네트워크 트래픽 제어 연구를 지원하는 테스트베드 구현

(Implementation of a Testbed Supporting the Network Traffic Control)

김 남 군 † 박 재 현 ††
(Namkun Kim) (Jaehyun Park)

요 약 본 논문은 테스트 환경의 재구축에 의한 번거로움을 제거하고 손쉽게 테스트 환경을 재구성할 수 있는 리눅스 기반의 테스트 베드인 Network Traffic Control Test-bed(NTCT)를 구현하고 이를 활용한 실험 결과를 보인다. 본 논문에서 제시하는 NTCT는 원하는 사용자가 지정하는 네트워크 트래픽을 발생시키는 NS2 연동 트래픽 발생기와, 네트워크의 대역폭을 관리하여 실제 네트워크 상황과 유사한 환경을 유지시키는 트래픽 제어기, 그리고 성능 평가를 위한 실시간 네트워크 모니터로 구성되어 있다. 본 논문에서는 제시하는 NTCT를 이용하여 네트워크 성능을 평가하는 예제를 포함한다.

키워드 : 트래픽 발생, 트래픽 제어, 네트워크 모니터링

Abstract This paper proposes architecture of Linux-based Network Traffic Control Test-bed (NTCT) that easily implements reconfigurable network environment. The proposed NTCT consists of traffic generator that uses the simulation results of NS2 simulator, traffic controller using Linux kernel, and traffic monitor. This paper also includes the analysis example using the proposed NTCT.

Key words : traffic generation, traffic control, network monitoring

1. 서 론

인터넷의 보급에 따라 다양한 종류의 네트워크 서비스들의 출현과 더불어, 네트워크상의 트래픽 증가에 따른 네트워크 서비스 품질, 즉 QoS(Quality of Service)에 대한 연구들이 활발히 진행되고 있다. 네트워크 QoS 연구는 공유 대역폭 상에서 트래픽을 제어하여 사용 가능한 대역폭을 보장함으로써 데이타의 지연과 지터(Jitter), 손실 등을 최소화하는 것이 목적이다. 이와 같은 네트워크 QoS 연구를 위해서는 실제 환경에서 발생하는 네트워크 트래픽을 모델링하고 이를 시뮬레이션을 통하여 분석하는 것이 선행되어야 한다. 네트워크 트래픽 모델에는 서비스 종류에 상관없이 모든 트래픽을 동등하게 취급하는 best-effort 방식의 IntServ(Integrated Service) 모델[1]이나 이 모델의 단점을 보완하고자 제안된 DiffServ (Differentiate Service) 모델[2], 그리고 이러한 모델들을 기반으로 한 큐 관리[3,4], 스케줄링, 트래픽 쉐이핑[5,3]에 대한 연구가 주를 이룬다. 모델링된 네트워크 트래픽 시뮬레이션을 위해서는 범용 네트워크 시뮬레이션 도구인 NS2[6]가 가장 널리 사용된다. NS2는 이벤트 시뮬레이터로써 뛰어난 확장성과 비교적 쉬운 구성 형태로 프로그램 된 네트워크 시뮬레이터로서 다양한 종류의 프로토콜들을 시뮬레이션 할 수 있는데, 인터넷을 이루는 유선네트워크는 물론, Ad-Hoc네트워크, 무선랜, 위성통신 등의 무선 네트워크까지 적용할 수 있다. 그러나 QoS 연구결과에 대한 정확한 예측과 네트워크의 성능 평가를 위해서는 NS2와 같은 소프트웨어에 의한 시뮬레이션만으로는 불충분하여, 실제 네트워크를 구성하고 실험을 통한 실증적 분석이 필요한 경우가 많다.

