TCP 선택을 위한 자동 적응 프레임워크

정회원 황 재 현*, 종신회원 유 혁*

Self-Adaptation Framework for TCP Selection

Jae-Hyun Hwang* *Regular Member*, Chuck Yoo* *Lifelong Member*

요 약

본 논문에서는 기존의 TCP 변종들을 바탕으로 종단 간의 경로 상에서 나타나는 네트워크 특성에 가장 적응이 잘 이루어진 변종의 알고리즘을 선택하는 TCP의 자동 적응 프레임워크를 제안한다. 프로토콜 선택의 문제가 중요한 이유는 모든 네트워크 환경에 적합한 단일 버전의 프로토콜이 존재하지 않기 때문이며, 이것은 각 네트워크 마다 TCP의 성능 저하 원인이 서로 다르기 때문이다. 이러한 판단 및 프로토콜의 적응이 가능하게 하기 위해 본 논문에서는 기존에 연구되어 왔던 여러 가지 네트워크 측정 기법들과 TCP 변종들을 하나로 합치는 과정을 거쳤으며, 여기에 각 TCP들의 성능 정보들을 제공하여 세션 중간에 적절한 전송 알고리즘을 선택하여 사용할 수 있도록 하였다. 시뮬레이션 실험을 통해 우리는 종단 간으로 여러 환경 하에서 높은 성능을 이끌어낼 수 있다는 것을 보였으며, 제안한 방법이 지금까지 연구되어온 여러 TCP 변종들이 실제로 적절하게 활용될 수 있도록 하는데 중요한 역할을 할 것으로 판단한다.

Key Words : Self-Adaptation, TCP Selection, TCP Variant, TCP Performance, Network Estimation

ABSTRACT

In this paper, we propose a self-adaptation framework that selects a TCP variant adapted to current end-to-end path among available TCP variants. There is no single version of TCP that is suitable to all network environments since the causes for performance degradation are different one another according to characteristics of network environments. Thus, determining that which TCP variants should be selected in order to get best performance is very important. To enable adaptation through such determination, we integrate the existing network estimation schemes and some TCP variants into our framework then make light-weight performance knowledge database for TCP selection. Through implementing and evaluating the proposed framework, we show that our solution can help TCP get high and stable performance on the various types of network environments by pure end-to-end.

Ⅰ. 서 론

TCP가 초기의 유선 인터넷 망을 가정하여 설계된 프로토콜이라는 것은 널리 알려진 사실이지만, 높은 안정성과 성능으로 인해 여전히 다양한 환경하에서 사용되고 있다. 그러나 최근 들어 네트워크의 속도와 대역폭이 증가하고, 여러 종류의 무선망이 등장함에 따라 설계 시 가정했던 사항들이 현실과 맞지 않는 경우가 발생하였고 그 결과 TCP의 성능 저하 문제가 발생하였다. 이러한 문제를 해결하기 위해 TCP를 새로운 환경에 적응시키고자 하는 연구들이 진행되었으며 대표적으로 무선망을 위

※ 이 연구에 참여한 연구자는 '2단계 BK21사업'의 지원비를 받았음.
* 고려대학교 컴퓨터학과 운영체제 연구실({jhhwang, hxy}os.korea.ac.kr)
논문번호 : KICS2008-11-499, 접수일자 : 2008년 11월 11일, 최종논문접수일자 : 2009년 2월 6일

한 무선용 TCP[1][2][3], High-BDP(Bandwidth Delay Product) 망을 위한 TCP 변종[4][5][6] 등이 있다. 또한 TCP는 내부적으로 휴리스틱에 의존한 알고리즘을 다수 가지고 있기 때문에 이를 좀 더 정교하게 개선한 변종들도 제안되었다[7][8]. 이처럼 새로운 환경에 TCP를 적응시키거나 TCP의 성능 개선을 위한 연구들이 진행되어 왔지만, 실제로 사용되고 있는 TCP의 버전은 지금까지도 표준 TCP와 SACK 옵션[9] 정도로 한정이 되어 있다. 이것은 우선적으로는 프로토콜의 배포 문제로써, 프로토콜의 배포가 빠르게 이루어지지 않는다는 점과 전송 측의 수정만으로 이득을 보기 어렵기 때문에 대부분 상대방과 동일한 프로토콜을 가져야 사용할 수 있다는 점이 그 원인이 될 수 있다. 이에 대한 해결방안으로 모바일 코드(mobile code)를 이용한 프로토콜의 업데이트에 대한 연구가 이루어진 바 있다[10]. 그러나 TCP의 변종, 배포 문제가 모두 해결이 된다 하더라도 어느 상황에 어떤 프로토콜을 사용할 지를 결정하지 못한다면 여전히 기존의 연구들은 실용화되기 어렵다. 실제로 몇몇 변종들은 리눅스와 같은 커널 내에 구현되어 상당수 배포가 이루어졌음에도 불구하고 이러한 변종들이 사용되는 경우는 극히 드물다. 우리는 이 문제를 지금까지 연구된 변종 알고리즘들을 모두 수용할 수 있는 TCP 프레임워크가 존재하지 않기 때문이라고 판단하고 있으며, 이 TCP 프레임워크의 주요 역할은 네트워크 환경에 잘 적응된 변종을 사용할 수 있도록 도와주는 것이 될 것이다.

프로토콜 선택의 문제가 중요한 보다 근본적인 이유는 모든 네트워크 환경에 적합한 단일 버전의 프로토콜이 존재하지 않기 때문인데, 이것은 각 네트워크마다 TCP의 성능 저하 원인이 서로 다르기 때문이다. 예를 들면, 무선망에 적응시킨 TCP 변종은 High-BDP 망에서는 표준 TCP와 동일한 성능 저하의 문제점을 갖는다. 뿐만 아니라 적응 방법 역시 두 경우 모두 혼잡 제어(congestion control) 알고리즘의 개선을 요구하므로 하나의 혼잡 제어 알고리즘으로 두 망에 대해 동시에 적응시키기란 쉽지 않은 문제이다. 즉, 두 가지의 서로 다른 적응 알고리즘을 놓고 시스템이 적절하게 스위칭하여 각 상황에 적절한 알고리즘을 선택해야 할 것이다. 본 논문은 이러한 프로토콜 결정의 문제를 해결하는데 초점을 맞추고 있으며, 우리가 알고 있는 바에 따르면 지금까지 이에 대한 적절한 솔루션은 제시된 바가 없다. 때문에 본 논문에서 제안하는 방법은 매우 직관적인 구조를 가지고 있으며, 제안한 방법이 지금까지 연구되어온 여러 TCP 변종들이 실제로 적절하게 활용될 수 있도록 하는데 중요한 역할을 할 것으로 믿는다.

자동 적응(Self-Adaptation) 기법에 대해 간단하게 설명하자면, 현재 사용 중인 네트워크의 환경을 몇 가지 파라미터를 통해 추정하고 측정된 환경 하에서 보유한 TCP 변종 중 처리량이 가장 높을 것으로 기대되는 변종을 선택하는 것이다. 즉, 가장 적응이 잘 된 알고리즘이 무엇인지 판단하는 것이라고 할 수 있다. 이러한 판단을 통한 프로토콜의 적응이 가능하게 하기 위해 본 논문에서는 기존에 연구되어온 여러 가지 네트워크 측정 기법들과 TCP 변종들을 하나로 합치는 과정을 거쳤으며, 여기에 각 TCP들의 성능 정보들을 제공하여 세션 중간에 적절한 전송 알고리즘을 선택하여 사용할 수 있도록 하였다. 실험을 통해 우리는 종단 간으로 송신 측의 수정만을 거쳐 다양한 환경 하에서 높은 성능을 이끌어낼 수 있다는 것을 보였다.

