무선 Ad-hoc 네트워크에서 TCP 성능 향상을 위한 동적 혼잡윈도우 조정 알고리즘

김관웅* · 배성환**

An Dynamic Congestion Window Tuning Algorithm for TCP Performance Improvement in Wireless Ad-hoc Network

Kwan-woong Kim* · Sung-hwan Bae**

요 약

TCP는 유선망을 위하여 설계되었기 때문에, 유선망과 다른 특성을 가진 무선망에서 서비스되는 경우 성능이 크게 저하된다. 무선 Ad-hoc 네트워크에서 혼잡윈도우(congestion window) 값은 TCP의 성능에 크게 영향을 준다. 네트워크 상황에 따라 적절한 크기의 혼잡 윈도우 값을 설정함으로서 TCP 성능을 향상시킬 수 있다. 본 논문에서는 무선 Ad-hoc 네트워크에서 TCP 성능을 향상시키는 새로운 TCP 알고리즘을 제안한다. 제안된 알고리즘은 TCP 수신측에서 최적의 윈도우 크기를 측정하고, ACK 패킷의 윈도우 필드를 사용하여 CWL(Congestion Window Limit)를 최적 값에 세팅하는 최대 혼잡윈도우 조정 기법이다. 무선 Ad-hoc 네트워크의 다양한 환경에서 컴퓨터 시뮬레이션을 수행한 결과 제안된 알고리즘이 기존 TCP new reno 프로토콜보다 전송률 및 패킷 손실에서 성능을 크게 향상하였다.

ABSTRACT

The TCP protocol is originally designed for wired network, however it performs very poor in wireless network due to different nature of wireless network from wired networks. In terms of TCP performance improvement in wireless Ad-hoc network, many researches show that small congestion window size of TCP connection can improve TCP performance. We propose a new TCP algorithm to improve TCP performance in wireless Ad-hoc network. The basic idea of our approach is that TCP receiver estimates the optimum window size and then sets congestion window limit of TCP sender to an optimum value by using the advertised window field in TCP ACK packet. From extensive computer simulation, the proposed algorithm shows superior performance than traditional TCP protocols in terms of packet delivery ratio and packet loss.

키워드

TCP, Congestion control, MANET(Mobile Ad Hoc-NETwork)

Ⅰ. 서론

무선 Ad-hoc 네트워크는 무선 링크에 의해 연결된 이동 노드들로 구성된 네트워크로, 중앙제어장비 없이 자동으로 네트워크를 구성한다[1, 2]. 각 이동 노드는 전송 범위 내의 인접 노드들과 대역폭이 제한된 무선 링크로

* 원광대학교 전기전자정보공학부

** 한려대학교 멀티미디어정보통신공학과

접수일자 2008. 03. 20

통신한다. 모바일 노드가 Ad-hoc 네트워크내의 다른 노드와 통신할 경우, 노드의 전송범위 제한 때문에 대부분의 연결은 중간 노드들을 경유하는 다중 홉 연결을 사용한다.

TCP는 인터넷에서 가장 많이 이용되는 전달 계층 프로토콜로써 대부분의 어플리케이션이 TCP를 이용하여 통신을 수행한다. 따라서 어플리케이션 호환성을 유지하기 위해서는 무선망에서도 TCP가 주요 전송계층 프로토콜로 이용된다.

그러나 TCP 프로토콜은 TCP는 유선망에 근거하여 설계되었기 때문에 유선망과 다른 특성을 가지는 무선 Ad-hoc 네트워크에 적용했을 경우, 성능이 크게 저하된다[3-5]. TCP 성능저하의 요인은 노드의 이동성, 무선채널의 감쇄, 간섭 현상 뿐만 아니라, TCP 자체의 특성도 성능저하의 주요 원인이 된다. Ad-hoc 네트워크에서 TCP 성능을 향상시키기 위해 수행된 관련 연구에 의하면, IEEE 802.11 계열 MAC 프로토콜에서 TCP 성능 저하의 주된 원인은 TCP의 크기가 큰 혼잡윈도우에 의해 유발된 과도한 매체 접근에 기인한 충돌 손실이다.

