ECN 마킹을 위한 최적의 Threshold

이 계 영[†] · 임 재 걸[††] · 장 익 현[†††]

요 약

ECN 방법은 통신 혼잡 발생 초기에 이 사실을 명백히 알려주기 때문에 통신 혼잡을 판단하는 정확한 방법이다. 그래서, 무선 TCP 분야에서는 ECN 방법이 심도 있게 연구되고 있다. 본 논문은 ECN 마킹을 위한 최적의 임계치(threshold)를 찾는 계산식을 제시하고, ECN 전략을 적용하는 TCP의 페트리 넷 모형을 구축한 다음, 시뮬레이션을 통하여 제시된 식의 타당성을 검증한다. 또한, 제시된 식의 실용화 방안도 제안한다.

본 논문의 주된 공헌은 ECN 마킹을 위한 최적의 임계치를 찾는 공식을 제공하는 것이지만, 본 논문에 소개된 ECN 전략을 적용한 TCP의 페트리 넷 모형도 네트워크 프로토콜을 연구하는 학자에게 도움을 주리라고 믿는다. 소개된 페트리 넷 모형은 기존의 TCP 모형에 ECN 전략을 추가하기 위하여 네트워크 부분을 주로 변형하였으며, 송신자와 수신자도 역시 변형되었다.

키워드 : TCP, ECN, 페트리 넷, 혼잡 윈도우, 무선통신

The Optimal Threshold for ECN Marking

Gyeyoung Lee[†] · Jaegeol Yim[††] · Ikhyeon Jang[†††]

ABSTRACT

ECN is accurate in determining traffic congestion since it explicitly notifies the incipient congestion. Therefore, ECN method has been thoroughly studied in the field of wireless TCP. This paper introduces a formula to find the optimal threshold for ECN marking. We have implemented a Petri net model of 'TCP with ECN strategy' and performed simulations on it in order to verify the validity of the formula. We have also introduced ideas of applying the formula in practice.

The primary contribution of this paper is proposing a formula to find the optimal threshold for ECN marking. However, introducing the Petri net model of 'TCP with ECN strategy' is no less valuable contribution because it can be helpfully used by the researchers in studying network protocols. We have built the Petri net model by modifying the existing Petri net model of TCP. In order to add ECN strategy to the existing model, we have mainly modified the network part. We have also modified sender part and receiver part as well.

Key Words : TCP, ECN(Explicit Congestion Notification), Petri Net, Congestion Window, Wireless Communication

1. 서 론

TCP는 프로세스간의 신뢰성 높은 연결을 구축해 주는 통신 규약[1]으로, 2003년 현재 인터넷 통신량의 90%가 TCP 정보이다[2]. TCP 규약은 연결부와 통신부로 구성되어 있다. 연결부는 연결을 설정하기 위하여 거쳐야 할 과정과 연결을 종료하기 위하여 거쳐야 할 과정으로 구성되며 비교적 간단하기 때문에 TCP의 성능 연구는 보통 통신부에 초점을 둔다.

TCP의 데이터 전송 단위는 세그먼트이므로, 송신 TCP는 각 세그먼트에 일련번호를 부착하여 송신한다. 수신 TCP는 순서대로 도착한 세그먼트에 대하여 ACK 세그먼트를 답신으로 보낸다. 송신 TCP는 접수확인이 도착해야 다음 번 세그먼트를 송신하며, 일정 시간이 지나도록 접수확인이 도착하지 않으면 그 세그먼트부터 다시 전송하게 된다. 이와 같은 이유로 TCP를 신뢰성이 높은 통신규약이라고 한다.

접수확인이 도착하기 전에 송신 TCP가 연속적으로 전송할 수 있는 세그먼트의 수를 혼잡 윈도우(wc : Congestion Window)라고 한다. 혼잡 윈도우가 x 이면 한꺼번에 x개의 세그먼트를 전송하기 때문에, 한 세그먼트만 전송하고 접수확인을 한 다음 또 한 세그먼트만 전송하는 것보다 x배 효율적일 수 있다. 그러나, 통신혼잡이 발생할 경우에는 경유하는 라우터에서 버퍼 넘침 현상이 발생할 수 있고, 버퍼 넘침이 발생하면 라우터는 나중에 도착하는 세그먼트를 그냥 버리게 된다. 따라서, 통신 상태를 고려하지 않고 혼잡 윈도우를 너무 크게 하게 되면 통신 효율을 감소시키는 결과를 낳게 된다.

혼잡 윈도우는 TCP의 통신 효율성을 결정하는 가장 중요한 요인이다. TCP는 다음과 같은 AIMD(Additive Increase Multiplicative Decrease) 방식을 통해 통신량을 조절한다. 혼

† 종신회원 : 동국대학교 공학대학 컴퓨터멀티미디어학과 교수
†† 종신회원 : 동국대학교 공학대학 컴퓨터멀티미디어학과 교수
††† 종신회원 : 동국대학교 공학대학 정보통신공학과 부교수
논문접수 : 2004년 10월 19일, 심사완료 : 2005년 6월 7일

잡 윈도우의 초기치는 1로 하고, 접수확인을 할 때마다 혼잡 윈도우의 크기를 1만큼 증가시킨다. 송신 TCP는 전송한 세그먼트에 대한 타임아웃이 발생하면 통신 혼잡으로 인한 손실이 발생하였다고 가정하여 그 세그먼트부터 다시 전송할 뿐 아니라, 통신량을 감소시키기 위하여 혼잡윈도우를 현재의 반으로 변경함으로써 전송 빈도수를 현저히 낮춘다. 이와 같이 타임아웃이 발생할 때마다 통신 혼잡으로 인한 손실이 그 원인이라고 가정하는 것은 네트워크 층 이하가 신뢰성이 높은 유선통신에서만 합당하다. 따라서, TCP는 효율성 면에서 통신 오류가 빈번한 무선통신에 적합하지 않다는 것을 알 수 있다.

현재 인터넷 통신량의 90%가 TCP 정보인 만큼, 사용중인 소프트웨어의 대부분이 TCP 기반으로 되어 있으며, 무선 인터넷, 무선 LAN이 보편화 되어감에 따라 무선통신 환경에서 TCP의 효율성을 개선하기 위한 연구가 활발히 진행되고 있다. 이러한 연구의 초점은 세그먼트 손실의 원인을 정확히 판단하여 통신 혼잡에 의한 손실일 경우에만 th=wc/2, wc=1을 수행하고, 통신오류에 의한 손실이면 wc와 th 값의 변경없이 재전송만 하는 것이다.

세그먼트 손실의 원인을 밝혀 혼잡을 회피하는 방법 중에는 ECN (Explicit Congestion Notification), EWLN (Explicit Wireless Loss Notification), DECbit, RED(Random Early Detection) 방식 등이 있다. ECN[3] 방식은 라우터나 게이트웨이에서 버퍼가 어느 정도 차서 폐기가 예상되면 세그먼트에 ECN 마킹을 하여줌으로써 송신 TCP에게 통신혼잡을 명백히 알린다. 이 방식을 적용하기 위하여 풀어야 할 문제는 '과연 버퍼가 어느 정도 찼을 때 ECN 마킹을 하는 것이 바람직한가' 하는 것이다.

본 논문에서는 무선통신 환경에서 TCP의 성능 향상을 위하여, ECN 마킹을 하는 시점을 찾는 계산식을 제시하고, 시뮬레이션을 통하여 제시된 식의 타당성을 검증하고 이의 적용방법을 제시한다. 시뮬레이션을 위하여 ECN 방식을 채용한 페트리 넷 모형을 구축하였다. 페트리 넷 모형은 통신 네트워크를 시뮬레이션하는 도구로 일반적으로 사용되는 ns-2 [4]보다 상세하기 때문에 본 논문이 제공하는 페트리 넷 모형은 ECN 관련 연구에 많은 도움을 줄 수 있을 것으로 본다.

본 논문의 구성은 다음과 같다. 2장에서 ECN과 관련된 연구를 살펴보고 3장에서 ECN 방식을 적용한 TCP의 페트리넷 모형을 구축하였다. 4장에서는 혼잡회피를 위한 최적의 ECN 마킹 위치를 찾는 식을 제시하였으며, 5장과 6장에서 모의실험을 통해 제시한 식의 타당성을 검증하고 이의 적용방법을 제시하였으며, 7장에서 결론을 맺는다.

2. 관련 연구

TCP의 혼잡제어 방식에는 SQM(Source Quench Message), ECN, DECbit, RED 등의 많은 방법이 제시되고 있다.