네트워크 트래픽의 모델링과 시뮬레이션을 통한 연구결과를 검증하고 개선하기 위해서는 실제 네트워크 환경에서의 실험을 통한 분석이 요구되는데, 이를 위해서는 실제 서비스 환경과 유사한 네트워크 환경을 구축하고 실제환경과 동일한 트래픽을 발생시키고, 연구에서 제시하는 알고리즘을 적용하여 트래픽 제어와 네트워크 모니터링을 통하여 그 결과를 분석하는 방법을 사용한

† 학생회원 : 인하대학교 정보통신공학부
nkkim@emcl.inha.ac.kr
†† 종신회원 : 인하대학교 정보통신공학부 교수
jhyun@inha.ac.kr
논문접수 : 2005년 12월 23일
심사완료 : 2007년 1월 30일

다. 그러나 다양한 종류의 네트워크 환경을 실제로 구축하기에는 많은 비용과 시간이 소요될뿐더러, 원하는 형태의 트래픽을 발생시키는 것이 매우 힘든 경우가 많다. 따라서 실제 네트워크 환경과 동일한 네트워크를 구성하지 않고서도 원하는 형태의 네트워크 트래픽을 발생시키는 네트워크 트래픽 테스트베드가 매우 유용하게 사용될 수 있다.

본 논문은 테스트 환경의 재 구축에 의한 번거로움을 제거하고 손쉽게 테스트 환경을 하여 관련 연구들에 적용할 수 있는 리눅스 기반의 테스트 베드인 Network Traffic Control Test-bed(NTCT)를 구현하고 이를 활용한 실험 결과를 보인다. 본 논문에서 구현하는 NTCT는 NS2를 이용한 시뮬레이션 결과를 그대로 이용할 수 있는 트래픽 발생기, 결과 분석을 돕는 트래픽 모니터 등으로 구성된다.

본 논문의 구성은 2장에서 본 논문에서 제시하는 NTCT의 구성을 제시하며, 3장에서는 제안된 시스템의 구현과 실험 결과를 보인다. 그리고 4장에서 결론을 맺는다.

2. NTCT의 구성

본 논문에서 제시하는 NTCT는 원하는 사용자가 지정하는 네트워크 트래픽을 발생시키는 NS2 연동 트래픽 발생기와, 네트워크의 대역폭을 관리하여 실제 네트워크 상황과 유사한 환경을 유지시키는 트래픽 제어기, 그리고 성능 평가를 위한 실시간 네트워크 모니터로 구성되어 있으며 그림 1에서 보이는 구성을 가진다.

2.1 NS2 연동 트래픽 발생기

NS2 연동 트래픽 발생기는 네트워크 시뮬레이션 도구인 NS2를 사용하여 테스트 환경에서 다룰 수 없는 다양한 구조의 네트워크 환경을 만들어 시뮬레이션을 하고 그 결과를 테스트 네트워크에 재현함으로써 실제 네트워크에서 발생할 수 있는 트래픽을 발생시킨다. 특히 NS2 연동 트래픽 발생기는 실제 서비스들을 사용하지 않고도 NS2의 시뮬레이션 결과로부터 이러한 서비스들에 대한 트래픽을 생성할 수 있으며, NS2의 시뮬레이션 과정에서 TrafficGenerator 클래스를 사용하여 파레토 분포, 지수 분포, CBR 형태를 갖는 트래픽이 생성 가능하기 때문에 확률분포가 적용된 트래픽을 발생시킨다는 장점을 갖는다. 트래픽 발생기에서는 특정 타입에 일정 길이의 패킷을 주기적으로 전송하도록 하여 실제 네트워크 환경에서 발생할 수 있는 트래픽을 재현하도록 에뮬레이션 하거나 정규 분포나 지수 분포 혹은 균등 분포(uniform distribution)와 같은 확률 분포 모델을 적용하여 TCP, UDP, ICMP, DNS, Telnet등의 프로토콜 타입의 트래픽을 발생시켜 실제 네트워크상과 유사하도록 하는 시뮬레이션을 하도록 한다.

