TCP의 동적 적응 프레임워크

황재현° 김세원 유혁
고려대학교 컴퓨터학과
{jhhwang°, swkim, hxy}@kiss.or.kr

Automatic Adaptation Framework of TCP using Variants

Jae-Hyun Hwang° Se-Won Kim Hyuck Yoo
Department of Computer Science, Korea University

요 약

본 논문에서는 기존의 TCP 변종들을 바탕으로 종단간 네트워크 환경에 가장 적응이 잘 이루어진 변종의 알고리즘을 선택하는 TCP 프레임워크를 제안한다. 프로토콜 선택의 문제가 중요한 이유는 모든 네트워크 환경에 적합한 단일 버전의 프로토콜이 존재하지 않기 때문이며, 이것은 각 네트워크마다 TCP의 성능 저하 원인이 서로 다르기 때문이다. 이러한 판단 및 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 end-to-end로 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 믿는다.

1. 서 론

TCP(Transport Control Protocol)는 오늘날 널리 사용되고 있는 신뢰성 있는 전송을 위한 end-to-end 프로토콜이다. TCP가 초기의 유선 인터넷 망을 가정하여 설계된 프로토콜이라는 것은 널리 알려진 사실이지만, 높은 안정성과 성능으로 인해 여전히 다양한 환경 하에서 사용되고 있다. 그러나 최근들어 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 무선 망을 위한 TCP[1], High-BDP 망을 위한 TCP 변종[2][3][4] 등이 있다. 또한 TCP는 내부적으로 휴리스틱에 의존한 알고리즘을 다수 가지고 있기 때문에 이를 좀 더 정교하게 개선한 변종들도 제안되었다[5]. 이처럼 새로운 환경에 TCP를 적응시키거나 TCP의 성능 개선을 위한 연구들이 진행되어 왔지만, 실제로 사용되고 있는 TCP의 버전은 지금까지도 표준 TCP와 SACK option[6] 정도로 한정이 되어 있다. 이것은 우선적으로는 프로토콜의 배포 문제로써, 프로토콜의 배포가 빠르게 이루어지지 않는다는 점과 송신측의 수정만으로 이득을 보기 어렵기 때문에 대부분 상대방과 동일한 프로토콜을 가져야 사용할 수 있다는 점이 그 원인이 될 수 있다. 이에 대한 해결방안으로 모바일 코드를 이용한 프로토콜의 업데이트에 대한 연구가 이루어진 바 있다[7]. 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.

본 논문에서 제안하는 TCP의 동적 적응 프레임워크에 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 종단간으로 송신 측의 수정만을 거쳐 다양한 환경 하에서 높은 성능을 이끌어낼 수 있다는 것을 보였다.

본 논문은 다음과 같이 구성된다. 먼저 2절에서는 TCP 적응 프레임워크의 전체 구조와 설계 목표를 설명한 뒤, 본 논문에서 사용한 네트워크 측정 기법과 TCP 변종에 대해 설명한다. 3절에서는 적절한 변종을 선택하기 위해 본 논문에서 사용한 판단 근거와 기준에 대해 자세히 설명하도록 한다. 4절에서는 변종들의 성능 정보를 측정하고 이를 바탕으로 제안된 기법을 적용, 구현하기 위한 과정을 서술한다. 5절에서는 제안된 기법에 대한 평가 과정을 거치고 그 결과를 분석한 뒤 6절에서 결론을 맺는다.

2. 구조 및 설계 목표

본 절에서는 TCP 적응 프레임워크의 전체 구조에 대해 설명한다. 제안하는 프레임워크의 구조는 일반적인 자동 적응 시스템과 마찬가지로 1) 사용 환경의 측정, 2) 프로토콜 성능

이 논문은 2004년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No.R01-2004-000-10588-0).