SQM은 라우터에 도착하는 메시지가 라우터가 처리할 수 있는 메시지보다 더 빈번하여 패킷이 폐기될 때 라우터가 생성하여 보내는 메시지로서, RFC 1009는 메시지 도착 빈도가 너무 높을 경우, 라우터는 반드시 SQM을 생성할 것을 요구하고 있다[5]. 버퍼 크기, 가상 경로의 대역폭, 통과하는 가상 경로의 수, 패킷의 평균 크기, 통신 지연시간 등을 기반으로 SQM을 생성하는 방법이 [6]에 소개되었다. ECN과 SQM을 혼합한 혼잡 회피 방법은 [7]에 소개되었으며, 이 방법은 TCP뿐 아니라 UDP 통신 조절에도 사용된다.

[8]은 SQM과 RED 방법을 혼합하여 사용하는 것이 실용적이고 효율적인 혼잡 회피 방법임을 보였으며, TCP를 위한 빠른 후방 SQM 방법이 [9]에 소개되었다. 이 방법에서는 송신 TCP와 수신 TCP의 경로상에 있는 라우터가 생성하는 ICMP SQM을 참조하여 수신 TCP가 접수확인 메시지를 발생하는 속도를 조절함으로써 통신혼잡을 피한다.

DECbit 방법에서는 네트워크 혼잡을 송신 TCP에게 알리기 위하여 게이트웨이가 패킷 헤더의 혼잡알림 비트를 사용한다[10, 11]. 통신 혼잡이 낮을 때에는 통신량이 증가함에 따라 처리량도 증가하지만, 통신 혼잡이 어느 경계선 이상 올라가게 되면 통신량을 증가시켜도 처리량은 증가하지 않는다. 이 경계선을 무릎이라 하며, 통신 혼잡이 무릎에 도달할 때 DECbit를 1로 표시한다.

RED 방법은 최소한계치와 최대한계치를 두어, 버퍼의 평균 사용량이 최소한계치 미만일 경우에는 패킷을 그대로 통과시키고, 최대한계치를 넘으면 패킷을 버리며, 최소한계치와 최대한계치 사이일 경우에는 최소한계치를 0으로 시작하고 최대한계치는 최대확률 값이 되도록 선형확률값을 적용하여 패킷을 버린다[12].

ECN 방법은 통신 혼잡 상황을 TCP 통신 경로상의 라우터나 게이트웨이가 세그먼트나 패킷에 표시하는 것으로 RED와 같은 원리로 혼잡을 예측하지만 해당 패킷을 버리는 대신 마킹을 하여 발신지에서 트래픽을 조절하도록 한다[21]. ECN과 관련된 연구로는 AIMD 방식의 증가분과 감소분을 얼마로 하는 것이 좋은지에 대한 연구[18-21]와 혼잡발생이 예상될 때 마킹을 하는 위치에 관한 연구[22] 등이 있다. ECN과 RED는 혼잡을 예측하는 방법이 유사하므로 같은 연구로 볼 수도 있다. 본 논문에서는 다른 연구와는 달리 라우터 버퍼에 존재하는 패킷의 수와 연관시켜서 혼잡 발생을 예측하도록 한다.

페트리 넷 모형은 처리량과 전파지연시간뿐 아니라 세그먼트 폐기 발생 수, 통신 오류 발생 수, 통신 혼잡으로 판단한 경우의 수 등 자세한 통계치를 시뮬레이션을 통하여 구할 수 있는 장점을 제공한다. TCP의 페트리 넷 모형은 [13-15]에 소개된 바 있는데, [13]은 확장 퍼지시간 페트리 넷으로 TCP 네트워크로 연결된 가상현실 시스템의 반응 시간을 평가한 사례를 소개하였고, [14]는 결정 통계 페트리 넷으로 짧은 TCP 연결의 완료시간을 평가하는 방법을 소개하며, [15]는 무한 통계 페트리 넷을 이용하여 WWW용 TCP 윈도우 방법 평가 사례를 소개하였다.

3. TCP의 페트리 넷 모형

3.1 TCP 요약

본 논문에서 구축하는 TCP 시뮬레이션 모델에 대한 설명을 위하여, 서론에 이어서 TCP에 대한 설명을 계속한다. 타임아웃 시간은 보통 RTT+4*D로 한다. RTT는 송신 TCP를 출발한 세그먼트가 수신 TCP에 도착하고, 그에 대한 수신확인이 다시 송신 TCP에 도착하는 데 걸리는 시간이고, D는 RTT의 표준편차이다. RTT는 시간에 따라 동적으로 변화하지만 본 논문에서는 평균전송 시간의 두 배로 하였다.

한편 수신자도 수신 TCP 윈도우를 가지고 있다. 본 논문에서는 수신 TCP의 초기치를 64로 하였고, 세그먼트를 수신할 때마다 1씩 감소하며, 세그먼트를 상위계층으로 전달할 때마다 1씩 증가시킨다. 수신 TCP의 ACK 메시지는 수신 세그먼트의 일련번호(an), 다음에 수신을 원하는 세그먼트의 일련번호, 그리고 수신자의 가용 윈도우(wr: Advertised Window)의 크기로 구성된다.

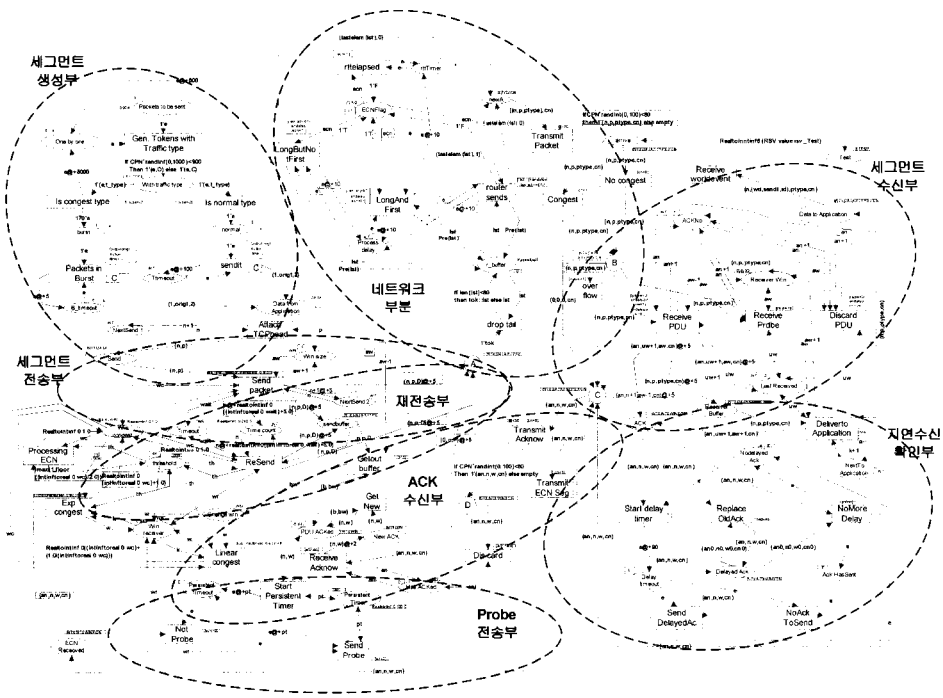
송신 TCP의 혼잡 윈도우(wc)는 송신 TCP가 전송할 수 있는 ACK 메시지를 받지 않은 세그먼트의 수이다. 수신자의 가용 윈도우(wr)는 수신자가 ACK 메시지에 붙여서 송신자 TCP에게 알려준다. 송신TCP는 송신되었지만 아직 수신확인이 되지 않은 세그먼트의 수(aw)를 기록한다. 결국 송신 TCP는 'aw < min(wr, wc)'일 때 새로운 세그먼트를 전송할 수 있다.

수신 TCP는 통신량을 경감시키기 위하여 지연 수신확인 정책을 채택한다. 즉, 새로운 세그먼트 i를 수신하자마자 i에 대한 수신확인을 송신TCP로 전송하는 것이 아니라, 일정 시간 동안 수신확인을 지연한다. 지연하는 동안에 i+1 세그먼트가 도착하면 i에 대한 수신확인은 취소하고 i+1에 대한 수신확인만 송신한다.