그림 2는 트래픽을 발생시키기 위한 순서도이다. 시뮬레이션의 참가 노드 선정은 처음에 구성된 가상 네트워크에서 트래픽을 재현할 구간에 포함된 노드들을 결정하는 과정을 거치며, trace 분석은 시뮬레이션 결과로 생성된 트레이스(trace) 결과로부터, 선정된 노드에 해당되는 trace 정보를 분석하여 패킷 생성에 필요한 정보를 확인하는 과정을 거친다. 시뮬레이션 결과로부터 트래픽을 발생시키기 위하여는 우선 trace 정보의 'from node'와 'to node' 필드가 모두 가상네트워크의 트래픽 발생 구간에 포함되고 'event' 필드가 '+'인 것만 사용한다. 위의 조건들을 만족하는 trace 정보의 'pkt type', 'pkt size'와 'time' 필드의 데이타를 확인한다.

패킷을 만들기 위하여 raw socket[7]을 사용한다. 우선 패킷의 길이가 'pkt size'인 비어있는 데이타그램을 만들고 'pkt type'에 맞도록 헤더를 만든다. 이 때 'src addr' 과 'dst addr' 필드를 사용하여IP주소와 port 번호

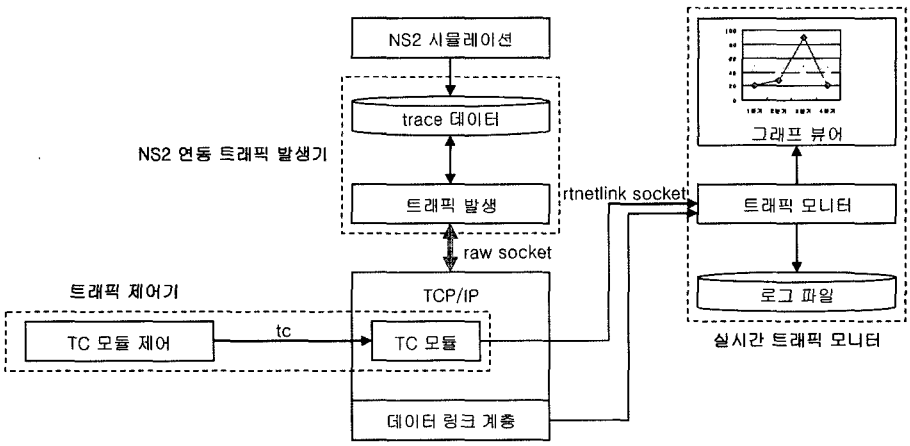


그림 1 NTCT의 구성

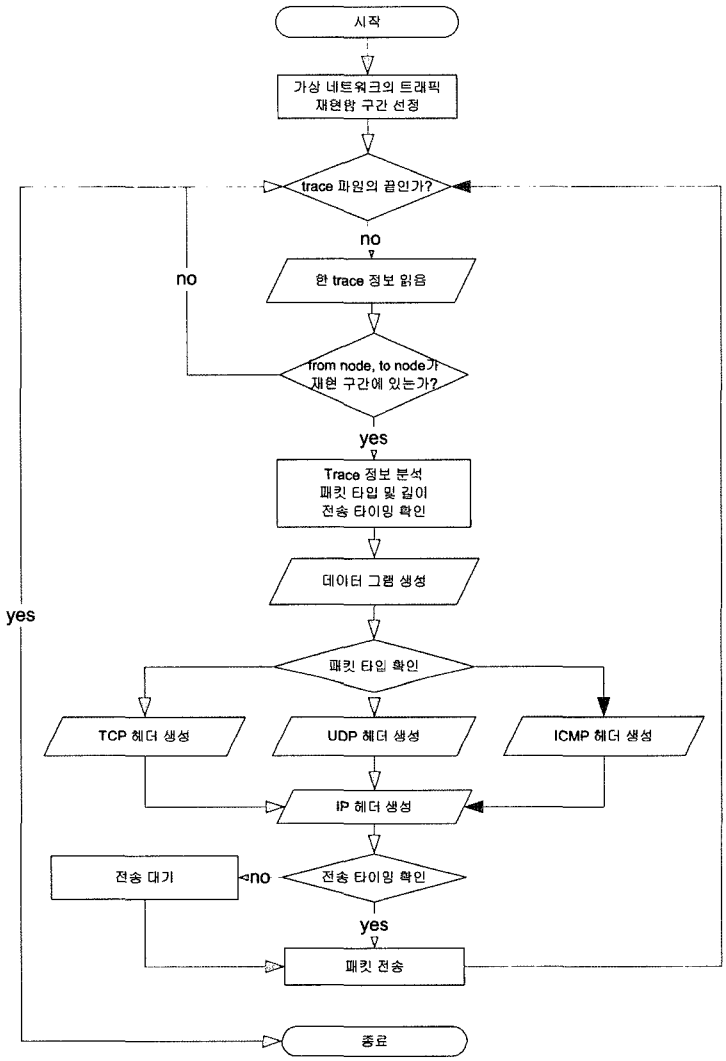


그림 2 NS2 연동 트래픽 발생 순서도

를 지정한다. 'src addr'과 'dst.addr'은 "노드 주소.포트 번호" 형식으로 되어 있다. 노드 주소는 실제 IP 주소 타입이 아니기 때문에, 노드 주소를 대신하여 테스트 네트워크에서 사용되는 IP 주소를 사용하며, 포트 번호 역시 Dynamic 포트 범위의 값을 사용한다.