데이터, 3) 전송 알고리즘 변경 가능한 TCP 구조로 나뉜다. 먼저 네트워크 환경 측정은 기존에 제안되어 왔던 종단간 네트워크 추정 기법들을 활용하여 현재 사용 중인 네트워크의 특성을 추정하는 부분이다. 이 부분의 설계 초점은 송신 측에서 추정이 가능해야 한다는 것과 측정이 빠르고 정확해야 한다는 것이다. TCP는 end-to-end 프로토콜이기 때문에 현재 사용중인 LAN의 특성보다는 종단간의 경로 상에서의 병목 지점의 특성이 성능에 주로 영향을 주게 된다. 이러한 네트워크 경로는 새로운 연결이 이루어질 때마다 바뀌기 때문에 네트워크 추정이 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 추정 기법이 정확하지 않다면 TCP의 적응 방법이 틀려질 수 있다. 따라서 추정 기법의 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 추정 방법은 기존에 제안 되어온 방식을 활용하였으며, TCP 역시 그 동안 제안되었던 변종들을 바탕으로 몇 가지를 선택하여 통합하였다. 이번 절에서는 본 논문에서 적용한 네트워크 파라미터 및 추정 기법과 TCP의 통합 방법을 간략히 소개하고 다음 절에서 프로토콜의 성능 데이터를 추출하는 과정과 실제 세션에서 적응하는 과정을 자세히 설명하도록 한다.

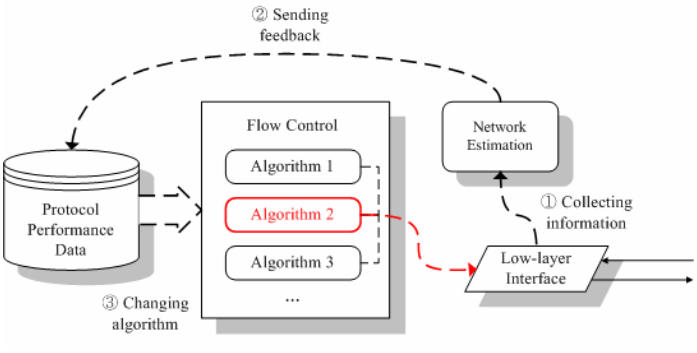


그림 1. 동적 적응 프레임워크의 개념적 구조

2.1 네트워크 추정 기법

네트워크의 특성을 나타내는 지표에는 여러 가지가 있지만 프로토콜의 성능에 영향을 미치면서 종단간 측정이 가능한 요소로는 대역폭, 지연시간, 그리고 손실률이 있다. 우리는 단말에서 바라보는 종단간 네트워크의 특성을 이 세가지 네트워크 파라미터로 정의하였으며, 기존에 제안된 추정 기법 중 송신 측에서 측정 가능한 기법을 TCP 적응 프레임워크에 도입하였다. 각 파라미터를 측정하는데 사용된 기법은 다음과 같다.

- 지연시간 : 기본적으로 지연 정보는 TCP에서 측정된다. RTT(Round-Trip Time)는 중간 라우터들의 큐잉 딜레이를 잘 반영하기 때문에 RTO(Retransmission TimeOut) 값을 설정하는데 사용되어 왔다. RTT는 네트워크의 혼잡 정도를 반영하는데 도움이 될 뿐 아니라, 최근 들어 위성 망과 같이 BDP (Bandwidth-Delay Product)가 높은 망에서 TCP의 효율성 저하 문제가 발생하면서 delay의 측정은 더욱 중요한 의미를 갖는다.
- 대역폭 : 본 논문에서는 네트워크의 중요한 특성 중 하나인 링크 대역폭를 측정하기 위해 TCP Probe[8] 기법을 동적 적응 프레임워크에 도입하였다. TCP Probe는 능동적인 링크용량 추정 기법 중 하나인 CapProbe[9]를 기반으로 한 것으로, TCP 패킷을 이용하여 수동적으로 대역폭 측정이 가능하므로 TCP에 적용이 가능한 측정 기법이다.
- 패킷 손실률 : 기존에 제안된 손실률 측정 기법들은 대부분 수신 측에서 측정이 이루어지며, 비신뢰성 프로토콜에 적용 가능한 것들이었다. TCP와 같은 신뢰성 프로토콜은 재전송과 같은 메커니즘이 포함되어 있기 때문에 기존의 기법들을 적용하기 어려운 단점이 있다. 본 논문에서는 송신 측에서 측정 가능하며 TCP에서 적용 가능한 추정 기법인 LEAST [10]를 도입하였다. 이것은 재전송 횟수를 이용하여 손실률을 측정하는 기법으로 대부분의 TCP 변종에 쉽게 적용 가능하다는 장점을 갖는다.

2.2 TCP 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 migrate-on)으로 인한 오버헤드를 최소화하는 것이다. 구현 방법과 관련하여 효율성 및 확장성과 관련된 다른 이슈가 존재할 수 있겠지만 본 논문에서는 간단히 함수 단위로 코드 분기를 사용하였다.

