초고속-장거리 네트워크에서 혼잡 제어 방안

양 은 호[†] · 함 성 일[††] · 조 성 호[††] · 김 종 권[†††]

요 약

초고속-장거리 네트워크(fast long-distance network)에서 TCP의 혼잡 제어(congestion control) 알고리즘은 대역폭을 효과적 사용하지 못하는 문제점을 가지고 있다. 이 문제를 해결하기 위하여 TCP 혼잡 제어 알고리즘을 수정한 여러 윈도우 기반 혼잡 제어 프로토콜(window-based protocol)이 제안되었다. 이러한 프로토콜들은 주로 확장성(scalability), TCP-친밀성(TCP-friendliness), 그리고 RTT-공평성(RTT-fairness) 등의 세 가지의 요구사항을 고려하고 있다. 하지만 기존에 제안된 프로토콜은 위 세 가지 특성의 균형관계(trade-off)로 인하여 이들 세 특성을 동시에 만족시키지 못한다. 본 논문에서는 EIMD (Exponential Increase/ Multiplicative Decrease)라고 하는 윈도우 기반 TCP 혼잡 제어 알고리즘을 제안한다. EIMD는 위의 세 가지 특성을 동시에 제공함은 물론이고, 초고속-장거리 네트워크에서 중요하게 고려해야 할 빠른 공정배분 수렴성(fair share convergence)도 제공한다. EIMD는 패킷 손실(packet loss)이 없는 한, 지수적으로 윈도우를 증가시켜 큰 대역폭을 효과적으로 사용하면서도, 혼잡제어 알고리즘의 반응 함수(response function)에 RTT를 반영하여 RTT-공평성와 TCP-친밀성을 제공한다. 또한 패킷 손실이 생기기 직전의 혼잡 윈도우 크기에 반비례하게 윈도우를 증가시킴으로써 공정배분(fair share) 값에 빠르게 수렴할 수 있다. 모의실험을 통해 제안된 프로토콜이 초고속-장거리 네트워크에서 위 4가지 특성들을 모두 만족하는지 검증하였다.

키워드 : 혼잡 제어, 초고속-장거리 네트워크, RTT-공평성, 공정배분 수렴, 프로토콜 디자인

Congestion Control Scheme for Wide Area and High-Speed Networks

Yang Eun Ho[†] · Ham Sung Il[††] · Cho Seongho[††] · Kim Chongkwon[†††]

ABSTRACT

In fast long-distance networks, TCP's congestion control algorithm has the problem of utilizing bandwidth effectively. Several window-based congestion control protocols for high-speed and large delay networks have been proposed to solve this problem. These protocols deliberate mainly three properties : scalability, TCP-friendliness, and RTT-fairness. These protocols, however, cannot satisfy above three properties at the same time because of the trade-off among them. This paper presents a new window-based congestion control algorithm, called EIMD (Exponential Increase/ Multiplicative Decrease), that simultaneously supports all four properties including fast convergence, which is another important constraint for fast long-distance networks; it can support scalability by increasing congestion window exponentially proportional to the time elapsed since a packet loss; it can support RTT-fairness and TCP-friendliness by considering RTT in its response function; it can support fast fair-share convergence by increasing congestion window inversely proportional to the congestion window just before packet loss. We evaluate the performance of EIMD and other algorithms by extensive computer simulations.

Key Words : Congestion Control, Fast Long-Distance Networks, RTT-fairness, Fair Share Convergence, Protocol Design

1. 서 론

네트워크가 점점 고속화되는 동시에 세계화되면서 초고속-장거리 네트워크에 대한 연구가 점차 주목을 받고 있다. 초고속-장거리 네트워크란 멀리 떨어져 있는 고성능 컴퓨터들을 초고속망으로 연결한 것으로 대역폭(bandwidth)뿐만 아니라 RTT(Round Trip Time)도 매우 큰 네트워크를 의미한다. 대표적인 초고속-장거리 네트워크로는 Grid Network나 Abilene, ESNet등이 있다. 이러한 네트워크는 High Energy Physics나 의료 영상처리(Medical Image Processing)과 같이 고성능 연산 능력뿐만 아니라 대용량의 파일 전송을 필요로 하는 응용분야의 출연으로 그 관심과 중요도가 더욱 커지고 있다.

이런 네트워크에서는 현재 지배적인 전송 계층 프로토콜인 기존의 TCP(Transmission Control Protocol)로 인한 성능 저하가 발생하게 된다. S. Floyd [1, 2]는 기존 TCP의 혼잡 제어 (congestion control) 알고리즘[3]이 패킷 손실이 일어나지 않은 상황에서는 1RTT 동안에 윈도우 크기를 1씩만 증가시키는 반면, 혼잡이 발생한 상황에서는 윈도우 크기를 절반으

† 준 회 원 : 서울대학교 컴퓨터공학부 석사과정
†† 준 회 원 : 서울대학교 컴퓨터공학부 박사과정
††† 정 회 원 : 서울대학교 컴퓨터공학부 교수
논문접수 : 2004년 12월 3일, 심사완료 : 2005년 5월 25일

로 급격히 줄이기 때문에 초고속-장거리 네트워크 상에서는 대역폭을 효과적으로 사용하지 못함을 지적하였다. 예를 들어, 패킷 (packet)의 크기가 1500 byte이고 RTT가 100ms인 TCP 플로우가 있다고 했을 때 10Gbps의 처리율을 얻기 위해서는 평균 윈도우 크기가 83,333이 되어야 한다. 이런 큰 평균 윈도우 크기를 유지하기 위해서는 대략 1시간 40분마다 하나 정도의 패킷만이 손실 되어야 하는데 이를 비트 에러율(Bit Error Rate)로 환산하면 대략 $2*10^{-14}$가 된다. 하지만 이 수치는 네트워크의 물리적인 비트 에러율 한계를 넘어서기 때문에 기존 TCP를 이용해서는 병목 링크(bottleneck link)의 대역폭이 아무리 크다고 해도 이를 충분히 활용할 수 없다[1, 2]. 따라서 현재 기존 TCP가 여전히 지배적인 전송계층 프로토콜임에도 불구하고, 초고속-장거리 네트워크에서는 새로운 초고속 프로토콜 (high-speed protocol)이 필요하다. 초고속-장거리 네트워크를 위한 새로운 프로토콜이 만족해야 하는 특성은 다음과 같다. [1, 4, 5]