2.2 트래픽 제어기

NTCT는 리눅스 커널 레벨의 모듈로 지원되는 TC를 이용하여 트래픽 제어기를 구현한다. 트래픽 제어기는 지정된 최대 데이타율 내에서 트래픽이 허용되도록 트래픽량과 트래픽률을 제어하는 트래픽 쉐이핑 기법을 적용하는데, 이를 위하여 token bucket 기반의 TBF 기법을 이용한다[5]. 이때 사용되는 트래픽의 형태는 사용자에 의해 선택된 확률 분포 모델의 형태를 갖도록 tc의 파라미터를 바꿈으로써 트래픽을 제어한다. 표 1은 TBF를 이용하여 트래픽을 제어할 때 적용할 수 있는 인자를 보이고 있다.

제어기에서 지원되는 분포 모델은 정규 분포, 지수 분포, 균등 분포, 코쉬 분포, 파레토 분포 등이 존재하며, 각 확률 분포 모델은 tick_cnt변수를 x축의 좌표로 하여 y축인 확률 밀도 함수인 P(x)를 계산하는 핸들러를 등록한다. 이후 트래픽 제어기를 동작시키는 동안 tick_cnt 변수를 증가시키며 확률 분포의 확률 밀도 함수를 등록한 핸들러를 호출하여 P(x)를 계산하도록 한다. 이 때 계산된 P(x)를 이용하여 해당 분포 모델의 히스토그램 형태의 트래픽 프로파일을 갖도록 제어할 수 있다. 그림 3은 트래픽 제어기의 순서도를 보인다.

표 1 TBF 제어 인자

limit/latency	큐에 삽입된 패킷이 큐에서 나갈 때 까지 기다려야 하는 바이트수/시간
burst/buffer	토큰이 담기는 버킷의 사이즈(바이트 단위)/버퍼의 크기
rate	토큰이 버킷에 담기는 평균 데이타율(bit/s)
peakrate	토큰이 버킷에 담기는 최대 데이타율(bit/s)
mtu	2번째 버킷 크기