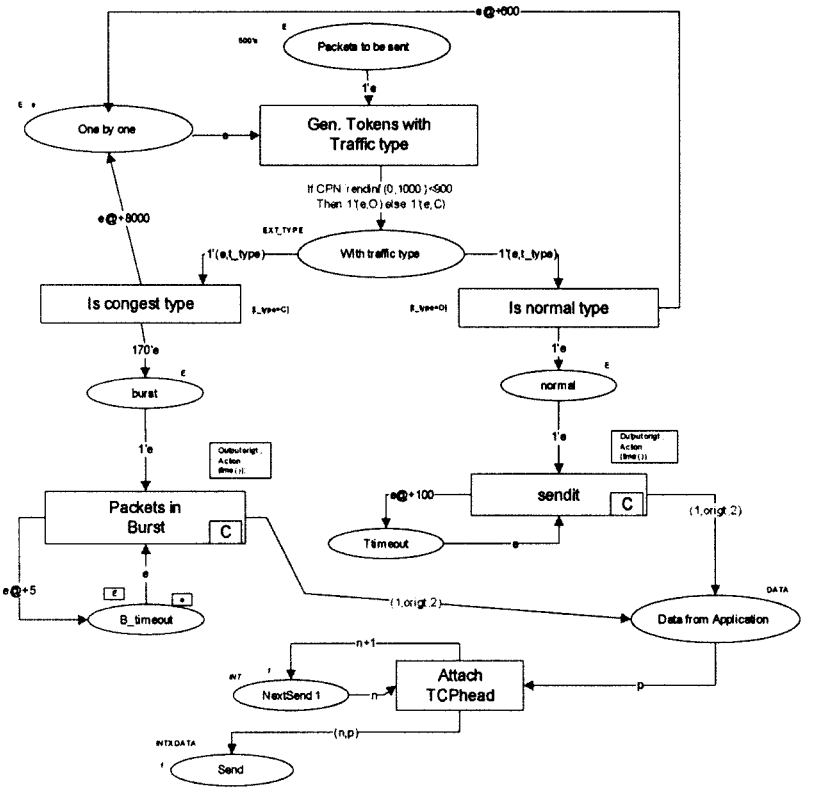
수신 TCP는 순서에 맞지 않는 세그먼트가 도착할 때, 그것을 버리고 접수를 기다리는 세그먼트의 번호를 즉각 송신 TCP에게 알려준다. 또한, 수신 TCP는 wr이 0이 되면 이를 송신 TCP에게 알린다. 그러면 송신 TCP는 wr이 0이므로 더 이상 세그먼트를 전송하지 못하게 된다. 추후에 수신 TCP가 세그먼트를 응용계층에 전달함에 따라 wr의 값이 양수가 되면, 수신 TCP는 새로운 wr 값을 송신 TCP에게 송신한다. 그런데, 전송 오류로 말미암아 이 세그먼트가 손실될 수 있다. 그러면 송신 TCP는 wr이 아직 0인줄 알고 전송을 멈추고, 수신 TCP는 새 값을 알렸기 때문에 새 세그먼트가 올 것으로 예상하고 기다리는 현상이 발생하게 된다. 이러한 현상을 방지하기 위하여 수신 TCP는 wr=0이라는 메시지를 접수하면 일정 시간 동안 새로운 wr 값이 도착하기를 기다렸다가, 그래도 wr=0이면 아직도 wr 값이 정말로 0인지 묻는 탐침메시지를 전송한다. 탐침메시지를 접수하면, 수신 TCP는 현재 wr 값을 즉각 송신TCP에게 전송한다.

3.2 페트리 넷 모형

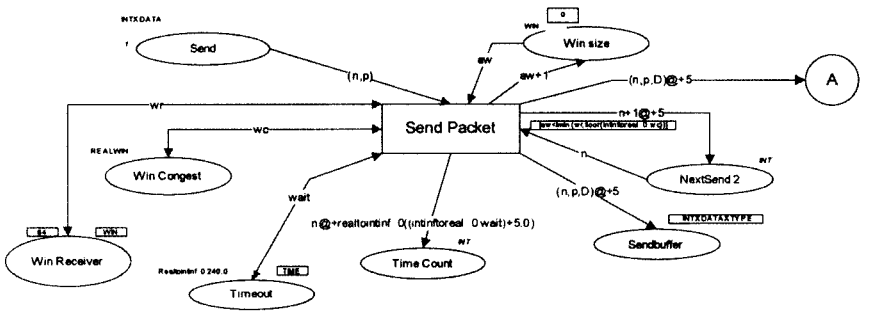
본 논문이 소개하는 TCP의 모형은 페트리 넷의 일종인 퍼지시간 고급 페트리 넷으로 (그림 1)과 같으며, Design/CPN을 이용하여 본 모형을 구축하였다. 페트리 넷 관련 용어는 [16], Design/CPN 관련 용어는 [17]을 참조하기 바란다. 모형에 나타낸 수치는 5절 실험에 사용된 수치를 반영한 것이다. (그림 1)은 송신 TCP, 네트워크, 그리고 수신 TCP로 구성된다. 송신 TCP는 세그먼트 생성부(그림 2), 세그먼트 전송부



(그림 1) ECN 전략을 사용하는 TCP의 페트리 넷 모형



(그림 2) 세그먼트 생성부

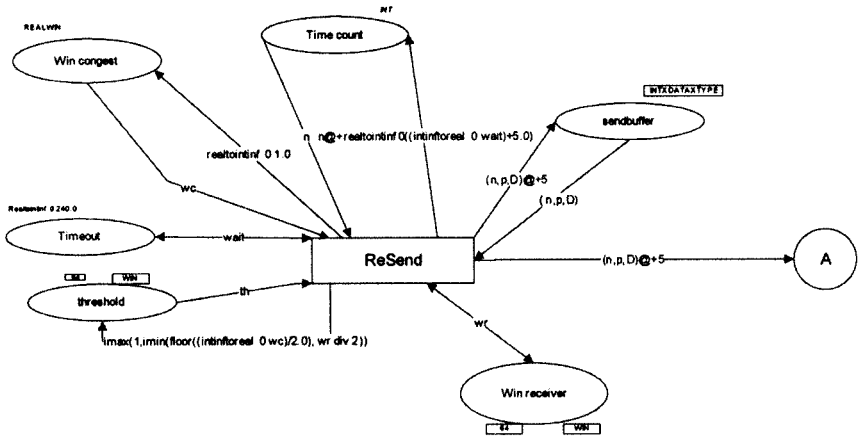


(그림 3) 세그먼트 전송부

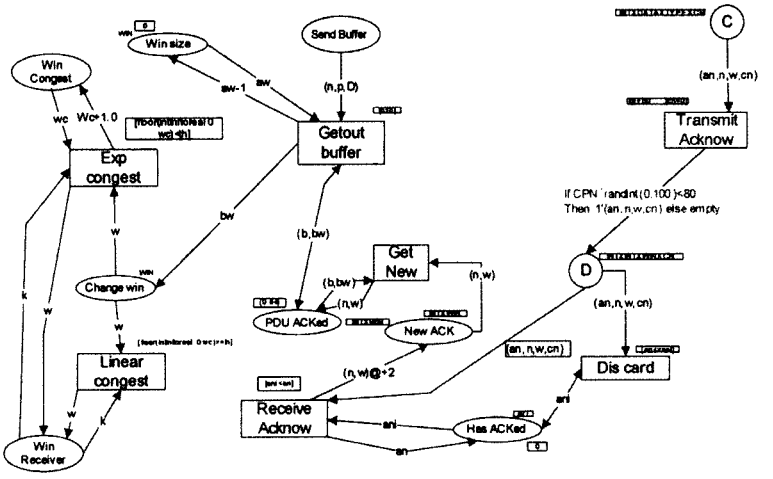
(그림 3), 재전송부(그림 4), ACK 수신부 (그림 5), 탐침전송부 (그림 6)로 구성된다. 네트워크 부분은 (그림 7)과 같다. 수신 TCP는 세그먼트 수신부(그림 8)와 지연 수신확인부(그림 9)로 구성된다.

(그림 2)에 보이는 세그먼트 생성부는 송신 TCP의 세그먼트 생성부분의 모델이다. 플레이스(Place) 'Packets to be sent'와 'One by one'에서 각각 하나의 토큰을 받아 트랜지션 'Gen. Tokens with Traffic type'은 트래픽 유형이 O(Ordinary)인 토큰이거나 C(Congestion) 인 토큰을 생성한다. 이때 C유형의 빈도는 10%이고 나머지 90%는 O유형이다. 트래픽 유형이 C인 토큰 하나는 'Is congest type' 트랜지션을 격발하면서 170 개의 토큰을 플레이스 'burst'에 놓는다. 동시에 타임스탬프가 +8000인 토큰을 플레이스 'One by one'에 놓는다. 어떤 토큰의 타임스탬프가 +8000이라는 것은 시뮬레이션 시간이 8000 단위시간 (Os) 경과하여야 그 토큰의 사용이 가능하게 된다는 의미가 있다. 한편 'burst'에 놓인 170 개의 토큰은 트랜지션 'Packets in Burst'의 입력으로 사용되어 170개의 세그먼트를 5 단위시간마다 하나씩 플레이스 'Data from Application'에 놓는 역할을 한다. 플레이스 'NextSend1'은 세그먼트에 붙여질 일련번호를 생성하기 위하여 존재한다.