3. TCP 변종의 결정

네트워크의 환경을 추정한 뒤 현재 환경에 적합한 TCP 알고리즘을 변경하여 사용가능 하다면, 어느 알고리즘을 선택하여 사용할 것인가에 대한 판단 근거와 기준이 필요하다. 선택의 기준이 되는 메트릭은 여러 가지가 있을 수 있겠지만, 본 논문에서는 망의 적응에 초점을 두어 처리량(throughput)으로 한정했다. 즉, 주어진 환경에서 처리량이 가장 높은 TCP를 그 망에 가장 잘 적응된 변종으로 판단하는 것이다. 그러나 TCP의 처리량은 동일한 환경에서도 네트워크의 동적 특성으로 인해 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 2은 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, 지연시간, 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 성능상에 유리함을 쉽게 알 수 있다.

그림 2(b)는 100Mbps의 대역폭, 350ms의 지연시간, 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 우월하다고 볼 수 있는지 판단하기 위해 가설검증(Hypothesis Test)[11] 방법을 도입했다. 가설검증 과정은 두 집단의 평균 및 편차가 주어진 경우, 실제 모집단의 평균을 알 수 없기 때문에 어느 정도 유의 수준을 두고 두 집단간의 우월 정도를 판단하는 통계 기법이다. 그림 2(b)에서 예로 든 TCP2와 TCP3을 위의 검증과정에 적용해 보도록 하겠다. 먼저 실험을 통해 얻어진 두 TCP에 대한 평균과 편차는 다음과 같다.

표 1. 그림 2(b)의 TCP2와 TCP3의 평균 및 편차

	측정횟수	$\bar{x}$	s
TCP2	100	2054.7	34.44
TCP3	100	2179.4	353.07

귀무가설과 대립가설은 각각 H_0: $u_A - u_B \geq 0$, H_a: $u_A - u_B < 0$가 된다. TCP3의 평균값이 높으므로 u_A로 두었으며, 이는 귀무가설이 TCP3가 TCP2보다 우월하다는 것을 의미하고 있다. 위의 샘플들은 1) 독립적으로 선택이 되었으며, 2) 샘플 크기가 충분히 크다는 가정을 만족하므로 t-분포 변환 후 t값을 다음과 같이 계산해낼 수 있다.

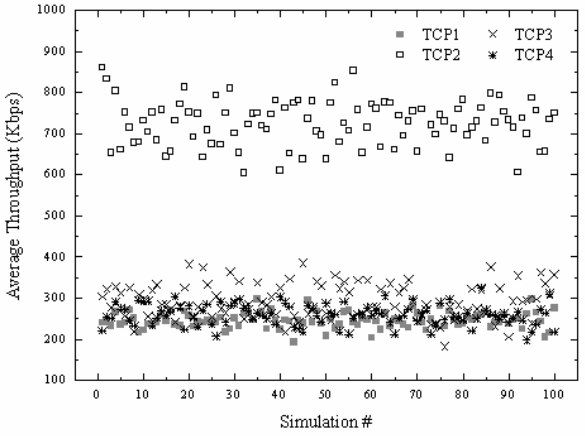
$$t = \frac{(2179.4 - 2054.7) - 0}{\sqrt{(353.07)^2/100 + (34.44)^2/100}} = \frac{124.7}{1258.445385} \approx 0.09909$$

이제 유의확률(p-value)을 구하기 위해서 자유도 df(degree of freedom)를 다음과 같이 계산한다.

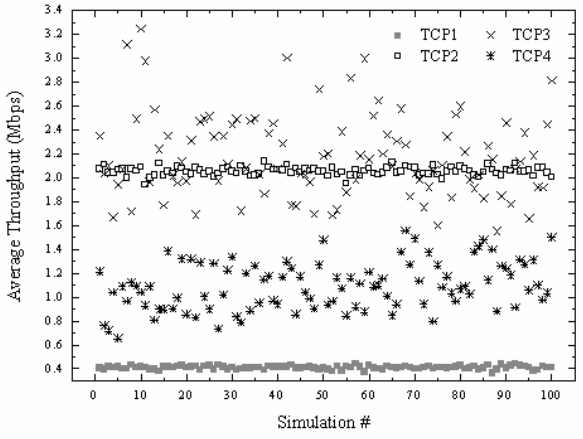
$V_{TCP3} = 11.861136$, $V_{TCP2} = 1246.584249$,

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

여기서 df값은 100으로 내림하였다. 이것을 미리 알려진 t 커브(curve)의 영역에서 도출해보면[11], 120 df를 가진 -0.1의 왼쪽 영역이라는 것을 알 수 있다. 즉, 유의확률이 0.46이 되며, 유의 수준을 0.05라고 했을 때 0.46>0.05 이므로 H_0는 기각되지 않는다. 따라서 TCP3의 성능이 TCP2보다 우월하다고 판단할 수 있다.