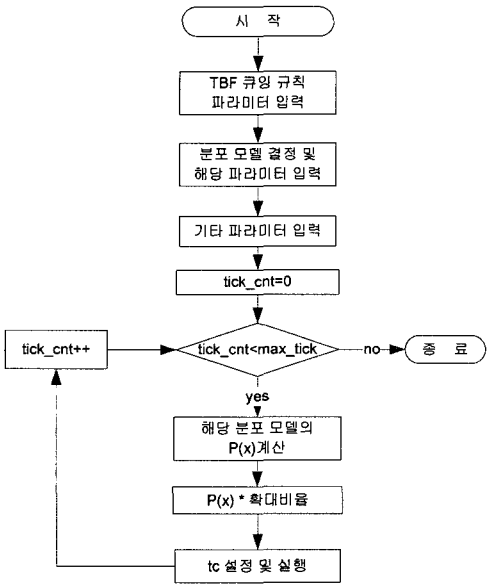


그림 3 트래픽 제어 순서도

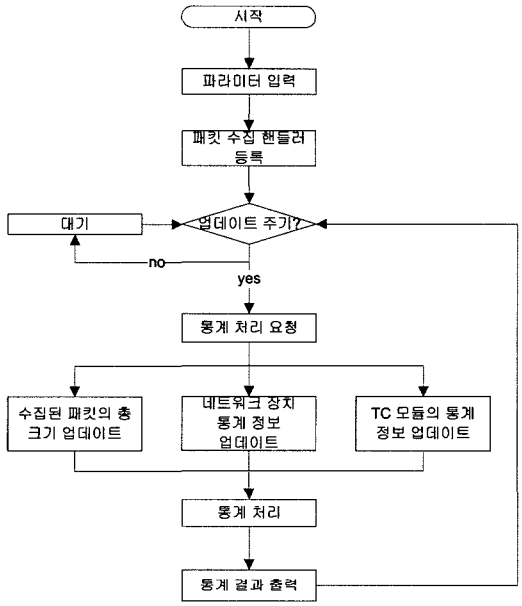


그림 4 실시간 트래픽 모니터 순서도

2.3 실시간 트래픽 모니터

실시간 트래픽 모니터는 발생·제어된 트래픽의 각 데이타율과 테스트가 진행 중인 전체 네트워크의 각 데이타율을 추적하여 통계를 보이도록 하는 것이 목적이며, 트래픽 발생기를 사용하여 발생된 트래픽을 추적할 수 있을 뿐아니라 TC 모듈에서 정의한 큐잉 규칙의 내부 큐의 크기나 스케줄링 정책에 의한 패킷 손실률을 분석할 수 있도록 한다. NTCT의 트래픽 모니터는 libpcap와 proc/net/dev의 bytes필드를 사용하여 현재 사용중인 대역폭과 최대 데이타율 및 평균 데이타율을 측정하며, 그리고 대역폭 제어에 있어서 전송되는 패킷을 위한 큐의 사이즈와 패킷 손실률의 관계를 분석한다. 그림 4는 실시간 트래픽 모니터의 순서도이다. 모니터는 모니터링을 위하여 필요한 모니터링에 참여할 타겟 네트워크 장치와 모니터링하여 얻은 자료를 업데이트 주기와 같은 파라미터를 입력하며, 입력된 네트워크 장치에 연결된 네트워크 망의 트래픽을 관찰하기 위한 패킷 수집 핸들러를 libpcap의 API인 pcap_loop함수를 사용하여 등록한다. 이 과정에서 모니터는 네트워크 장치에 의해서 패킷이 수집될 때마다 수집된 패킷의 총 byte 합계를 구한다. 이렇게 수집된 총계와 proc/net/dev파일의 해당 장치에 속한 transmit 필드 값, rtnetlink 소켓 통신을 이용하여 TC 모듈로부터 net_device->qdisc 구조체 내부에 등록된 stats 구조체에 저장된 패킷의 폐기량, TC 모듈을 통과한 모든 패킷의 총 개수의 정보를 읽고 업데이트한다.

통계 처리 과정은 업데이트 된 각 정보를 바탕으로 각 데이타율과 패킷 손실률을 계산한다. 통계 결과 출력 과정은 계산된 각 통계 결과를 그래프나 로그 파일로 나타내는 과정이다. 패킷 손실률은 로그파일로 저장되고, 각 데이타율은 그래프로 출력된다.