(그림 3)은 세그먼트 전송부로 트랜지션 'Send Packet'이 중심이며, (그림 2)와 플레이스 'Send'를 공유한다. 'Send Packet'에는 [aw<min(wr, floor(IntInftoreal 0 wc))]라는 guard(실행조건을 나타내는 Design CPN 용어임)가 붙어 있다. aw, wr, wc는 각각 2.1절에 소개된 '송신되었지만 아직 수신확인이 되지 않은 세그먼트의 수(aw)', '수신자의 가용 윈도우(wr)', '송신 TCP의 혼잡 윈도우(wc)'를 의미한다. 플레이스 'NextSend2'는 'Send Packet'이 세그먼트를 순서대로 내 보내는 것을 확실히



(그림 4) 재전송부

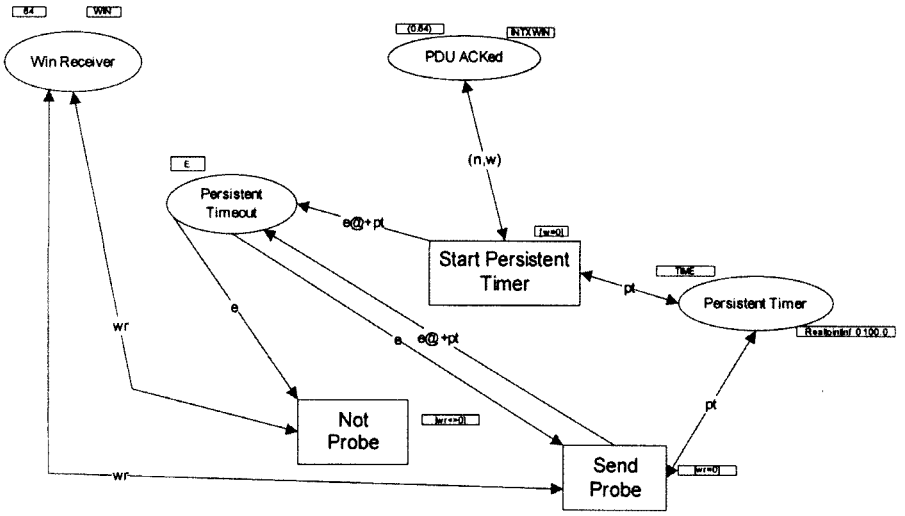


(그림 5) ACK 수신부

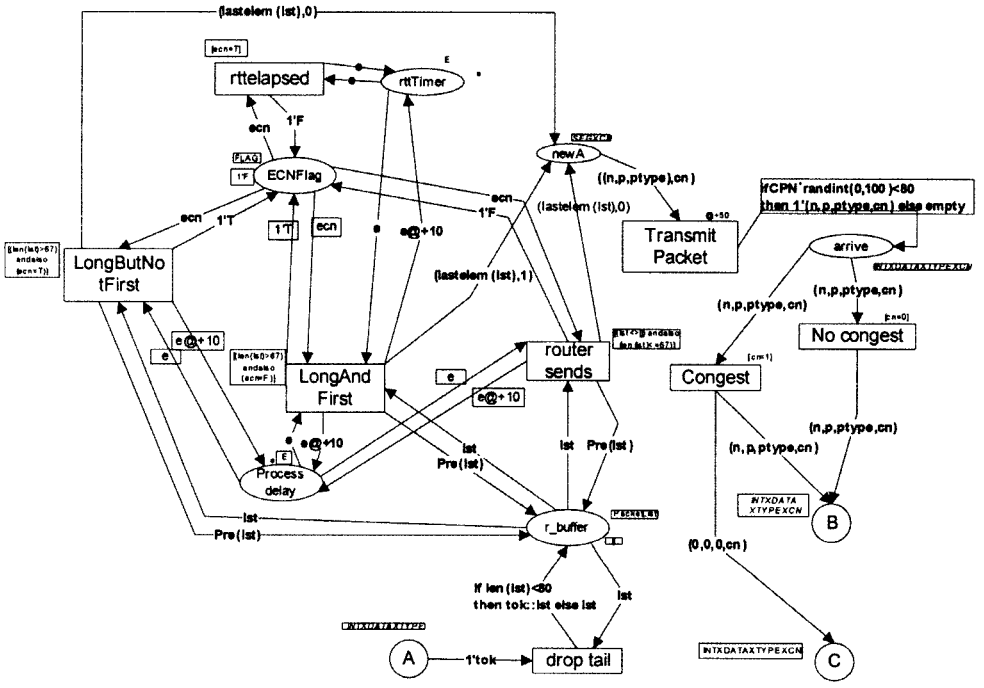
보증하는 역할을 수행한다. 송신된 세그먼트는 플레이스 'Send-buffer'에 보관되어, 타임아웃이 발생하여 재전송 되든지, 아니면 수신확인이 접수되어 버려지게 된다. 플레이스 'Timeout'에 있는 토큰은 240이라는 수다. 이 값이 'wait' 변수에 할당되어 'Send Packet'이 격발되는데 사용되고, 다시 플레이스 'Timeout'로 되돌아온다. 값이 240인 'wait'는 플레이스 'Time Count'로 들어가는 토큰 n의 타임스탬프를 계산하는데 사용된다. 즉, 플레이스 'Time Count'에는 전송된 세그먼트의 일련번호 n이 들어오게 되는데 이 n의 타임스탬프는 240+5.0 = 245이다. 타임스탬프는 앞에서 언급한 바와 같이 해당 토큰을 타임스탬프 시간 동안 무력하게 만드는 역할을 한다.

(그림 4)의 재전송부는 트랜지션 'ReSend'가 중심이며, (그림 3)과 플레이스 'sendbuffer', 'Time count', 'Timeout', 'Win congest', 'Win receiver'를 공유한다. 'ReSend'는 타임아웃이 발생하면 플레이스 'sendbuffer'에 보관된 이미 전송되었지만 아직 접수확인이 도착하지 않은 세그먼트 n을 재전송한다. 세그먼트 번호 n은 플레이스 'Time count'에 보관된다. 재전송된 세그먼트일지라도 타임아웃이 발생하면 또다시 재전송이 가능하기 때문에 플레이스 'Time count'로부터 들어 온 토큰 n의 타임스탬프를 'wait' ((그림 3) 참조) 시간 증가시켜서 다시 'Time count'로 되돌려 보낸다. 마찬가지 이유로 재전송된 세그먼트, (n, p, D)를 플레이스 'sendbuffer'에 보관한다. 타임아웃이 발생하면 'wc'를 1로 고친다. 혼잡회피 단계에서는 수신확인을 접수하면 wc=wc+(1/wc)을 한다. 따라서 wc의 데이터 유형은 실수라야 한다. 그런데 Design/CPN 현재 버전에서는 실수 유형을 사용하지 않기 때문에 플레이스 'Win congest'의 데이터 유형을 IntInf 유형으로 정의하였고, 실수 1.0을 사용하기 위하여 realtointinf라는 형변환 함수를 사용하였다. 재전송이 발생하면 th 값도 wc의 반이나 wr의 반 중 작은 값으로 변경한다. 단, 1보다 작지 않도록 한다.

(그림 5)에 보이는 ACK 수신부는 트랜지션 'Receive Acknow', 'Discard', 'Get New', 'Getoutbuffer' 등으로 구성되며, (그림 3)과 플레이스 'Win size', 'SendBuffer'를 공유하고, (그림 4)와 플레이스 'SendBuffer', 'Win Congest', 'Win Receiver'를 공유한다. 'Receive Acknow'는 수신확인 메시지에 기록된 an 번호가 이미 접수한 번호보다 큰 것(ani<an)만 접수함으



(그림 6) 탐침전송부



(그림 7) 네트워크 부분

로써 중복 접수를 피한다. 'Receive Acknow'의 격발은 (n, w)를 플레이스 'New Ack'에 놓는데 여기에서 n은 수신TCP가 수신하기를 원하는 세그먼트의 일련번호이고, w는 수신TCP의 가용 윈도우의 크기이다. 트랜지션 'Get New'는 이 (n,w)를 플레이스 'PDU Acked'에 놓는 역할을 하고, 트랜지션 'Getout buffer'는 플레이스 'SendBuffer'에 보관중인 일련번호가 n (여기에서는 b로 표현됨)보다 작은 세그먼트들을 모두 꺼내 버리고, 수신 TCP의 윈도우 크기(여기에서는 bw임)를 플레이스 'Change win'에 놓고, 송신되었지만 아직 수신확인이 되지 않은 세그먼트의 수(aw)를 1 감소시킨다. 페트리 넷에서 변수의 영역은 트랜지션이기 때문에 변수 이름을 반드시 달리 써야 하는 경우가 발생한다. 예를 들어 'Getout buffer'의 입출력 간선에 사용된 변수 중, b와 n은 모두 세그먼트 넘버지만 값이 다를 수 있기 때문에 상이한 변수 이름을 사용해야 한다. 비슷한 이유로 수신자의 가용 윈도우를 나타내는 변수로 wr, bw 혹은 w가 쓰인다.

