論 文
53D-12-4

타임아웃 패턴에 기반한 TCP 혼잡 제어

TCP Congestion Control Based on Timeout Patterns

林 甲 珠* · 延 昌 桓** · 南 尙 沅*** · 李 寅 煥†
(Gabjoo Lim · Changhwan Youn · Sang-Won Nam · Inhwan Lee)

Abstract - This paper infers two patterns of timeout from the characteristics of TCP Reno and confirms the existence of the patterns by conducting actual experiments. All timeouts can be classified into either of these patterns by using the history of RTT value. Based on the observed patterns, the paper proposes two algorithms to improve the performance of TCP Reno. Experimental results show that, when compared with TCP Reno, the proposed algorithms improve the bandwidth utilization by 3 to 12 percent. The paper provides good examples of how timeout-based and delay-based congestion control can efficiently work together.

Key Words : TCP, Congestion Control, Timeout Pattern, Round Trip Time

1. 서 론

컴퓨터 네트워크에서 혼잡(congestion)이라 함은 네트워크 노드에 과부하가 걸린 상태를 의미하며, 혼잡제어(congestion control)라 함은 네트워크 노드들이 과부하를 예방하거나 과부하 상태로부터 빠져나오기 위한 노력을 지칭한다. 인터넷에서는 Transmission Control Protocol (TCP) 계층이 혼잡제어를 담당한다. TCP에서 혼잡제어를 위해 가장 널리 사용하는 알고리즘은 TCP Reno [1,2] 및 TCP Vegas이고 [3], 이들은 각각 타임아웃 기반 및 지연시간 기반 혼잡제어의 대표적인 사례이다. 이 외에도 TCP 혼잡제어의 성능을 향상시키기 위해 대단히 많은 수의 알고리즘들이 제안되었다 [4-8].

이렇게 TCP 혼잡 제어를 위한 많은 연구가 이루어져 왔으나 그 성능 개선의 여지는 여전히 남아 있다. 그 주된 이유는 인터넷에서는 엔드 호스트만을 이용하여 혼잡제어를 수행함으로 인해 완벽한 알고리즘이 제시되기 어렵기 때문이다. 그리고 또 다른 이유는 네트워크의 대역폭이 확장되고, 멀티미디어와 동등계층(peer-to-peer) 통신과 같은 새로운 형태의 트래픽 및 통신 형태가 지속적으로 출현함에 따라, 혼잡제어 알고리즘이 이에 적응할 필요가 있기 때문이다. 본 논문에서는 현재 대부분의 시스템이 사용하고 있는 TCP Reno의 특성을 재조명하고, 이를 개선하기 위한 새로운 알고리즘을 제안하고 평가한다.

† 교신저자, 正會員 : 漢陽大 電子電氣컴퓨터 工學 副教授
E-mail : ihlee@hanyang.ac.kr
* 正 會 員 : 三星電子 半導體事業部
** 學生會員 : 漢陽大 電子通信電波 工學 碩士過程
*** 正 會 員 : 漢陽大 電子電氣컴퓨터 工學 教授
接受日字 : 2004年 10月 13日
最終完了 : 2004年 11月 23日

2. TCP Reno/Vegas

TCP Reno에서는 타임아웃이 발생하면 혼잡이 발생한 것으로 보고 혼잡 윈도우(congection window; cwnd)를 감소시켜 패킷 전송률을 낮춤으로써 혼잡을 해소한다. TCP Reno는 크게 슬로우 스타트, Additive Increase Multiplicative Decrease(AIMD), 그리고 fast retransmission 및 fast recovery로 구성 된다 [1,2].

슬로우 스타트는 TCP 컨넥션이 새로 연결된 경우 또는 타임아웃에 의한 재전송 후에, 사용가능한 대역폭을 찾기 위해 cwnd 값을 빠르게, 즉 acknowledge(ACK)가 도착할 때마다 cwnd값을 1씩 증가시킨다. 이러한 과정에서 타임아웃이 발생하면 cwnd를 1로 설정하고, 슬로우 스타트 임계값(slow start threshold; ssthresh)을 타임아웃 발생 당시 cwnd의 1/2로 설정한 후, 다시 슬로우 스타트를 수행한다.

슬로우 스타트에 의해 cwnd가 빠르게 증가하다가 ssthresh보다 커지면, Additive Increase(AI)는 혼잡을 피하기 위하여 cwnd 만큼의 ACK가 수신되면 cwnd를 1씩 증가시킨다. 따라서 AI에서는 cwnd가 상대적으로 느리게 증가한다. AI 과정에서 타임아웃이 발생하면 Multiplicative Decrease(MD)에 의해 ssthresh를 현재 cwnd의 1/2로 설정하고, cwnd를 1로 감소시킨 후, 다시 슬로우 스타트를 수행한다.

TCP Reno는 데이터를 전송한 후 타임아웃이 발생하기 전에 동일한 시퀀스 번호를 갖는 duplicated ACK(dupack)을 3개 이상 수신하면 (본 논문에서는 이 상황을 간단히 'triple dupack'이라고 표시함), 해당 패킷이 손실되었다고 판단하고 ssthresh를 cwnd의 1/2로 설정하고, fast retransmission에 의해 즉시 해당 패킷을 재전송한다. 이후 새로운 세그먼트를 요구하는 ACK가 수신되면 fast recovery에 의해 cwnd를 ssthresh로 설정하고 AI를 수행한다.