본 논문은 다음과 같이 구성된다. 먼저 Ⅱ장에서는 네트워크 추정을 통한 적응과 관련된 관련 연구를 살펴본다. Ⅲ장에서는 TCP 적응 기법의 전체 구조와 설계 목표를 설명한 뒤, 본 논문에서 사용한 네트워크 측정 기법과 TCP 변종에 대해 설명한다. Ⅳ장에서는 적절한 변종을 선택하기 위해 본 논문에서 사용한 판단 근거와 기준에 대해 자세히 설명하도록 한다. Ⅴ장에서는 변종들의 성능 정보를 측정하고 이를 바탕으로 제안된 기법을 적용, 구현하기 위한 과정을 서술한다. Ⅵ장에서는 제안된 기법에 대한 성능평가 과정을 거치고 그 결과를 분석한다. Ⅶ장에서는 제안된 기법과 관련된 몇 가지 이슈에 대해 논의한 뒤 Ⅷ장에서 결론을 맺는다.

Ⅱ. 관련연구

STP(Self-spreading Transport Protocols)[10]는 현재 보유하고 있지 않는 TCP의 소스 코드를 다운로드하여 업그레이드를 할 수 있게 해주는 프레임워크이다. 이들은 기존의 TCP 변종들이 실제로 잘 쓰이지 않는 이유가 프로토콜의 배포문제라고 파악하고 있다. 즉, 한쪽 단말이 사용하고 있는 TCP 변종을 상대방이 보유하고 있지 않을 때 모바일 코드 형태로 소스 코드를 다운로드 받게 하고, 이 후 이를 컴파일 및 로드하여 세션을 재 연결하는 방식을 취한다. 그러나 앞서 언급한 바와 같이 배포의 문제

가 해결된다 하더라도 모든 환경에 적응 가능한 단일 변종이 존재하지 않기 때문에 여전히 선택의 문제가 발생하며, STP는 이에 대한 해결책은 제시하지 않고 있다.

한편으로 네트워크 환경을 측정하여 해당 정보를 기초로 TCP의 적응을 가능하게 하려는 연구가 제안되어왔다. TCP의 버퍼 튜닝(buffer tuning) 기법이 그 중 하나로, 대표적으로 Automatic TCP Buffer Tuning[11]과 Dynamic Right-Sizing(DRS)[12]이 있다. 이들의 궁극적인 목표는 TCP 패킷 헤더의 타임스탬프(timestamps) 정보 등을 통해 종단 간 경로 상의 대역폭 지연시간 곱을 추정하여 적절하게 버퍼의 크기를 조절하는 것이다. 그러나 버퍼의 크기를 조절하는 것은 네트워크 환경에 따른 TCP의 적응을 다룬다고 보기 어려우며, 버퍼 크기가 성능 저하의 주요 원인이 아닌 환경에서는 적용하기 어렵다.

네트워크의 손실률에 기반한 적응 알고리즘 역시 다수 제안되었다. 대표적으로 LDA+ 알고리즘[13], TFRC[14] 등이 있으며, 종단 간으로 링크의 손실률을 측정하여 이를 바탕으로 TCP 친화적으로 전송률을 조절하는 기법들이다. 이들 역시 손실률이라는 네트워크 환경 정보를 바탕으로 적응을 시도하고 있지만, 주로 손실률 정보에만 초점을 맞추고 있으며 네트워크 환경에 적응을 한다기보다 TCP와의 성능 상 친화관계(TCP-friendly)에 목표를 두고 있다는 한계가 있다.

즉, 대부분의 네트워크 측정을 통한 적응과 관련된 연구들은 단편적인 정보에 의존하고 있으며, 기존의 적응 알고리즘들을 통합하여 다양한 환경에 모두 적응할 수 있는 메커니즘을 제안하지 못하고 있다. 우리는 본 논문에서 각 TCP 변종들이 특정 환경에서 높은 성능을 보인다는 점을 바탕으로 적응된 환경이 각각 다르다는 사실을 인지하였으며, 이들을 모두 통합하여 각 환경에 적합한 알고리즘을 적용하여 사용한다면 다양한 환경 하에서 꾸준히 높은 성능을 보일 수 있다는 것을 보였다. 이에 더 나아가 본 논문에서는 실제 TCP에 적용 가능한 네트워크 측정 기법들을 이용하여 제안된 기법이 실용적일 수 있음을 보여줄 것이다.

Ⅲ. 적응 프레임워크의 구조 및 설계 목표

본 장에서는 자동 적응 프레임워크의 전체 구조에 대해 설명한다. 제안하는 프레임워크의 구조는 일반적인 자동 적응 시스템과 마찬가지로 1) 사용환경의 측정, 2) 프로토콜 성능 데이터, 3) 전송 알고리즘 변경 가능한 TCP 구조로 나뉜다. 먼저 네트워크 환경 측정은 기존에 제안되어 왔던 종단 간 측정 기법들을 활용하여 현재 사용 중인 네트워크 경로의 특성을 추정하는 부분이다. 이 부분의 설계 초점은 송신측에서 측정이 가능해야 한다는 것과 측정이 빠르고 정확해야 한다는 것이다. TCP는 종단 간 프로토콜이기 때문에 현재 사용 중인 LAN의 특성보다는 종단 간의 경로 상에서의 병목 지점의 특성이 성능에 주로 영향을 주게 된다. 이러한 네트워크 경로는 새로운 연결이 이루어질 때마다 바뀌기 때문에 네트워크 측정이 매번 이루어져야 하며, 따라서 네트워크의 특성을 빠르게 측정해내는 것은 해당 세션에서의 적응 시간을 단축시키므로 성능 향상에 중요한 요소가 된다. 또한 프로토콜의 성능 데이터는 네트워크 환경 별로 측정된 결과를 사용하기 때문에 측정이 정확하지 않다면 TCP의 적응 방법이 틀려질 수 있다. 따라서 측정의 정확성 역시 성능 향상에 중요한 요소가 된다. 다음으로 TCP는 그 동안 제안되어 왔던 변종 몇 가지를 합쳐서 세션 중간에 다른 알고리즘을 적용할 수 있도록 구현되어야 한다. 대부분의 TCP 변종들은 기존 코드에서 일부 알고리즘만을 개선하여 제안된 것들이기 때문에 몇 개의 변종들을 하나의 TCP로 합쳐도 전체 코드의 크기는 크게 증가하지 않으며, 여러 개의 TCP 변종들을 각각 독립적으로 구현하는 것보다 코드 크기에 있어서 보다 효율적이다. 마지막으로 프로토콜의 성능 데이터는 각 환경에서 어떤 TCP의 적응 방법을 따를 것인지 선택하는 중요한 기준이 되며, 본 논문의 주요 기여도라 할 수 있다. 반면 환경 측정 방법은 기존에 제안 되어온 방식을 활용하였으며, TCP 역시 그 동안 제안되었던 변종들을 바탕으로 몇 가지를 선택하여 통합하였다. 이번 장에서는 본 논문에서 적용한 네트워크 파라미터 및 측정 기법과 TCP의 통합 방법을 간략히 소개하고 다음 장에서 프로토콜의 성능 데이터를 추출하는 과정과 실제 세션에서 적응하는 과정을 자세히 설명하도록 한다.