(그림 5)의 트랜지션 'Exp congest'는 느린 시작 모드에서 'wc'가 지수적으로 증가하는 것을 나타낸다. 즉, wc<th 일 때 플레이스 'Change win'에 토큰이 도착하면 wc의 값을 1 증가시키는 일을 한다. 한편 트랜지션 'Linear congest'는 혼잡회피 모드에서 'wc'가 선형적으로 증가하는 것을 나타낸다. 즉, wc>=th 일 때 플레이스 'Change win'에 토큰이 도착하면

wc를 1/wc 만큼 증가시키는 일을 한다.

(그림 6)은 탐침전송부이며, (그림 5)와 플레이스 'Win Receiver', 'PDU ACKed'를 공유한다. 수신TCP의 가용 윈도우 크기 w가 0이면 트랜지션 'Start Persistent Timer'가 격발하여 플레이스 'Persistent Timeout'에 타임스탬프가 pt 만큼 증가한 토큰 e를 놓는다. pt는 플레이스 'Persistent Timer'에서 오는 토큰으로 본 논문에서는 그 값이 100으로 설정되었다. 따라서 트랜지션 'Start Persistent Timer'가 격발한지 100 단위시간 후에 트랜지션 'Not Probe' 혹은 'Send Probe'가 격발하게 된다. 트랜지션 'Not Probe'는 wr이 0이 아닐 때 탐침 세그먼트를 그냥 버리는 역할을 하고, 'Send Probe'는 wr이 아직 0일 때 탐침 세그먼트를 전송하는 역할을 수행한다.

(그림 7)은 네트워크 부분이며, (그림 3)과 플레이스 'A'를 공유한다. 플레이스 'r_buffer'는 라우터 버퍼를 나타내며, 초기에는 이 버퍼에 아무 세그먼트도 없다. 세그먼트가 도착하면 트랜지션 'drop tail'이 버퍼의 끝에 이 세그먼트를 첨가한다. 단, 버퍼에 이미 있는 세그먼트의 수가 80이 넘으면 새 세그먼트를 그냥 버린다. 트랜지션 'Transmit Packet'에는 @+50이라는 시간이 붙어있는데, 이것은 전송에 소요되는 시간을 나타낸다. 트랜지션 'Transmit Packet'과 플레이스 'arrive' 사이의 간선에는 0부터 100사이의 난수를 생성하여 생성된 난수가 80보다 작으면 플레이스 'arrive'에 세그먼트를 전달하고 그렇지 않으면 세그먼트를 버리는 조건문이 붙어 있다. 트랜지션 'LongAndFirst'는 버퍼의 길이가 일정 길이(67)보다 길 때, 세그먼트에 ecn 마킹을 한다. 트랜지션 'LongButNotFirst'는 버퍼의 길이가 BT보다 길더라도 이미 ecn 마킹이 된 세그먼트를 전송하였을 경우에는 세그먼트에 더 이상 ecn 마크를 붙이지 않는 일을 한다. 트랜지션 'rttelapsed'는 세그먼트에 ecn 마킹을 하고 RTT 시간이 경과하였음을 나타내는 트랜지션이다.

수신TCP의 세그먼트 수신부는 (그림 8)에 보이며, (그림 7)과 플레이스 'B', 'C'를 공유하고, (그림 5)와 플레이스 'C'를 공유한다. 세그먼트의 유형은 데이터(D)와 탐침(P)로 구분되는데 트랜지션 'Receive PDU'와 'Discard PDU'가 D유형을 수신하고 'Receive Probe'가 P 유형을 수신한다. 트랜지션 'Receive PDU'는 세그먼트의 일련번호(n)가 지난번에 수신한 일련번호(uw)와 n=uw+1이라는 관계식을 만족하는 세그먼트를 수신한다. 이렇게 성공적으로 수신된 세그먼트는 응용계층과 지연수신 확인부로 전달된다. 수신TCP는 수신한 모든 세그먼트의 수를 플레이스 'ACKNo'에 기록한다. 데이터 유형 세그먼트 중, n<=uw인 것은 'Discard PDU'가 접수하여 버리고, ACK를 즉시 전송하여 송신TCP에게 원하는 세그먼트 번호를 알린다. 'Receive Probe'는 P 유형의 세그먼트를 접수하여 송신TCP에게 수신자의 가용 윈도우(wr)의 크기를 즉시 알려준다.

지연수신 확인부는 (그림 9)에 보이며, (그림 8)과 플레이스 'ACK'를 공유하고, (그림 5)와 플레이스 'C'를 공유한다. 트랜지션 'Start delay timer'는 타임스탬프가 90 증가한 토큰을 플레이스 'Delay timeout'에 놓음으로써 수신확인 메시지가 90 단위시간 동안 플레이스 'Delayed Ack'에 머물도록 한다. 머무는 동안 '다음에 수신을 원하는 세그먼트의 일련번호'가 더 큰 새로운 수신확인 메시지가 도착하면 기다리던 메시지를 버리고 새로운 수신확인 메시지를 트랜지션 'NoMore Delay'를 격발하여 전송한다. 한편 'Delayed Ack'에서 기다리는 동안 지정된 시간이 경과하면 트랜지션 'Send DelayedAc'를 격발하여 기다리던 메시지를 송신한다.

4. 임계치

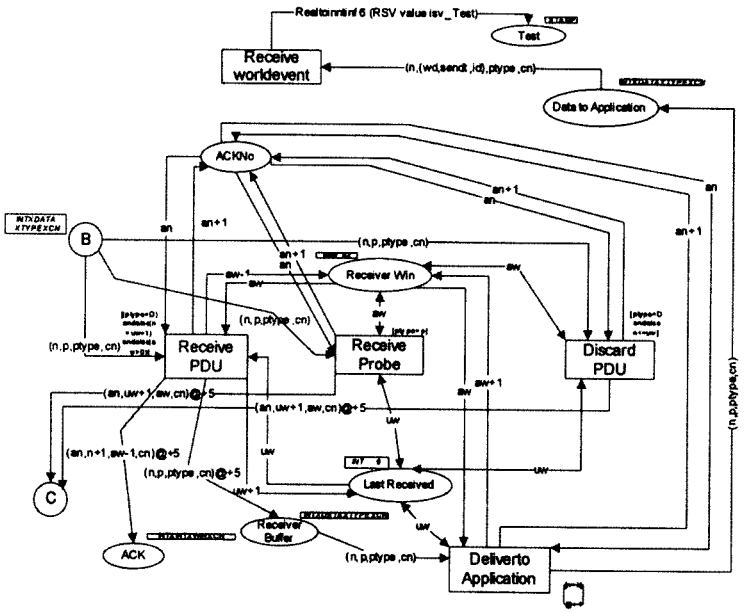
TCP 성능을 극대화시키는 방법으로 흐름제어와 혼잡제어가 있다. 흐름제어는 전송자가 너무 빈번하게 데이터를 전송하여 수신자의 버퍼에서 넘침이 발생하는 것을 방지하는 것으로 수신자가 사용 가능한 버퍼의 크기를 전송자에게 전달함으로써 비교적 쉽게 해결할 수 있는 문제다. 혼잡제어는 송신자와 수신자 간의 연결을 구성하는 라우터에서 버퍼 넘침이 발생하는 것을 방지하여 TCP의 효율성을 극대화하는 것을 목적으로 하는 문제이며 아직 풀리지 않은 대표적인 문제 중의 하나이다.

혼잡의 정의에 비추어 볼 때, 혼잡을 탐지하는 가장 적당한 장소는 라우터이다. 혼잡이 발생하는 라우터에서 혼잡을 탐지하여 혼잡을 회피하기 위한 많은 연구가 있다[6-12, 18-22]. 이들은 크게 DECbit, RED, ECN으로 분류될 수 있다. 그러나 이들 중 아직 완전한 혼잡제어 방식은 없다. DECbit 방식은 경계선인 무릎을 결정하는 식을 제공하지 못하며, RED와 ECN 방식은 최소한계치와 최대한계치를 결정하는 최적의 식을 제공하지 못한다.