TCP Reno의 AIMD는 타임아웃이 발생하여 혼잡을 감지하면 cwnd를 1/2로 급격하게 감소시킴으로써 혼잡을 빠르게 해소할 수 있는 이점을 제공하는 반면, 급격한 cwnd의 감소로 인해 cwnd의 값이 출렁거리게 되고 해당 컨넥션들의 대역폭 이용률이 낮아질 수 있다. 또한 AI는 네트워크 트래픽을 고려하지 않고 cwnd를 꾸준히 증가시키기 때문에, 필연적으로 타임아웃을 유발하고 이에 따라 대역폭 이용률이 떨어질 수 있다. 어떻게 보면 이러한 단점들은 타임아웃 기반 혼잡제어의 한계라 할 수 있다. 본 논문에서는 TCP Reno의 AIMD의 특성을 보완하는 새로운 혼잡제어 알고리즘을 제안한다.

TCP Vegas[3]는 지연시간 기반의 혼잡제어 알고리즘이다. Vegas는 패킷을 전송할 때의 Round Trip Time(RTT) 정보를 이용하여 네트워크에서 처리량(throughput)의 기대값과 실제값을 추정하고, 이들 두 값의 차이를 이용하여 네트워크 혼잡의 정도를 판단하며, 그 정도에 따라 cwnd의 값을 조정한다. 실제 Vegas 알고리즘의 구조와 파라미터들은 많은 실험을 통해 타임아웃을 줄이고 처리량을 늘이도록 최적화 되었다.

타임아웃이나 triple dupack 정보에 의존하는 타임아웃 기반 방식과는 달리, TCP Vegas는 RTT를 측정하여 얻은 네트워크 트래픽에 대한 정보를 사용하기 때문에 혼잡 상황에 능동적으로 대응할 가능성이 있다. 그리고 Vegas에서의 cwnd의 변화는 Reno의 그것에 비해 상당히 매끄러운 모습을 보인다. 그러나 Vegas와 같은 지연시간 기반 알고리즘에서는 RTT가 잘못 측정되면 네트워크 혼잡의 정도를 올바르게 판단할 수 없기 때문에, asymmetry 문제, rerouting 문제 및 persistent congestion 문제 등이 발생할 수 있다[9,10].

3. 타임아웃 패턴

인터넷에서 트래픽이 점차 증가하면 병목 라우터(bottleneck router; 과부하가 걸린 라우터)가 출현하고, 이 라우터는 일부 패킷을 버리게 되어 결국 타임아웃이 발생한다. 본 연구의 첫 번째 착안점은, AIMD 하에서는, 이러한 상황을 네트워크를 공유하던 기존의 컨넥션들이 cwnd를 증가시켜 트래픽이 증가하거나 또는 새로운 컨넥션이 연결되어 트래픽이 증가하는 두 가지로 크게 나누어 생각할 수 있다는 것이다. 첫 번째 상황에서는 cwnd가 AI에 의해 느리게 증가하다 혼잡을 유발하므로 비교적 완만한 트래픽의 증가를 보이다가 타임아웃이 발생하는 것으로 볼 수 있다. 반면, 새로운 컨넥션이 연결되는 경우는 슬로우 스타트에 의해 cwnd가 빠르게 증가되기 때문에 비교적 급격하게 트래픽이 증가하며 타임아웃이 발생할 수 있다.

본 연구의 두 번째 착안점은 만일 TCP가 이러한 두 상황을 실시간으로 구별할 수 있다면 TCP는 AIMD보다 개선된 형태의 혼잡제어를 수행할 수 있다는 것이다. 본 연구에서는 TCP 내부에 존재하는 RTT 정보를 이용하여 앞서 제시한 두 상황을 구분하고자 한다. 일반적으로 RTT는 전송시간, 전송지연시간, 그리고 큐잉지연시간의 합으로 표시할 수 있다. 이 중에서 앞의 두 항은 트래픽의 변화에 민감하지 않으나, 큐잉지연시간은 라우터의 큐 길이에 비례하므로 해당 컨넥션이 경유하고 있는 네트워크의 트래픽에 따라 변화한다. 따라서 TCP는 RTT의 변화를 관찰함으로써, 두 가지의 타임아웃 상황의 식별을 시도할 수 있다.

표 1 타임아웃 패턴의 측정을 위한 실험 조건
Table 1 Experimental conditions for evaluating timeout patterns

	FTP 서버 위치	거리	대역폭	RTT (평균)	RTT (최대)
실험 01	USA	22 hop	20 KBps	250 ms	450 ms
실험 02	Korea	17 hop	50 KBps	30 ms	230 ms
실험 03	Korea	13 hop	70 KBps	20 ms	210 ms

3.1. 실험 환경

우선 위에서 제시한 두 가지의 타임아웃 패턴을 실제 인터넷에서 관찰할 수 있는지를 확인하기 위한 실험을 수행하였다. 인터넷에서는 수많은 TCP 컨넥션들이 네트워크를 공유하고, 트래픽이 수시로 변화하며, 여러 가지 조건을 인위적으로 제한할 수 없으므로, 정확한 실험조건을 설정할 수 없다. 하지만 본 실험의 목적은 AIMD 하에서의 두드러진 타임아웃 패턴을 관찰하는데 그 목적이 있다.