- -확장성(scalability) : 새로운 프로토콜은 패킷 손실률(loss event rate)가 낮을 때, 전체 대역폭을 효과적으로 사용할 수 있어야 한다. HSTCP [2]와 같은 기존의 프로토콜은 패킷 손실률이 10^{-7}일 때, RTT가 100ms인 플로우의 처리율이 10Gbps가 되는 것을 목표로 하고 있다.
- -TCP-친밀성(TCP-friendliness) : 초고속 플로우와 공존하는 기존 TCP 플로우가 가능한 많은 대역폭을 할당 받아야 한다. TCP-친밀성은 대역폭이 작은, 즉, 패킷의 손실률이 높은 환경에서 더욱 중요하다.
- -RTT-공평성(RTT-fairness) : 경쟁하는 플로우들이 RTT 값에 따라서 불공평하게 대역폭을 할당 받지 않아야 한다. RTT에 따라서 처리율 차이가 생기는 주 원인은 동기 손실(synchronized loss) 때문이다. 패킷 손실이 균등하게 분포하는 네트워크에서는 모든 플로우가 RTT에 무관하게 패킷 손실률을 겪게 되고, 결과적으로 같은 크기의 혼잡 윈도우를 갖게 된다. 하지만 대역폭이 커져서 동시 손실 발생하게 되면, RTT에 비례하여 패킷 손실률이 증가하게 된다. 결과적으로 RTT가 큰 플로우가 상대적으로 작은 크기의 혼잡 윈도우를 가지게 되고, 처리율의 차이가 발생하게 되는 것이다. 기존의 TCP의 경우도 RTT에 따라 심각한 처리율 차이를 겪게 된다. 예를 들어, 병목 링크의 대역폭이 2.5Gbps이고 RTT가 6배 차이가 나면, RTT가 큰 플로우의 경우는 전체 중 오직 67Mbps의 처리율을 얻게 된다. 이것은 TCP의 처리율의 비가 $1/RTT^2$이기 때문이다 [21].
- -공정배분 수렴(Fair share convergence) : 새로 시작하는 플로우가 자신의 공정 배분 값에 빠르게 수렴할 수 있어야 한다. 초고속-장거리 네트워크의 경우는 i)각 플로우의 공정배분 대역폭 값이 크고 ii)RTT가 커서 혼잡 윈도우의 증가 속도가 느리므로 빠르게 자신의 공정배분 값에 수렴하는 것이 중요하다.

이러한 특성을 제공하기 위하여 TCP 튜닝 [17, 18, 19]에서부터, 윈도우 기반(window-based) 프로토콜 [2, 4, 5], 지연 기반(delay-based) 프로토콜 [6], 그리고 단순히 TCP 혼잡제어 알고리즘 수정의 범위를 넘어서는 새로운 프로토콜 [7, 22]까지 다양한 기법이 제안되었다. 이러한 프로토콜 중에서 우리는 윈도우 기반 혼잡 제어 프로토콜에 초점을 맞추도록 한다. 윈도우 기반 프로토콜은 라우터의 수정을 필요로 하지 않고, 점차적 배치(incremental deployment)에 유리하다는 장점이 있기 때문이다 [9]. 기존의 윈도우 기반 프로토콜의 경우는 윈도우 증가량을 현재의 윈도우 크기를 바탕으로 매 RTT마다 새롭게 계산한다. 즉, 현재 혼잡 윈도우의 크기가 큰 경우에는 윈도우 증가량을 크게 하여 대역폭 이용효율(bandwidth utilization)을 좋게 하고 반대로 윈도우가 작을 때는 상대적으로 윈도우 증가량을 작게 하여 네트워크의 혼잡을 경감시키고 보다 작은 RTT를 가지는 기존 TCP 플로우(현재 인터넷 플로우의 대다수)에 주는 피해를 최소화 한다.

하지만 HSTCP [2], Scalable TCP [4] 및 BIC [5]와 같은 기존의 윈도우 기반 프로토콜은 확장성, TCP-친밀성과 RTT-공평성을 동시에 만족시키지 못하는 문제점을 가지고 있다. 확장성과 TCP-친밀성을 동시에 만족시키기 위한 메커니즘이 RTT-공평성을 심각하게 손상시키기 때문이다. 이 세 가지 요구 사항을 동시에 만족시키지 못하는 것 외에도, 기존의 윈도우 기반 프로토콜은 공정배분 값에 수렴하는데 시간이 오래 걸리는 문제점을 가지고 있다. 이는 현재 윈도우 크기에 비례하여 윈도우 증가시키기 때문이다.

본 논문에서는 EIMD(Exponential Increase/ Multiplicative Decrease)라는 초고속-장거리 네트워크를 위한 새로운 윈도우 기반 프로토콜을 제안한다. EIMD는 현재의 윈도우에 비례하여 윈도우를 증가시키는 대신에, 패킷 손실 이후에 윈도우가 증가하기 시작한 시간에 지수적으로 비례하게 윈도우를 증가시킴으로써 확장성을 제공한다. 패킷 손실 이후에 경과된 시간 이외에 윈도우를 증가시키는 두 개의 요소가 더 존재한다. i) 반응함수에 RTT를 반영하여 RTT-공평성과 TCP-친밀성을 제공하고 ii) 패킷 손실 직후의 윈도우 크기, w_{max}에 반비례하게 윈도우를 증가시킴으로써 공정배분 값에 빠르게 수렴할 수 있도록 한다. EIMD는 패킷 손실률에 상관 없이 네 가지 요구사항을 동시에 만족시킨다.

본 논문의 구성은 다음과 같다. 우선 2장에서는 관련 연구에 대해서 설명하고 3장에서는 초고속 TCP 프로토콜의 특성에 대해서 살펴본다. 4장에서는 EIMD의 반응함수와 이에 맞는 윈도우 혼잡제어 알고리즘의 윈도우 증가/감소 규칙(increase/decrease rule)을 설명하고 5장에서는 모의실험을 통해 EIMD가 위에서 언급한 4가지의 특성들이 만족함을 보인다. 마지막으로 6장에서 본 논문의 결론을 맺는다.

2. 관련 연구

초고속 네트워크 네트워크를 위한 전통적인 해결책은 TCP 조율(TCP tuning)을 기본으로 하고 있다. PSocket [17], GridFTP [18]와 combined Parallel TCP [19] 등과 같이 응용

계층에서 병렬 TCP(parallel TCP)를 이용하는 방법이 대표적인 예이다. 이러한 방법은 여러 개의 병렬 TCP 연결을 이용하여 높은 처리율을 얻는다. 하지만 광범위한 조율을 필요로 하고, 네트워크의 동적인 특성으로 인하여, 대역폭을 효과적으로 사용하면서도 혼잡상황을 야기시키지 않을 수 있는 적절한 TCP 연결의 수를 사전에 정하는 것이 어렵다는 단점이 존재한다 [20].

윈도우 기반 프로토콜은 TCP의 반응함수를 수정함으로써 성능을 향상시키는 방법을 사용한다. HSTCP는 목표로 하는 반응 함수를 따르도록 윈도우 증가 및 감소 파라미터를 선택한다. Scalable TCP는 AIMD 대신에 MIMD(Multiplicative Increase/Multiplicative Decrease) 방법을 사용하고 있다. BIC는 적절한 윈도우 크기를 정하는 것을 탐색(searching) 문제로 보고 선형탐색(linear search) 대신에 이진탐색(binary search)를 사용하였다. 각 프로토콜의 반응 함수와 자세한 특성은 다음 장에서 살펴보도록 한다.

LTCP [16]는 주로 계층적 비디오 전송(layered video transmission) 사용되는 계층 개념을 도입하였다. LTCP는 2차원의 혼잡 제어 알고리즘을 갖는다. 거시적인 수준에서는 네트워크 상태에 따라서 계층의 개수를 조절하고, 미시적인 수준에서는 각 계층에서 동작하는 혼잡 윈도우를 통제하는 방법을 정의하였다. FAST TCP [6]는 TCP-Vegas의 초고속 네트워크 버전으로 식에 기반하여 윈도우를 조절한다. FAST TCP는 RTT로부터 대기 지연(Queueing delay)를 예측하고 이를 기반으로 전송률을 정하게 된다. XCP [7]는 ECN (Explicit Congestion Notification)을 일반화하여 초고속-장거리 네트워크에 맞도록 수정 하였다. XCP는 좋은 효율(utilization)과 수렴 특성에도 불구하고 중간 라우터들의 수정을 필요로 한다. SABUL [22]과 같은 프로토콜은 오직 제어 정보만 TCP를 이용하여 전송하고 신뢰적 데이터 전송은 UDP를 이용하는 방식을 사용한다.