(a) 병목링크: 5Mbps, 300ms, 1.5% 손실률



(b) 병목링크: 100Mbps, 350ms, 0.1% 손실률

그림 2. 네 개의 서로 다른 TCP 변종에 대한 평균 처리량

4. TCP 적응 프레임워크의 구현

이번 장에서는 TCP 적응 프레임워크의 구현하는 과정에 대해 설명한다. 한 가지 중요한 문제는 프레임워크 내에 실제로 어느 변종들을 포함시켜 구현할 것인가 이다. 물론 궁극적인 목표는 기존에 제안된 변종들을 모두 포함시키는 것이지만, 본 논문에서는 Reno를 포함하여 최근에 제안된 변종들 중 적응 알고리즘을 소스 코드 수준에서 쉽게 구할 수 있었던 Westwood[5], CUBIC[3] 그리고 Veno[12] 이 네 종류의 TCP만을 포함시켰다. 사실 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 동일 하게 적용되며 따라서 어느 변종이라도 같은 과정을 거쳐 확장가능하다.

4.1 TCP 변종들의 처리량 측정

적절한 TCP 변종을 선택하기 위해서는 주어진 환경에서 각 TCP에 대한 처리량의 기대값을 미리 알고 있어야 한다. 이 과정은 프레임워크를 구현하기 전에 선행되어야 하며, 측정 방법으로는 크게 시뮬레이션을 이용한 방법과 실제 인터넷 환경에서 실측을 하는 방법이 있다. 그러나 실제 네트워크 환경에서 이루어지는 실측의 경우, 네트워크 환경 값들을 직접 조절해 내기가 어렵기 때문에 시뮬레이션을 통한 측정이 좀 더 실용적이라 판단하였다. 따라서 본 논문에서는 다양한 환경에서 각 TCP의 처리량 기대 값을 얻어내기 위해서 NS-2 version 2.26[19]을 이용하여 집중적인 시뮬레이션 실험을 거쳤다. 측정 시 환경에 대한 통제 요인은 앞서 언급한 대역폭, 지연시간, 손실률 세 가지이며 이들의 범위는 다음 표에 정리하였다.

표 2. 네트워크 파라미터의 범위

대역폭 (Mbps)	1, 2, 5, 10, 20, 30, 50, 70, 100, 200, 500, 1000
지연시간 (ms)	10, 20, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500
패킷 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. 처리량 측정을 위한 시뮬레이션 토폴로지

그림 3의 토폴로지에서 variation area의 환경변수를 각각 표 2의 값들로 변화시켜 가며 4가지 TCP 변종에 대한 처리량을 구해내었다. 패킷 크기는 1500bytes로 고정시켰으며, 각 측정마다 200초 동안의 평균 처리량을 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.

4.2 TCP 변종간의 선택

이제 각 TCP의 성능 정보를 토대로 TCP의 결정 맵을 생성해야 한다. 실제 네트워크 환경을 측정한 뒤에는 가공 하지 않은 처리량 정보로 성능의 우월성을 판단하려면 추가 시간이 소요되기 때문에 프로토콜의 결정 맵은 어느 것으로 교체할 것인지에 대한 정보만을 담아야 한다. 먼저, 본 논문에서는 교체의 판단을 위해 각 TCP마다 우선순위를 다음과 같이 정의했다.

표 3. 교체 판단을 위한 TCP 우선순위

우선순위	의 미
1	평균 throughput이 가장 높음
2	우선순위 1보다 throughput은 약간 떨어지나, 교체할 필요가 없음
3	우선순위 1의 TCP로 교체해야 함

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 가설검증 과정을 거친다. 이 결과가 기각되면 우선순위 2를, 가결되면 우선순위 3을 각각 부여한다.

