논문 2005-42CI-2-1

MANET용 TCP의 성능 개선을 위한 단-대-단 방법

(End-to-End Method for Improving TCP Performance for MANET)

임 재 걸*

(Jaegeol Yim)

요 약

인터넷을 위하여 현재 사용되는 TCP는 MANET(Mobile Ad Hoc Network)에서 사용될 때 효율성이 떨어진다. 그 이유는 무선네트워크에서 패킷 손실의 주된 원인이 통신 오류임에도 불구하고, TCP는 모든 패킷 손실을 통신혼잡에 기인한다고 가정하기 때문이다. 이러한 현상을 해소하고 성능을 제고하기 위하여, 본 논문은 전송지연시간과 전송지연시간들의 차이를 고려하여 패킷 손실의 원인을 규명하는 단-대-단 (end-to-end) 방법을 제안한다. 제안한 방법은 두 가지 특성을 지닌다. 첫째, 제안된 방법은 패킷 손실이 발생할 경우에만 작동하기 때문에 에너지 효율적이다. 둘째, 단 방향 통신에 소요된 시간을 사용함으로 왕복시간을 고려하는 기존의 방법보다 정확성이 더 높다. 제안된 방법의 페트리 넷 (Petri net) 모형을 구축하고, 시뮬레이션을 실행한 결과를 분석하여, 제안된 방법이 표준 TCP 보다 생산성(throughput)과 통신지연시간 면에서 월등히 우수함을 보인다.

Abstract

The current implementation of TCP for the Internet is not efficient when used for Mobile Ad hoc Networks (MANETs). This is because TCP assumes that all packet losses are caused by congestion, whereas transmission errors are a main reason for packet losses in wireless networks. To remedy this situation and increase performance, we propose an end-to-end method of using propagation delays and the differences between propagation delays to distinguish the causes for packet losses. The proposed method has two characteristics: Firstly, it is energy-efficient because this solution is only initiated when a packet loss is detected. Secondly, our approach considers only the one way propagation delay and is more accurate in determining causes for packet losses than existing methods which consider round trip time. Petri net models of the proposed TCP and of the standard TCP have been built and simulations have been performed on them. Our simulation results show that the proposed approach increases throughput and reduces propagation delay compared with standard TCP.

Keywords: TCP, MANET, Wireless Network, Petri net, Congestion Control

I. 서 론

인터넷에서 신뢰성 높은 데이터 전달을 위하여 전송계층에 주로 사용되는 프로토콜은 TCP이다. TCP는 원래 유선 환경에서 개발되었기 때문에 하부 네트워크 계층의 신뢰성이 비교적 높다고 가정한다.

따라서, 패킷 손실이 발생하면, 송신 TCP는 이것을 혼잡의 증거로 보고, 혼잡 윈도우를 크게 감소시킴으로써 혼잡 조절을 한다. 패킷 손실의 원인이 주로 통신 혼잡인 유선 네트워크에서는 이 방법이 회선을 공유하는 연결(connection)들 간의 공평성도 유지시키고, 혼잡 조절도 효율적으로 잘 한다.

그러나, 이 방법은 비트 오류율이 높은 무선 네트워크에서 통신 성능을 저하시키는 주요 원인이 된다. 왜냐하면, 무선네트워크에서는 패킷 손실의 주요 원인이 통신 혼잡이 아닌 통신 오류임에도 불구하고, TCP는 패킷 손실의 원인을 밝히지 않고 모든 패킷 손실의 원인을 혼잡으로 간주하여, 혼잡윈도우를 크게 감소시킴으로써 회선의 용량을 허비하기 때문이다. 따라서, 통신 오류율이 높은 무선네트워크의 TCP 성능을 개선하려면, 패킷 손실의 원인을 규명하도록 TCP가 변형되어야

* 평생회원, 동국대학교 컴퓨터학과
(Department of Computer Science, Dongguk University)
접수일자: 2004년10월29일, 수정완료일: 2005년3월7일

한다.

패킷 손실의 원인을 규명하는 방법으로 단-대-단 방식(end-to-end)[1], ECN (Explicit Congestion Notification)[2] 방식, ELN (Explicit Loss Notification)[3] 방식 등이 있다. 단-대-단 방식은 경로상의 라우터나 게이트웨이가 제공하는 정보에 의존하지 않고 RTT(Round Trip Time)나 생산성(throughput) 등 TCP가 수집할 수 있는 정보를 바탕으로 통신 혼잡을 유추하는 방법이다. ECN 방식은 라우터나 게이트웨이에서 버퍼가 어느 정도 차서 세그먼트 폐기(drop)가 예상되면 세그먼트에 ECN 마킹을 하여줌으로써 송신TCP가 통신혼잡을 감지하고 윈도우의 크기를 조절하게 하는 것이다. ELN 방법은 통신 오류의 발생을 알린다. 즉, IP 계층에서 통신 오류를 탐지할 때, 오류 발생 패킷을 버리는 대신 오류발생 사실을 TCP에게 전달하는 것이다. 그러므로, TCP는 세그먼트 손실을 통신 혼잡으로 확신할 수 있는 것이다

이 논문은 통신 오류율이 높은 무선 링크로 연결된 MANET를 지원하는 TCP의 성능 개선을 위한 단-대-단 방식을 제안한다. 제안된 방식의 특징은 패킷 손실이 발생할 경우에만 작동하기 때문에 에너지 효율적이라는 것과, 단 방향 전송지연시간을 참조하기 때문에 RTT를 사용하는 방법보다 더 정확하다는 것을 들 수 있다.

본 논문의 전개 방향은 다음과 같다. Ⅱ절에서 관련 연구를 고찰하여 본 연구의 중요성을 강조하고, Ⅲ절에서 제안하는 방법 및 제안하는 방법을 실제 TCP에 반영하기 위한 방안을 소개하고, Ⅳ절에서는 본 논문에서 사용한 시뮬레이션 모델을 구축하기 위한 이론적 바탕을 제공하는 페트리 넷 이론을 간단히 소개하고, 제안하는 TCP에 대한 페트리 넷 모형도 소개한다. Ⅴ절에서 시뮬레이션 결과를 소개하고, Ⅵ절에서 시뮬레이션 결과를 분석하며, 끝으로 Ⅶ절 결론에서 본 논문의 목적과 중요성을 다시 한번 강조한다.