3. 구현 및 실험결과

본 장에서는 실제 테스트 네트워크를 구성하여 NTCT의 각 도구인 NS2 연동 트래픽 발생기, 트래픽 제어기, 실시간 트래픽 모니터를 이용하여 테스트를 수행하고 이에 대한 결과를 제시한다. 그림 2는 NTCT를 네트워크 시스템에 적용한 한 예를 보인 것이다. 각 네트워크의 호스트나 NTCT의 트래픽 발생기에 의해 발생된 트래픽은 NTCT를 거치면서 제어된다. 네트워크 경로 상의 제어되거나 발생된 트래픽을 가지고 NTCT는 포함

된 실시간 트래픽 모니터를 통하여 해당 경로로 전송되는 트래픽의 최소 및 최대, 평균, 현재 데이타율을 측정하고, 트래픽 제어기의 TC 모듈로부터 큐잉 규칙 내부 큐의 크기와 제어된 패킷의 손실률 등의 정보를 확인한다.

본 논문에서 제안하는 NTCT의 시험 환경은 그림 5과 같다. 트래픽은 PC1또는 라우터에서 발생되어 PC2로 전송하도록 한다. PC1, 라우터, PC2 모두는 리눅스 OS를 사용하며 각 호스트간의 연결은 100Mbps Ethernet을 사용한다.

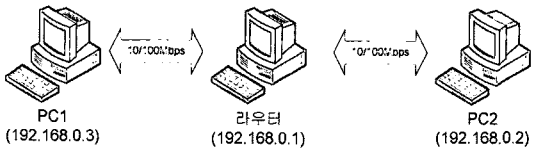


그림 5 테스트 네트워크 환경

3.1 TC 기반의 트래픽 제어기 테스트

NTCT의 트래픽 제어기는 단순히 허용 가능한 최고 데이타율을 설정하고 그 범위 안에서 트래픽을 제어하는 것뿐만 아니라 제어된 트래픽의 프로파일이 사용자 정의가 가능한 형태로 나타낸다. 본 테스트는 이를 위하여 트래픽 제어기에 의해 제어된 트래픽의 프로파일이 정규 분포, 지수 분포, 코쉬 분포, 파레토 분포의 히스토그램의 형태를 갖도록 설정한다. 트래픽을 발생시키기 위한 NS2의 가상 네트워크와 시나리오는 그림 6과 같다. 이 때, 가상 네트워크는 N0부터 N5로 구성된 LAN 환경으로 설정하며, 트래픽 발생을 위하여 N7와 N5, N1과 N4가 각각 통신을 할 때 LAN 환경에 나타나는 트래픽 상황을 사용한다. 따라서 트래픽 발생 참여 노드는 N0, N1, N4과 N5이다.

그림 6의 테스트 네트워크에서 PC1은 NS2 시뮬레이션 결과를 사용하여 패킷을 발생시켜 PC2로 전송하며, 전송 경로상의 라우터에서 트래픽 제어 및 500ms 주기로 모니터링을 한다. 표 2는 각 확률 분포 모델에 적용