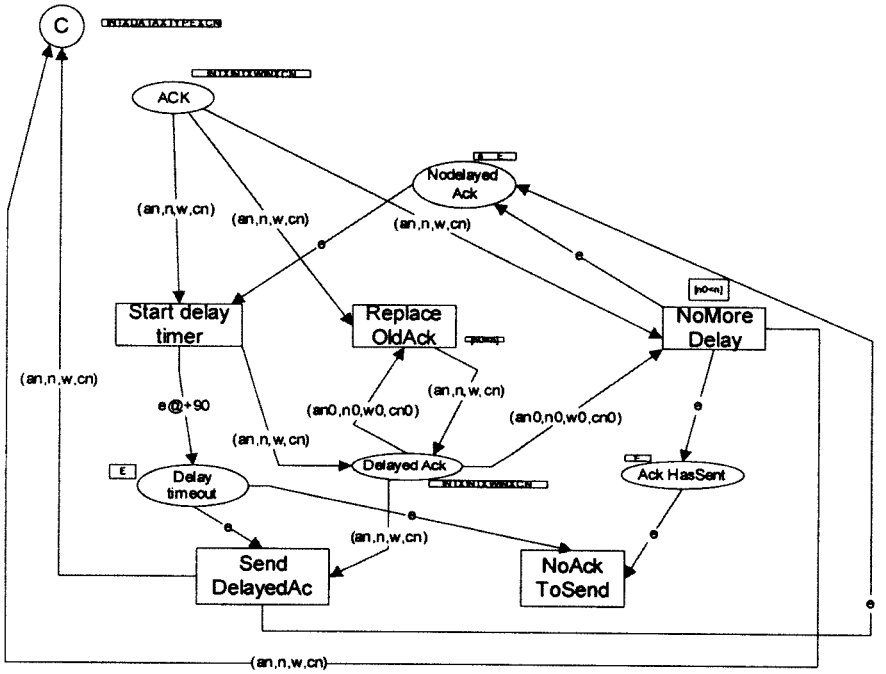
ECN 방식의 효율성은 혼잡 예측의 정확성과 시간성에 좌우된다. 시간성이라 함은 혼잡을 피하기 위한 조치를 취할 수 있는 충분한 시간 이전에 혼잡이 예측되어야 한다는 것이다. 라우터는 RTT 이전에 혼잡을 탐지해야 송신자에게 적시에 혼잡을 알릴 수 있다. 그 이유는 혼잡을 알리는 메시지가 수신자에게 전달되었다가 다시 송신자에게 전달될 것이며, 그 동안에도 송신자는 평상시와 같이 계속 세그먼트를 전송하기 때문이다.

제안하는 방법은 적절한 시간성을 확보하기 위하여 실제 라우터 버퍼의 크기와 라우터 버퍼에 있는 메시지의 수를 고려하여 ECN 마킹을 하도록 한다. 한편 이의 정확성은 시뮬레이션을 통해 검증하도록 한다.

라우터에서 혼잡이 발생하는 원인은 라우터를 떠나는 세그먼트의 빈도보다 라우터로 들어오는 세그먼트의 빈도가 더 높기 때문이다. 세그먼트 i가 라우터를 떠나는 시각에서 세그먼트 i-1이 라우터를 떠난 시각을 뺀 차를 TBL_i라 하고, 모든 세그먼트들에 대한 TBL_i의 평균을 TBL (Time Between Leave)이라 하자. 비슷한 방법으로 i번째 세그먼트가 라우터에 도착한 시각에서 i-1번째 세그먼트의 도착 시각을 뺀 차를 TBA_i라 하고 모든 세그먼트들에 대한 TBA_i의 평균을 TBA (Time Between Arrival)라 하자. 그러면 RTT 시간 동안 버퍼에 쌓이는 세그먼트의 수는 (RTT/TBA)-(RTT/TBL)이다.



(그림 8) 세그먼트 수신부



(그림 9) 지연수신 확인부

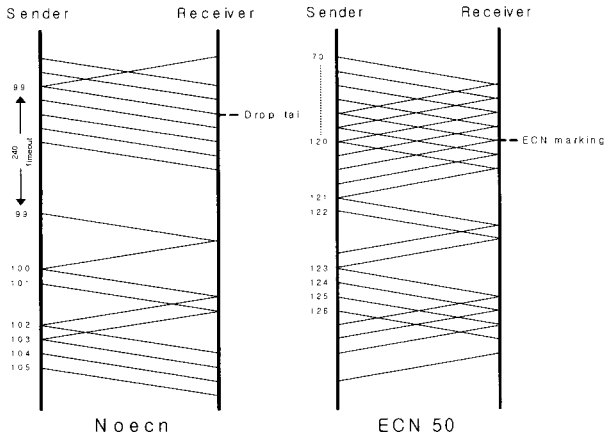
따라서 버퍼에 남은 공간이 이보다 적게 되면 혼잡 경고를 발생해야 한다. 즉, 라우터의 버퍼에 들어 있는 세그먼트의 수가 (식 1)에 제시된 임계치를 넘으면 ECN 마크를 부착하는 것이 가장 효율적이다. 이렇게 하면 충분한 시간성을 확보하고 혼잡을 예측하므로 네트워크에 혼잡이 발생할 확률은 그만큼 줄어든다고 할 수 있다. TBA와 TBL, RTT가 모두 동적으로 변하는 값이므로 임계치도 동적으로 변하게 되고 이는 기존 방법에서의 확률적으로 ECN 마킹을 하는 것보다 효율적이고 융통성이 높은 방법이다.

임계치 = 버퍼의 크기-((RTT/TBA)-(RTT/TBL)) (식 1)

(그림 10)은 ECN을 사용하지 않는 경우에 99번 세그먼트가 폐기되어 타임아웃이 발생한 후 99번 세그먼트부터 재전송을 하는 것과, ECN 방식에서 미리 혼잡 윈도우의 크기를 감소시킴으로써 재전송을 피하는 경우의 회선 사용을 보인다.

이와 같이 ECN 방식은 재전송을 피함으로써 회선 사용의 효율성을 높인다.

ECN 마크를 하는 임계치가 (식 1)보다 작은 경우에는 회선의 용량을 충분히 사용하기 전에 미리 혼잡윈도우를 감소시켜 통신 용량을 허비하는 결과를 낳는다. 그러므로 버퍼의 사용량이 (식 1)에 의하여 구한 임계치 값일 때 ECN 마크를 하는 것이 가장 효율적임을 알 수 있다.



(그림 10) Noecn과 ECN방식의 재전송 발생 비교

효율적인 혼잡제어를 위한 많은 연구가 수행되어, 효율성을 제고하는 다양한 전략이 제시되었지만 아직도 혼잡제어는 미해결 과제이다. 혼잡제어의 해를 아직 발견하지 못한 이유는 링크의 비트 오류율이 다양하고 세그먼트 크기가 일정하지 않기 때문에 세그먼트의 손실율도 다양하며, 대역폭과 세그먼트의 크기에 따라 세그먼트가 라우터에 머무는 시간이 다양하고, 더구나 라우터가 관련된 연결의 수가 동적으로 변화하는 등 혼잡과 관련된 변수가 너무 많기 때문이다.

본 논문이 제시한 (식 1)은 혼잡이 발생하는 장소인 라우터 자체에 생각을 고정시켜 비트 오류율, 손실율, 대역폭, 세그먼트의 길이, 등 다양한 변인이 모두 반영되어 라우터에 나타나는 측정 가능한 현상만을 사용하여 혼잡을 예측한다는 장점이 있다.

5. 실 험

라우터가 혼잡을 예측하기 위하여 사용할 임계치를 구하는 (식 1)의 타당성을 보이려면 (식 1)의 실용성과 정확성을 보여야 한다. 실용성은 6절에서 다루기로 하고, 본 절에서는 (식 1)의 정확성을 보이는 실험 결과를 제시한다.

5.1 전송 오류율이 제로인 경우

본 실험의 목적은 라우터가 ECN 마킹을 하는 기준으로 사용될 임계치를 구하는 (식 1)의 정확성을 보이는 것이다. (식 1)에 사용된 인수는 버퍼의 길이, RTT, TBL 그리고 TBA 뿐이다. 이 인수들은 모두 라우터에서 측정 가능한 값들이다. 이 인수들의 값을 변형시키면서 (그림 1)의 페트리넷에서 시뮬레이션을 수행한 결과 이들이 어떤 값을 갖더라도 (그림 11)과 <표 1>에 보이는 결과와 비슷한 경향을 보였으며, 이로부터 (식 1)이 타당하다는 결론을 얻었다. 여러 가지 실험 중, 다음과 같은 실험 환경에서 얻은 실험 결과를 소개한다.