Ⅱ. 관련연구

이동무선네트워크를 지원하는 TCP의 성능 개선을 위하여 기존에 제안된 방법은[4, 5] 중간 노드가 혼잡을 발견하여 혼잡 발생을 알리는 메시지를 생성하여 주어야 함으로, 혼잡 제어 과정에 중간 노드가 반드시 참여하여야 한다. 예를 들어, Shagdar 등은 ELN의 무선네트워크 버전으로, [4]에서 MAC(Medium Access) 계층으로부터 오는 정보를 바탕으로 패킷 손실의 원인을 구분하는 EWLN(Explicit Wireless Loss Notification) 방식을 제안하였고, Chawla와 Nandi는 [5]에서 혼잡을 제어하기 위하여 ECN 방법을 사용하는FECN(Freeze and Explicit Congestion Notification)을 제안하였다. 이러한 방법들은 MANET에서 TCP의 성능을 개선하는 효과를 발휘하지만, 통신 경로상의 모든 이동 노드들이 항상 통신 혼잡 상황을 테스트하는 작업을 수행하기 위하여 에너지를 소모해야 하는 단점이 있다. 그런데 MANET에서는 이동노드들의 에너지가 제한적이므로 에너지를 많이 소모하는 이러한 기존의 방법들은 실용화하기 어려움이 있다. 따라서, MANET에서는 송신자와 수신자만 통신 혼잡을 판단하는 작업을 수행하는 단-대-단 방식을 채용하는 것이 필요함으로, 본 논문은 MANET용 TCP의 성능 개선을 위한 단-대-단 방식을 제안한다.

지금까지 많은 수의 효율적인 단-대-단 방식이[1, 6] 제안되었다. Barman 등은 [1]에서 RTT를 참조하여 혼잡을 추정하는 방법을 소개하였고, Fu는 [6]에서 여러 가지 측정치를 포괄적으로 참조하여 혼잡을 추정하는 방법을 제안하였다. 그러나 [1]의 방법은 RTT를 사용하기 때문에 혼잡이 송신 경로에서 발생하였는지 아니면 수신 경로에서 발생하였는지 판단하기가 불가능하고, [6]의 방법은 송신자에게 세그먼트가 도착할 때마다 여러 가지 측정치를 계산하기 때문에 에너지 효율성에서 세그먼트 손실이 발생할 때만 측정치를 계산하는 본 논문이 제안하는 방법보다 비효율적이다.

Ⅲ. 제안하는 방법

세그먼트 손실의 원인은 병목 라우터(bottleneck router)에서 발생하는 세그먼트 폐기(drop)와 통신오류로 구분된다. 라우터의 처리 능력보다 라우터에 도착하는 세그먼트의 빈도가 더 높으면, 라우터의 버퍼에 세그먼트가 쌓이기 시작한다. 버퍼의 길이가 차츰 길어짐에 따라 세그먼트가 버퍼에 머무는 시간이 차츰 길어지고 따라서 세그먼트의 전송지연시간도 차츰 길어지게 된다. 더 나아가서, 쌓이는 시간이 어느 정도 지속되면 버퍼의 용량을 넘치게 되고, 그 이후에 도착하는 세그먼트는 폐기된다. 따라서, 본 논문은 전송지연시간과 전송지연시간들 사이의 차이의 변화를 참조하여 세그먼트 손실의 원인을 판단한다. 제안하는 방법은 특히 이동단말기의 에너지 소모를 최소화하기 위하여, 세그먼트 손실이 발생할 경우에만 작동한다.

1. 전송 지연시간

MANET에서는 전송지연시간이 RTT (Round Trip Time) 보다 혼잡을 가리키는 지표로 더 적당하다. 그 이유는 무선 링크가 비대칭적이기 때문이다. 즉, 송신 TCP를 출발한 세그먼트가 수신 TCP에 도착하는데 걸리는 시간과 수신 TCP의 ACK 메시지가 수신 TCP를 출발하여 송신 TCP에 도착하는데 걸리는 시간과의 차이가 매우 크기 때문이다. 따라서, 전송지연시간은 RTT의 반이라고 할 수가 없다. 이러한 비대칭성을 감안할 때, 전송지연시간이 RTT보다 혼잡 탐지에서 더 정확한 지표가 되는 것이다.

제안하는 방법은 전송지연시간의 평균을 바탕으로 임계치를 구하여 사용한다. 전송지연시간의 평균 $\overline{P}$는 다음과 같이 RTT를 구하는 식과 비슷한 식에 의하여 계산된다.

$$\overline{P} = \alpha \overline{P} + (1-\alpha) P_{\neq w}$$

단, α는 0과 1 사이의 실수

P_{new}는 최근 도착한 전송지연시간.

실제 전송지연시간은 세그먼트가 전송 TCP를 출발하여 수신 TCP에 도착하는데 걸리는 시간이므로, 전송 TCP의 시계와 수신 TCP의 시계가 정확히 일치하지 않는 한, 수신 TCP의 시계로 측정한 도착 시각에서 송신 TCP의 시계로 측정한 출발 시각을 뺀 차이를 실제 전송지연시간이라고 할 수 없다. 그러나, 제안한 방법은 수신 TCP의 시계로 측정한 도착 시각에서 송신 TCP의 시계로 측정한 출발 시각을 뺀 차이를 전송지연시간이라고 한다. 본 방법은 이렇게 구한 전송지연시간의 평균을 바탕으로 임계치를 설정하고, 이러한 전송지연시간들 사이의 차이를 고려하여 혼잡을 판단하기 때문에 송신 TCP와 수신 TCP의 시계가 일치하지 않더라도 혼잡 판단에 영향을 미치지 않는다.

2. 전송지연시간들 간의 차이

전송지연시간들 간의 차이의 정도가 또한 혼잡의 지표가 된다. 즉, 혼잡이 발생하면 TCP 연결상의 병목 라우터 버퍼에 점점 더 많은 세그먼트들이 쌓여, 통신 혼잡이 해소될 때까지 세그먼트의 전송지연시간이 점점 더 길어진다. 이러한 상태가 계속되면, 결국 버퍼 넘침이 발생하여 세그먼트를 폐기하기까지 이르게된다. 그래서 d_i를 i 번째 세그먼트의 전송지연시간에서 i-1번째 세그먼트의 전송지연시간을 뺀 차라고 할 때, 혼잡이 발생하면, $d_i - d_{i-1}$의 차이가 0이 아니고, 더 나아가서 차이가 점점 증가하게 된다. 즉, $d_{i+1} - d_i > d_i - d_{i-1}$가 된다.