여기서는 저자가 소속된 대학교에 위치한 호스트가 TCP 프로토콜을 사용하는 인터넷상의 다른 FTP 서버에 접속하여 10MB 크기의 파일을 전송하면서 RTT, cwnd, 타임아웃, triple dupack 등을 측정하였다. FTP 클라이언트로 사용되는 호스트는 x86 CPU를 사용하고 Linux 2.2.10[11]을 OS로 사용하였고, 인터넷의 특성상 다양한 거리와 경로, 대역폭, RTT가 나타날 수 있기 때문에 다양한 조건에서 실험하였다. 표 1은 아래에서 데이터를 제시할 세 가지의 측정조건을 보인다.

이 실험에서는 독일 Technical University의 Telecommunication Networks Group에서 개발한 SNUFFLE[12]이라는 인터넷 프로토콜 측정 도구를 사용하였다. SNUFFLE은 TCP/IP 프로토콜 스택을 모니터할 수 있고, 시퀀스 번호, 포트 번호, 플래그 등의 헤더 값과 윈도우 크기, ssthresh, RTT, Retransmission Timeout(RTO) 등의 내부 상태 뿐 아니라, 타임아웃, triple dupack, 재전송 등의 TCP 이벤트들을 측정할 수 있다.

3.2. 실험 결과

그림 1은 실제 RTT 값의 시간적 변화를 보이는 예로서 '실험 03'에서 측정된 것이다. 이 그림에서 보듯이 RTT는 어느 정도의 변화폭 내에서 변화하는 경우와 급격하게 증가하여 피크를 나타내는 경우로 구분되는 유형을 보인다. 그림 2는 이러한 두 가지 상황의 실제 예를 측정된 데이터를 이용하여 예시한다. 그림 2(a)는 RTT가 급격히 증가하여 피크 값을 보이며 타임아웃이 발생한 경우이고, 그림 2(b)는 RTT 피크가 나타나지 않으며 타임아웃이 발생한 경우이다.

이러한 두 가지 상황을 정량적으로 식별하기 위하여 RTT의 변화 정도를 나타내는 RTT 피크율(RTT peak

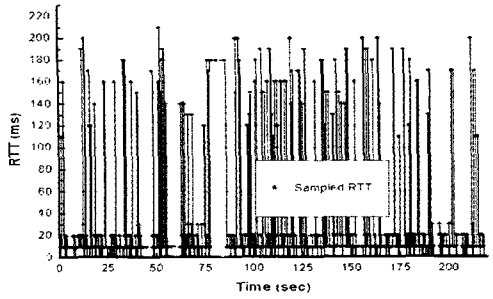
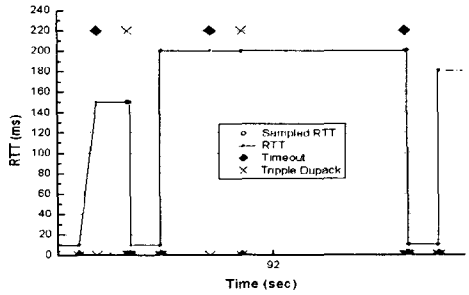
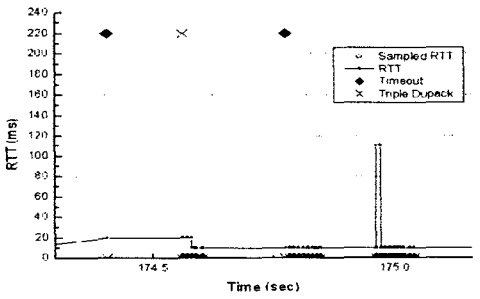


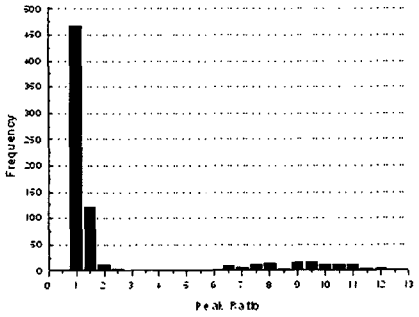
그림 1 RTT 값의 시간적 변화
Fig. 1 Measured RTT



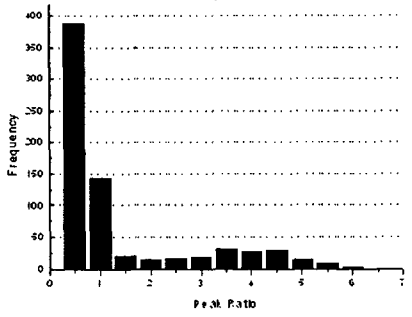
(a) RTT 피크와 함께 타임아웃 발생



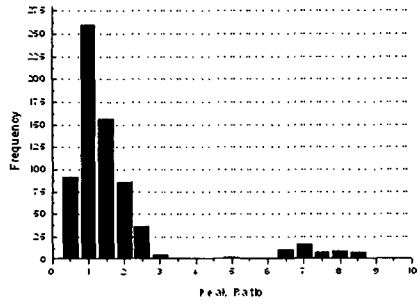
(b) RTT 피크 없이 타임아웃 발생
그림 2 타임아웃의 실제 예
Fig. 2 Examples of timeouts



(a) 실험 01



(b) 실험 02



(c) 실험 03
그림 3 RTT 피크율의 분포
Fig. 3 Distribution of RTT peak ratio

ratio)을 다음과 같이 정의하였다.

$$RTT\ 피크율 = \frac{RTT - BaseRTT}{RTT\ average - BaseRTT} \quad (1)$$

식(1)에서 BaseRTT는 측정된 RTT의 최소값으로 TCP 컨넥션의 전송지연시간을 의미한다. BaseRTT를 도입하지 않고 RTT의 평균값에 대한 측정된 RTT의 비율을 RTT 피크율로 정의하면, 전송지연시간이 큰 컨넥션 일수록, 즉 송신자와 수신자 사이의 거리가 멀수록 RTT의 평균값이 증가하여 상대적으로 RTT 피크율이 감소하게 되므로, 인터넷에서와 같이 다양한 전송지연시간을 갖는 환경에서 정량화된 기준으로 사용할 수 없다.