3.1 네트워크 추정 기법

네트워크의 특성을 나타내는 지표에는 여러 가지가 있지만 프로토콜의 성능에 영향을 미치면서 종단 간 측정이 가능한 요소로는 네트워크 대역폭, 지연시간, 그리고 패킷 손실률이 있다. 우리는 단말에서 바라보는 종단 간 네트워크의 특성을 이 세 가

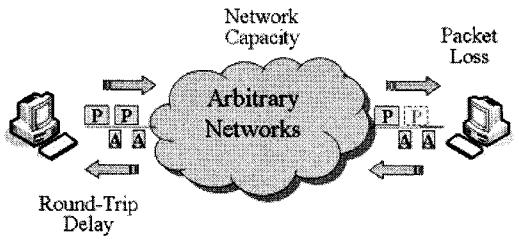


그림 1. 단말에서 인지 가능한 네트워크의 특성

지 네트워크 파라미터로 정의하였으며(그림 1), 기존에 제안된 측정 기법 중 송신 측에서 측정 가능한 기법을 해당 프레임워크에 도입하였다. 각 네트워크 파라미터를 측정하는데 사용된 기법은 다음과 같다.

- 지연시간 : 기본적으로 지연시간 정보는 TCP에서 측정된다. RTT는 중간 라우터들의 큐잉 딜레이를 잘 반영하기 때문에 RTO(Retrasmission TimeOut)값을 설정하는데 사용되어 왔다. RTT는 네트워크의 혼잡 정도를 반영하는데 도움이 될 뿐 아니라, 최근 들어 위성 망과 같이 대역폭 지연시간의 곱이 높은 망에서 TCP의 효율성 저하 문제가 발생하면서 지연시간의 측정은 더욱 중요한 의미를 갖는다. 본 논문에서는 TCP 내에서 측정한 평균 RTT 값을 네트워크 파라미터의 지연시간 값으로 사용하였다.
- 대역폭 : 본 논문에서는 네트워크의 중요한 특성 중 하나인 대역폭을 측정하기 위해 TCP Probe[15]기법을 자동 적응 프레임워크에 도입하였다. TCP Probe는 능동적인 용량 측정 기법 중 하나인 CapProbe[16]를 기반으로 한 것으로, TCP 패킷을 이용하여 수동적인 네트워크 용량 측정이 가능하므로 TCP에 적용이 가능한 측정 기법이다. 기존 TCP 변종에서 사용하던 대역폭 측정 기법은 Westwood에 적용된 Bandwidth Estimation 기법이 있었다. 이미 기존 연구[15]에서 이 둘의 성능 차이를 비교한 바 있지만, TCP Probe를 도입한 가장 큰 이유는 Bandwidth Estimation은 병목 지점이 존재하지 않으면 정확한 네트워크 용량을 측정하지 못한다는 점이다. 예를 들어 간단한 단일 연결로 테스트를 해보면 송신자의 전송률보다 네트워크의 대역폭이 훨씬 클 때, Bandwidth Estimation은 보통 보다 적게 측정하게 된다.
- 손실률 : 기존에 제안된 패킷 손실률의 측정 기법들은 대부분 수신 측에서 측정이 이루어지며, 비신뢰적인 프로토콜에 적용 가능한 것들이었다. TCP와 같은 신뢰적인 프로토콜은 재전송과 같은 메커니즘이 포함되어 있기 때문에 기존의 기법들을 적용하기 어려운 단점이 있다. 본 논문에서는 송신 측에서 측정하며 TCP에서 적용 가능한 측정 기법인 LEAST[17]를 도입하였다. 이것은 재전송 횟수를 이용하여 손실률을 측정하는 기법으로 대부분의 TCP 변종에 쉽게 적용 가능하다는 장점을 갖는다.

3.2 TCP 변종들의 통합 버전

앞서 언급한 바와 같이 최근까지도 TCP의 전송 알고리즘을 개선한 다양한 변종들이 제안되고 있다. 이러한 변종들은 새로운 네트워크의 특성을 반영하여 성능을 개선시키고 있지만, 실제 사용하고 있는 TCP는 새로운 알고리즘을 포함한 단일 버전이 아닌 개별적인 변종으로 구현되고 있다. 이것은 세션 연결 전에 사용할 TCP의 종류를 미리 결정해야 한다는 것을 의미한다. 물론 어느 TCP를 사용하는 것이 효율적일지 미리 판단할 수 있다면 독립적으로 구현해도 상관이 없겠지만, 특정 환경을 목표로 사용하지 않는 한 결정하기 어려운 문제이다. 따라서 세션 연결 중 망의 환경을 측정한 뒤 적절한 변종 알고리즘을 판단하고 적용해야 하며, 이는 기존의 모든 변종 알고리즘을 하나의 통합된 단일 버전으로 구현하여 원하는 시점에 적용할 알고리즘을 변경할 수 있게 해야 한다는 것을 의미한다. 즉, TCP 통합과 관련된 설계 목표는 모든 변종들을 수용하여 세션 연결 후 원하는 시점에 다른 전송 알고리즘으로 스위칭이 가능하게 하는 것이며, 결과적으로 세션 마이그레이션(session migration)으로 인한 오버헤드를 최소화하는 것이다. 구현 방법과 관련하여 효율성 및 확장성과 관련된 다른 이슈가 존재할 수 있겠지만 본 논문에서는 간단히 함수 단위로 코드 분기를 사용하였으며, 이에 대해서는 5장에서 자세히 설명하도록 하겠다.

Ⅳ. TCP 변종 간의 선택 메커니즘

네트워크의 환경을 측정한 뒤 현재 환경에 적합한 TCP 알고리즘을 변경하여 사용가능 하다면, 어느 알고리즘을 선택하여 사용할 것인가에 대한 판단 근거와 기준이 필요하다. 선택의 기준이 되는 메트릭(metric)은 처리량, 공평성(fairness), 시간성

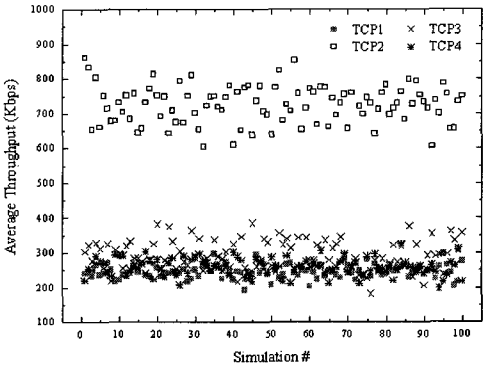
(timeless) 등 여러 가지가 있을 수 있겠지만, 본 논문에서는 망의 적응에 초점을 두어 처리량으로 한정했다. 즉, 주어진 환경에서 처리량이 가장 높은 TCP를 그 망에 가장 잘 적응된 변종으로 판단하는 것이다. 그러나 처리량이라는 메트릭은 동일한 환경에서도 네트워크의 다이나믹스(network dynamics)에 따라 어느 정도의 편차가 존재하기 때문에 단순히 평균값으로만 판단 기준으로 삼기에는 그 신뢰도가 떨어진다. 다음의 시뮬레이션 결과를 통해 이 부분에 대하여 좀 더 살펴보기로 하겠다.

그림 2는 두 가지 서로 다른 환경 하에서 4개의 TCP 변종(Reno 포함)에 대한 100초간의 평균 처리량을 100회 반복하여 측정한 그래프이다. 그림 2(a)는 대역폭, 왕복 지연, 손실률이 각각 5Mbps, 300ms, 1.5%인 병목 링크 하에서 측정되었다. 이 환경에서는 TCP2가 다른 변종들에 비해 2~3배 높은 처리량을 뚜렷하게 보여주고 있다. 즉, TCP2의 전송 알고리즘이 이 환경에 보다 잘 적응되었다는 것을 의미하며, TCP2를 선택하여 사용하는 것이 성능 상에 유리함을 쉽게 알 수 있다.

