http://www.isi.edu/nsnam/ns [9] K. Fall and K. Varadhan, Eds., "ns notes and documentation," 1999; available from http://www.isi.edu/nsnam/ns.

저 자 소 개

오 세 덕 (정회원)

2001년 : 호서대학교 전자공학과 졸업 (공학사)
2003년 : 명지대학교 대학원 전자공학과 (공학석사)
2006년 : 명지대학교 대학원 전자공학과 (공학박사)
<주관심분야> Ad-Hoc Network, Embedded System

김 재 호 (학생회원)

2006년 : 명지대학교 전자공학과 졸업 (공학사)
2008년 : 명지대학교 대학원 전자공학과 (공학석사)
<주관심분야>GMAHN, Automatic Robot, Computer Network

배 진 승 (학생회원)

2002년 : 명지대학교 전자공학과 졸업 (공학사)
2004년 : 명지대학교 대학원 전자공학과 (공학석사)
2004년 3월 ~ 현재: 명지대학교 대학원전자공학과 (박사과정수료)
<주관심분야> Embedded system, Automatic Robot

정 찬 혁 (정회원)

2000년 : 명지대학교 전자공학과 졸업 (공학사)
2002년 : 명지대학교 대학원 전자공학과 (공학석사)
2006년 : 명지대학교 대학원 전자공학과 (공학박사)
2005년 ~ 현재 : 정보통신 연구진흥원 연구원
<주관심분야> Ad Hoc Network, 센서네트워크, 이동통신

이 치 문(정회원)

1997년 : 명지대학교 전자공학과 졸업 (공학사)
1999년 : 명지대학교 대학원 전자공학과 (공학석사)
2008년 현재 : 명지대학교 대학원 전자공학과 (박사과정수료)
<주관심분야> VoIP, NGN, Mobile IP

하 재 승 (정회원)

1991년 : 명지대학교 전자공학과 졸업 (공학사)
1993년 : 명지대학교 대학원 전자공학과 (공학석사)
2001년 : 명지대학교 대학원 전자공학과 (공학박사)
2005년 ~ 현재 : 명지전문대학 정보통신과 부교수
<주관심분야> SDR(Software Defined Radio), Mobile IP, Ad-Hoc

유 충 렬 (정회원)

1988년 : 명지대학교 전자공학과 졸업 (공학사)
1991년 : 명지대학교 대학원 전자공학과 (공학석사)
1996년 : 명지대학교 대학원 전자공학과 (공학박사)
2003년 ~ 현재 : 명지대 방목기초교육대학 조교수
<주관심분야> Microprocessor, Computer Architecture

이 광 배 (정회원)

1979년 : 고려대학교 전자공학과
졸업 (공학사)
1981년 : 고려대학교 대학원
전자공학과 (공학석사)
1984년 ~ 1986년 : Univ. of Southern California, 컴퓨터공학 공학석사
1986년 ~ 1991년 : Arizona state Univ.컴퓨터공학
공학박사
1992년 ~ 현재 : 명지대학교 전자공학과 정교수
<주관심분야> Embedded system, Ad Hoc Network, Computer Network

김 현 욱 (정회원)

1978년 : 고려대 전자공학과
졸업 (공학사)
1980년 : 고려대 대학원
전자공학과 (공학석사)
1987년 : 고려대 대학원
전자공학과 (공학박사)
1980년 ~ 1981년 : 동양전문대 전자공학과 전임강사
1981년 ~ 1988년 : 명지대 전자공학과 전임강사
1990년 ~ 현재 : 명지대 전자공학과 정교수
<주관심분야> Embedded system, Ad Hoc Network, MPEG-4