그림 3은 앞서 제시한 세 가지 실험 조건에서 측정한 RTT 피크율의 분포를 나타낸다. 이 그림은 RTT 피크율이 1 주변의 값을 갖는 경우 (즉, 측정된 RTT가 RTT 평균값과 유사한 경우)와 매우 큰 값을 갖는 경우로 구분할 수 있음을 보인다. 실험 조건에 따라 RTT 피크율의 최대치는 차이가 있지만, 대부분의 경우 RTT 피크율은 2 이하의 값으로 나타나고, 나머지 큰 값들은 긴 꼬리의 형상으로 분포하고 있다. 실제 실용화 단계에서는 많은 실험을 통해 관련 파라미터를 최적화하여야 하나, 본 연구에서는 타임아웃 발생시 RTT 피크율의 값이 2 이상인 경우 RTT가 급격히 변하여 피크 값을 보인 것으로 간주한다.

실험을 통해 측정된 RTT 피크의 발생빈도, 그리고 타임아웃과 triple dupack이 발생했을 때의 RTT 피크의 발생 빈도는 표 2와 같다. 본 실험에서 RTT 피크의 발생 빈도는 5%에서 20% 사이의 변화를 보이고 있고, 타임아웃과 triple dupack이 발생한 조건에서의 RTT 피크 발생빈도는 13%에서 20%로서, 단순한 RTT 피크 발생빈도에 비해 높고 그

표 2 RTT 피크와 타임아웃 및 triple dupack과의 관계
Table 2 Relationship among RTT peak, timeout, and triple dupack

	RTT 피크 발생	타임아웃시 RTT 피크 발생	triple dupack시 RTT 피크 발생
실험 01 (20KBps)	19.6%	20%	19.7%
실험 02 (50KBps)	5.04%	14.4%	12.5%
실험 03 (70KBps)	6.2%	15.3%	18.5%

표 3 타임아웃 또는 triple dupack 발생 후 cwnd 값의 상승
Table 3 Recovery of cwnd after timeout or tiple dupack

	Cwnd의 증가 평균 (RTT 피크 발생하지 않은 경우)	Cwnd의 증가 평균 (RTT 피크 발생한 경우)
실험 01 (20KBps)	4.37	3.69
실험 02 (50KBps)	5.53	4.92
실험 03 (70KBps)	7.01	5.63

표 4 타임아웃과 triple dupack이 발생할 확률
Table 4 Probability of timeout or triple dupack occurring

	RTT 피크 발생하지 않은 경우		RTT 피크 발생한 경우	
	타임아웃	triple dupack	타임아웃	triple dupack
실험 01 (20KBps)	3.97%	7.31%	4.08%	7.26%
실험 02 (50KBps)	5.43%	5.39%	17.18%	14.43%
실험 03 (70KBps)	9.09%	4.79%	24.83%	16.44%

변화폭이 작다. 표 2는 실제 타임아웃과 triple dupack이 발생할 때 RTT 피크가 관찰되지 않는 경우가 더 많음을, 즉 이들 대부분이 점진적인 트래픽의 증가에 의해 발생한 것임을 시사한다. 실제 인터넷에는 많은 컨넥션이 존재하기 때문에 한 컨넥션의 cwnd가 급격히 증가하더라도 나머지 모든 컨넥션의 입장에서는 완만한 트래픽의 증가로 보일 수 있다.

TCP Reno에서 타임아웃 또는 triple dupack이 발생하면 cwnd는 크게 감소하였다가 다음 타임아웃 또는 triple dupack까지 다시 증가한다. 표 3은 이러한 두 사건 사이에 cwnd가 얼마나 증가하는지를 보인다. 이 표에서 보듯이 RTT 피크를 동반한 경우에 cwnd의 값이 회복되는 폭은 RTT 피크를 동반하지 않은 경우의 그것에 비해 80% 정도이다. 이것은 RTT 피크를 동반한 타임아웃 또는 triple dupack에서는 급격한 트래픽의 증가로 인해 사용가능한 대역폭이 RTT 피크를 동반하지 않은 경우보다 더 많이 감소한다는 것을 나타내고, 또한 RTT 피크의 존재 여부가 트래픽의 변화 정도를 반영하고 있음을 의미한다.

표 4는 RTT 피크가 발생했을 때와 발생하지 않았을 때의 타임아웃과 triple dupack의 발생빈도를 보인다. 이 표는 RTT 피크가 발생하면 타임아웃 혹은 triple dupack이 발생할 가능성이 크다는 것을 뚜렷이 보여준다. 그리고 대역폭이 증가할수록 RTT 피크의 발생과 함께 타임아웃과 triple dupack이 발생한 빈도가 점점 증가한다. 이러한 결과는 RTT 피크를 TCP 컨넥션이 경유하고 있는 네트워크의 혼잡의 징후로 볼 수 있음을 의미한다.