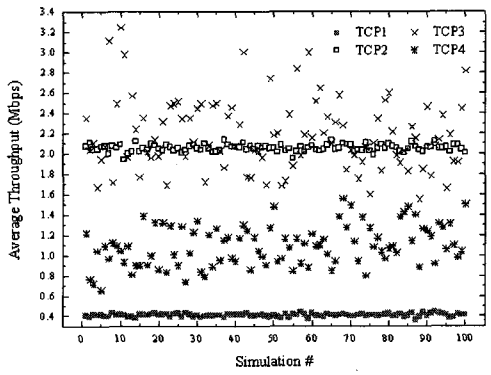
그림 2(b)는 100Mbps의 대역폭, 350ms의 왕복 지연, 0.1%의 손실률을 가진 링크 하에서 측정된 그래프이다. 이 환경에서 가장 높은 처리량을 보인 변종은 TCP3이다. 그러나 (a)의 경우와는 달리 TCP3의 처리량은 큰 편차를 보이고 있다. 특히, 몇몇 실험에서는 TCP2보다 낮은 처리량을 보였다. 평균의 관점으로 본다면, 100회 반복한 결과의 평균을 계산했을 때 TCP3가 TCP2보다 약간 높게 나타나고 있다. 그러나 실제 모집단의 평균값은 알 수 없으므로 이렇게 편차가 높은 상황에서는 TCP3가 TCP2보다 우월하다고 직접적으로 판단하기 어렵다.

본 논문에서는 특정 환경 하에서 두 TCP의 평균 처리량을 측정한 뒤 어느 TCP의 처리량이 보다 우월한지 판단하기 위하여 가설 검증 기법(test of hypothesis procedure[18])을 도입했다. 이는 일부 표본 집단으로부터 전체 모집단 간의 우월 정도를 추정할 수 있게 해주는 방법이다. 먼저 TCP A와 B라는 두 변종에 대해 동일한 환경 하에서 각각의 처리량에 대한 모집단과 표본 집단의 평균 및 분산을 표 1과 같이 정의하였다.

n_A와 n_B는 측정 횟수를 의미하며 모두 100으로 고정되어 있다. 검증하고자 하는 내용은 두 TCP의 평균 처리량의 차이가 되며 귀무가설(null hypothesis)과 대립가설(alternative hypothesis)은 다음과 같이 표현할 수 있다.



(a) 병목링크: 5Mbps, 300ms, 1.5% 손실률



(b) 병목링크: 100Mbps, 350ms, 0.1% 손실률

그림 2. 네 개의 다른 TCP 변종에 대한 평균 처리량

표 1. 각 TCP 처리량 집단에 대한 평균 및 분산 표기법

	평균	분산	샘플 수
집단 A	μ_A	σ_A^2	N/A
집단 B	μ_B	σ_B^2	N/A
A의 샘플	$\overline{x_A}$	s_A^2	n_A
B의 샘플	$\overline{x_B}$	s_B^2	n_B

$$H_0 : \mu_A - \mu_B \geq 0$$
$$H_a : \mu_A - \mu_B < 0$$

귀무가설은 두 TCP의 평균 처리량의 차이가 0보다 높다는 것이다. 즉, 귀무가설이 가결 되면 A가 B보다 우월하다는 것을 의미하며, 기각되는 경우는 A의 평균 처리량이 B보다 높다고 하더라도 A가 우월하다고 볼 수 없다는 것을 의미한다. 검증통계량은 두 TCP 변종 간의 평균 처리량의 차이, 즉 $(\overline{x_A} - \overline{x_B})$가 된다. 그러면 이는 중심 극한 정리

(Central Limit Theorem)에 의해 정규분포를 따르게 되며 다음과 같이 표현할 수 있다.

$$\overline{x_A}-\overline{x_B} \sim N\left(\mu_A-\mu_B, \frac{\sigma_A^2}{n_A}+\frac{\sigma_B^2}{n_B}\right)$$

그리고 이의 표준화 변수는

$$z=\frac{\overline{x_A}-\overline{x_B}-(\mu_A-\mu_B)}{\sqrt{\frac{\sigma_A^2}{n_A}+\frac{\sigma_B^2}{n_B}}}$$

이 된다. 여기서 모집단의 분산들인 σ_A^2과 σ_B^2은 알 수 없는 값들이므로 측정된 결과로부터 얻어진 s_A^2과 s_B^2를 적용하여 표준화 분포를 t-분포로 변형시켜야 한다. 이때의 가정 조건인 n_A와 n_B가 충분히 크므로(일반적으로 30이상) 변형 가능하다. 즉, 표본분산을 적용하면 표준화 정규분포 변수 z는

$$t=\frac{\overline{x_A}-\overline{x_B}-(\mu_A-\mu_B)}{\sqrt{\frac{s_A^2}{n_A}+\frac{s_B^2}{n_B}}}$$

로 바꾸어 표현 되며, t는 다음의 자유도(degree of freedom)를 가지고 분포하게 된다.

$$df=\frac{(V_A-V_B)^2}{\frac{V_A^2}{n_A-1}+\frac{V_B^2}{n_B-1}},$$
$$where\ V_A=\frac{s_A^2}{n_A},\ V_B=\frac{s_B^2}{n_B}$$

이제 t와 df를 통해 미리 알려진 p-value를 구할 수 있게 된다. 이 값은 t-분포 내부 α영역의 크기를 의미하며 일반적으로 0.05 이하로 나올 때 H_0가 기각된다. 즉 p-value가 0.05 이상이 되어야 A가 B보다 평균 처리량이 높다는 것을 의미하게 되며, 이 결과는 95% 이상의 신뢰 수준을 가진다.

그림 2(b)에서 예로 든 TCP2와 TCP3을 위의 검증과정에 적용해 보도록 하겠다. 실험을 통해 얻어진 두 TCP에 대한 평균과 편차는 표 2와 같다. 귀무가설과 대립가설은 각각 $H_0: \mu_A-\mu_B \geq 0$, $H_a: \mu_A-\mu_B<0$가 된다. TCP3의 평균값이 높으므로 μ_A로 두었으며, 이는 귀무가설이 TCP3가 TCP2보다 우월하다는 것을 의미하고 있다. 위의 샘플(즉, 각각의 처리량 측정값)들은 1) 독립적으로 선택이 되었으며 2) 샘플 크기가 충분히 크다는 가정 사항

표 2. 그림 2(b)의 측정값에 대한 평균과 분산

	n	$\overline{x}$	s
TCP2	100	2054.7	34.44
TCP3	100	2179.4	353.07

을 만족하므로 t-분포 변환 후 t값을 다음과 같이 계산해낼 수 있다.

$$t=\frac{(2179.4-2054.7)-0}{\sqrt{(353.07)^2/100+(34.44)^2/100}}$$
$$=\frac{124.7}{1258.445385}\approx 0.09909$$

이제 p-value를 구하기 위해서 df를 다음과 같이 계산한다(df값은 100으로 내림하였다).

$$V_{TCP3}=11.861136,\ V_{TCP2}=1246.584249,$$
$$df=\frac{(11.861136+1246.584249)^2}{\sqrt{(11.861136)^2/99+(1246.584249)^2/99}}$$
$$=\frac{1583684.787}{15698.11}\approx 100.8838$$

이것을 이미 잘 알려진 t 커브의 영역에서 도출해보면[18], 120 df를 가진 -0.1 의 왼쪽 영역이라는 것을 알 수 있다. 즉, $p-value \approx 0.460$이 되며, 유의수준(significance level) α를 0.05라고 했을 때 0.46>0.05 이므로 H_0는 기각되지 않는다. 따라서 TCP3의 성능이 TCP2보다 우월하다고 판단할 수 있다.