그림 1과 같이 여러 개의 패킷이 다중 홉(multi-hop)을 거쳐서 가는 경우에 전송중인 패킷 간에 충돌이 일어날 수 있으며, 또한 TCP 데이터 패킷과 경쟁하는 역 방향 ACK 패킷이 Hidden terminal 현상으로 충돌이 일어날 수 있다. 종종 TCP 데이터 패킷은 무선채널 매체를 점유하여 ACK 패킷이 목적지에 도착하는 것을 방해할 수 있다[6, 7].

이와 관련된 연구에서 작은 혼잡윈도우 알고리즘이 TCP 성능을 향상시키고, 최적 Congestion Window Limit (CWL)은 연결의 홉-수(hop count)에 관계된다고 알려졌다[3, 8]. Fu[9]은 TCP를 위한 최대의 수율(throughput)은 윈도우 크기의 최대 값이 종단 간 홉수의 1/4일 때 얻어짐을 시뮬레이션을 통해 보였다. 다중-홉 Ad-hoc망에서 TCP 성능을 향상하기 위한 하나의 방법은 CWL의 최적 값을 구하여 과도한 매체접근에 의해 유발 된 충돌을 줄이는 기법이다.

본 논문에서는 CWL 값을 측정된 최적 값으로 설정하여 Ad-hoc 망에서 TCP 성능을 향상시키는 기법을 제안한다. 제안된 알고리즘은 TCP 수신측에서 Ad-hoc 망으로부터 ECN(Explicit Congestion Notification)[10] 정보를 수집함으로써 window limit의 최적 값을 측정한다. CWL을 측정된 최적 값에 설정하는 작업은 ACK 패킷에 배달된 advertised window를 사용하여 수행된다. 제안된 알고리즘은 적절하게 망 상태를 고려하는 최대 혼잡윈도우 조정 방식이며, 종단 사용자간(end-to-end)에 TCP 표준을 침해하지 않는다.

본 논문의 구성은 2장에서 Ad-hoc 네트워크에서 TCP와 관련된 이전 연구와 제안된 알고리즘에 관해서 살펴보고, 3장에서 컴퓨터 시뮬레이션을 통해 제안된 알고리즘의 성능 평가를 수행하여 기존의 다른 알고리즘과 성능을 비교 분석한다. 마지막으로 4장에서 결론을 맺는다.

Ⅱ. 다중-홉에서 최대 혼잡윈도우의 계산

2.1 관련연구

Ad-hoc 네트워크에서 TCP 성능을 향상시키기 위한 많은 관련 연구를 살펴보면 Gerla[5]는 MAC 계층이 ACK 보호를 제공하지 않을 경우에 CWL이 1-패킷보다 크면 TCP 성능이 저하됨을 컴퓨터모의 실험을 통해 증명하였다.

J. Li[11]은 다중-홉 Ad-hoc망에서 IEEE 802.11 MAC 프로토콜의 공간적 재사용 속성(Spacial Reuse Property)에 관해서 심도 있는 해석을 하였다. 공간적 재사용 속성은 다중-홉 무선망에서 동시에 전송될 수 있는 패킷의 수를 제한하는 것으로, 최대 혼잡윈도우의 증가는 무선링크에서 패킷 손실률이 낮을지라도 TCP의 수율을 감소시킨다. 결론적으로 노드 체인의 이상적인 용량은 효율적인 라디오 채널 대역폭의 1/4이고, 전송측 TCP에서 여러 개의 패킷을 발생시키면 무선 채널의 점유 확률이 높아지므로, 목적지에 더 가까운 노드가 마지막 노드로 데이터를 전송하기 위한 무선채널은 접근이 어려워져 실제 용량은 홉-수의 1/7까지 저하된다.

Fu[9]은 최대 혼잡윈도우에 대한 최적 값을 조사했는데, TCP의 최대 산출량은 TCP 윈도우 크기 제한이 TCP 연결의 h/4(h : 홉-수)일 때 성취됨을 보였다. 또한 다중-홉 흐름의 TCP 성능을 향상시키기 위해서 RED(Random Early Detect) 와 adaptive pacing 알고리즘에 기초한 두 가지 방법을 제안했다. LRED(Link RED)는 패킷의 ECN 비트를 마킹하는데 사용되는 링크에서의 패킷폐기 확률을 계산하기 위해 IEEE 802.11 MAC 계층에서 패킷 전송의 재시도 수를 모니터링 한다. 이러한 메커니즘은 링크 손실은 망에서 버퍼 오버플로어에 의한 패킷 손실보다

자주 발생한다는 실험결과에 근거한다.