3. 초고속 TCP 혼잡제어 알고리즘의 특성

확장성, TCP-친밀성 및 RTT-공평성 : TCP 혼잡제어 알고리즘의 반응 함수는 손실률 p에 따른 혼잡 윈도우(congestion window)의 크기(또는 전송률)를 의미한다. 기존 초고속 프로토콜의 반응함수는 식 (1)로 근사화할 수 있다.

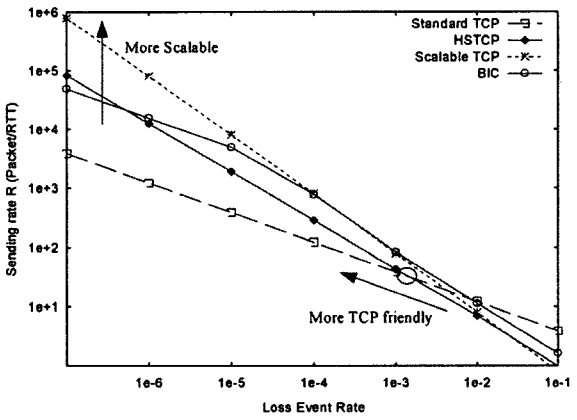
$$w = A / p^{s} \tag{1}$$

여기서, w는 평균 윈도우의 크기이고 A와 s는 프로토콜 종속적인 상수 값이다.

반응함수를 통하여 확장성, TCP-친밀성 그리고 RTT-공평성와 같은 프로토콜의 기본적인 특성을 알 수 있다. (그림 1)은 TCP와 여러 초고속 프로토콜의 반응함수를 로그-로그 스케일로 보인 것이다. 손실률이 낮을수록 네트워크의 대역폭이 크다는 것을 의미하므로, 전송률이 높아야만 가용 대역폭(available bandwidth)을 효율적으로 사용할 수 있다. 따라서 손실률이 낮은 구간에서 높은 전송률을 가질수록 프로토콜이 확장성이 가장 뛰어나다고 할 수 있다. 그림에서 볼 수 있듯이, Scalable TCP가 확장성이 가장 뛰어나고 HSTCP와 BIC가 그 뒤를 따른다.

하지만 손실률이 높은 환경에서는 기존 네트워크와 유사한 환경이므로 기존 TCP의 처리율을 저하시키지 않아야 한다. 이를 위하여 초고속 프로토콜은 TCP의 반응함수와의 교점보다 손실률이 더 높은 구간에서는 기존 TCP의 혼잡 제어 알고리즘을 그대로 사용하게 된다. 그러므로 즉, 두 그래프가 만나는 교차점이 왼쪽으로 이동할수록 보다 TCP-친밀성이 높은 프로토콜이라 할 수 있다. HSTCP가 가장 TCP-친밀성이 높고 Scalable TCP, BIC가 그 뒤를 따른다.

RTT-공평성은 반응함수의 기울기에 의해서 표현될 수 있다. 동시 손실 모델(synchronized loss model)을 가정했을 때 두 플로우의 RTT를 각각 RTT_1, RTT_2라고 한다면, 처리율의 비는 $(RTT_1 / RTT_2)^{1/(1-s)}$이 된다. 이 때 $-s$는 로그-로그 스케일에서 반응함수 그래프의 기울기에 해당하며 절대값이 작을수록 RTT-공평성이 높아진다. 기존 TCP, HSTCP 그리고 Scalable TCP의 s 값은 각각 0.5 0.82, 1이 되고, BIC의 s 값은 손실률이 감소함에 따라 1에서 0.5로 감소한다. 즉, 손실률이 낮은 환경에서는 BIC가 가장 RTT-공평성이 높고 HSTCP와 Scalable TCP가 그 뒤를 따르는 반면에, 손실률이 높은 환경에서는 HSTCP가 가장 RTT-공평성이 뛰어나다. 예를 들어, RTT의 비율이 1/2인 두 플로우의 처리율 비는 각각 BIC는 4~∞, HSTCP는 43, Scalable TCP는 ∞가 된다.

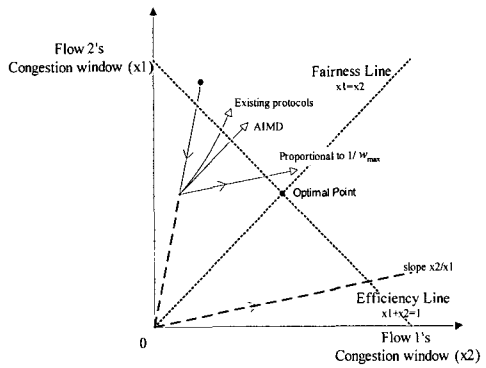


(그림 1) 기존 TCP와 high-speed 프로토콜의 반응 함수

위의 관점에서 볼 때, 기존의 반응함수 형태로는 확장성, TCP-친밀성 및 RTT-공평성을 동시에 만족시킬 수 없다. 다시 말하면, 기존의 반응함수는 셋 중 하나의 특성을 손상시키는 대가로 나머지 두 특성을 만족시키게 된다. 반응함수가 $a(p_1, w_1)$와 $b(p_2, w_2)$ $(p_1 >> p_2, w_2 >> w_1)$ 두 점을 지나는 프로토콜을 가정해보자. TCP-친밀성을 위해서는 w_1을 줄여야 하는 반면에, 확장성을 위해서는 w_2를 증가시켜야만 한다. 하

지만 w_1을 줄이거나 w_2를 증가시키는 것은 식(1)과 같은 어떠한 반응함수를 사용하더라도 RTT-공평성을 해치게 된다. 이는 코쉬의 중앙값 정리에 의해 패킷 손실률이 $(\log(w_2)-\log(w_1))/(\log(p_1)-\log(p_2))$가 되는 지점이 반드시 존재하게 되어 기울기가 증가하기 때문이다. 예를 들어, HSTCP처럼 a = (0.0015, 31)와 b = (10^{-7}, 83333)가 된다면, S =0.82가 되는 지점이 반드시 존재하게 된다.

공정배분 수렴: (그림 2)는 Chiu & Jain의 모델 [15]에 기반한 혼잡 윈도우의 궤적을 그린 것이다. 최적지점에 수렴하기 위해서는 최악의 경우에 AIMD와 같게 증가해야 한다. 하지만 기존의 프로토콜들은 현재 윈도우의 값에 따라서 AIMD와 같거나 최적 지점에서 멀어지는 방향으로 증가하게 된다. 이는 현재 윈도의 크기에 비례하여 윈도우를 증가량을 결정하기 때문이다. 최적지점에 가장 빠르게 수렴하는 방법은 증가량을 $1/w_{max}$에 비례하게 선택하는 것이다 [10].



(그림 2) 혼잡 윈도우의 증가 궤적

4. Exponential Increase/Multiple Decrease (EIMD)

이번 장에서는 확장성, TCP-친밀성, RTT-공평성 및 빠른 공정배분 수렴을 모두 만족시키는 Exponential Increase/Multiple Decrease (EIMD)를 제안한다. 우선 프로토콜의 목표와 EIMD의 반응함수를 살펴보고, 혼잡 제어 규칙에 대해서 설명하도록 한다. 마지막으로 실제적 구현 이슈(implementation issue)에 대해서 토론해 본다. 본 논문에서는 실제 초고속-장거리 네트워크에 부합되는 동시손실 모델을 가정한다. 또한 초고속 프로토콜을 사용하는 플로우와 기존 TCP 플로우가 공존하는 상황에서 개조되지 않은 기존의 TCP 플로우의 RTT는 초고속 플로우의 RTT보다 상대적으로 작다는 것을 가정한다 [5, 21].