위에서 소개한 두 가지 지표를 이용하여 혼잡이 발생하였는지 판단하는 기준을 다음과 같이 제안한다.

혼잡의 기준: 다음 두 가지 조건이 만족할 때 네트워크가 혼잡이라고 한다 ;

가. 전송 지연시간이 임계치보다 크다. 단, P가 전송지연시간의 평균치이고 β 가 네트워크 응용을 반영하는 변수라면, 임계치는 βP이다.

나. 전송 지연시간들 간의 차이가 점차 증가한다.

세그먼트 손실이 탐지되면, 송신 TCP는 위에 제시된 혼잡의 기준을 계산하여 손실의 원인이 혼잡인지 아닌지 판단한다. 혼잡이라고 판단되면 혼잡윈도우를 반감하지만, 그렇지 않으면 송신 TCP는 세그먼트 손실의 원인이 통신오류라고 간주하고 혼잡윈도우의 크기를 그대로 둔다.

제안된 방법을 구현하기 위하여, 송신 TCP와 수신 TCP를 각각 다음과 같이 변경한다.

수신 TCP: 세그먼트를 수신하면 $T_{recv} - T_{send}$를 계산한다. 단, T_{recv}는 세그먼트의 도착시각이고 T_{send}는 송신 TCP가 이 세그먼트를 전송한 시각이다. 이렇게 구한 전송지연시간을 해당 세그먼트의 ACK 세그먼트에 첨부하여 보낸다. 또한, 송신 TCP가 네트워크의 최신 상황을 나타내는 정보를 중단 없이 받아 볼 수 있도록 세그먼트를 수신할 때마다 ACK를 지체없이 보낸다. 한 가지 언급해야 할 사항은 $T_{recv} - T_{send}$를 구하기 위하여 송신 TCP의 시계와 수신 TCP의 시계가 동기화되어 있어야할 필요는 없다는 것이다. 그 이유는 혼잡의 기준에서 전송지연시간 자체를 사용하는 것이 아니라, 전송지연시간의 평균에 기반한 임계치와 전송지연시간들 간의 차이를 참조하기 때문이다.

송신 TCP: ACK를 받으면 전송지연시간을 버퍼에 저장하고, 동시에, 평균 전송지연시간을 갱신한다. 전송시간을 저장할 버퍼의 길이는 응용에 따라 달라지지만 비교적 작은 용량으로 충분하다. 본 연구에서는 버퍼의 크기를 8로 하였다.

Ⅳ. 제안된 TCP의 페트리 넷 모형

제안하는 방법의 효율성을 보이기 위하여 제안하는 방법을 적용한 TCP 모형을 페트리 넷(Petri net)으로 구축하고, 시뮬레이션을 통하여 표준 TCP와 제안하는 TCP의 생산성(throughout)을 비교한다.

본 논문에서 사용한 TCP 모형은 네트워크 연구에서 일반적으로 사용되는 ns-2[7]로 구축된 것이 아니라 시스템 분석에 일반적으로 사용되는 페트리 넷의 형태로 구축되었다. 따라서, 시뮬레이션 도중 라우터에서 발생한 세그먼트 폐기의 수, 재전송을 위하여 송신자 측에 대기 중인 세그먼트의 수, 순서에 어긋나게 수신자 측에 도착한 세그먼트의 수, 등 각종 세세한 데이터를 구하여 볼 수 있다는 장점이 있다. 페트리 넷[8]을 간략히 소개하면 다음과 같다.

1. 페트리 넷 소개

페트리 넷은 1962년에 처음 소개된 이후 시스템 성능 테스트와 통신 규약의 일관성 및 타당성 테스트 등을 비롯한 컴퓨터 관련 전 분야에서 시스템 모델과 분석 도구로 널리 사용되고 있다. 이와 같이 페트리 넷을 이용한 연구들이 활발히 진행되는 이유는 페트리 넷 모델은 구축하기가 용이하고 구축된 모델을 분석하는 수학적 방법이 널리 연구되었기 때문이다.

페트리 넷의 정의는 표 1과 같이 정의되는 이분그래프다[8]. 여타의 그래프가 그렇듯이 페트리 넷도 그림으로 표현하면 더 직관적이다. 그림에서 P의 원소 장소는 원이나 타원으로, T의 원소 변천은 사각형으로 표현된다. F의 정의에서 보듯이 간선은 장소와 변천, 혹은 변천과 장소를 연결하고, 장소끼리 혹은 변천끼리는 연결하지 않기 때문에 페트리 넷은 이분그래프이다. 페트리 넷의 역동적 성질에 대해서 모의실험을 가능케 하여 주는 변천 격발 (fire)의 정의는 다음과 같다.

표 1. 페트리 넷의 정의
Table 1. The Definition of Petri net.

페트리 넷 N은 5가지 요소로 구성된다.
N = (P, T, F, W, M_0) where:
P={p_1, p_2, ..., p_n}은 장소 (place)라는 유한 집합,
T={t_1, t_2, ..., t_m}은 변천 (transition)이라는 유한 집합,
F : 유향간선이라는 (P×T)와 (T×P)의 합집합의 부분집합,
W: F → {1, 2, 3, ...}는 유향간선의 가중치를 결정하는 함수,
M_0: P → {0, 1, 2, 3, ...}은 초기에 각 장소에 놓인 토큰의 수를 표현하며, 초기 마킹(marking)이라고 함.
단, P와 T의 교집합은 공집합이고, P와 T의 합집합은 공집합이 아님.

가. 하나의 변천 t에 대하여, 입력 장소 p가 최소한 W(p, t) 만큼의 토큰을 갖고 있으면, t는 장전되었다고 한다.
나. 장전된 t는 격발될 수도 있고, 아니 될 수도 있다.
다. t의 격발은 W(p, t) 만큼의 토큰을 각 입력 장소 p에서 제거하고, W(t, p) 만큼의 토큰을 각 출력 장소에 더하여 준다.