그림 1은 신호의 간섭에 의한 링크 손실 예를 보여준다. IEEE 802.11 네트워크에서 동시에 데이터를 전송하기 위해서는 데이터를 송·수신하는 노드들 사이가 4-홉 이상이어야 한다. Carlos[7]은 다중-홉 무선망의 TCP 성능 향상을 위해서 충돌이 발생했을 때 재전송에 사용되는 MAC 계층 back-offs 수를 고려하는 COPAS라 불리는 contention-balancing 알고리즘을 제안했다.

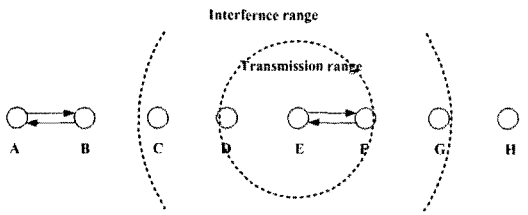

그림 1. IEEE 802.11 네트워크에서 Hidden terminal 현상
Fig. 1. Hidden Terminal Problem in IEEE 802.11 MAC

Chen[6]은 802.11 MAC 프로토콜을 위한 TCP 연결의 Round-Trip Hop-Count의 1/5로 주어지는 adaptive Congestion Window Limit(CWL)을 제안했다. Adaptive CWL은 다중-홉 망에서 경로의 대역폭-지연 곱을 계산하여 얻을 수 있다. 그러나 이 방식의 알고리즘은 홉 정보를 전달하기 위해서 IP나 TCP 헤더에 홉-수를 위한 새로운 필드를 필요로 하는 단점이 있다. 또한 CWL을 세팅하기 위해서 라우팅 모듈, 링크 계층과 TCP 계층 사이에 정보교환이 필요하여, Cross-Layer Interaction 기법이 도입되어야 하는 제한사항이 있다.

2.2 제안된 알고리즘

Ad-hoc 네트워크에서 TCP 송신측이 CWL을 최적 값으로 설정한다면 TCP 성능이 향상된다. 그러나 앞 절에서 언급한 것처럼 링크 계층에서 TCP 연결들의 홉-수를 세는 일은 효율적이지 않고 토폴로지의 변화에 따라 가변적이기 때문에 정확하지 않을 수 있다. 또한, TCP 흐름에 CWL을 세팅하기 위해서는 링크 계층과 TCP 계층 사이에 정보 교환을 위한 인터페이스의 변경이 요구된다.

제안된 알고리즘의 기본 아이디어는 TCP 수신측에서 수신 패킷의 ECN 필드를 모니터링하여 망의 혼잡 상태를 인지할 수 있다. TCP 수신측이 최적 윈도우를 발견하면 advertised window를 이용하여 TCP 송신측에 ACK 패킷을 보냄으로서 CWL을 세팅한다.

본 논문에서는 링크 계층에서 충돌 상황을 감지하기 위해서 LRED[9]을 채택한다. LRED에서 링크 계층은 패킷 재전송 개수의 평균을 유지한다. 패킷이 다음 홉에 보내지기 전에 링크 계층은 평균 재시도 수를 가지고 패킷의 ECN 비트 마킹 확률을 계산하고, 여기서 얻은 확률을 이용하여 패킷을 마킹한다. TCP 수신측은 ECN의 빈도수에 의해서 현재 흐름의 최적 윈도우를 조절한다. TCP가 ACK 패킷을 수신할 때마다 ECN 빈도의 평균(ecn_avg)을 측정한다. ecn_avg는 수식 (1)과 같이 정의된다.

$$ecn\_avg = \alpha \cdot ecn\_avg + (1-\alpha) \cdot ecn\_rate \quad (1)$$

(여기서, ecn_rate는 전체 수신된 데이터 패킷 수에 대한 ECN 마크된 패킷의 수이고, α는 weight 값으로 0.9이다.)