4.1 설계 목표

EIMD는 다음과 같은 특성을 만족시킨다.

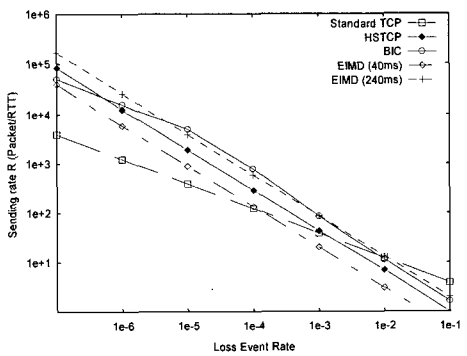
- **RTT-공평성**: 처리율은 RTT^{-k}에 비례한다. 따라서 k 값을 조절함으로써 EIMD는 네트워크 상황에 맞는 RTT-fairness를 제공할 수 있다. 본 논문에서는 k를 1로 하여 RTT에 반비례하는 처리율을 갖는다. 이는 RTT에 무관하게 모든 플로우가 같은 크기의 혼잡 윈도우를 갖게 됨을 의미한다.
- **확장성**: RTT가 100ms인 플로우가 10^{-7}의 손실률에서 10Gbps의 처리율을 얻는다. 이 때 다른 RTT를 갖는 플로우들은 RTT가 100ms인 플로우를 기준으로 RTT에 반비례하는 처리율을 갖는다.
- **TCP-친밀성**: EIMD는 패킷 손실률에 관계없이 기존 high-speed 프로토콜보다 보다 TCP-친밀성이 뛰어나다.
- **공정배분 수렴**: 기존 프로토콜보다도 각 플로우가 자신의 공정배분 값에 빠르게 수렴한다.

4.2 EIMD의 반응 함수

3절에서 언급한대로 기존의 반응함수의 형태로는 확장성과 TCP-친밀성을 만족시키면서 RTT-공평성을 동시에 만족시킬 수 없다. 이 특성들 사이의 균형관계(trade-off)를 해결하기 위하여 식 (2)와 같은 RTT를 고려한 새로운 반응함수를 제안한다

$$w_i = \frac{A}{p^s} RTT^r \tag{2}$$

여기서 r은 RTT-공평성의 정도를 결정하는 파라미터이다. 식 (2)와 기존의 반응 함수의 차이점은 다른 RTT를 가지는 플로우는 다른 반응 함수를 따른다는 점이다. 이는 동시 손실로 인한 RTT가 큰 플로우가 높은 패킷 손실률을 받게 되는 현상을 보상하기 위한 것이다. A가 1.75이고 r과 s가 0.82 일 때, 40ms, 240ms인 두 플로우의 반응함수는 (그림 3)과 같다.[1] 기존의 반응함수는 RTT를 고려하지 않는 반면, EIMD는 RTT를 고려하였기 때문에 반응함수 자체는 EIMD의 특성을 정확히 보여주지 못한다. 따라서 EIMD의 특성을 보다 정확하게 파악하기 위하여 RTT가 반영된 처리율을 직접 비교해야 한다.



(그림 3) EIMD의 반응함수 vs. 기존 high-speed 프로토콜의 반응함수

1) 그림 3은 RTT가 40ms와 240ms인 경우의 반응 함수만 그린 것이다. 다른 RTT에 대한반응함수 이 두 그래프를 기준으로 하여 균일하게 분포한다.

동시손실 모델에서 r에 따른 EIMD의 RTT-공평성을 분석해보면 다음과 같다. w_i와 RTT_i를 각각 플로우 i의 혼잡 윈도우 크기와 RTT 값을 나타낸다고 하고 t는 평형 상태(steady-state)에서 두 개의 연속적인 패킷 손실 사이의 시간(초(second) 단위)이라고 하자. 여기서 t는 동시 손실을 가정하였으므로, 모든 플로우에 대해서 일정하다. 플로우 i의 손실이 p_i의 확률로 균일하게 분포되어있다고 하면 두 개의 연속적인 패킷 손실 동안 보낸 총 패킷의 수는 $1/p_i$가 된다. 이때 총 필요한 RTT의 개수가 t/RTT_i번 이므로 다음 식을 얻을 수 있다.

$$w_i = \frac{1/p_i}{t/RTT_i} = RTT_i/(t \cdot p_i) \tag{3}$$

식(2), (3)를 이용하면,

$$w_i = \frac{RTT_i^{1-r/s}}{t \cdot \sqrt[s]{A/w_i}} \Rightarrow w_i = \left(\frac{RTT_i^{(s-r)s}}{t \cdot \sqrt[s]{A}}\right)^{s/(s-1)}$$

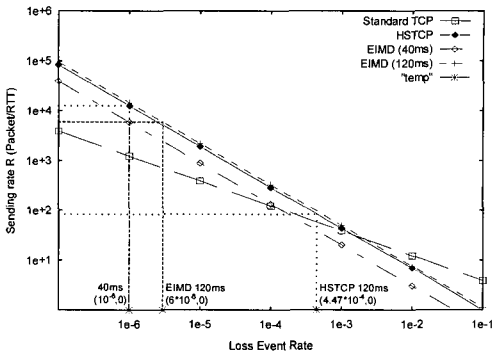
임을 알 수 있다. 그러므로 두 플로우의 처리율의 비는

$$\frac{(w_1/RTT_1)/(1-p_1)}{(w_2/RTT_2)/(1-p_2)} \approx \frac{w_1/RTT_1}{w_2/RTT_2} = \left(\frac{RTT_2}{RTT_1}\right)^{\frac{1-r}{1-s}} \tag{4}$$

이 된다. 이 때 패킷 손실률이 거의 0에 가깝기 때문에 $(1-p_i)$는 근사적으로 1이라고 할 수 있다. 또한 손실률과 손실률의 비는 식(2), (3)으로부터 다음과 같이 계산된다.

$$p_i = \left(\frac{RTT^{1-r}}{A \cdot t}\right)^{1/(1-s)}, \quad \frac{p_1}{p_2} = \left(\frac{RTT_1}{RTT_2}\right)^{(1-r)/(1-s)}$$

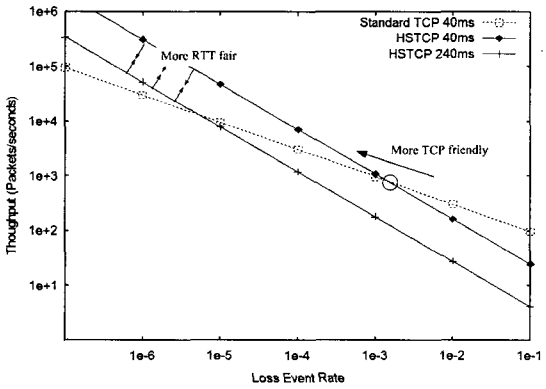
패킷 손실이 균등하게 일어나는 환경에서는 RTT에 무관하게 같은 패킷 손실률을 경험하는데 비해서 초고속 장거리 네트워크에서는 동시손실로 인하여 패킷 손실률이 RTT가 클수록 더 큰 패킷 손실률을 겪게 된다. EIMD는 식 (2)에서 보는 것처럼 RTT에 따라서 다른 반응 함수를 따르도록 함으로써 이러한 문제를 해결한다. 만약 r과 s가 같은 값을 갖게 되면, 윈도우의 크기는 RTT에 무관해지고, 손실률은 RTT에 비례하게 된다. (그림 4)는 40ms의 RTT를 갖는 플로우의 손실률이 10^{-6}일 때, 120ms의 RTT를 갖는 EIMD와 HSTCP 플로우의 혼잡 윈도우의 크기를 나타내고 있다. EIMD 플로우는 같은 크기의 혼잡 윈도우를 갖는 반면에 HSTCP는 손실률의 차이로 인하여 불공정한 혼잡 윈도우를 갖게 된다 (다른 high-speed 프로토콜의 경우도 HSTCP와 크게 다르지 않다).