V. 자동 적응 TCP 프레임워크의 구현

이번 장에서는 앞에서 설명한 TCP의 자동 적응 프레임워크를 위한 3가지 구성요소인 1) TCP 변종들의 성능 측정 정보 구축, 2) 네트워크 측정 기법 및 변종들의 통합, 3) 최적의 변종 선택을 구현하는 과정을 설명한다. 본 논문에서는 실험의 편의를 위해 NS-2 (버전 2.26[19]) 시뮬레이터를 사용하였으며, 제안하는 프레임워크 역시 시뮬레이터 내에 구현되었다. 한 가지 중요한 문제는 제안하는 프레임워크 내에 실제로 어느 변종들을 포함시켜 구현할 것인가이다. 물론 궁극적인 목표는 제안되어온 변종들을 모두 포함시키는 것이지만, 본 논문에서는 Reno를 포함하여 최근에 제안된 변종들 중 적응 알고리즘을 소스코드 수준에서 쉽게 구할 수 있었던 Westwood[8], CUBIC[6] 그리고 Veno[3] 이 네 종류의 TCP만을 포함시켰다. 실제로 High-BDP 환경에 대한 변종들이

최근 많이 제안되었지만, 같은 적응 문제를 해결하고 있기 때문에 실험의 편의를 위해 CUBIC 한 변종만을 포함시켰다. 이 후 설명할 과정들은 모든 변종들에 동일하게 적용되며 따라서 어느 변종이라도 같은 과정을 거쳐 확장시킬 수 있다.

5.1 변종간의 성능 측정

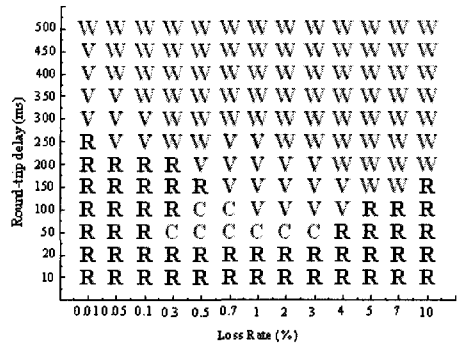
적절한 TCP 변종을 선택하기 위해서는 주어진 환경에서 각 TCP에 대한 처리량의 기대 값을 미리 알고 있어야 한다. 이 과정은 자동 적응 TCP 프레임워크를 구현하기 전에 선행되어야 하며, 다양한 환경에서 각 TCP의 처리량 기대 값을 얻어내기 위해서 집중적인 시뮬레이션 과정을 수행하였다. 측정 시 환경에 대한 통제 요인은 앞서 언급한 대역폭, 지연시간, 손실률 세 가지이며 단일 병목 지점을 갖는 Dump-Bell 토폴로지 상에서 이들을 다음 표 3과 같이 변화 시켜가며 성능을 측정하였다.

그림 3의 토폴로지에서 변동 구간(variation area)의 환경변수를 각각 표 3의 값들로 변화시켜 가며 4가지 TCP 변종에 대한 처리량을 구해내었다. 패킷 크기는 1500bytes로 고정시켰으며, 각 측정마다 200초 동안의 평균 처리량 값을 구하여 같은 변종, 같은 환경에 대한 실험을 매번 100회 반복하였다. 이렇게 구해진 100개의 평균 처리량 샘플 값은 특정 환경에 대한 해당 TCP의 성능을 나타내는 지표가 된다. 그림 4는 이렇게 구해진 샘플들을 바탕으로 단순히 높은 평균 처리량을 보이는 TCP를 적응 TCP로 결정했을 때 나타나는 성능 맵(performance map)이다. 지면상 대역폭이 각각 1, 10, 100, 1000Mbps일 때의 4가지 경우만을 실었다. 이 성능 맵은 각 TCP의 특성을

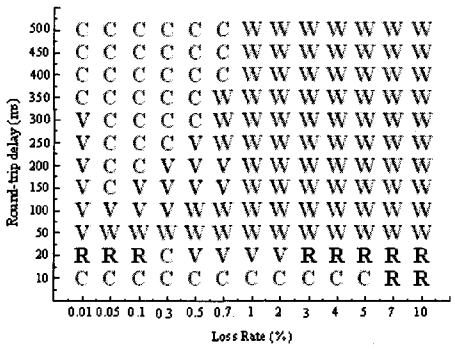
표 3. 네트워크 파라미터의 범위

파라미터	변화 값
대역폭 (Mbps)	1, 2, 5, 10, 20, 30, 50, 70, 100, 200, 500, 1000
왕복 지연 (ms)	10, 20, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500
손실률 (%)	0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 1.0, 2.0, 3.0, 4.0, 5.0, 7.0, 10.0

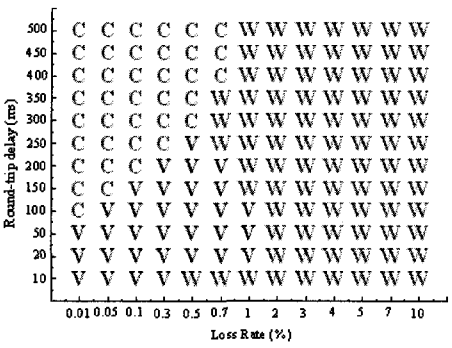
그림 3. 성능 측정을 위한 시뮬레이션 토폴로지



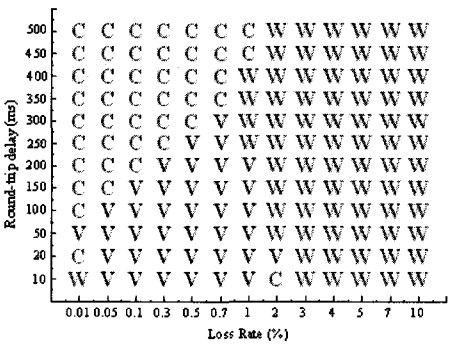
(a) 대역폭: 1Mbps



(b) 대역폭: 10Mbps



(c) 대역폭: 100Mbps



(d) 대역폭: 1000Mbps

그림 4. 성능 맵: R-Reno, W-Westwood, C-Cubic, V-Veno

그대로 반영하고 있다. 예를 들어 High-BDP 네트워크에 적응시킨 Cubic의 경우 상대적으로 낮은 손실률과 지연시간이 큰 망에서 좋은 성능을 나타낸다. Westwood 역시 손실률이 높은 환경에서 다른 변종들에 비해 높은 성능을 보이고 있다. 그러나 앞서 지적한 바와 같이 단순히 처리량의 평균이 높다고 하여 다른 변종들보다 확실히 우월하다고 보기 어렵다. 그림 4(d)에서도 보면 1000Mbps, 10ms의 delay와 2%의 손실률을 가지는 환경에서 갑자기 Cubic의 성능이 가장 좋게 나오고 있다. 이는 실제로 평균값은 비슷하나 샘플링 시 조금 더 나은 처리량 값이 나와 생긴 결과이다. 즉 서로 우열을 가리기 어렵다는 것을 의미하며, 실제로 가설 검증을 거친 결과도 Cubic이 다른 TCP보다 우월하다는 귀무가설이 기각되었다. 사용할 TCP의 알고리즘을 자주 교체하는 것은 해당 함수의 내부 변수가 계속해서 리셋(reset)이 되는 것을 의미하기 때문에 이런 상황에서는 굳이 다른 변종을 쓰고 있다가 Cubic으로 교체하지 않는 것이 좋다.

5.2 변종간의 선택

이제 각 TCP의 성능 정보를 토대로 TCP의 결정 맵(determination map)을 생성해야 한다. 실제 네트워크 환경을 측정한 뒤에는 순수 처리량 정보로 성능의 우월성을 판단하려면 추가시간이 소요되기 때문에 결정 맵은 어느 것으로 교체할 것인지에 대한 정보만을 담아야 한다. 먼저, 본 논문에서는 교체의 판단을 위해 각 TCP마다 우선순위를 표 4와 같이 정의했다. 그 후, 각각의 환경에 대한 4가지 TCP 변종들의 처리량 값에 대한 평균과 분산 값을 얻어내고, 먼저 가장 평균 처리량이 높은 TCP에게 우선순위 1을 부여한다. 그 후 나머지 변종들을 각각 우선순위 1의 TCP와 비교하여 평균과 분산 값으로 가설 검증 과정을 거친다. 이 결과가 기각되면 우선순위 2를, 가결되면 우선순위 3을 각각 부여한다.