3.3. 토 론

본 절에서는 TCP 컨넥션에서 타임아웃이 발생하는 패턴을 두 가지로 구분할 수 있음을 보였다. 첫 번째 상황에서는 cwnd가 AI에 의해 비교적 느리게 증가하다 혼잡을 유발하므로 비교적 완만한 트래픽의 증가를 보이다가 타임아웃이 발생한다. 반면, 새로운 컨넥션이 연결되는 경우는 슬로우 스타트에 의해 cwnd가 빠르게 증가되기 때문에 비교적 급격하게 트래픽이 증가하며 타임아웃이 발생한다. 이것은 하나의 컨넥션의 관점에서 타임아웃을 해석한 것이다. 실제 인터넷에서 네트워크를 공유하고 있는 TCP 컨넥션의 수는 매우 많고 또한 시간적으로 크게 변화한다. 그리고 네트워크를 공유하고 있는 컨넥션의 수가 많을수록 새로운 컨넥션이 추가되어도 각각의 컨넥션이 받는 영향은 작아진다. 즉, 컨넥션의 수가 많아지면, 새로운 컨넥션에 의해 트래픽이 급격하게 증가하는 것과 기존의 컨넥션에 의해 트래픽이 점진적으로 증가하는 것의 차이가 타임아웃 패턴을 결정하는 것이 아니라, 타임아웃이 발생한 시점에서 공유하고 있는 네트워크의 전체 트래픽의 증가량이 급격한지 아닌지가 타임아웃 패턴을 결정하게 된다.

결론적으로 두 가지 타임아웃 패턴은 네트워크를 공유하고 있는 컨넥션들이 얼마나 많은 대역폭을 더 요구하고 있는가를 구분하는 것이다. 그리고 RTT 피크율은 이러한 추가적으로 요구되는 대역폭의 정도를 나타내는 척도로 볼 수 있다.

4. 알고리즘 제안 및 성능 평가

본 절에서는 앞 절의 결과를 바탕으로 TCP Reno의 AIMD의 특성을 개선하기 위한 두 가지 알고리즘을 제안하고 평가한다. 제안한 알고리즘은 RTT 정보를 이용하여 TCP Reno의 특성을 개선하는 것으로, 타임아웃 기반 및 지연시간 기반의 혼잡제어를 결합한 형태를 갖는다.

4.1 선택적 감소 (Selective Decrease; SD)

본 논문에서 제안하는 SD 알고리즘은 타임아웃 패턴을 기반으로 동작한다. 먼저 RTT 피크를 동반한 타임아웃이 발생하면 기존의 MD와 동일하게 ssthresh를 cwnd의 50%로 설정하고 cwnd를 1로 감소시킨다. 그러나 RTT 피크를 동반하지 않은 타임아웃이 발생하면, MD와 달리, ssthresh를 cwnd의 70%로 설정하고 cwnd를 1로 감소시킨다. 즉, SD의 요체는 타임아웃이 발생한 시점에서 네트워크에서 추가적으로 요구되는 대역폭이 적을 때에는, AIMD에 비해

cwnd와 이에 따른 패킷 전송률의 회복속도를 빠르게 해 주어 네트워크의 대역폭을 보다 적극적으로 사용하자는 것이다. 물론 cwnd를 너무 빠르게 회복시키면 혼잡이 해소되지 않을 수 있으므로 감소율을 적절히 선택하여야 한다. 여기서 70%라는 값은 본 연구에서의 실험을 통해 선택된 값으로, 실용화를 위해서는 보다 광범위한 실험을 통해 RTT 피크가 발생한 경우와 그렇지 않은 경우에 대한 ssthresh 값의 최적치를 결정하여야 할 것이다.

이러한 선택적 감소는 fast retransmission에도 적용된다. 즉, triple dupack을 감지하게 되면 RTT 피크의 발생여부에 따라 ssthresh를 cwnd의 50% 혹은 70%로 설정하고 cwnd를 1/2로 감소시킨다. 그림 4는 TCP Reno에서 SD가 적용되는 부분을 나타낸다.

4.2. 능동적 감소 (Proactive Decrease; PD)

기존의 TCP는 사용가능한 대역폭을 파악하기 위해 타임아웃이 발생할 때까지 AI에 의해 cwnd를 꾸준히 증가하기 때문에 불가피하게 타임아웃이 발생한다. 앞 절에서는 RTT 피크가 발생한 경우에 타임아웃 혹은 triple dupack이 발생할 빈도가 높고, RTT 피크를 네트워크 혼잡의 척도로 볼 수 있음을 보였다. Proactive Decrease (PD) 알고리즘은 네트워크의 혼잡을 예측하고, 혼잡이 발생하기 전에 cwnd와 이에 따른 패킷 전송률을 능동적으로 감소시킴으로써, 불필요한 타임아웃이나 triple dupack을 줄이고자 한다. 하지만 PD가 혼잡을 지나치게 적극적으로 예측하면 필요한 것보다 빨리 cwnd를 감소시킴으로써 대역폭 이용률이 떨어질 위험성이 있다.