이 과정까지가 미리 만들어져 프레임워크 내 프로토콜 정보로 사용된다. 실제 TCP 세션이 연결되면, 네트워크 측정 기법을 통해 얻어진 파라미터 값을 바탕으로 프로토콜 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 수행하면 된다.

4.3 TCP 변종의 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. 즉, 프레임워크에서 사용하는 TCP 변종들을 정수로 각각 정의한 뒤, *tcp_cong_avoid()*와 같은 함수가 호출될 때 현재 사용하고 있는 TCP의 변종을 확인한다. 이 후 switch문과 같은 분기 코드를 사용하여 해당 변종의 알고리즘을 포함한 함수를 간접 호출하도록 하였다. 디폴트로는 Reno의 코드를 수행하도록 하고 있다. 더불어 이와 같은 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.

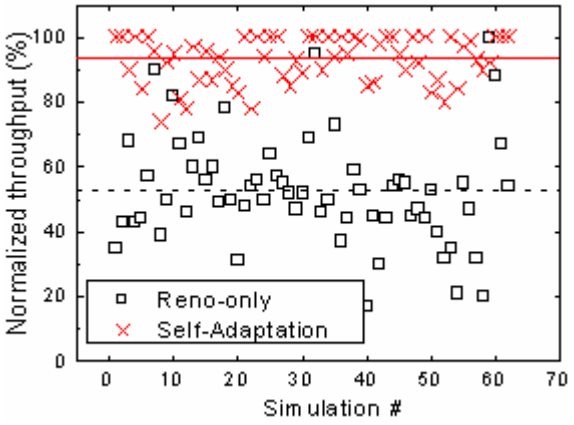
5. 실험 및 분석

이번 절에서는 네트워크 추정 기법, 프로토콜 성능 정보를 반영한 결정 맵, 그리고 각 TCP 변종들의 알고리즘을 통합한 TCP 적응 프레임워크의 성능을 측정하기 위한 실험 과정을 거쳤다. 먼저 4장에서 설명한 과정을 NS-2에 그대로 구현하였으며, 병목 링크의 대역폭, 지연시간, 패킷 손실률을 각각 1~1000Mbps, 10~500ms, 0.01~10% 사이에서 임의로 생성하여 각 변종들과 성능 차이를 분석하였다.

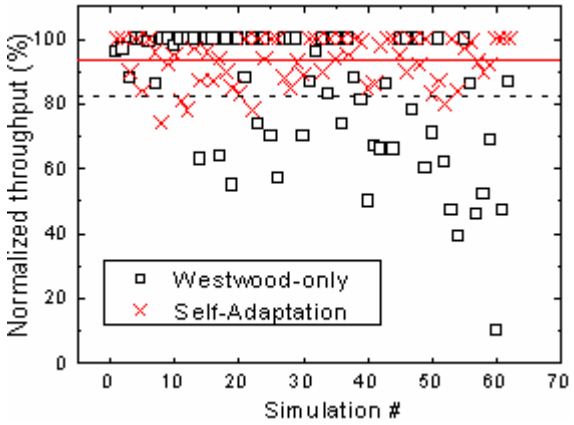
그림 10은 각각의 단일 변종만을 사용한 경우와 네트워크 추정 기법을 통해 측정된 환경에 따라 적절한 변종의 알고리즘으로 스위칭한 적응 프레임워크(그래프 상에서 Self-Adaptation으로 표기)와의 처리량을 비교한 그래프이다. 동시에 수행된 결과를 편의상 각 변종과 적응 프레임워크로 비교한 네 개의 그래프로 분리하였으며, 매 환경마다 처리량 값의 스케일이 다르게 나타나기 때문에 *normalized throughput*을 사용하였다. 즉, 프레임워크를 포함한 5개의 TCP에서 보인 처리량 중 가장 높은 값을 기준 (100%)으로 나머지 값들을 normalization 한 것이며, 이를 통해 해당 변종의 적응 정도를 파악할 수 있다. 이 결과 그래프에서 주목할 만한 점은 제안된 적응 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 환경 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 본 논문에서 사용한 네 가지의 TCP 변종 중 Westwood가 다른 변종에 비해 적응된 환경이 보다 넓기 때문이다. 그러나 이들 단일 변종들의 적응 정도는 제안된 적응 프레임워크에 비해 높은 편차를 보이고 있다. 즉 환경에 따라서 성능의 변화가 심하다는 것을 의미하며, 이 역시 한 가지의 적응 방법으로 모든 환경에 적응시키기 어렵다는 사실을 뒷받침해주고 있다.