TCP 수신측은 advertised window가 최소 임계치(min_th)와 최대 임계치(max_th)사이에서 ecn_avg을 유지하도록 조절한다. 만약 ecn_avg가 max_th보다 크면 TCP 수신측은 advertised window를 감소하고, ecn_avg가 min_th보다 작으면 advertised window를 증가 시킨다. 이러한 방법으로 수신측 TCP는 최적 CWL을 계산하고, 계산된 CWL 값을 ACK 패킷의 advertised window 필드를 통해 송신측 TCP에 통보할 수 있다. 따라서 최적 산출량이 성취되고 TCP가 overshooting 문제[6]을 피하게 할 수 있다. CWL을 설정하기 위해 TCP 송신측과 수신측은 다음과 같은 프로세스를 수행한다.

(1) 송신측으로부터 수신측에 congestion window를 전송 : TCP 송신측은 현재 congestion window을 데이터 패킷의 window 필드를 통해 전송한다.
(2) 링크 계층에서 패킷 마킹 : 패킷이 다음 홉으로 포워딩하기 전에 링크 계층은 평균 재시도 수를 가지고 마킹확률을 계산하고 구해진 마킹확률을 가지고 패킷의 ECN 비트를 세팅한다.
(3) 평균 ECN 율(average ecn rate)을 측정 : TCP 수신측은 수식 (1)을 이용하여 M 패킷마다 평균 ECN 율(ecn_avg)을 계산한다.
(4) TCP 수신측에서 advertised window 설정 : 만약 ecn_avg > min_th이고 awnd < def_wnd이면 awnd를 증가시킨다. 만약 ecn_avg < max_th이고 awnd > 1이

면 awnd를 감소시킨다.

(5) Advertised window를 이용하여 congestion window limit 세팅 : TCP 송신측은 ACK 패킷을 수신할 때마다 CWL을 advertised window로 갱신한다.

TCP 수신측에서 CWL 계산 알고리즘은 표 1과 같다.

Ⅲ. 성능 평가

제안된 알고리즘의 성능 평가를 위해 NS2 시뮬레이터[12]를 사용하여 기존 TCP 프로토콜과 성능 비교를 다양한 네트워크 환경에서 수행하였다.

표 1. CWL 계산 알고리즘의 의사코드
Table 2. Pseudo-code of CWL Procedures

Parameter definitions
ecn_rate : ecn 마크된 ACK 패킷의 비율, *ecn_avg* : ecn 마크된 ACK 패킷의 평균 비율
min_th : 최소 임계치, *max_th* : 최대 임계치, *flag* : ecn 경험의 유무, *M* : 상수 값
def_wnd : 기본 advertised window 크기

TCP behavior: TCP_RecvDATA(packet p)

```
    /* set last_seen to current congestion window of TCP sender */
      last_seen ← window of p;
      pkt_count = pkt_count + 1;        /* counts number of recv, packets */

      if ecn bit of p is marked then
      ecn_count = ecn_count + 1;  /* increase ecn_count */
            if flag = 0 then
                  awnd = last_seen ;flag ← 1;
            end if
      else
            if flag = 0 then /* find max congestion window */
                  last_seen = max {awnd, last_seen}
            end if
        end if
if pkt_count => M then        /* calculate average ecn rate every M packet */
      /* calculate current ecn_rate */
      ecn_rate = ecn_count / pkt_count;
      /* initialize pkt_count & ecn_count */
      ecn_count = pkt_count = 0;
   /* calculate the average of ecn rate */
      ecn_avg = α · ecn_avg + (1 - α) · ecn_rate;
      if ecn_avg < min_th then
            /* increase advertised window */
            awnd = min {awnd + 1, def_wnd};
      else if ecn_avg > max_th then
            /* decrease advertised window */
            awnd = max {awnd - 1, 1};
      end if
end if
```

3.1 체인 토폴로지(Chain Topology)

멀티-홉 환경에서 홉-수가 TCP에 미치는 영향을 알아보기 위해서 그림 2와 같은 체인 네트워크에서 시뮬레이션을 수행하였다. 인접 노드들 간의 거리는 200 m이고 각 노드의 전송 거리는 250 m, 간섭 범위는 550 m로 설정하였다. 하나의 FTP 연결이 패킷을 생성하여 마지막 노드로 전송한다. 패킷의 크기는 1 Kbyte이며, 라우팅 프로토콜은 AODV[13]를 사용하고 TCP는 가장 광범위하게 사용되는 TCP Reno를 사용하였다. 시뮬레이션 시간은 500 초로 설정하고 실험을 수행하였다.