이 과정까지가 오프라인으로 만들어져 성능 맵 형태의 결정 맵이 생성된다. 결정에 필요한 인자는 대역폭, 지연시간, 그리고 손실률 세 개의 값이며, 이전 절에서 처리량 측정에 사용한 환경들이 이산

표 4. 변종 선택 시 각 TCP들의 우선순위와 그 의미

우선순위	의미
1	평균 처리량이 가장 높음
2	우선순위 1보다 처리량은 약간 떨어지나, 교체할 필요 없음
3	우선순위 1의 변종으로 교체요구

```
ON receiving new ACK:
    bandwidth = TCP_PROBE();
    delay = TCP_RTT();
    loss = LEAST();

    priority =
determination_map[bandwidth][delay][loss];
    SWITCH(priority[current_tcp])
        CASE 1:
        CASE 2:
            break;
        CASE 3:
            FOR(i=0;i<tcp_num;i++)
                IF(priority[i]==1)
                    break;
                END
            END
            current_tcp=i;
            break;
    END
```

그림 5. 변종 선택에 대한 결정

값이기 때문에 가장 가까운 값으로 근사 시킨다. 즉, 현재 사용하는 네트워크의 환경이 각각 48Mbps, 60ms, 0.12%로 측정되었다면 이는 50Mbps, 50ms, 0.1%로 근사하여 결정 맵에서 해당 TCP의 우선순위 값을 찾는다. 이 때 사용하고 있는 TCP의 우선순위가 1 또는 2라면 교체할 필요가 없으며, 3일 경우 우선순위 1에 해당하는 TCP로 교체를 수행한다. 그림 5는 이에 대한 수도 코드(pseudo code)를 나타낸 것이다. 시스템이 보유하고 있는 모든 변종들은 Reno부터 0으로 시작하여 차례로 1씩 증가하여 정의되어 있으며, *current_tcp* 전역 변수가 현재 사용하고 있는 변종에 대한 정보를 유지한다. *tcp_num*은 보유하고 있는 변종의 개수를 나타낸다.

5.3 변종간의 통합

사용할 TCP 변종의 결정과 더불어 가장 중요한 사항은 여러 변종 알고리즘을 통합한 단일 TCP를 구현하는 것이다. 앞서 언급한 바와 같이 이에 대해서는 별도의 구현과 관련된 이슈가 존재하겠지만, 통합 시의 요구사항은 세션 중간에 동적으로 다른 변종으로 스위칭 할 수 있어야 한다는 점이다. 따라서 본 논문에서는 간단히 함수 단위로 해당 변종의 함수를 호출할 수 있도록 각 함수의 초반 부분에 분기점을 두는 방식을 사용하였다. 그림 6은 이를 반영한 수도 코드로, 예를 들어 *tcp_cong_avoid()*라

```
ON calling tcp_cong_avoid() function:
    SWITCH(current_tcp)
        CASE RENO:
            goto DEFAULT;
        CASE WESTWOOD:
            tcp_cong_avoid_westwood();
            return;
        CASE CUBIC:
            tcp_cong_avoid_cubic();
            return;
        CASE VENO:
            tcp_cong_avoid_veno();
            return;
    END

    DEFAULT:
        ··· Reno's code ···
```

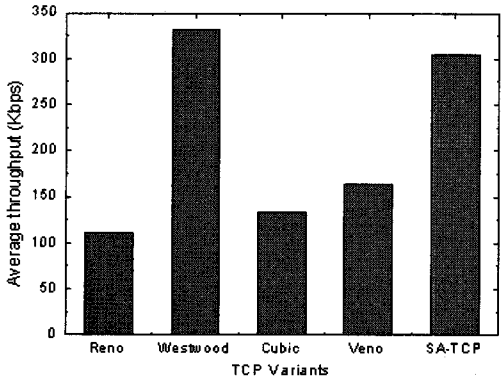
그림 6. 변종 전환의 예

는 함수가 호출될 때 현재 사용하고 있는 TCP의 변종을 확인한 후 해당 변종의 알고리즘을 포함한 함수를 간접 호출하도록 하고 있다. 디폴트로는 Reno의 코드를 수행하도록 하고 있다. 더불어 그림 6의 코드 분기는 모든 함수가 아닌, 변종 TCP가 해당 함수를 변형하여 사용하고 있을 때에 한해 삽입되어 있다. 또한 각 변종 별로 사용하고 있는 내부 변수를 초기화 해주는 *variant_initialize()* 함수를 두어, TCP가 바뀔 때마다 먼저 호출되도록 하였다.

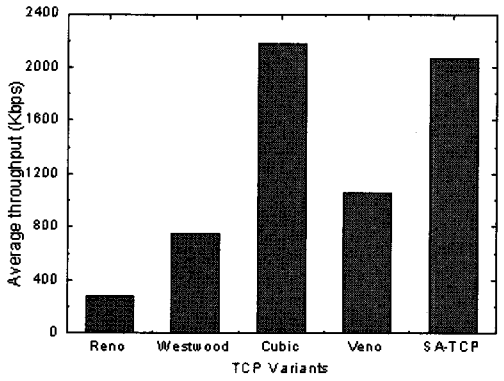
VI. 성능평가

이번 장에서는 네트워크 측정 기법, 결정 맵 그리고 각 TCP 변종들의 알고리즘을 통합한 자동 적응 TCP 프레임워크의 적절한 동작 여부와 그 성능을 측정하기 위해 몇 가지 성능 평가 과정을 거쳤다. 먼저 5장에서 설명한 과정을 NS-2에 그대로 구현하였으며, 각 알고리즘으로의 변환이 제대로 이루어지는지 확인하기 위해 그림 7(a)-(c)의 그래프와 같이 Reno에서 각각 Westwood, Cubic 그리고 Veno로 전환되는 환경 하에서 각 TCP의 평균 처리량을 100초간 측정하였다.

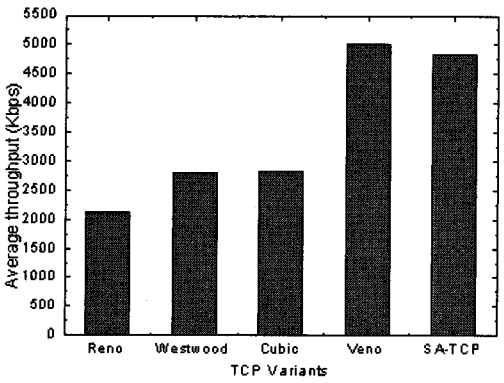
그림 7(a)는 50Mbps의 대역폭, 300ms의 왕복지연, 그리고 3%의 손실률을 가지는 환경에서 측정한 그래프로, Westwood가 가장 높은 처리량을 보이고 있다. 제안하는 자동 적응 기법(SA-TCP로 표기)은 Reno에서 시작하여 5.17초가 지난 후 네트워크 환



(a) 병목링크: 50Mbps, 300ms, 3.0%, 선택 시간: 5.17s



(b) 병목링크: 200Mbps, 500ms, 0.1%, 선택 시간: 5.5s



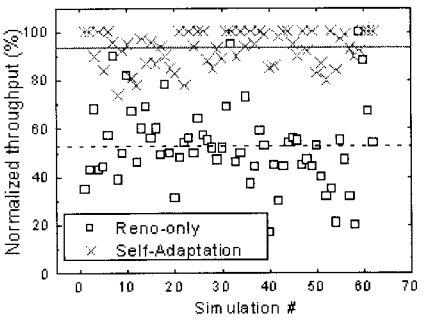
(c) 병목링크: 50Mbps, 80ms, 0.25%, 선택 시간: 5.066s