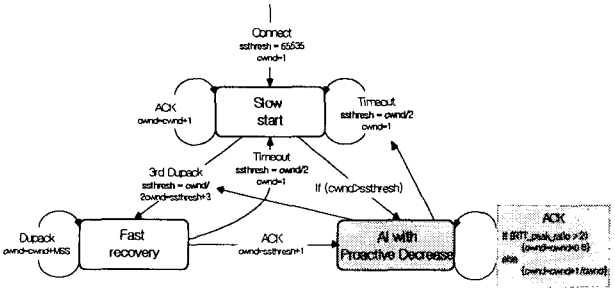


그림 5 TCP Reno와 PD
Fig. 5 TCP Reno and PD

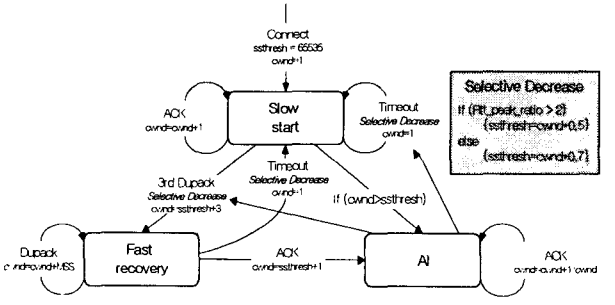


그림 4 TCP Reno와 SD
Fig. 4 TCP Reno and SD

구체적으로 PD는 ACK가 도착하여 측정된 RTT의 값에서 피크가 발생하면 동작한다. 만약 RTT 피크가 발생하면 타임아웃이나 triple dupack의 발생 여부와 관계없이 cwnd를 20% 감소시킴으로써, AI의 꾸준한 cwnd의 증가에 의한 타임아웃이나 triple dupack을 줄이고자 한다. 그림 5는 기존의 TCP Reno에서 PD가 적용되는 부분을 나타낸다.

4.3 성능 평가

앞서 언급했듯이 인터넷에서는 실험 조건들을 임의로 설정할 수 없으므로 인터넷 실험을 통한 엄밀한 수치적인 성능평가는 어렵다. 따라서 본 연구에서는 제안한 알고리즘의 성능을 평가하고 기존의 TCP Reno와 비교하기 위해 그림 6과 같은 통제된 실험 환경을 구축하였다. 이 환경에서는 병목 라우터의 역할을 담당하는 게이트웨이, FTP 서버의 역할을 담당하는 호스트 2개, 그리고 FTP 클라이언트 역할을 담당하는 호스트 2개로 네트워크를 구성하여, 여러 개의 컨넥션이 네트워크를 공유할 수 있도록 하였다. 물리적 네트워크는 100Mbps Ethernet이고, 게이트웨이에는 Traffic Shaper[13]를 설치하여 병목 라우터의 대역폭을 조정할 수 있도록 하였다.

이 실험 환경에서 게이트웨이의 대역폭이 유일한 bottleneck으로 작용할 수 있도록, 서버와 클라이언트 간의 처리량이 1MBps 이상이 되는 것을 확인하였고, 1MBps 이하의 대역폭에서 모든 실험을 진행하였다. 모든 호스트는 Linux 2.2.10을 OS로 사용하였고 제안한 알고리즘들도 Linux 2.2.10 커널의 TCP 코드를 기반으로 구현하였다[14]. 다양한 조건에서 실험을 수행하였으며, 여기서는 대표적인 결과를 보이는 두 실험조건(표 5)에서의 결과를 논한다. 각 실험에서 혼잡제어알고리즘은 Reno, SD 및 PD를 사용하였다. 그리고 SD와 PD를 동시에 적용한 경우도 실험하였다.

표 6은 두 실험에서의 결과 (처리량, 타임아웃의 수효, triple dupack의 수효 및 RTT 피크 발생빈도)를 보인다. 이

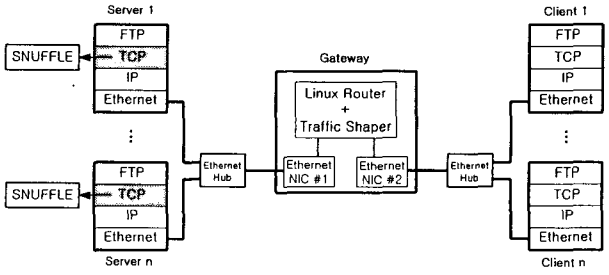


그림 6 실험 네트워크의 구성
Fig. 6 Experimental network

표 5 성능 평가를 위한 실험 조건
Table 5 Experimental conditions for evaluating performance

	네트워크	대역폭 (병목 라우터)	컨넥션
실험 41	그림 6	150KBps	다중접속
실험 42	그림 6	1MBps	다중접속

표 6 측정된 처리량, 타임아웃, triple dupack, RTT 피크
Table 6 Measured throughput, timeout, triple dupack, and RTT peak

	처리량 (상대값)	타임아웃 (상대값)	triple dupack (상대값)	RTT 피크 발생빈도
실험 41 (Reno)	126.5 KBps (100.0)	72.5 (100.0)	670.5 (100.0)	15.8%
실험 41 (SD)	130.9 KBps (103.5)	81.3 (112.1)	874.0 (130.4)	19.7%
실험 41 (PD)	133.1 KBps (105.2)	77.1 (106.3)	589.3 (87.9)	10.5%
실험 41 (SD, PD)	142.7 KBps (112.8)	73.1 (100.8)	617.4 (92.1)	12.1%
실험 42 (Reno)	894.5 KBps (100.0)	9.8 (100.0)	54.2 (100.0)	13.6%
실험 42 (SD)	927.9 KBps (103.7)	14.4 (146.7)	71.0 (131.1)	16.9%
실험 42 (PD)	935.5 KBps (104.6)	9.8 (99.4)	50.2 (92.6)	12.1%
실험 42 (SD, PD)	951.3 KBps (106.3)	8.4 (85.6)	61.0 (112.6)	12.8%