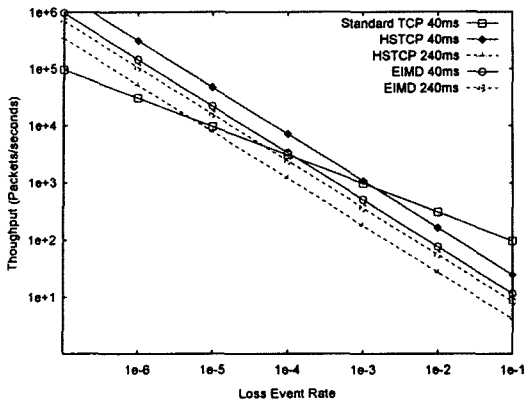
(그림 4) 다른 RTT를 갖는 플로우의 손실률

다음은 EIMD의 TCP-친밀성에 대해서 살펴보자. (그림 5)는 RTT가 각각 40ms와 240ms인 HSTCP의 처리율을 보여주고 있다. 프로토콜의 TCP-친밀성은 기존 TCP 직선과 HSTCP의 두 직선이 만나는 점에 의해서 결정된다. 하지만 그림에서 볼 수 있는 것처럼, 상대적으로 작은 RTT를 가지는 HSTCP 플로우가 TCP-친밀성을 해치는 주요 원인이 된다. 그러므로 상대적으로 작은 RTT를 갖는 플로우를 덜 공격적 (aggressive)으로 만드는 것은 RTT-공평성 뿐만 아니라 TCP-친밀성도 좋게 할 수 있다. 즉, RTT-공평성과 TCP-친밀성을 동시에 증진시키기 위해서는 서로 다른 RTT를 가지는 플로우들의 처리율 간의 간격을 줄여야 한다. (그림 6)에서 EIMD가 HSTCP에 비해서 처리율 간의 차이가 더 좁다는 것을 보여준다. 즉, EIMD는 RTT-공평성를 해결하면서도 HSTCP보다 TCP-친밀성을 향상시켰다.

마지막으로 EIMD의 확장성에 대해서 살펴보자. 위에서 언급한 것처럼, RTT가 상대적으로작은 플로우가 낮은 패킷 손실률을 겪게 된다. 그 결과 높은 확장성을 가지게 되고, RTT-형평성을 손상시키게 된다. 이러한 문제를 해결하기 위해서 EIMD는 다른 RTT를 가지는 플로우들 사이의 처리율 차이를 줄여주었다. 그럼에도 불구하고, 100ms RTT를 갖는 EIMD 플로우는 같은 RTT를 같는 HSTCP 플로우와 같은 처리율을 갖고, (그림 6)에서 볼 수 있는 것처럼 100ms 이하의 RTT를 갖는 플로우도 10^{-7} 구간에서 10Gbps이상의 처리율을 얻을 수 있다.



(그림 5) HSTCP의 처리율



(그림 6) EIMD와 HSTCP의 처리율

4.3 EIMD의 증가/감소 규칙(Increase/Decrease Rule)

4.2절의 반응 함수로부터 EIMD의 증가/감소 규칙을 유도해보자. 1장에서 언급한 것처럼 EIMD는 패킷 손실이 일어난 이후에 경과한 시간에 지수적으로 비례하여 혼잡 윈도우를 증가시킨다. 이를 식으로 표현하면

$$w(T) = w(0) + cT^u \tag{5}$$

이 된다. 여기서 $w(T)$는 윈도우가 증가하기 시작한 경과 시간 T(RTT 단위)에서의 혼잡 윈도우의 크기를 연속적으로 근사한 값이다. w_T를 시간 T에서의 혼잡 윈도우의 크기라고 하고, $\alpha = u \cdot c^{1/u}$, $w_0 = w(0)$ 할 때, 1RTT 동안의 EIMD의 증가 규칙을 $w_{T+1} \leftarrow w_T + \alpha(w_T - w_0)^{1-1/u}$와 같이 정의하면, $\frac{dw(T)}{dT} = \alpha(w(T) - w_0)^{1-1/u}$를 얻을 수 있고, $(1/(w(T) - w_0)^{1-1/u}) \cdot dw(T) = \alpha \cdot dT$ 로부터 양변을 적분하면 근사적으로 식(5)와 같이 혼잡 윈도우가 증가함을 알 수 있다. EIMD는 패킷 손실이 일어난 직후에는 AIMD와 비슷하게 혼잡 윈도우를 $w_{\max}(1-\beta)$로 줄여준다. EIMD의 증가/감소 규칙을 요약하면 다음과 같다.

증가 규칙(1RTT 당) : $w_{T+1} \leftarrow w_T + \alpha(w_T - w_0)^{1-1/u}$

감소 규칙 : $w_{T+\delta} \leftarrow w_T - \beta w_T$

여기서 $\beta = 0.125$이고, δ는 패킷 손실을 감지하는데 걸리는 시간이다.

평형 상태(steady-state)의 처리율은 c와 u값에 의해 정해진다는 사실을 바탕으로 EIMD가 앞에서 제안한 반응 함수를 따르도록 c와 u값을 정하여 보자. [11]에서 제안한 방법과 비슷한 방법으로 c(사실은 증가 규칙의 α 값) 값과 u값을 $w_{\max}$ 및 RTT에 관한 함수로 만들어보자. 평형 상태에서는 윈도우의 증가량과 감소량이 똑같게 되므로 $cT^u = \beta w_{\max}$가 성립한다. 이 때 T 기간 동안 전송된 패킷의 개수는 식(5)로부터 $\int_0^T ((w_{\max} - \beta w_{\max}) + cT^u)dT$ 와 같이 계산될 수 있다. 그러므로 우리는 다음과 같은 식을 얻을 수 있다.

$$T = \left(\frac{\beta w_{\max}}{c}\right)^{1/u} \tag{6}$$

$$Total\ number\ of\ packets = \frac{1}{p} = (w_{\max} - \beta w_{\max})T + \frac{c}{u+1}T^{u+1} \tag{7}$$

식 (3)과 $T = t/RTT$로부터 평균 윈도우 $w = 1/(p \cdot T)$이 됨을 알 수 있다. 이 값이 앞에서 정의한 반응함수 $\frac{A}{p^s}RTT^r$와 같아야 하므로 다음과 같은 식을 얻을 수 있다.

$$\frac{1}{p \cdot T} = \frac{A}{p^s}RTT^r \tag{8}$$

식(7), (8)를 이용하여 c와 T의 관계를 얻을 수 있다.