그림 7. 평균 처리량 비교 그래프

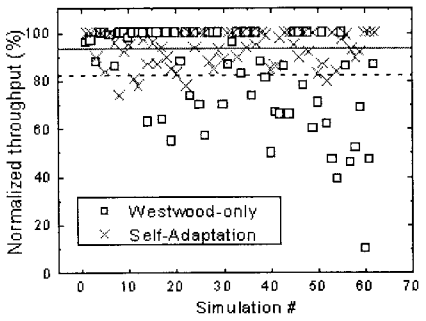
경을 동일하게 측정해내었으며, 그 후 Westwood로 알고리즘을 곧바로 교체하였다. 그림 7(b)와 7(c)는 각각 Cubic 그리고 Veno가 가장 높은 처리량을 보이는 환경이며, 자동 적응 TCP는 마찬가지로 5.5초, 5.066초 후에 네트워크 환경을 측정한 뒤 Cubic과 Veno 알고리즘으로 교체되었다. 자동 적응 TCP의

경우, 초반에 네트워크 환경을 측정하는데 걸리는 시간과 새 알고리즘이 반영되는데 걸리는 시간으로 인해 각 환경의 최적의 TCP보다 약간 평균 처리량은 낮아졌으나, 세 가지 환경에서 모두 꾸준히 높은 처리량을 보이고 있다. 이를 통해 파악할 수 있는 사실은 1) 하나의 변종 알고리즘이 여러 환경에 적응될 수 없다는 점과 2) 본 논문에서 제안한 방법이 세 환경에 모두 적응 가능하도록 적절한 알고리즘 스위칭을 해 준다는 점이다.

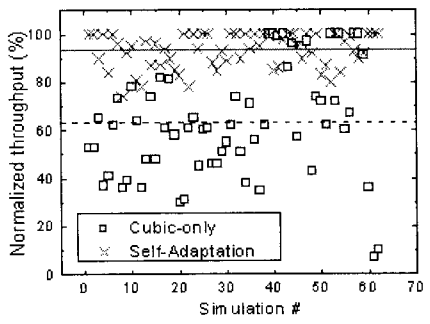
제안한 TCP 프레임워크의 적응력을 보다 일반화하기 위해 병목 링크의 대역폭, 왕복 지연시간, 손실률을 각각 1~1000Mbps, 10~500ms, 0.01~10% 사이에서 임의로 생성하여 각 변종들과 성능 차이를 분석하였다. 그림 8은 각각의 단일 변종만을 사용한 경우와 네트워크 측정을 통해 측정된 환경에 따라 적절한 변종의 알고리즘으로 스위칭한 자동 적응 TCP 프레임워크와의 처리량을 비교한 그래프이다. 동시에 수행된 결과를 편의상 각 변종과 자동 적응 기법(self-adaptation)으로 비교한 네 개의 그래프로 분리하였으며, 매 환경마다 처리량 값의 스케일이 다르게 나타나기 때문에 표준화된 처리량을 사용하였다. 즉, 자동 적응 기법을 포함한 5개의 TCP에서 보인 처리량 중 가장 높은 값을 기준(100%)으로 나머지 값들을 정규화(normalization)한 것이며, 이를 통해 해당 변종의 적응 정도를 파악할 수 있다. 이 결과 그래프에서 주목할 만한 점은 제안하는 기법이 최대 처리량 대비 평균 93%의 처리량을 보이며, 각각의 변종을 잘 활용하여 꾸준히 높은 성능을 보인다는 것이다(그래프 내의 실선으로 표시). 즉, 다양한 환경에서 자신이 보유한 적응 알고리즘으로 낼 수 있는 최대 성능 중 93% 정도의 수준을 항상 기대할 수 있음을 의미한다. 이것은 초기의 네트워크 측정에 의해 적절한 알고리즘으로 변경되기까지 지연되는 시간을 감안한다면 충분히 실용적이라고 판단된다. 반면에 한 가지의 적응 알고리즘만을 사용하는 경우에는 Reno가 평균 약 53%로 가장 낮은 적응력을 보였으며, Westwood가 약 83%로 단일 알고리즘 중 가장 높은 적응력을 보였다. (그래프 내의 점선으로 표시). Westwood가 가장 높게 나타난 까닭은 그림 4에 제시된 성능 맵에서도 알 수 있듯이 실험에서 사용한 네 가지의 변종 중 Westwood가 다른 변종에 비해 적응된 영역이 보다 넓기 때문이다. 그러나 이들 단일 변종들의 적응 정도는 자동 적응 기법에 비해



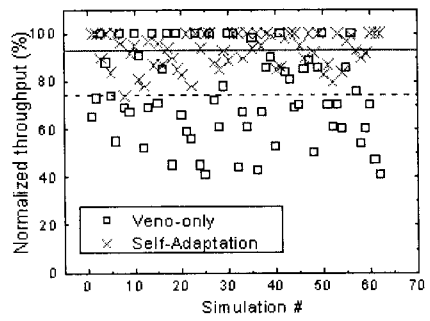
(a) Reno-only vs. Self-Adaptation



(b) Westwood-only vs. Self-Adaptation



(c) Cubic-only vs. Self-Adaptation



(d) Veno-only vs. Self-Adaptation

그림 8. 최대 처리량에 정규화된 처리량 비교

높은 편차를 보이고 있다. 즉 환경에 따라서 성능의 변화가 심하다는 것을 의미하며, 이 역시 한 가지의 적응 방법으로 모든 환경에 적응시키기 어렵다는 사실을 뒷받침 해주고 있다. 결과적으로 본 논문에서 제안한 자동 적응 기법은 기존에 제안되어온 여러 적응 알고리즘을 활용하여 단일 변종만 사용한 경우와 비교했을 때 평균적으로 최적의 성능에 가장 근접할 뿐 아니라 최대 약 40%의 성능 향상을 안정적으로 제공해줌을 알 수 있었다.

Ⅶ. 논의사항

앞서 언급한 바와 같이 적응 알고리즘을 선택하기 위해서는 네트워크 환경의 측정이 수행되어야 한다. 본 논문에서 종단 간 경로 상의 대역폭, 지연시간, 손실률을 측정하기 위해 사용한 기법은 TCP Probe, RTT 측정, LEAST이며, 측정 시간과 정확도가 적절한 변종의 선택과 선택에 따른 성능 상의 이득에 영향을 미치게 된다. 실험 시 환경 측정에 소요된 시간은 대부분의 경우 5~10초 사이였으며, 이것이 적응 상의 오버헤드가 되어 최적의 경우에 못 미치는 원인으로 작용하였다. 그러나 각 측정 기법에 대한 정확도는 별개의 연구 이슈이므로 본 논문에서 논하지 않았으며, TCP에서 사용 가능하며 보다 빠르고 정확한 측정 기법이 TCP 선택에 보다 도움이 될 것임은 자명하다.