표에서 상대값이라 함은 Reno를 사용한 해당 실험의 결과를 100으로 볼 때 이에 따른 환산값을 나타낸다. 그리고 제시한 결과는 각 조건에서의 실험을 10회 이상 반복하여 측정한 결과의 평균치이다. 여기서 병목 라우터의 대역폭을 표 5와 같이 설정하고 처리량을 측정하였으므로, 측정된 처리량은 대역폭 이용률을 나타낸다.

표 6은 SD와 PD 모두 대역폭 이용률을 향상시키는 효과가 있음을 보인다. 대역폭 이용률은, SD 또는 PD의 경우 각각 3%에서 5%까지 향상되었고, SD와 PD를 함께 적용한 경우에는 6%에서 12%까지 향상되었다. 이러한 결과는 RTT 피크율을 기반으로 한 네트워크 혼잡의 판단 및 이에 따른 SD와 PD의 적용이 타당함을 보이는 것이다.

타임아웃과 triple dupack을 보면, SD의 경우 30% 정도 증가하였다. 이것은 SD가 AI에 비해 공격적으로 cwnd 값을 관리하기 때문에 부분적으로 과부하 상황이 발생하였음을 나타낸다. PD의 경우 타임아웃은 큰 변화가 없지만 triple dupack이 약 10% 감소하는 것을 볼 수 있다. 이것은 PD가 능동적으로 cwnd를 감소시켜 불필요한 재전송을 줄이고 대역폭 이용률을 향상시키고 있음을 나타낸다.

한 가지 재미있는 사실은 SD나 PD를 각각 독립적으로 적용한 경우보다 이들을 함께 적용한 경우 대역폭 이용률이 더 크게 향상된다는 것이다. 그것은 SD와 PD가 갖고 있는 특성에서 기인한다. SD는 RTT 피크가 발생하지 않은 타임아웃이나 triple dupack에 대하여 cwnd를 MD에 비해 적게 줄이기 때문에 MD보다 적극적으로 대역폭을 이용하고자 한다. 반면에 PD는 RTT 피크가 발생하면 타임아웃이나 triple dupack이 일어나기 전에 cwnd를 감소시켜 혼잡을 회피하므로, AI에 비해 사용가능한 대역폭을 충분히 사용하지 않을 수 있다. 표 6의 결과는 상반되는 성격을 갖는 SD와 PD가 함께 적용되면 SD의 공격적 특성과 PD의 방어적 특성이 상호 보완되어 대역폭 이용률이 보다 많이 향상된다는 것을 보인다.

5. 결 론

본 논문은 TCP Reno의 특성을 분석하고, 그 결과를 이용하여 TCP Reno의 성능을 향상시키기 위한 새로운 알고리즘을 제안하였다. 제안한 알고리즘은 타임아웃 기반 및 지연시간 기반의 혼잡제어가 어떻게 효율적으로 결합하여 상호 보완할 수 있는지를 잘 보여 준다.

구체적으로 본 논문에서는 TCP Reno의 AIMD 특성으로부터 두 가지의 타임아웃 패턴의 존재를 유추하고, 실제 인터넷상의 실험을 통해 이러한 패턴을 확인하였다. 주어진 타임아웃이 어떤 패턴에 속하는가 하는 것은 그 타임아웃이 발생한 시점에서 트래픽의 증가량이 얼마나 급격한가에 의해서 결정된다. 본 연구에서는 이러한 트래픽의 증가량을 RTT의 변화 정도를 이용하여 추정할 수 있음을 보였고, RTT의 변화의 정도를 정량화하기 위해 RTT 피크율이라는 파라미터를 도입하였다. 그리고 실제 실험을 통해 RTT 피크율의 값이 2일 때를 경계로 타임아웃 패턴을 구분하였다.

이러한 타임아웃 패턴을 기반으로 본 논문에서는 TCP Reno의 AIMD의 특성을 보완하기 위한 두 가지 알고리즘을 제안하였다. 선택적 감소(SD) 알고리즘에서는 RTT 피크를 동반하지 않는 타임아웃의 경우, 즉 네트워크 트래픽의 증가가 완만할 때에 발생한 타임아웃의 경우, 패킷 전송률을 기존의 MD에 비해 적게 감소시킴으로써 대역폭 이용률을 향상시키고자 한다. 구체적으로 이 경우, SD는 ssthresh를 cwnd의 70%로 설정하고 cwnd를 1로 감소시킨다. 능동적 감소(PD)에서는 AI의 꾸준한 cwnd의 증가에 의해 발생하는 타임아웃과 triple dupack을 억제하기 위해, RTT 피크가 발생하는 경우, 타임아웃이 발생하지 않더라도 cwnd를 20% 감소시킨다.