$$w_{\max} - \beta w_{\max} + \frac{c}{u+1}T^u = (A \cdot RTT^r)^{1/(1-s)}T^{s/(1-s)} \tag{9}$$

(6), (9)번 식을 이용하여 c값과 그에 따른 α값을 구할 수 있다.

$$c = \left(\frac{(A \cdot RTT^r)^{1/(1-s)}}{1-(u/(u+1))\beta}\right)^{u(1-s)/s} \cdot \frac{\beta}{\Gamma(u+1)} w_{\max}^{1-u(1-s)/s} \tag{10}$$

$$\alpha = u\left(\frac{(A \cdot RTT^r)^{1/(1-s)}}{1-(u/(u+1))\beta}\right)^{(1-s)/s} \cdot \left(\frac{\beta}{\Gamma(u+1)}\right)^{1/u} w_{\max}^{\frac{1}{u}\frac{1-s}{s}} \tag{11}$$

이 된다.[2)]

식 (10)으로부터, 윈도우의 크기가 $w_{\max}^{1-u(1-s)/s}$에 비례함을 알 수 있다. u가 $1-u(1-s)/s = -1$를 만족하도록 함으로써 (즉, $u = 2s/(1-s)$), $c \propto 1/w_{\max}$이 성립하도록 하였다. 즉 혼잡 윈도우의 증가가 $1/w_{\max}$에 비례하도록 u를 선택하였다. 이로 인해 EIMD는 빠른 공정분배 특성을 갖는다. r, s 그리고 A 값을 (그림 3)과 같이 선택하면, EIMD는 성공적인 각 패킷의 전송에 대해서 다음과 같은 증가 규칙을 갖게 된다.

2) α 지나치게 커지거나 작지지 않도록 식 9에 들어가는 RTT에 최대 및 최소 임계값(threshold)을 적용할 수 있다

증가 규칙 : $w_{new} \leftarrow w_{old} + \alpha(w_{old} - w_0)^{1-1/u} / w_{old}$, $\alpha = 1.8RTT / w_{\max}^{0.11}$, $u = 9.11$ (12)

4.4 구현 이슈

EIMD 알고리즘을 실제로 구현하기 위해서는 송신측에서 기존 TCP를 식 (10)과 같이 증가 규칙만을 변경하면 된다. 하지만 EIMD의 증가 규칙은 w_t와 w_0의 차이를 이용하여 계산하기 때문에, 패킷 손실 후 첫 번째 ACK에 대한 윈도우 증가량이 0이 된다(∵ 현재 윈도우 크기 $w_t = w_0$).

이러한 문제를 해결하기 위하여 식 (12) 대신에 윈도우를 증가시킬 수 있는 다른 증가 규칙을 제안한다.

Increase : $w_{new} \leftarrow w_{old} + \alpha'(w_{old} - w_0')^{1-1/u} / w_{old}$, $w_0' = w_0 - 1$ (13)

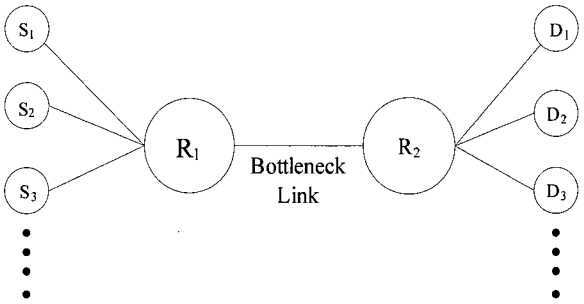
이런 증가 규칙을 만족시키는 α'은 다음과 같이 계산된다.

$$\alpha' = u\left(\frac{(A \cdot RTT^r)^{1/(1-s)}}{1-\left(\frac{(\beta w_{\max})^{1/u}}{(u+1)((\beta w_{\max})^{1/u}-1)}-1\right)\beta}\right)^{(1-s)/s}((\beta w_{\max})^{1/u}-1)w_{\max}^{\frac{1-s}{s}} \quad (14)$$

Appendix 에서, 윈도우가 클 때 위의 α' 값을 적용한 EIMD가 식 (2)과 같은 반응함수를 따른다는 것을 증명한다. 만약 $w_{\max}$가 충분히 커서 $(\beta w_{\max})^{1/u} - 1$이 $(\beta w_{\max})^{1/u}$에 근사 되면, 식 (11)과 식 (14)은 정확히 일치한다. 보다 간단한 구현을 위해서 다음 장의 모의실험에서는 식 (13)의 증가 규칙과 α를 사용하도록 한다.

5. 모의 실험

EIMD의 성능을 평가하기 위하여 (그림 7)과 같은 덤벨 토폴로지 (dumbbell topology)에서 NS-2 시뮬레이터를 이용하여 HSTCP, BIC와 비교 실험하였다.[3] 일반적인 인터넷에서의 병목 링크와는 다르게 초고속-장거리 환경에서는 초고속 링크가 만나는 곳에서 주로 병목 현상이 생기므로 덤벨 토폴로지가 실제 초고속-장거리 네트워크와 흡사하다고 할 수 있다. 병목 링크의 큐의 상황을 보다 정확하게 고려하기 위하여 R_1과 R_2사이의 링크만 오직 병목 링크이고, 이외의 다른 링크는 무한대의 대역폭을 가지고 있다고 가정하였다. 각 플로우들은 (S_i,D_i)를 소스(source)와 목적지(destination)으로 하여 병목 링크를 지나가게 된다. 이 때, S_i와 R_1과의 지연(delay)을 각기 다르게 하여 플로우의 RTT값을 조절하였다. 위상 효과 (Phase effect) [12]를 줄이기 위해서, 각 플로우들은 임의로 시작하고 종료하도록 하였다. 각 실험에서는 EIMD, BIC 그리고 HSTCP를 사용하는 초고속 플로우(high-speed flow)와 수정되지 않은 TCP를 사용하는 TCP 플로우, 마지막으로 배경 트래픽(Background traffic)으로 혼잡 윈도우의 최대 크기가 64이하로 제한된 small TCP와 웹 트래픽(web traffic) 등의 세 가지 타입의 플로우를 사용하였다 [13, 14]. 아래의 각 실험에서 RTT-공평성, TCP-친밀성 및 대역폭 이용효율, 공정 배분 속도에 대해서 살펴보았다. 모든 실험에서 $r = s = 0.82$, $A = 1.75$ 값을 사용하였다. HSTCP와 BIC의 파라미터들은 기본값을 사용하였다. 각각의 실험은 5번의 결과를 평균한 것이다.



(그림 7) 모의 실험 토폴로지

5.1 RTT-공평성

이 실험에서는 혼잡 상황에 따른 각 프로토콜의 RTT-공평성을 살펴보기 위하여 서로 다른 RTT를 갖는 두 개의 초고속 플로우의 처리율의 비율을 측정하였다. 하나의 플로우는 40ms로 고정되어 있고, 다른 하나는 40ms, 120ms, 240ms로 변경하며 두 플로우의 RTT 비율을 다르게 조절하였다. 또한 윈도우 크기에 따른 영향을 실험하기 위하여, 병목 링크의 대역폭을 200Mbps, 2.5Gbps로 변경하며 실험하였다. 표 1과 2는 각각 200Mbps와 2.5Gbps에서 RTT의 비율에 따른 처리율의 비율을 나타낸 것이다. 두 가지 경우에 모두 EIMD의 처리율의 비는 RTT에 반비례함을 알 수 있다. 반면 HSTCP와 BIC의 경우 EIMD보다 훨씬 RTT-공평성이 낮음을 알 수 있다. EIMD의 경우 목표로 했던 RTT-공평성보다 더 좋은 결과를 얻었는데 이는 윈도우가 작을수록 또는 RTT가 커질수록 동시 손실이 생길 확률이 줄어 상대적으로 RTT가 큰 플로우들이 피해를 적게 받았기 때문이다.