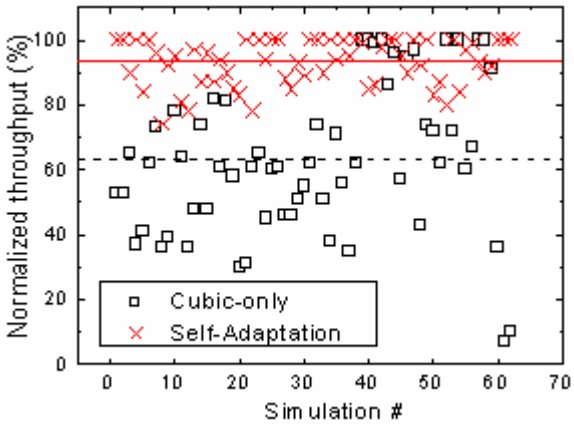
결과적으로 본 논문에서 제안한 TCP 적응 프레임워크는 기존에 제안되어온 여러 적응 알고리즘을 활용하여 단일 변종만 사용한 경우와 비교했을 때 평균적으로 최적의 경우에 가장 근접할 뿐 아니라 최대 약 40%의 성능 향상을 안정적으로 제공해줌을 알 수 있었다.



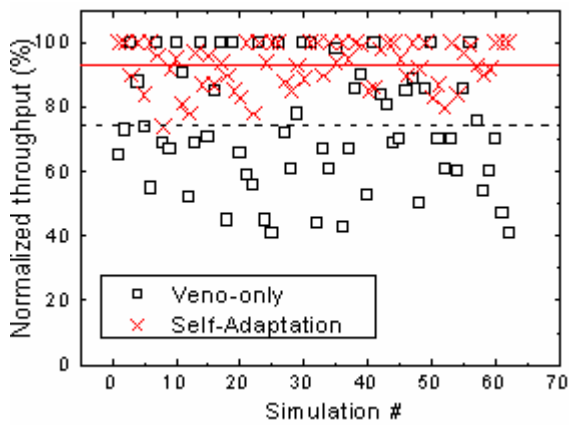
(a) TCP Reno와 적응 프레임워크



(b) TCP Westwood와 적응 프레임워크



(c) TCP Cubic과 적응 프레임워크



(d) TCP Veno와 적응 프레임워크

그림 4. TCP 적응 프레임워크와 각 단일 변종과의 처리량 비교

표 4. TCP 적응 프레임워크와 각 단일 변종 normalized throughput의 평균과 표준편차

	Reno	Westwood	Cubic	Veno	TCP 적응 프레임워크
평균(%)	52.56	82.3	63.1	74.19	92.98
표준편차	16.884	20.735	23.231	18.89	7.146

6. 결 론

본 논문에서는 기존의 TCP 변종들을 바탕으로 종단간의 네트워크 환경에 가장 적응이 잘 이루어진 변종의 알고리즘을 선택하는 TCP 프레임워크를 제안하였다. 이러한 선택을 통한 프로토콜의 적응이 가능하게 하기 위해 본 논문에서는 기존에 연구되어온 대역폭, 지연시간, 패킷 손실률의 측정 기법들과 TCP 변종들을 하나로 합쳤으며, 여기에 각 TCP들의 성능 정보들을 제공하여 교체 여부를 결정하는 프로토콜 결정 맵을 구성하였으며, 이를 통해 세션 중간에 적절한 전송 알고리즘을 선택하여 사용할 수 있도록 하였다. 실험을 통해 우리는 end-to-end로 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.

참고문헌

[1] 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.

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

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

[4] L. Xu, K. Harfoush and I. Rhee, "Binary increase congestion control for fast long-distance networks", In Proceedings of IEEE INFOCOM, 2004.

[5] 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.

[6] S. Floyd, M. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to the Selective Acknowledgement(SACK) option for TCP", RFC 2883, IETF, 2000.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] J. H. Hwang, J. H. Choi, and C. Yoo, "A Decision Maker for Transport Protocol Configuration", In Proceedings of International Conference on Computational Science, May 2006.

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

[15] P. Patel, D. Wetherall, J. Lepreau, and A. Whitaker, "TCP Meets Mobile Code", In Proceedings of USENIX HotOS IX, 2003.

[16] 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.

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

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

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