표 1에 보인 페트리 넷에는 시간이라는 개념이 나타나 있지 않다. 본 논문의 목적이 제안하는 TCP와 표준 TCP의 성능을 비교하는 것이므로 TCP의 모델을 구축하기 위하여 시간 개념이 가미된 변형된 페트리 넷이 필요하다. 더 나아가서, 토큰을 까만 점으로 제한하면 표현할 수 있는 현상이 매우 제한적임으로 토큰에 이름을 부여하는 것을 허용한다. 시간을 표시할 수 있고, 토큰에 이름이 부여된 페트리 넷으로 퍼지 시간 고급 페트리 넷 (Fuzzy Timing High-Level Petri net)이 [9]에 소개되었다.

퍼지 시간 고급 페트리 넷의 예로 그림 1의 일부를 이해하기 쉽게 자세히 설명하지만, 혹 페트리 넷 용어와 개념에 대한 보충 설명이 필요하면 [8]을 참고하기 바란다. 그림 1의 장소에는 'Packets to be sent', 'One by one', ... 등이 있으며, 변천에는 'Gen. Tokens with Traffic type', 'Is congest type', ... 등이 있다. 장소 'Packets to be sent'의 외부에 붙어 있는 E는 이 장소에 놓일 수 있는 토큰의 유형을 제한하고, 500'e는 이 장소에 초기에 놓인 토큰의 수를 의미한다. 장소 'One by one'의 외부에 붙어 있는 e는 이 장소에 초기에 놓인 토큰의 수가 하나임을 의미한다. 1'e와 e는 동일한 의미를 갖는다. 장소 Packets to be sent와 변천 Gen. Token with Traffic type을 연결하는 간선에는 1'e라는 레이블이 붙어 있고, 장소 One by one과 변천 Gen. Tokens with Traffic type을 연결하는 간선에는 e라는 레이블이 붙어 있다. 변천 Gen. Tokens with Traffic type은 위에 소개된 격발 정의에 의하여 장전되었음을 알 수 있다. 이 변천이 격발하면 격발 정의에 의하여 장소 One by one과 Packets to be sent에서 각각 하나의 토큰을 삭제하고, 변천 Gen. Tokens with Traffic type

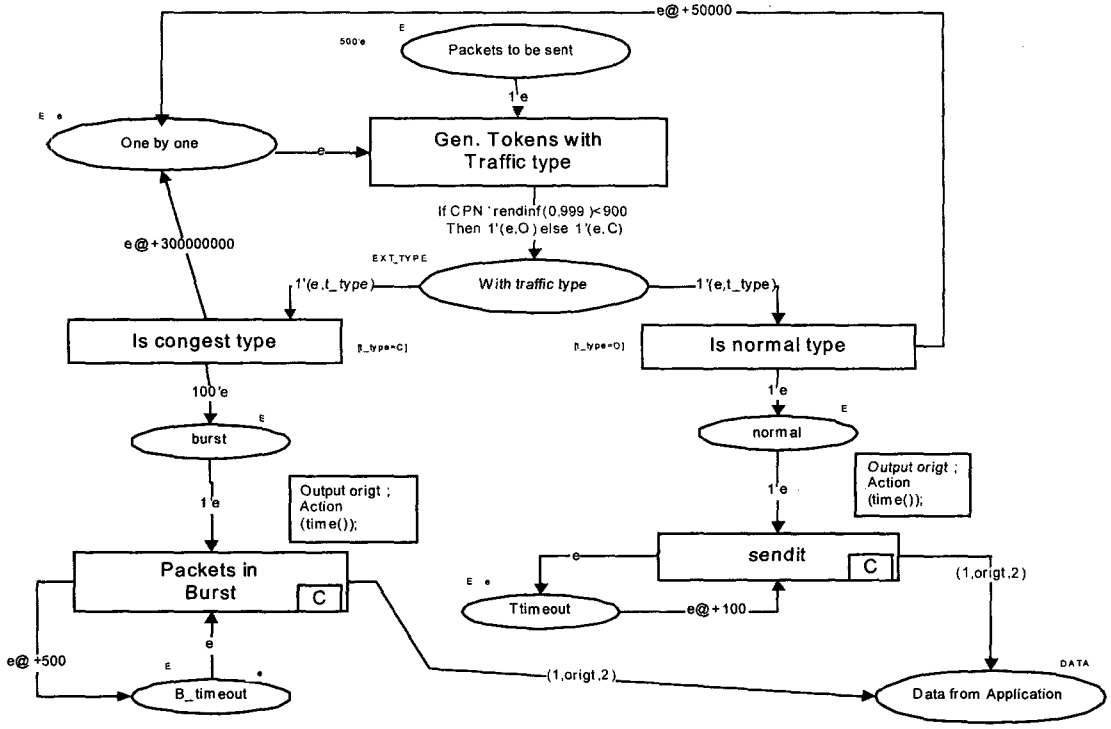

그림 1. 제안하는 TCP 모형의 세그먼트 생성부
Fig. 1. Segment generation part of the proposed TCP model.

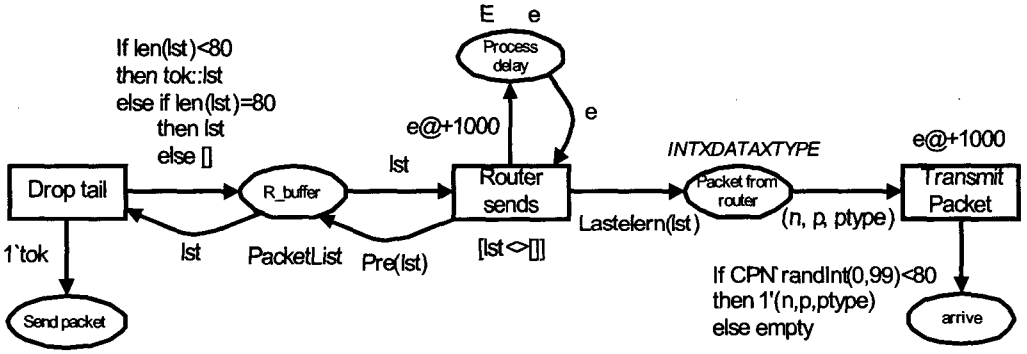

그림 2. 제안하는 TCP의 네트워크 모듈
Fig. 2. Network module of the proposed TCP.