비록 제한된 실험이지만, 본 연구에서 수행한 실험에 의하면, 제안한 알고리즘들은 TCP Reno에 비해 대역폭 이용률을 각각 3%에서 5%까지 향상시킨다. 이러한 결과는 RTT 피크율을 기반으로 한 네트워크 혼잡의 판단 및 이에 따른 SD와 PD의 적용이 타당함을 보인다. 특히 재미있는 것은 SD나 PD를 각각 독립적으로 적용한 경우보다 이들을 함께 적용한 경우 대역폭 이용률이 더 크게 향상된다는 것이다. 이것은 SD의 공격적 특성과 PD의 방어적 특성이 상호 보완되는 방향으로 작용함을 의미한다. 구체적으로 SD와 PD를 같이 적용한 경우, 대역폭 이용률은 6%에서 12%까지 향상되어, 이들 알고리즘을 독립적으로 사용한 것에 비해 약 2배의 향상을 보이며, 타임아웃의 수효도 상당히 감소한다.

타임아웃 패턴과 트래픽의 관계는 본 논문에서 다룬 요인 외에도 여러 가지 요인의 영향을 받는다. 예를 들면, 네트워크를 공유하고 있는 다른 엔드 호스트들과 병목 라우터 사이의 거리, 병목 라우터의 패킷 관리 정책과 각각의 컨넥션이 경험하게 되는 패킷 드롭 확률, 그리고 여러 개의 병목 라우터가 존재할 때의 타임아웃 등, 실제 인터넷에는 측정하기 어려운 요인들이 매우 많다. 이러한 요인들이 타임아웃 패턴에 어떠한 영향을 미치는가에 대한 연구, 그리고 제안한 알고리즘에 대한 보다 광범위한 성능 평가 및 관련 파라미터들의 최적화에 대한 연구가 앞으로 재미있는 과제가 될 것이다.

Acknowledgement

This study was supported by a grant of the Korea Health 21 R & D Project, Ministry of Health & Welfare, Republic of Korea (02-PJ3-PG6 -EV08-0001).

참 고 문 헌

[1] V. Jacobson, "Congestion avoidance and control," ACM SIGCOMM '84, pp. 314-329, August 1988.

[2] M. Allman, V. Paxon, and W. R. Stevens, TCP Congestion Control, RFC2581, April 1999.

[3] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," ACM SIGCOMM '94, pp. 24-35, August 1994.

[4] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," ACM SIGCOMM '96, pp. 270-280, August 1996.

[5] R. Jain, "A delay based approach for congestion avoidance in interconnected heterogeneous computer networks," ACM Computer Communication Review, vol. 19, issue 5, pp. 56-71, October 1989.

[6] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in 4.3-Tahoe BSD TCP congestion control algorithm," ACM Computer Communication Review, vol. 22, issue 2, pp. 9-16, April 1992.

[7] Z. Wang and J. Crowcroft, "A new congestion control scheme: slow start and search (Tri-S)," ACM Computer Communication Review, vol. 21, issue 1, pp. 32-43, January 1991.

[8] C. Parsa and J. J. Garcia-Luna-Aceves, "Improving TCP congestion control over Internet with heterogeneous transmission media", International Conference on Network Protocols, pp. 213-221, November 1999.

[9] T. V. Lakshman, U. Madhow, and B. Suter, "Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance," INFOCOM '97, vol. 3, pp. 1199-1209, April 1997.

[10] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," INFOCOM '99, vol. 3, pp. 1556-1563, March 1999.

[11] Linux kernel 2.2.10 available from http://ftp.kernel.org/

[12] B. Rathke, T. Assimakopoulos, R. Morich, G. Schulte, and A. Wolisz, "SNUFFLE: integrated measurement and analysis tool for Internet and its use in wireless in-house environment," Computer Performance Evaluation, pp. 340-343, September 1998.

[13] Linux traffic shaper available from http://lwn.net/1998/1119/shaper.html

[14] J. Crowcroft and I. Phillips, TCP/IP and Linux Protocol Implementation: System Code for the Linux Internet, Wiley, 2001.

저 자 소 개

임 갑 주(林 甲 珠)

1975년 10월 6일생. 2001년 한양대학교 기계공학부 졸업. 2003년 한양 대학교 전자통신전파공학과 졸업(석사). 삼성전자 반도체사업부. 본 연구는 한양대학교 대학원 재학시 수행하였음.

Tel : (02) 2299-9798, Fax : (02) 2281-9912

E-mail : gabju.lim@samsung.com

연 창 환(延 昌 桓)

1977년 6월 28일생. 2003년 한양대학교 전자전기공학부 졸업. 2003년~현재 한양대학교 전자통신전파공학과 석사과정

Tel : (02) 2299-9798, Fax : (02) 2281-9912

E-mail : chyoun@csl.hanyang.ac.kr

남 상 원(南 尙 沅)

1957년 2월 24일생. 1981년 서울대학교 전자공학과 졸업. 1990년 Univ. of Texas at Austin 졸업(공박). 1991년~현재 한양대학교 전자전기컴퓨터공학부 교수

Tel : (02) 2281-0588, Fax : (02) 2298-1957

E-mail : swnam@hanyang.ac.kr

이 인 환(李 寅 煥)

1957년 1월 27일생. 1979년 서울대 전기공학과 졸업. 1994년 Univ. of Illinois at Urbana-Champaign 졸업(공박). 1997년~현재 한양대 전자전기컴퓨터공학부 부교수

Tel : (02) 2290-0378, Fax : (02) 2281-9912

E-mail : ihlee@hanyang.ac.kr