두 번째로 결정 맵은 각 변종들의 처리량에 대한 기대 값을 기반으로 하고 있는데, 이들은 모두 NS-2 시뮬레이션을 통해 얻어진 정보이다. 우려되는 사항은 시뮬레이션 성능과 실측으로 얻어진 성능 간의 차이가 존재할 수 있다는 점인데, 이는 NS에 구현된 리눅스 TCP[20]를 사용하여 어느 정도 극복이 가능하다고 판단된다. 이것은 리눅스 TCP의 구현 코드를 시뮬레이터에 그대로 이식한 것으로, 실측과 유사한 성능과 동작을 보이는 것으로 알려져 있다. 따라서 네트워크 환경을 쉽게 제어할 수 있는 시뮬레이터의 장점을 수용하여 각 변종들의 정확한 실측 성능을 측정하는 것이 가능하다. 또한 결정 맵은 TCP 코드 외에 별도의 추가적인 데이터가 필요하다는 것을 의미하는데, 이것은 5장에서 설명한 바와 같이 각 변종마다 해당 환경에 대한 우선순위 정보를 담고 있다. 하나의 우선순위 값을 1 바이트로 표현했을 때 전체 맵의 크기는 약 7.5 Kbytes 정도였다. 여기에 하나의 변종이 추가될 때마다 약 1.87 Kbytes가 늘어난다. 이는 TCP 전체 코드의 크기에 비해 매우 작은 값이며, 우선순위는 2 비트로 표현이 가능하기 때문에 실제로는 1/4로 줄어든 용량으로도 성능 정보를 유지할 수 있다.

마지막으로 Ⅴ장에서 TCP들의 성능을 측정하기 위해 사용된 네트워크 파라미터의 값들의 간격이 이산적으로 떨어져 있음을 알 수 있는데, 실제로 네트워크 측정을 통해 구해진 값들은 연속적인 실수로 나타나기 때문에 보유하고 있는 정보와 얻어진 값이 일치하지 않게 된다. 일반적으로 성능 맵에서 나타난 바와 같이 각 변종이 적응하고 있는 환경은 일정한 영역으로 나타나기 때문에 치명적이지는 않지만, 보유 변종의 수가 늘어난다면 선택의 정확도가 떨어질 수 있다. 이를 보완할 수 있는 방법은 추가적인 측정을 통해 각 파라미터들의 적절한 간격을 얻어내거나, 일단 구축된 맵 정보를 초기 값으로 두고 시스템이 경험을 통해 추가적으로 얻어지는 정보로 맵을 확장해나가는 것이다. 두 방법 모두 최종 결과는 보다 견고해진 맵 정보가 될 것이며 이는 보다 안정적이고 적응 능력을 높이는데 도움이 될 것이다.

Ⅷ. 결 론

본 논문에서는 기존의 TCP 변종들을 바탕으로 종단 간 경로의 환경에 가장 적응이 잘 이루어지는 변종의 알고리즘을 선택하는 TCP 프레임워크를 제안하였다. 이러한 선택을 통한 프로토콜의 적응이 가능하게 하기 위해 본 논문에서는 기존에 연구되어온 대역폭, 지연시간, 손실률의 네트워크 측정 기법들과 TCP 변종들을 하나로 합쳤으며, 여기에 각 TCP들의 성능 정보들을 제공하여 교체 여부를 결정하는 결정 맵을 구성하였고, 이를 통해 세션 중간에 적절한 전송 알고리즘을 선택하여 사용할 수 있도록 하였다. 실험을 통해 우리는 종단 간으로 여러 환경 하에서 꾸준히 높은 성능을 안정적으로 이끌어낼 수 있다는 것을 보였다.

본 논문이 가지는 가장 큰 기여도는 지금까지의 연구들은 대부분 특정 환경에 대한 적응 알고리즘만을 연구해 왔으나 이들을 통합하여 각 변종들의 장점들을 모두 수용할 수 있는 프레임워크를 제공하였다는 점이며, 제안한 방법이 TCP 변종들이 단지 연구로써 만이 아닌 실제로 적절하게 활용될 수 있도록 하는데 중요한 역할을 할 것으로 믿는다.

참 고 문 헌

[1] B. S. Bakshi, P. Krishna, N. H. Vaidya, D. K. Pradhan, "Improving performance of TCP over wireless networks," In Proceedings of *the 17th International Conference on Distributed Computing Systems*, 1997.

[2] P. Sinha, N. Venkitaraman, R. Sivakumar, V. Bharghavan, "WTCP: a reliable transport protocol for wireless wide-area networks," In Proceedings of *5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999.

[3] C. P. Fu, S. C. Liew, "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks," *IEEE Journal of Selected Areas in Communications*, Vol.21, No.2, Feb. 2003.

[4] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," In Proceedings of *IEEE INFOCOM*, 2004.

[5] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *Computer Communication Review*, Vol.32, No.2, Apr. 2003.

[6] I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," In Proceedings of *PFLDnet*, Feb. 2005.

[7] L. S. Brakmo, and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communications*, Vol.13, No.8, Oct. 1995.

[8] M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo, "TCP Westwood: Congestion Window Control Using Bandwidth Estimation," In Proceedings of *IEEE GLOBECOM 2001*, Nov. 2001.

[9] S. Floyd, M. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to the Selective Ac-knowledgement(SACK) option for TCP," *RFC 2883, IETF*, 2000.

[10] P. Patel, A. Whitaker, D. Wetherall, J. Lepreau, and T. Stack, "Upgrading Transport Protocols using Untrusted Mobile Code," In Proceedings of *19th ACM Symposium on Operating Systems Principles*, 2003.

[11] J. Semke, J. Mahdavi, and M. Mathis, "Automatic TCP Buffer Tuning," In Proceedings of *ACM SIGCOMM*, Oct. 1998.

[12] E. Weigle, and W. Feng, "Dynamic Right-Sizing: A Simulation Study," In Proceedings of *IEEE ICCCN*, 2001.

[13] D. Sisalem, and A. Wolisz, "LDA+ TCP-friendly adaptation : A measurement and comparison study," In Proceedings of *International Workshop on NOSSDAV*, Jun. 2000.

[14] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," In Proceedings of *ACM SIGCOMM*, Aug. 2000.

[15] A. Persson, C. A. C. Marcondes, L. Chen, M. Y. Sanadidi, and M. Gerla, "TCP Probe: A TCP with built-in Path Capacity Estimation," In Proceedings of *8th IEEE Global Internet Symposium*, 2005.

[16] R. Kapoor, L. Chen, Li Lao, M. Gerla, and M. Y. Sanadidi, "CapProbe: A Simple and Accurate Capacity Estimation Technique," In Proceedings of *ACM SIGCOMM*, 2004.

[17] M. Allman, W. Eddy, and S. Ostermann, "Estimating Loss Rates With TCP," *ACM Performance Evaluation Review*, Vol.31. No.3, Dec. 2003.

[18] G. W. Cobb, "Introduction to Design and Analysis of Experiments," *Springer*, Mar. 1998.

[19] ns2 Network Simulator version 2.26. http://www.isi.edu/nsnam/ns

[20] D. X. Wei, and P. Cao, "NS-2 TCP-Linux: An NS-2 TCP Implementation with Congestion Control Algorithms from Linux," In Proceedings of *ValueTool'06 - Workshop of NS-2*, Oct. 2006.

황 재 현 (Jae-Hyun Hwang) 정회원

2003년 2월 가톨릭대학교 컴퓨터공학과 졸업

2005년 8월 고려대학교 컴퓨터학과 석사

2005년 9월~현재 고려대학교 컴퓨터학과 박사과정

<관심분야> 커널 네트워킹, 네트워크 프로토콜 설계 및 구현

유 혁 (Chuck Yoo) 종신회원

1982년 2월 서울대학교 전자공학과 학사

1984년 2월 서울대학교 전자공학과 석사

1986년 8월 Univ. of Michigan 전산학 석사

1990년 2월 Univ. of Michigan 전산학 박사

1990년~1995년 Sun Microsystems 연구원

1995년~현재 고려대학교 컴퓨터학과 교수

<관심분야> 운영체제, 시스템 가상화, 네트워크, 멀티미디어 스트리밍