와 장소 With traffic type을 연결하는 간선에 붙어 있는 다음 함수를 수행하여 결과를 장소 With traffic type에 놓는다.

If CPN'randinf(0,999)<900

Then 1'(e,O) else 1'(e,C)

장소 One by one에 있던 하나의 토큰을 삭제하였음으로 변천 Gen. Tokens with Traffic type은 더 이상 격발이 불가능하게 된다. 한편, 위의 함수는 (e,O) 혹은 (e,C)라는 토큰을 생성하여 장소 With traffic type에 놓는다. 이때, CPN'randinf(0,999)가 0부터 999 사이의 정수를 무작위로 생성하기 때문에, (e,O)가 생성될 확률은 90%이고 (e,C)가 생성될 확률은 10%이다.

변천 Is normal type에는 [t_type = O]라는 레이블이 붙어 있는데, 이러한 레이블을 변천의 가드(guard)라고 하며 격발가능의 조건이 된다. 변천 Is congest type의 가드는 [t_type = C]이다. 따라서 장소 With traffic type에 (e,O) 토큰이 놓이면 변천 Is normal type이 격발가능하게 되고, (e,C) 토큰이 놓이면 변천 Is congest type이 격발가능하게 된다. 가령 변천 Is congest type이 격발하면, 장소 One by one과 장소 Burst에 각각 토큰을 놓게 되는데, 연결하는 간선에 붙어있는 레이블에 따라 장소 One by one에는 e@+300,000,000이라는 토큰

표 2. 각 장소와 토큰 유형
Table 2. Token of each place.

장소	Color	초기마킹	설명
Packets	colorless and timed (E)	500`e	이 장소의 토큰은 응용 프로그램에서 온 패킷에 해당함.
one by one	colorless and timed (E)	one colorless token, e	토큰 생성 속도를 조절하기 위한 장소임.
with traffic type	product of E and T_TYPE, EXT_TYPE	없음	통신유형은 일상형(O)과 혼잡유형(C)임: {O, C}.
burst	colorless and timed (E)	없음	혼잡유형 패킷이 100개의 토큰으로 변화됨.
normal	colorless and timed (E)	없음	일상형 패킷이 한 개의 토큰으로 됨.
B_timeout	colorless and timed (E)	one colorless token, e	혼잡유형 세그먼트의 생성 속도를 조절하는 장소임..
Timeout	colorless and timed (E)	one colorless token, e	일상형 세그먼트의 생성 속도를 조절하는 장소임.

표 3. 각 변천의 역할
Table 3. Operation of each transition.

변천	Code or Guard	역할
Gen. tokens with traffic type	없음	응용 프로그램이 가끔 통신을 요청하듯, 주기적으로 토큰을 생성함.
Is congest type	[traffic_type = C]	통신유형이 혼잡 유형인가를 테스트함.
Is normal type	[traffic_type = O]	통신유형이 일상형인가를 테스트함.
Packets in burst	output origt; action (time());	500 시간단위마다 토큰을 생성하고, 생성시각을 'origt'에 기록함.
SendIt	output origt; action (time());	하나의 토큰을 생성하고, 생성 시각을 'origt'에 기록함.

표 4. 몇몇 중요 간선에 대한 설명
Table 4. Explanation of some inscriptions.

간선	Inscription	설명
('Gen. tokens with traffic type', 'with traffic type')	if CPN`randint(0, 999) < 900 then 1`(e, O) else 1`(e, C)	90%가 일상유형, 10%가 혼잡유형의 토큰임.
('Is congest type', 'one by one')	e@+300,000,000	혼잡유형 토큰생성후 이 시간만큼 쉬었다가 다음 토큰을 생성함.
('Is normal type', 'one by one')	e@+50,000	일상형 토큰 생성후 이 시간만큼 쉬었다가 다음 토큰을 생성함.
('is congest type', 'burst')	100`e	혼잡유형 패킷 하나가 100개의 세그먼트로 됨.
('Send packets in burst', 'B_timeout')	e@+500	500 단위시간마다 세그먼트를 생성함.
('SendIt', 'Data From Applocation')	(1, origt, 2)	토큰 생성 시각이 'origt'에 기록됨.

을 놓고, Burst에는 100 개의 e 토큰을 놓는다.

e@+300,000,000의 300,000,000은 해당 토큰이 300,000,000 단위시간 동안 격발에 사용될 수 없도록 (disable) 하는 역할을 한다. 한편,
Burst에 놓인 100 개의 e 토큰은 매 500 단위시간마다 변천 Packets in burst를 격발하여 통신혼잡을 유발한다. 그림 1에 대한 자세한 설명은 다음 절에서 계속된다.

2. 페트리 넷 모형

표준 TCP의 페트리 넷 모형은 [10]에 소개된 모형을 사용한다. 제안하는 방법을 채용한 TCP의 페트리 넷 모형은 [10]에 소개된 모형의 송신 TCP와 수신 TCP 부분을 3절에서 소개한 바와 같이 수정하고, 그림 1과 같은 세그먼트 생성부와 그림 2와 같은 네트워크부를 첨부하여 구축한다. 페트리 넷 모형 구축과 시뮬레이션 수행을 위하여 Design/CPN[11]을 사용한다. Design/CPN은 페트리 넷 모형을 구축하는 에디터와 시뮬레이션을 수행하는 시뮬레이터를 비롯하여 시뮬레이션 결과를 분석하는 모듈을 포함한 다양한 모듈을 제공하는 페트리 넷 모형 구축을 위한 통합소프트웨어다.