1) 송신 TCP는 매 5 단위시간마다 한 세그먼트를 전송하여 총 300 세그먼트를 전송하고 종료함. 즉, TBA를 5 단위시간으로 함.
2) 라우터 버퍼의 크기를 80으로 함.
3) TBL을 10 단위시간으로 함.
4) 전송 오류율을 0으로 함.

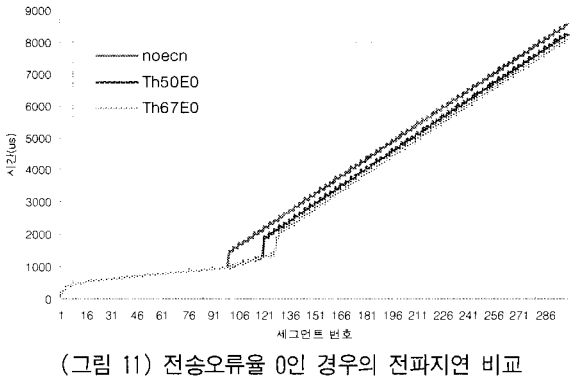
실험 결과 <표 1>과 같은 처리량(Throughput)과 전파지연(Propagation Delay)을 얻었다. Noecn, Th50, Th67은 각각 ECN을 채용하지 않은 경우, 임계치(Threshold)를 50으로 한 경우, 임계치를 67로 한 경우를 의미한다. 임계치 67은 (식 1)에 의하여 얻은 값이다.

300 세그먼트를 '하나 전송하고 접수확인을 받으면 또 다른 하나를 전송하는 방식'으로 전송한다면 모두 전송하는데 걸리는 시간은 299*120+61 = 35,940이므로, ECN 방식을 채용하지 않은 TCP는 하나씩 전송하는 방법에 대하여 35940/10163 = 3.54 배 속도가 빠른 것을 보인다. 예상한 바와 같이 임계치를 67로 한 경우에 처리율과 전파지연 모두가 좋다는 것을 알 수 있다. 각 세그먼트에 대한 전파지연 시간은 (그림 11)에 보이는 차트와 같다.

실험 결과, Noecn, Th50, Th67의 성능 차이가 미미한 것을 볼 수 있는데, 그 이유는 (그림 11)에 보이는 바와 같이 Noecn이나 Th50 모두 비슷한 횟수의 '느린 시작'을 하기 때문이다. 차이점은 Noecn의 경우 세그먼트 폐기 후 타임아웃 기간 동안 전송했던 세그먼트들을 재전송하는데 반하여, Th50과 Th67은 재전송 없이 '느린 시작'을 한다는 것이다.

<표 1> 전송 오류율이 0일 경우, ECN 마킹을 위한 임계치에 따른 성능 비교

	noecn	Th50	Th67
300개 전송에 걸린 시간 (throughput)	10,163	9,768	8,052
한 세그먼트 전송에 필요한 평균 전파지연	3583.247	3290.197	2754.9



(그림 11) 전송오류율 0인 경우의 전파지연 비교

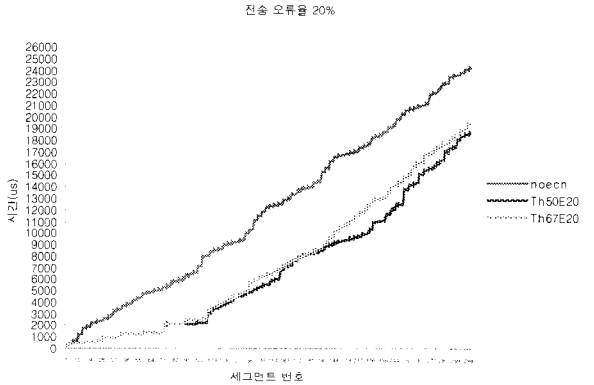
5.2 전송 오류율이 20%인 경우

실제 세계에서는 전송 오류가 빈번하게 발생한다. 특히, 무선 네트워크의 전송 오류율은 매우 높다. 본 실험에서는 전송 오류율이 매우 높은 경우에도 (식 1)이 타당한지 보이기 위하여 전송 오류율을 20%로 설정하였고, 300 개의 세그먼트를 모두 전송하는 데 걸리는 시간을 측정하여 <표 2>와 같은 결과를 얻었다. 앞에서와 마찬가지로, noecn, Th50, Th67은 각각 ECN 마킹을 사용하지 않는 경우, 라우터 버퍼에 세그먼트가 50(67)개일 때 ECN 마킹을 하는 경우를 나타낸다.

실험 결과 Th50과 noecn을 비교할 때, 처리율은 22% 개선 효과가 있으며, 평균 전파지연은 42% 개선 효과를 보인다. Th50과 Th67을 비교할 때 Th50의 경우가 성능이 우세함을 보인다. 이것은 전송오류에 따른 재전송으로 말미암아, 라우터에 도착하는 세그먼트의 빈도가 높아짐에 따라 (식 1)의 값이 67보다 50에 더욱 가깝기 때문이다.

〈표 2〉 전송 오류율이 0.2일 경우, ECN 마킹을 위한 임계치에 따른 성능 비교

	noecn	Th50	Th67
300개 전송에 걸린 시간 (throughput)	25,900	20,270	21,148
한 세그먼트 전송에 필요한 평균 전파지연	11,921.26	6,865.78	7459.69



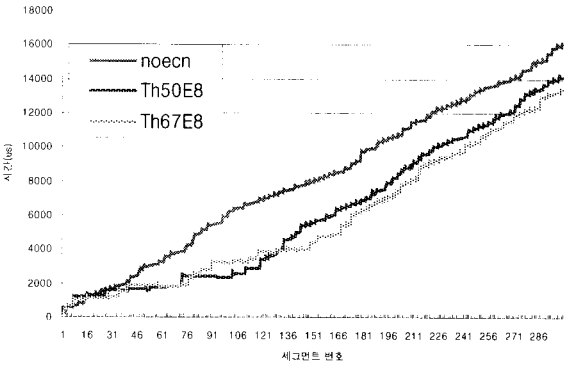
(그림 12) 전송오류율 0.2인 경우의 전파지연 비교

5.3 전송 오류율이 8%인 경우

본 실험에서는 실제 세계를 반영하기 위하여 전송 오류율을 8%로 설정하였고, 300 개의 세그먼트를 모두 전송하는 데 걸리는 시간을 측정하여 <표 3>과 같은 결과를 얻었다. 앞에서와 마찬가지로 noecn, Th50, Th67은 각각 ECN 마킹을 사용하지 않는 경우, 라우터 버퍼에 세그먼트가 50(67)개일 때 ECN 마킹을 하는 경우를 나타낸다. 실험 결과 Th50과 noecn을 비교할 때, 처리량은 약 11% 개선 효과가 있으며, Th67과 noecn을 비교하면 처리량이 약 15% 정도 개선됨을 알 수 있다.

〈표 3〉 전송 오류율이 0.08일 경우, ECN 마킹을 위한 임계치에 따른 성능 비교

	noecn	Th50	Th67
300개 전송에 걸린 시간 (throughput)	17,669	15,732	14,986
한 세그먼트 전송에 필요한 평균 전파지연	8,120.34	6,091.67	5,705.78



(그림 13) 전송오류율 0.08인 경우의 전파지연

6. 적용 방법

앞 절에서, ECN 마킹을 위한 적당한 임계치를 구하는 식으로 (식 1)을 제시하고, 실험을 통하여 이의 타당성을 보였다. (식 1)을 실제에 적용하려면 TCP 연결에 할당된 라우터 버퍼의 길이, RTT, TBA, TPS를 라우터가 알아야 한다. 라우터의 버퍼 운용 전략은 매우 복잡한 문제이지만, 운용 전략에 따라 TCP 연결에 할당된 라우터 버퍼의 길이를 측정하는 것은 가능하다.

RTT는 송신자가 세그먼트에 부착하여 라우터에게 알려주어야 한다. Option 필드에 RTT를 기록하고, 이 사실을 예약 필드에 표시하여 줌으로써 라우터에게 필요한 정보를 전달할 수 있다. TBA와 TPS는 라우터에서 통계적으로 비교적 쉽게 구할 수 있는 값이다. 이러한 값들을 동적으로 구하여 (식 1)을 평가함으로써 가장 효율적인 ECN 마킹 전략을 운용할 수 있다.