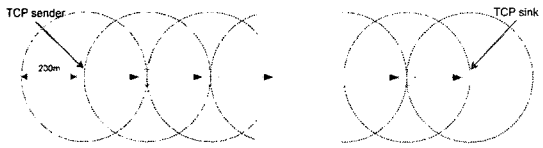

그림 2. 체인 토폴로지
Fig. 2. Chain Topology

그림 3과 그림 4는 홉-수가 14일 때 TCP Reno와 제안된 알고리즘의 TCP 혼잡윈도우 변화를 보여준다. TCP reno의 경우 그림 3과 같이 혼잡윈도우의 크기가 톱니모양으로 오르내리며 수렴하지 않는다. 제안된 알고리즘은 초기 패킷을 전송할 때, 혼잡윈도우의 값이 8 패킷까지 증가하나 이후, 수신측 TCP에 의해 윈도우 최적 값인 3-2사이로 수렴한다.

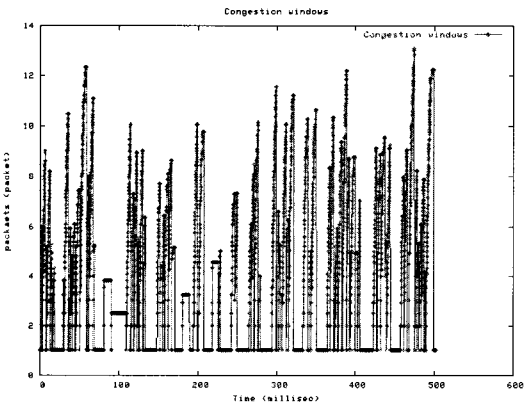

그림 3. 홉-수가 14일 때 TCP Reno의 혼잡윈도우 추이
Fig. 3. Congestion Wondow of TCP Reno(Hops = 14)

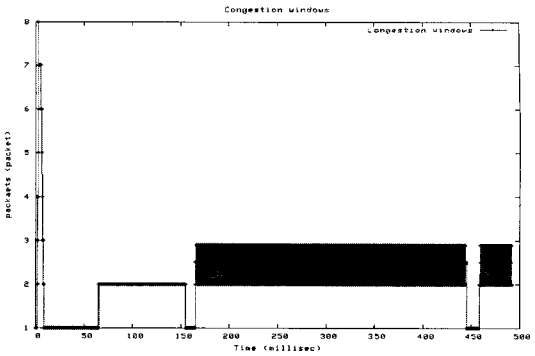

그림 4. 홉-수가 14일 때 제안된 알고리즘의 혼잡윈도우 추이
Fig. 4. Congestion Window of TCP Reno with proposed scheme(Hops = 14)

그림 5는 제안된 알고리즘의 TCP 최대 혼잡윈도우 크기이다. 송신측 TCP는 TCP 수신측에서 보낸 ACK 패킷의 advertised window 값으로 최대 혼잡윈도우 크기를 설정한다. TCP 수신측은 데이터 패킷의 ECN 필드의 빈도가 min_th에서 max_th 값 사이가 되도록 advertised window값을 조정한다. 시뮬레이션 결과 홉-수가 14인 경우에 최적 윈도우 값인 3으로 수렴됨을 알 수 있다.

표 2는 홉-수가 14인 TCP의 성능 결과를 보여준다. 제안된 알고리즘의 경우 네트워크 상황에 따라 수신측 TCP에서 송신측 윈도우 크기를 조절하여 약 50%의 수율 향상을 보였다. TCP Reno의 경우 평균 혼잡윈도우 값이 6.313이고 제안된 알고리즘의 경우 2.456이었다.

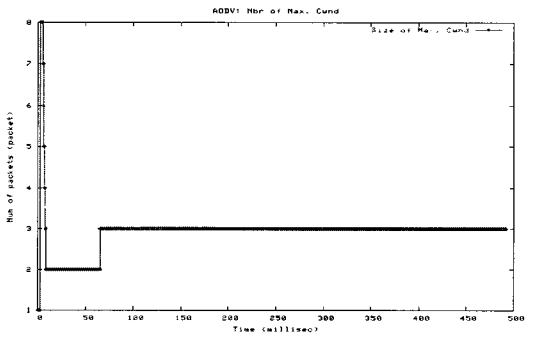

그림 5. 홉-수가 14일 때 제안된 알고리즘의 최대 혼잡윈도우 크기
Fig. 5. Maximum Congestion Window of TCP Reno with proposed scheme(Hops = 14)