페트리 넷 모형 설명에서, Design/CPN 관련 용어는 [11]을 참조하고, 페트리 넷 관련 용어는 [8]을 참조하기 바란다. 그림 1의 세그먼트 생성부는 변천 'Gen. Tokens with traffic type'을 격발하는 것으로 세그먼트 생성 작업을 시작한다. 이 변천의 격발 결과 하나의 토큰이 장소 'With traffic type'에 놓이는데, 90%는 (e, O)이고 나머지 10%는(e, C)이다. 토큰 (e, C)는 변천 'Is congest type'을 격발하여 100 개의 토큰을 장소 'burst'에 놓으면서 동시에 장소 "one by one'에 타임스탬프가 +300,000,000 단위시간인 토큰을 놓는다. 어떤 토큰의 타임스탬프가 +300,000,000이라는 것은 시뮬레이션 시간이 300,000,000 단위시간 경과해야 그 토큰의 사용이 가능하게 된다는 의미가 있으므로, 변천 'Gen. Tokens with traffic type'의 격발을 이 시간 동안 정지하는 역할을 수행하게 된다. 한편 'burst'에 놓인 100 개의 토큰은 변천 'Send packets in burst'의 입력으로 사용되어 100개의 세그먼트를 500단위시간마다 하나씩 장소 'Data from application'에 놓음으로써 통신 혼잡을 유발하는 역할을 한다.

각 장소의 역할, 각 변천의 역할, 그리고 간선의 역할은 각각 표 2, 표3, 표 4에 요약된 바와 같다.

제안한 TCP의 성능 평가를 위한 모형의 네트워크 모듈은 그림 2와 같다. MANET의 통신 오류율이 비교적 높다는 것은 잘 알려진 사실임으로 본 모델에서는 오류율을 0.2로 설정하였다. 통신 오류율은 변천 'Transmit Packet'에서 장소 'arrive'로 가는 간선에 연합된 다음에 보이는 함수로 모델된다.

```
if CPN'randInt(0, 99) < 80
then 1'(n, p, ptype)
else empty
```

병목 현상을 보이는 라우터도 그림 2에 보인다. 라우터 버퍼는 선입선출 방법으로 세그먼트를 처리한다. 버퍼의 크기는 80으로 설정하고, 선입선출 방법은 변천 'Drop tail'에서 장소 'r_buffer'로 가는 간선에 연합된 다음 함수에 의하여 모델된다.

```
if len(lst) < 80
then tok::lst
else if len(lst) = 80
    then lst
    else []
```

```
update_prop_list()
1  when (receives an acknowledgment)
2     update propagation delay list;

predict_loss_cause()
3  if (packet is lost)
4     bool congest   false
        // true means a network is in congestion state
5  if ((the weighted sum of propagation delay in
the list > threshold) and ( for (모든 전송지연시간 di
에 대하여 di>di-1)))    then congest □ true
```

그림 3. 제안하는 방법의 송신 TCP의 혼잡제어 알고리즘

Fig. 3. Congestion control algorithm of the proposed TCP at sender.

'lst'는 라우터 버퍼에 들어 있는 세그먼트들을 가리키는 변수이며, 'tok'는 새로 도착하는 세그먼트를 가리키는 변수다. 'lst'의 길이가 80보다 작으면 tok를 연결하고 그렇지 않으면 tok를 폐기한다.

본 논문이 제안하는 송신 TCP의 혼잡 판단 알고리즘은 그림 3에 보이는 바와 같다. 문장 1, 2는 수신확인 세그먼트가 송신 TCP에 도착하면, 전송지연 시간을 추출하여 리스트에 첨가하고, 리스트에 기존에 있던 전송지연시간 중 가장 오래된 것을 제거하여 최근 몇 개(본 실험에서는 8 개)의 세그먼트를 리스트에 보관한다. 세그먼트 손실이 탐지되면 문장 5에서 전송지연시간 리스트에 있는 전송지연시간들의 합이 임계치를 넘고 전송지연시간이 점차 증가하면 혼잡이라고 판단한다. 전송지연시간들의 합을 구할 때에는 최근 것에 더 많은 무게를 둔다.

V. 시뮬레이션

본 시뮬레이션은 네트워크 전용 시뮬레이션 도구인 ns-2가 아니라, 일반적인 시스템 분석 도구인 Design/CPN [11]으로, 두 가지 관점에서 시뮬레이션을 실행하고자 한다. 한 가지는 혼잡판단 함수의 정확도 측정이고 다른 한 가지는 생산성 비교이다.

1. 혼잡도 판단

본 시뮬레이션에서는 제안하는 방법이 얼마나 정확하게 혼잡을 판단하는지 분석하려고 한다. 이를 위하여

```
input(n, lst)
...
val world1 = open_append "droptailed"
val world2 = open_append "notdroptailed"
...
(if len(lst) >= 80 then
    output (world1, makestring n ...
```

그림 4. 변천 Drop tail에 첨부된 코드
Fig. 4. The code attached to the transition Drop tail.

표 5. 비교하는 차이의 수가 정확도에 미치는 영향을 분석하기 위한 실험 결과.
Table 5. Simulation results for an analysis of the effect of the number of differences compared.

	ST	10	20	30	40
폐기	0	0	0	17	46
통과	153	186	272	711	674
혼잡재전송	53	75	138	405	296
오류재전송	0	14	34	223	324

그림 1의 세그먼트 생성부를 100 개의 세그먼트를 매 5 단위시간마다 하나씩 생성하도록 변형한다. 그림 2의 라우터 처리 시간을 10 단위 시간으로 변형하고, 전송시간은 50 단위시간으로 한다.

정확도 비교를 위하여 라우터에서 폐기되는 세그먼트와 제대로 전송되는 세그먼트를 각각 기록하고, 재전송시에도 혼잡으로 판단하는지 아니면 전송오류로 판단하는지 기록한다. 이를 위하여 그림 4와 같은 코드를 변천 Drop tail에 첨부하고, 비슷한 코드를 재전송 변천에도 첨부한다.

'전송지연시간들 간의 차이가 점차 증가하는 것'이 얼마나 정확히 혼잡을 판단하는지 보이기 위하여 β = 10으로 고정하고 비교하는 차이의 수를 10, 20, 30, 40으로 설정하여 각각 시뮬레이션을 실행하여, 표 5와 같은 결과를 얻었다. 폐기는 라우터에서 폐기된 세그먼트의 수, 통과는 폐기되지 않고 라우터를 통과한 세그먼트의 수, 혼잡재전송은 재전송시 혼잡으로 판단된 경우의 수, 오류재전송은 재전송시 혼잡이 아니라고 판단된 경우의 수이다. 한편, ST는 표준 TCP, 10, 20, 30, 40은 각각 비교하는 지연시간의 차이의 수를 의미한다.