〈표 1〉 병목 링크의 대역폭이 200Mbps일 때의 처리율의 비율

RTT ratio	1	3	6
EIMD	1.04	2.27	2.93
BIC	0.98	18.54	43.49
HSTCP	0.96	10.45	26.51

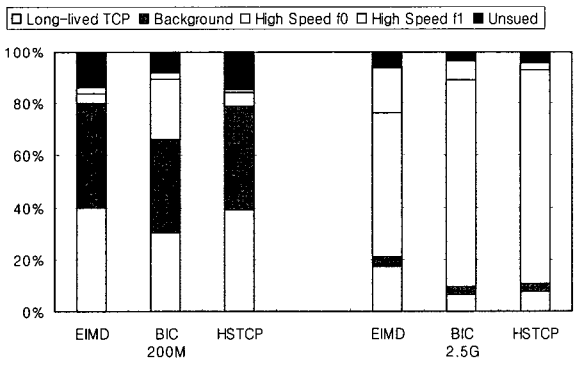
3) Scalable TC RT공평성만족하지 못하므 본 논문에서 실험 고려하지 않기로 한다

〈표 2〉 병목 링크의 대역폭이 2.5Gbps일 때의 처리율의 비율

RTT ratio	1	3	6
EIMD	1.03	3.2	4.91
BIC	0.87	10.63	35.08
HSTCP	1.02	28.03	108.08

5.2 TCP-친밀성

이번 실험에서는 TCP-친밀성을 살펴보기 위해서 초고속 플로우와 함께 존재하는 기존 TCP 플로우의 처리율을 살펴보았다. 200Mbps의 경우는 10개의 TCP 플로우를 2.5Gbps의 경우는 4개의 TCP 플로우를 사용하였다. 초고속 플로우 f_1과 f_2는 각각 40ms, 120ms의 RTT를 갖는다. (그림 8)은 각 플로우 종류에 할당된 대역폭의 비율을 나타낸 그래프이다. 여기서 TCP 플로우와 배경 트래픽의 합이 각 프로토콜의 TCP-친밀성을 나타내게 된다. BIC는 사용 가능한 대역폭을 효과적으로 사용한 반면, 윈도우의 크기에 상관없이 TCP 플로우들로부터 많은 양의 대역폭을 뺏어 오는 단점이 있다. 패킷 손실이 높은 경우에도 하나의 BIC 플로우가 10개의 TCP 플로우들이 사용하는 대역폭을 사용하게 되어 TCP-친밀성이 떨어짐을 알 수 있다. 반면에 EIMD는 패킷 손실에 상관없이 가장 TCP-친밀성이 가장 뛰어남을 알 수 있다. EIMD의 경우 미사용 대역폭이 다른 두 프로토콜에 비해서 상대적으로 약간 높지만, 병목 링크의 대역폭이 커질수록 미사용 대역폭이 점점 감소하는 사실을 다음 실험을 통하여 확인할 수 있다.



(그림 8) 병목 링크가 200Mbps와 2.5Gbps일 때 플로우 종류마다 할당된 대역폭 비율

5.3 대역폭 이용 효율

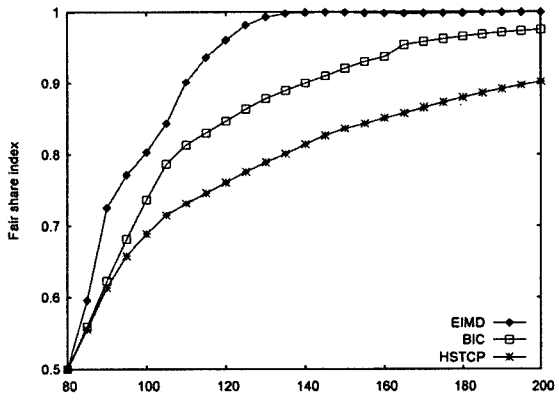
EIMD가 초고속 환경에 잘 맞는지 살펴보기 위하여 대역폭에 따른 이용 효율(utilization)을 측정해 보았다. 병목 링크의 대역폭을 200Mbps에서 5Gbps까지 바꿔가며 실험하였다. 기타의 환경 설정은 실험 4.2와 동일하게 하였다. 표 3은 병목 링크의 대역폭에 따른 병목링크의 전체 이용 효율을 나타내고 있다. 대역폭이 작을 때, 이용 효율이 낮은 이유는 EIMD 플로우가 TCP 플로우와 비슷한 처리율을 얻게 되기 때문이다. 하지만 대역폭이 커질수록, 전체 대역폭에서 사용되지 않는 부분이 점점 작아짐을 확인할 수 있다. 사용되지 않는 대역폭이 반대 경로(reverse path)의 ACK까지 포함하고 있다는 것을 고려할 때, EIMD가 초고속-장거리 네트워크에서 효과적으로 대역폭을 사용하고 있음을 알 수 있다. HSTCP와 BIC의 경우도 병목 링크의 대역폭이 증가할수록 95% 이상의 이용 효율을 보여준다.

〈표 3〉 EIMD의 대역폭 이용 효율

Bottleneck(bps)	200M	2.5G	5G
Utilization(%)	86.22	93.34	96.95

5.4 공정 배분 수렴

이번 실험에서는 각 프로토콜의 공정배분 수렴 속도에 대해서 살펴본다. 병목 링크의 대역폭이 2.5Gbps일 때, 100ms의 RTT를 가지는 초고속 플로우가 0초에서 20초 사이에 임의로 전송을 시작한다. 이 플로우의 혼잡 윈도우가 충분히 커졌다고 판단되는 80초부터 같은 RTT를 갖는 다른 초고속 플로우가 새로 전송을 시작하여 병목 링크를 공유하도록 하였다. 80초 이후부터 5초 간격으로 각 플로우의 누적 처리율을 계산하였다. (그림 9)는 이 누적 처리율을 바탕으로 공정 배분 지표(fair share index) [15] 값을 시간에 따라 나타낸 것이다. 그림에서 볼 수 있는 것처럼, EIMD가 가장 빠르게 공정 배분 값으로 접근하며, 그 이후에도 공정한 상태를 잘 유지하고 있음을 확인할 수 있다.



(그림 9) high-speed 프로토콜들의 fair share index

6. 결 론

본 논문에서는 초고속-장거리 네트워크에서 확장성과 TCP-친밀성을 제공하면서도 RTT에 따른 불공평한 처리율 배분 문제를 근본적으로 해결할 수 있는 EIMD라고 하는 새로운 전송계층 프로토콜을 제안하였다. 이들 세가지 요구사항을 만족시키기 위하여, EIMD는 RTT와 혼잡 윈도우가 증가하기 시작한 시간에 관한 정보를 이용하는 혼잡 제어 알고리

즘을 갖는다. 뿐만 아니라 $1/w_{max}$에 비례하여 혼잡 윈도우를 증가시킴으로써 빠르게 공정배분 값으로 수렴하는 특성도 갖는다.