표2. 홉-수가 14일 때 TCP 성능 비교표
Table 2. TCP Performance comparison(Hops = 14)

	TCP Reno	TCP Reno with proposed algorithm
TCP Thoughput(Kbps)	40.146	65.273
Average CWnd(packets)	6.313	2.456

그림 6과 그림 7은 홉-수를 1부터 증가하면서 실험을 수행한 결과를 보여준다. 홉-수가 증가할수록 TCP의 수율은 홉-수에 반비례하여 감소함을 알 수 있다. 이는 무선 채널을 인접노드 간에 공유하며, 4-홉 이내의 공간에서는 한 번에 하나의 패킷을 전송할 수 있는 무선 네트워크의 특성에 기인한다. 모든 경우에서 제안된 알고리즘이 TCP Reno보다 10~ 60 %의 성능이 향상되었다.

3.2 랜덤 토폴로지(Random Topology)

다수의 TCP 연결이 존재할 때 제안된 알고리즘의 성능 평가를 위하여, 60개의 모바일 노드로 이루어진 Ad-hoc 네트워크를 구성하고 시뮬레이션을 수행하였다. FTP 연결의 수는 5~20 개까지 순차적으로 증가하여 수행하였으며, 네트워크상의 모바일 노드는 최대 5 m/sec의 속도로 이동한다. 그림 8은 랜덤 토폴로지에서 FTP 연결 수에 따른 TCP 수율을 보여주고 있다.

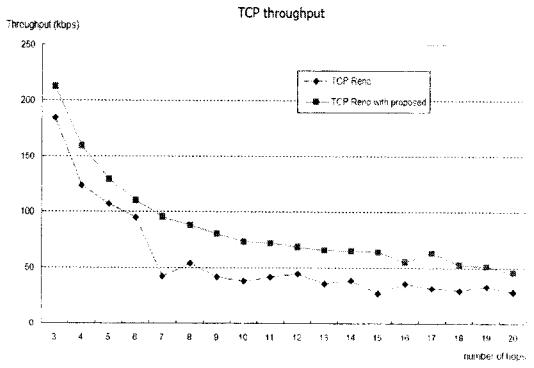

그림 6. 홉-수에 따른 TCP 수율
Fig. 6. TCP throughput vs. Hops

시뮬레이션 결과, 모든 상황에서 제안된 알고리즘이 TCP Reno보다 우수한 성능을 보여준다. 제안된 알고리즘은 네트워크의 상황에 따라 TCP 윈도우의 크기를 최적 값으로 설정하므로 기존 TCP의 높은 혼잡윈도우 크기에 따른 충돌을 방지하여 종단간 전송 효율을 높일 수 있었다.

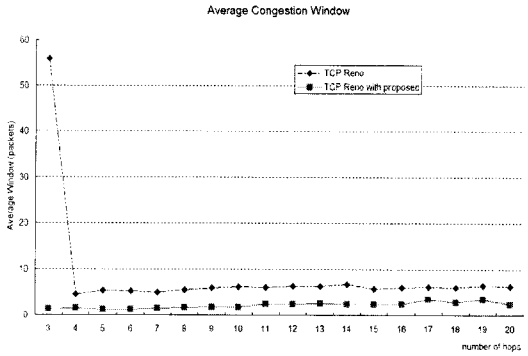

그림 7. TCP의 평균 혼잡윈도우 값
Fig. 7. Average Congestion Window of TCP

TCP 연결 수가 늘어날수록 제안된 알고리즘과 TCP Reno와 성능차가 줄어드는데, 이는 연결 수에 비례하여 트래픽이 증가한다. 그 결과, 제안된 알고리즘의 TCP 혼잡윈도우 크기를 최적 값에 설정하더라도, 증가한 트래픽에 의한 충돌이 발생되어 성능향상이 감소한 결과를 나타낸다.