비교하는 차이의 수가 증가함에 따라 혼잡이라고 판단되는 경우의 확률이 감소하게 되고, 오류재전송으로 판단하는 확률이 증가한다. 그 결과 혼잡윈도우의 크기가 증가되어 한꺼번에 전송되는 세그먼트의 수가 증가된다. 그런데 전송오류율을 0.2로 하였기 때문에 이들은 대부분이 재전송된다. 그래서 재전송되는 세그먼트의 수도 증가하게 된다.

표 6. 30의 경우 제안하는 방법의 정확도 분석을 위한 실험결과
Table 6. Simulation results for the precision analysis of our prediction in the case of 30.

판단 / 사실	오류(223)	혼잡(405)
오류(611)	223 (0.36)	388 (0.64)
혼잡(17)	0 (0.0)	17 (100%)

표 7. β 값이 정확도에 미치는 영향을 분석하기 위한 실험 결과.
Table 7. Simulation results for an analysis of the effect of β.

β 값 / 통계치	1	1.3	1.5	1.7	2
폐기	0	0	0	0	231
통과	153	178	178	178	964
혼잡재전송	53	8	8	8	0
오류재전송	0	70	70	70	1095

30의 경우의 정확도를 살펴보면 표 6과 같다. 송신된 세그먼트가 100개인데 혼잡윈도우의 크기가 증가함에 따라 대부분이 재전송되어 라우터를 통과한 세그먼트의 수는 총 711개이고, 그 중 100개는 성공적으로 접수확인이 된 것이므로 나머지 611개가 전송오류이다. 세그먼트 폐기는 혼잡의 결과임으로 실제 혼잡은 17이다. 출력결과를 살펴본 결과 이들은 모두 정확히 혼잡으로 판단되었음을 알 수 있었다. 따라서 비교하는 지연시간의 수가 30일 경우에는 혼잡을 100% 혼잡으로 판단함을 알 수 있다. 10과 20의 경우에는 폐기된 세그먼트의 수가 0임으로 30이하의 경우에는 모두 혼잡을 100% 혼잡으로 판단한다고 할 수 있다. 오류를 오류로 판단하는 비율은 0.36이고, 오류를 혼잡으로 잘못 판단하는 비율은 0.64이다.

다음은 비교하는 전송시간의 차이들의 수를 0으로 고정시키고 α 역시 0으로 고정시킨 후, β 값을 변경하면서 시뮬레이션을 실행하여 표 7과 같은 결과를 얻었다. β 값이 1일 경우에는 모든 재전송의 원인을 혼잡재전송으로 판단하고, β 값을 2로 증가시키면 모든 재전송의 원인을 오류재전송으로 판단하는 것을 볼 수 있으며, β 값이 1.3이든, 1.5이든, 1.7이든 동일한 결과를 보이는데, 이것은 본 실험 환경을 5 단위시간마다 세그먼트를 생성하고 라우터 처리시간이 10 단위시간이며 전

표 8. 제안하는 방법과 기존 방법의 생산성 비교
Table 8. Throughput Comparison for the existing methods and the proposed method.

방법 / 생산성	30	RTT	β =1.5	β = 1.5 이고 30
통신 완료에 소요된 시간	8,153	9.234	7,998	7,960

송 지연시간은 50 단위시간으로 설정하였기 때문이다.

RTT 시간을 비교하는 기존의 방법[1]과 가는 방향의 전송시간만 비교하는 제안하는 방법의 성능을 비교하기 위하여 30의 경우를 선택하여, 양방향 전송시간을 모두 다음과 같이 변경한 후, 100개의 세그먼트를 전송 완료하는데 걸리는 시간을 측정하였다.

@+CPN'randInt(48, 52)

제안하는 방법은 8,153 단위시간을 소요한데 반하여, RTT 방법은 9,234 단위시간을 소요하였다. 이로부터 통신혼잡이 실제로 발생하는 가는 방향의 전송시간만 고려하는 방법이 RTT를 고려하는 방법보다 우월하다는 것을 알 수 있다.

다음은 비교하는 전송시간의 차이들의 수를 30으로 하고 또한 β =1.5로 하여 100 세그먼트 전송을 완료하는데 걸리는 시간을 관찰한 결과 표 8에 보이는 바와 같이 7,960 단위시간이 소요됨을 알 수 있었다. 이로부터 '전송시간이 임계치보다 크고' '전송시간의 차이가 증가하는' 두 가지 조건을 모두 만족할 때 혼잡이라고 판단하는 본 논문이 제안하는 방법이 두 가지 중 한 가지만으로 혼잡을 판단하는 기존의 방법보다 생산성을 증대시킨다는 것을 알 수 있다. 연결 초기에는 '전송시간의 차이'를 참조하는 방법의 정확도가 높고, 전송이 어느 정도 진행되면 전송시간을 임계치와 비교하는 방법의 정확도가 높게 됨으로 두 가지 기준을 모두 고려하는 제안하는 방법이 생산성을 더욱 제고하는 것이다.

2. 생산성 비교

본 시뮬레이션에서, 세그먼트 생성 모듈에는 평상 모드와 혼잡 모드의 두 가지 선택이 있으며, 장소 'With traffic type'으로 들어오는 간선에 연합된 함수에 반영된 확률에 따라 모드가 결정된다. 본 시뮬레이션에서는 혼잡모드의 토큰을 10%, 평상 모드의 토큰을 90% 생성한다. 혼잡모드의 토큰 하나는 100 개의 세그먼트를 500 단위시간마다 하나씩 생성함으로써 틀림없이 혼잡을 유발하고, 평상 모드의 토큰은 50,000 단위시간에 하나의 세그먼트를 생성한다.

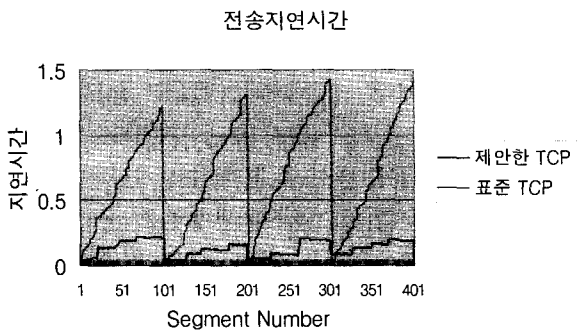

그림 5. 제안하는 TCP와 표준 TCP의 전송지연시간 비교
Fig. 5. Comparison of Propagation Delay for standard TCP and the proposed TCP.