또한 본 논문에서는 정적(static) 환경에서뿐만 아니라 동적(dynamic) 환경에서 BIC 및 HSTCP와 비교하여 EIMD의 특성을 살펴보았다. 모의 실험 결과 EIMD가 확장성을 제공하면서도 다른 프로토콜들 보다 TCP-친밀성, RTT-공평성 및 공정배분 수렴 특성이 뛰어남을 확인하였다. 향후 연구로 실제 구현을 통하여 EIMD의 성능을 보다 정확하게 평가해보고자 한다.

References

[1] S. Floyd, S. Ratnasamy, and S. Shenker, "Modifying TCP's Congestion Control for High Speeds", http://www.icir.org/floyd/hstcp.html, May, 2002.

[2] S. Floyd, "HighSpeed TCP for Large Congestion Windows", RFC 3649, December, 2003.

[3] J. Nagle, "Congestion Control in IP/TCP," IETF RFC 896, Jan., 1984.

[4] T. Kelly, "Scalable TCP : Improving performance in highspeed wide area networks", ACM SIGCOMM Computer Communication Review, Vol.33, Issue 2, pp.83-91, April, 2003.

[5] L. Xu, K. Harfoush, I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks", IEEE INFOCOM '04, March, 2004.

[6] C. Jin, D. Wei, and S. H. Low, "FAST TCP : Motivation, Architecture, Algorithms, Performance", IEEE INFOCOM '04, March, 2004.

[7] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", ACM SIGCOMM '02, August, 2002.

[8] Y. Gu, X. Hong, M. Mazzucco, and R. L. Grossman, "SABUL : A High Performance Data Transfer Protocol", Submitted for publication, 2002.

[9] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms", SIGCOMM '01, August, 2001.

[10] S. Jin, L. Guo, I. Matta, A. Bestavros, "TCP-friendly SIMD Congestion Control and Its Convergence Behavior", ICNP'2001 : The 9th IEEE International Conference on Newtork Protocols, Riverside, CA, November, 2001.

[11] S. Jin, L. Guo, I. Matta, and A. Bestavros, "A Spectrum of TCP-friendly Window-based Congestion Control Algorithms", IEEE/ACM Transactions on Networking, Vol.11, No 3, pp.341-355, Jun., 2003.

[12] S. Floyd, E. Kohler, "Internet research needs for better models", http://www.icir.org/models /bettermodels.html, October, 2002.

[13] HighSpeed TCP Simulation Reports, http://www-itg.lbl.gov/~evandro/hstcp/simul/simul.html

[14] Simulation Code and Scripts for BI-TCP Paper, http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/scripts/scripts.htm

[15] D.-M. Chiu, R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", Computer Networks and ISDN systems, 17 : 1-14, 1989.

[16] S. Bhandarkar, S. Jain, A. L. N. Reddy, " LTCP : A Layering Technique for Improving the Performance of TCP in Highspeed Networks", draft-bhandarkar-ltcp-01.txt, August, 2004.

[17] H. Sivakumar, S. Bailey, and R. L. Grossman, "PSockets : The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks", High-Performance Network and Computing Conference, November, 2000.

[18] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke, "Data Management and Transfer in High-Performance Computational Grid Environments", High-Performance Network and Computing Conference, November, 2002.

[19] T. Hacker, B. Noble, and B. Athey, "Improving Throughput and Maintaining Fairness using Parallel TCP", IEEE INFOCOM '04, March, 2004.

[20] T. Hacker, B. Athey, and B. Noble, "The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network", International Parallel and Distributed Processing Symposium, April, 2002.

[21] T. V. Lakshman, U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", IEEE/ACM Transactions on Networking, Vol.5, No 3, pp.336-350, July, 1997.

[22] Y. Gu, X. Hong, M. Mazzucco, and R. L. Grossman, "SABUL : A High Performance Data Transfer Protocol", Submitted for publication, 2002.

Appendix

다음은 식 (14)의 α'의 근사값을 구하는 과정이다. Increase rule에 의해서 T와 X의 값이 다음과 같이 얻어진다.

$$T = \frac{(\beta w_{\max}+1)^{1/u}-1}{\alpha'(k+1)} \tag{15}$$

$$X = \int_0^T (w_{\max} - \beta w_{\max} - 1 + (\alpha'(k+1)t+1)^u dt$$

$$= (w_{\max} - \beta w_{\max} - 1)T + \frac{1}{\alpha'(k+1)(u+1)} T^{u+1}.$$

식 (8)에 의해서,

$$(w_{\max} - \beta w_{\max} - 1) + \frac{(cT+1)^{u+1}}{\alpha'(k+1)(u+1)\cdot T} \approx (A \cdot RTT^r)^{1/(1-s)} T^{s/(1-s)}$$

이 성립하고, 다시 T에 식 (15)를 대입하면

$$(w_{\max} - \beta w_{\max} - 1) + \frac{(\beta w_{\max}+1)^{(u+1)/u}}{(u+1)((\beta w_{\max}+1)^{1/u}-1)} = (A \cdot RTT^r)^{1/(1-s)} \left(\frac{(\beta w_{\max}+1)^{1/u}-1}{\alpha'(k+1)} \right)^{s/(1-s)}$$

을 얻을 수 있다. 만약 $w_{\max} >> 1$, $\beta w_{\max} >> 1$이 성립하게 된다면, α'은 다음과 같이 표현된다.

$$\alpha' = u\left(\frac{(A \cdot RTT^r)^{1/(1-s)}}{1 - \left(\frac{(\beta w_{\max})^{1/u}}{(u+1)((\beta w_{\max})^{1/u}-1)} - 1 \right)\beta}\right)^{(1-s)/s} ((\beta w_{\max})^{1/u} - 1) w_{\max}^{\frac{1-s}{s}}$$

양 은 호

e-mail : ehyang@popeye.snu.ac.kr
2004년 서울대학교 컴퓨터공학부(학사)
2004년~현재 서울대학교 전기·컴퓨터공학부 석사과정
관심분야 : 무선 MAC 프로토콜, 차세대 인터넷, 전송계층 프로토콜, QoS 등

함 성 일

e-mail : siham@popeye.snu.ac.kr
2002년 한동대학교 전산·전자공학부(학사)
2004년 서울대학교 대학원 전기·컴퓨터공학부(공학석사)
2004년~현재 서울대학교 대학원 전기·컴퓨터공학부 박사과정
관심분야 : 무선 MAC 프로토콜, Resource Management, Quality-of-Service (QoS), 차세대 인터넷 등

조 성 호

e-mail : shcho@popeye.snu.ac.kr
1999년 서울대학교 전산과학과(학사)
2001년 서울대학교 대학원 전기·컴퓨터공학부(공학석사)
2001년~현재 서울대학교 대학원 전기·컴퓨터공학부 박사과정
관심분야 : 차세대 인터넷, 프로토콜 디자인 및 성능분석, 이동성 관리, 전송계층 프로토콜, QoS(Quality-of-Service) 등

김 종 권

e-mail : ckim@popeye.snu.ac.kr
1981년 서울대학교 산업공학과(학사)
1982년 미국 조지아 공대(공학석사)
1987년 미국 일리노이대학(공학박사)
1984년~1987년 IBM 산 호세 연구소 연구조원
1987년~1991년 미국 벨 통신 연구소 연구원
1991년~현재 서울대학교 전기·컴퓨터공학부 교수
관심분야 : 차세대 인터넷, 초고속 라우터, 이동통신 등