(식 1)의 단점으로 RTT가 너무 크고, TBA와 TBL의 차이가 크면 (식 1)의 결과가 음수가 되거나, 아주 작은 값이 된다는 것이다. 이럴 경우에도 (식 1)의 결과는 타당하다. 왜냐하면, 이러한 경우에는 (식 1)을 적용하는 방법이 버퍼에 세그먼트가 얼마나 쌓였는지는 고려하지 않고, 라우터에 도착하는 세그먼트의 빈도가 처리 능력을 초과하면 ECN 마킹을 해야 한다는 SQM 방법이 되기 때문이다.

본 논문이 제공한 모델을 이용하여 <표 4>에 보이는 시뮬레이션 방법을 적용하여 최적 임계치를 구할 수 있다. <표 4>의 단계1과 단계2는 자명하므로 설명을 생략한다. 단계 3에서는 추정 한계치를 수정한다. 이때, 앞 단계에서 사용한 추정 한계치가 단계2에서 테스트한 세 가지 경우 중 가장 우수하면, 범위만 축소하여 단계2를 반복하고, 아니면 추정 한계치만 수정하여 단계2를 반복한다.

예를 들어, 추정한계치=0.7, 범위=0.1의 경우 단계2에서 속도3이 가장 우수하였다면, 다음 반복에서는 추정한계치=0.8, 범위=0.1이 되어 단계2를 반복하게 된다. 같은 예에서 단계2의 결과 속도1이 가장 우수하였다면 추정 한계치는 0.7로 그대로 두고, 범위를 0.05로 축소하여 단계2를 반복한다. 속도 1, 2, 3간의 차이가 충분히 작으면 추정 한계치를 최적 한계치로 결정한다.

〈표 4〉 시뮬레이션 방법으로 최적 한계치를 찾는 과정

단계 1: 초기화. 추정한계치 = 0.7; 범위 = 0.1;
단계 2: 시뮬레이션
  속도1 = '추정한계치'를 이용한 시뮬레이션 결과;
  속도2 = '추정한계치'-범위를 이용한 시뮬레이션 결과;
  속도3 = '추정한계치'+범위를 이용한 시뮬레이션 결과;
단계 3: 추정한계치 및 범위 갱신:
  추정한계치 = 속도1, 속도2, 속도3을 비교하여 가장 좋은 결과를 얻은 경우의 한계치;
  속도1이 가장 좋은 결과인 경우에는 범위 = 범위/2; (* 아니면, 불변 *)
단계 4: 속도1, 속도2, 속도3 간의 편차가 작으면 추정한계치를 최적 한계치로 하고 종료, 아니면 단계 2로 점프.

7. 결 론

본 논문은 ECN 마킹을 위한 적당한 임계치를 구하는 식으로 (식 1)을 제시하고, ECN 마킹을 적용하는 TCP 모형을 페트리 넷으로 구축한 다음, 시뮬레이션을 통하여 (식 1)의 타당성을 보였다. 또한, (식 1)의 실용화 방안도 제시하였다.

혼잡에 영향을 주는 요인은 링크의 비트 오류율, 세그먼트의 손실율, 대역폭과 세그먼트의 크기, 라우터가 관련된 연결의 수 등 너무 많다. 더구나 이들의 값은 대부분 동적으로 변화한다. 이러한 요인들을 한꺼번에 고려하면 일목요연한 결과를 얻기가 힘들다. 이러한 다양한 요인들이 복합적으로 작용하여 라우터에서는 TBL과 TBA라는 통계적으로 측정 가능한 변인으로 나타난다. (식 1)의 특징은 통계적으로 측정 가능한 변인으로 구성되어 적용이 용이하다는 것이다.

여러 가지 실험을 통하여, 본 논문에 소개된 TCP 모형이 실제 TCP를 잘 반영한다는 것을 검증하였다. 본 모형은 각 사건에 대하여 어느 세그먼트에서 그 사건이 발생하였는지 기록하는 것과 같은 비교적 세밀한 수준의 모형으로 쉽게 변화될 수 있다는 장점을 가지고 있기 때문에, 네트워크 연구자에게 좋은 시뮬레이션 도구로 사용될 수 있으리라고 사료된다.

참 고 문 헌

[1] J. Postel, "Transmission control protocol," RFC 793, IETF, Sept., 1981

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

[3] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP," RFC 2481, Jan., 1999

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

[5] R. Braden and J. Postel, "Requirements for internet gateways," RFC 1009, IETF, June, 1987

[6] T.Y. Lin and Y.C. Chen, "A congestion control approach for LAN/MAN interconnection via ATM," Proc. IEEE 13th Networking for Global Communications, Vol.2, 1994, Toronto, Ont., Canada, pp.892-901.

[7] A. Dracinschi and S. Fdida, "Congestion avoidance for unicast and multicast traffic," 1st European Conference on Universal Multiservice Networks, 2000, Colmar, France, pp.360-368.

[8] Z. Chen, S. Jiang, S. Zheng, and C. Ko, "Performance comparison between ECN and BECN," ICATM 2001, 2001, Seoul, South Korea, pp.305-309.

[9] F. Peng and V. Leung, "Fast backward congestion notification mechanism for TCP congestion control," 21st IEEE International Performance, Computing, and Communications Conference, 3-5 April, 2002, pp.419-424.

[10] K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," ACM Trans. On Computer Systems, Vol.8 No.2, 1990, pp.158-181.

[11] R. Jain, "Congestion control in computer networks: issues and trends," IEEE Network, Vol.4, Issue: 3, pp.24-30, May, 1990.

[12] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Trans. on Networking, Vol.1, Issue: 4, pp.397-413, Aug., 1993.

[13] Y. Zhou, T. Murata, and T. DeFanti, "Modeling and performance analysis using extended fuzzy-timing petri nets for networked virtual environments," IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol.30, No.5, pp.737-756, October, 2000.

[14] R. Gaeta, M. Gribaudo, D. Manini, and M. Sereno, 'On the use of petri nets for the computation of completion time distribution for short TCP transfers,' ICATPN 2003, Lecture Notes in Computer Science, Vol.2679, Springer, New York, pp.181-200, 2003.

[15] A. Ost and B. Haverkort, "Analysis of windowing mechanisms with infinite-state stochastic petri nets," ACM Performance Evaluation Review, Vol.26, No.8, pp.38-46, 1998.

[16] 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* , Springer Verlag, New York, pp.88-114, 1999

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

[18] M. Kwon and S. Fahmy, "On TCP reaction to explicit congestion notification," Journal of High Speed Networks, Vol.13, pp.123-138, 2004

[19] S. Floyd, M. Handley, J. Padhye, and J. Widner, "Equation-based congestion control for unicast applications," Proc. ACM SIGCOMM, 2000, pp.43-56

[20] Y. Yang and S. Lam, "General AIMD congestion control," Proc. IEEE ICNP 2000, Osaka, Japan, http://www.cs.utexas.edu/users/lam/Vita/IEEE/YangLam00.pdf

[21] S. Floyd, "TCP and explicit congestion notification," ACM Computer Communication Review, Vol.24, No.5, pp.10-23, Oct., 1994

[22] C. Liu and R. Jain, "Improving explicit congestion notification with the mark-front strategy," Computer Networks, Vol.35, No.2-3, pp.185-201, 2001

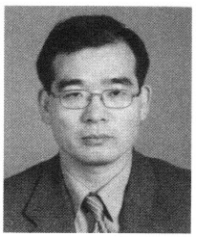
이 계 영

e-mail : lky@dongguk.ac.kr
1980년 동국대학교 전자계산학과(학사)
1983년 동국대학교 대학원 전산학과(석사)
1992년 단국대학교 전자공학과(박사)
1980년~1981년 한국후지쯔 OS 개발부
1988년~1991년 동국대학교 전자계산 소장
1996년~1997년 미국 워싱턴주립대학 방문연구교수
198년5~현재 동국대학교 공학대학 교수
2005년~현재 정보처리학회 이사
관심분야 : 운영체제, 음성처리, 멀티미디어, 이동 에이전트 등

임 재 걸

e-mail : yim@dongguk.ac.kr
1974년 인천교육대학(학사)
1981년 동국대학교(학사)
1987년 일리노이대 시카고 캠퍼스(석사)
1990년 일리노이대 시카고 캠퍼스(박사)
1992년~현재 동국대학교 컴퓨터학과 교수
관심분야 : 페트리넷 이론과 응용, 컴퓨터 네트워크, 시스템 설계 분석, 인공지능

장 익 현



e-mail : yim@dongguk.ac.kr
1984년 서울대학교 계산통계학과(학사)
1986년 한국과학기술원 전산학과(석사)
1998년 한국과학기술원 전산학과(박사)
1986년~1999년 (주) 데이콤 책임연구원
1999년~현재 동국대학교 정보통신공학과 부교수
관심분야 : 컴퓨터통신, 분산시스템, 인터넷 프로토콜 등