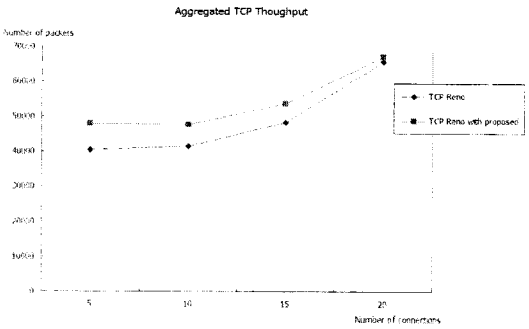

그림 8. 랜덤 토폴로지에서 TCP 수율
Fig. 8. TCP Throughput in Random Topology

V. 결 론

TCP는 대부분의 인터넷 어플리케이션에서 사용되는 프로토콜로 유선망에서 뿐 아니라, 무선망에서도 신뢰성 있는 전송 프로토콜로 지배적인 위치를 점할 것으로 예측된다. 그러나 IEEE 802.11 계열의 프로토콜 특성상 Ad-hoc 네트워크와 같은 다중 홉 환경에서는 크게 성능이 저하된다. 기존 연구에서 무선 Ad-hoc 네트워크상에 다중 홉을 경유하는 TCP 성능은 경로의 홉-수와 밀접한 관계가 있고, 또한 TCP의 송신 최대 혼잡윈도우를 경로의 홉 수에 따라 제한함으로써 무선채널 상에서 충돌을 줄일 수 있음을 보였다. 따라서 적절한 크기의 혼잡윈도우는 TCP 성능을 향상 시킨다. 본 논문에서는 경로의 홉 정보 없이 TCP 계층에서 최대 혼잡윈도우를 최적 값으로 조정하는 알고리즘을 제안하였다. 컴퓨터 시뮬레이션을 통해 제안된 알고리즘의 성능을 검증하기 위하여 체인 네트워크와 랜덤 네트워크 등의 다양한 환경에서 성능 평가를 수행하였으며 그 결과 제안된 알고리즘이 TCP 성능을 기존 TCP 알고리즘에 비해 패킷의 전송률 및 손실률을 크게 향상시켰다.

참고문헌

[1] C.E. Perkins, *Ad Hoc Networking*, Addison Wesley, Boston, MA, USA, January 2001.

[2] IEEE. Wireless lan medium access control(mac) and physical layer(phy) specifications - std 802.11. The Institute of Electrical and Electronics Engineers, 1999.

[3] V. Bharghavan, "Performance Analysis of a Medium Access Protocol for Wireless Packet Networks," *IEEE Performance and Dependability Symposium*, 1998.

[4] Z. Fu, X. Meng, and S. Lu, "How bad tcp can perform in mobile ad hoc networks," in *Proc. IEEE International Symposium on Computers and Communications (ISCC'02)*, Taormina, Italy, July 2002.

[5] M. Gerla, K. Tang, and R. Bagrodia, "Tcp performance in wireless multi-hop networks," in *Proc. IEEE International Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, New Orleans, Louisiana, USA, Feb. 1999.

[6] K. Chen, Y. Xue, and K. Nahrstedt. "On setting tcp's congestion window limit in mobile ad hoc networks," *IEEE International Conference on Communications(ICC 2003)*, Anchorage, Alaska, May 2003.

[7] C. Cordeiro, S. R. Das and D. P. Agrawal, "COPAS: Dynamic Contention-Balancing to Enhance the Performance of TCP over Multi-hop Wireless Networks," *Proc.of the 10th Int. Conf. on Computer Communication and Networks (IC3N)*, Miami, October 2002.

[8] Z. Fu, X. Meng, S. Lu, "How bad TCP can perform in mobile ad hoc networks," *Proc. IEEE International Symposium on Computers and Communications (ISCC'02)*, Taormina, Italy, July 2002.

[9] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on tcp throughput and loss," in *Proc. IEEE Infocom 2003*, San Francisco, California, USA, Apr. 2003.

[10] Floyd, S., "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, Vol. 24 No. 5, pp. 10-23, October 1994.

[11] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. "Capacity of ad hoc wireless network," *ACM MOBICOM'01*. Rome, Italy, July 2001.

[12] The network simulator ns-2. http://www.isi.edu/nsnam/ns/

[13] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, "Ad Hoc On-demand Distance Vector Routing", IETF Draft, 33 pages, Oct 1999.

저자소개

김 관 웅(Kwan-woong Kim)

한국해양정보통신학회논문지
제12권 제2호 참조

배 성 환(Sung-hwan Bae)

한국해양정보통신학회논문지
제12권 제2호 참조