표 9. 제안하는 TCP와 표준 TCP의 생산성 비교
Table 9. Throughput comparison for the proposed TCP and standard TCP.

모형 / 생산성	제안하는 TCP	표준 TCP
전송된 세그먼트의 수	1069	201

그림 5는 제안된 방법을 채용한 TCP에서 세그먼트 전송지연시간이 표준 TCP의 전송지연시간보다 훨씬 개선됨을 보인다. 이것으로 미루어 제안된 방법이 세그먼트 손실의 원인을 어느 정도 정확히 판단함을 알 수 있다. 제안한 방법이 혼잡에 의한 세그먼트 손실을 통신 오류에 의한 손실로 잘못 판단하였다면, 송신 TCP는 혼잡윈도우를 축소하지 않았을 것이고, 결과적으로 통신 혼잡이 더욱 가중하게 됨에 따라 후속 세그먼트들이 계속 폐기되어 여러 차례 재전송 됨에 따라 전송지연시간이 매우 커지게 되었을 것이다. 그림 5는 제안하는 TCP가 그렇게 하지 않음을 잘 보여준다.

표 9는 제안하는 방법과 표준 TCP의 생산성을 비교한다. 동일한 세그먼트 생성 모듈을 사용하고 시뮬레이션 시간을 12억 단위시간으로 고정하여 성공적으로 전송된 세그먼트의 수를 카운트한 결과, 표준 TCP는 201 세그먼트, 제안하는 TCP는 1,069 세그먼트를 각각 전송한 것을 확인하였다. 이와 같은 생산성의 비교는 제안하는 TCP가 세그먼트 손실의 원인을 어느 정도 정확히 판단한다는 또 다른 증거가 된다. 만일 제안하는 TCP가 세그먼트 손실의 원인을 잘못 파악하였다면 필요 이상으로 혼잡윈도우를 반감하여 생산성이 떨어져야 하는데 실험 결과는 그렇지 않다.

VI. 결 론

본 논문은 MANET에 사용되는 TCP의 성능을 제고하기 위한 방안을 제안하였다. 제안한 방법은, MANET의 특성을 고려하여, 송신 TCP와 수신 TCP만 약간의 작업을 하고, 세그먼트 손실이 발생했을 때에만 송신 TCP가 간단한 계산을 수행하기 때문에 기존의 방법에 비하여 에너지 소모를 적게 하는 장점이 있다.

또한, 제안한 방법은 송신 TCP에서 수신 TCP로 전송되는 패킷 전송 지연 시간의 증가를 혼잡의 징후로 간주함으로써 왕복시간을 고려한 기존의 방법보다 정확도를 개선하였다. 나아가서, 전송지연시간들 간의 차이가 증가하는 것까지 고려함으로써 정확도를 더욱 증가시킴으로써, 제안한 방법이 표준 TCP보다 약 5배 생산성을 향상시키는 것을 보였다.

이동통신에서는 단말기의 이동성을 고려한 무선 통신의 효율성 개선 문제가 심각하다. 향후에는 TCP 혼잡제어 방법과 더불어 이동성을 고려한 통신 효율성 개선 방법에 대하여 연구하고자 한다.

감 사 말 씀

바쁘신 가운데 귀한 심사의견을 제시하여 논문의 질을 제고하여 주신 무명의 심사자들에게 감사의 말씀을 전한다.

참 고 문 헌

[1] D.Barman and I.Matta, "How well can TCP infer network state?" Technical Report, Computer Science Dept. Boston University, BUCS-TR-2003-011, May 16, 2003.

[2] B. Zhan and M. N. Shirazi, "Implementation of explicit wireless loss notification using MAC-layer information," IEEE Wireless Communica-tions and Networking, Vol. 2, pp. 1339-1343, March 2003.

[3] D. Barman and I. Matta, "Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks," Proceedings of the 10th IEEE International Conference on Network Protocols, pp. 2-11, Nov. 12-15, 2002.

[4] O. Shagdar, M. N. Shirazi and B. Zhang, "Improving ECN-based TCP performance over wireless networks using a homogeneous implementation of EWLN", Proceedings of the 10th International Conference on Telecommuni-cations, Vol. 1, pp. 812-817, 2003.

[5] R Chawla, and S. Nandi, "TCP FECN: a unified solution for wireless networks" Proceedings of The 8th International Conference on Communica-tion Systems, Vol. 2, pp. 815-819, 2002.

[6] Z. Fu, B. Greenstein, X. Meng and S. Wu, "Design and Implementation of a TCP-friendly Transport Protocol for Ad hoc Wireless Networks", Proceedings of The 10th IEEE International Conference on Network Protocols (ICNP'02), pp. 216-225, Nov. 2002.

[7] The Network Simulator ns-2 [Online]. http://www.isi.edu/nsnam/ns.

[8] T. Murata, "Petri nets: Properties, analysis and applications," Proceedings of the IEEE, Vol. 77. no. 4, pp. 541-580, April 1989.

[9] T. Murata, T. Suzuki and S. Shatz, "Fuzzy-Timing High-Level Petri Nets (FTHNs) for Time-Critical Systems," in J. Cardoso and H. Camargo (editors) "Fuzziness in Petri Nets" Vol. 22 in the series "Studies in Fuzziness and Soft Computing" by Springer Verlag, New York, pp. 88-114, 1999.

[10] Y. Zhou, T. Murata, and T. DeFanti, "Modeling and performance analysis using extended fuzzy-timing Petri nets for networked virtual environments," IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol.30, No.5, pp. 737-755, October 2000.

[11] K. Jensen, Design/CPN [Online]. Dept. Computer Science, Univ. Aarhus, Denmark. Available: http://www.daimi.au.dk/designCPN/.

저 자 소 개

임 재 걸(평생회원)
1974년 인천교육대학 졸업
1981년 동국대학교 졸업
1987년 일리노이대 시카고 캠퍼스 석사
1990년 일리노이대 시카고 캠퍼스 박사
1992년~현재 동국대학교 컴퓨터학과 교수
<주관심분야 : 페트리 넷 이론과 응용, 컴퓨터 네트워크, 시스템 설계 분석, 인공지능>