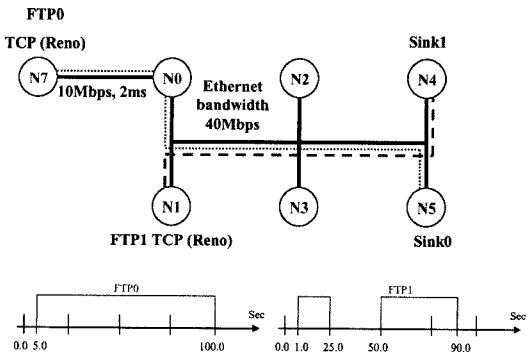


그림 6 확률 분포 모델의 트래픽 제어를 위한 가상 네트워크 및 시나리오

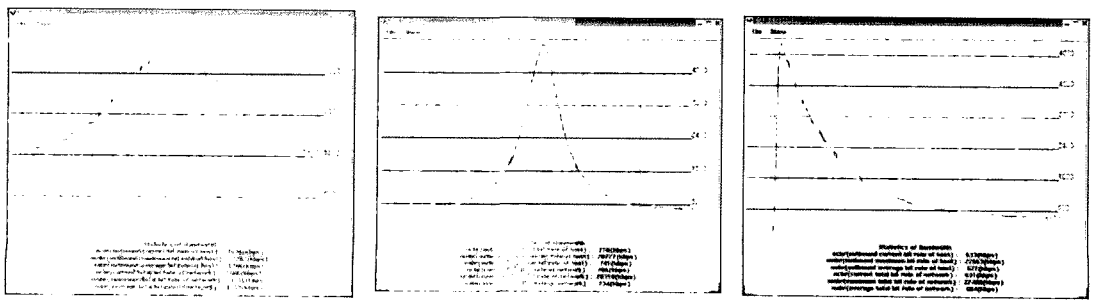
표 2 각 확률 분포 모델에 적용된 파라미터

정규 분포		코쉬 분포		지수 분포	
분산	400	분산	8	모수(λ)	1
평균(tick)	100	평균(tick)	100	tick 증가	0.05
확대 비율	1000	확대 비율	10000	확대 비율	50

된 파라미터를 정리한 것으로써 정규 분포, 코쉬 분포, 지수 분포는 1tick을 500ms로 설정하였다. 그림 7은 각 확률 분포 별 모니터링 결과를 보이는 것으로 제어된 트래픽의 형태가 각 확률 분포가 같은 히스토그램과 동일한 모습을 나타낸다. 따라서 제어된 트래픽 프로파일이 사용자가 원하는 형태로 나타날 수 있다.

3.2 NTCT를 이용한 패킷 손실률 분석

본 테스트는 NTCT 과정을 전체적으로 적용한 것으로써 라우터의 대역폭 제어에 따른 패킷의 손실률을 테스트한 것이다. 라우터에서 대역폭 제어 시 허용되는 최대 데이타율과 큐의 길이에 의해서 처리되는 패킷 손실률이 영향을 받는다. 본 테스트를 위한 가상네트워크와 시나리오는 그림 8과 같다. 그림 8로부터 노드 N0, N2, N4, N5는 ftp서비스에 참여하며 N1, N3는 cbr서비스에 참여한다는 것을 알 수 있다. 여기서 트래픽 발생을 위



(a) 정규 분포 적용 (b) 코쉬 분포 적용 (c) 지수 분포 적용

그림 7 확률 분포 별 트래픽 제어 결과

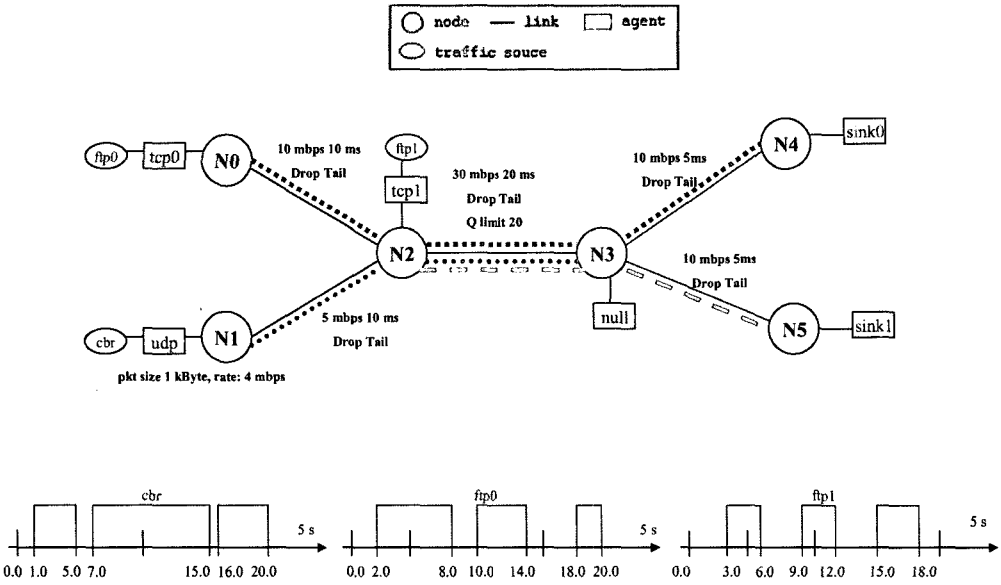


그림 8 트래픽 발생 가상 네트워크

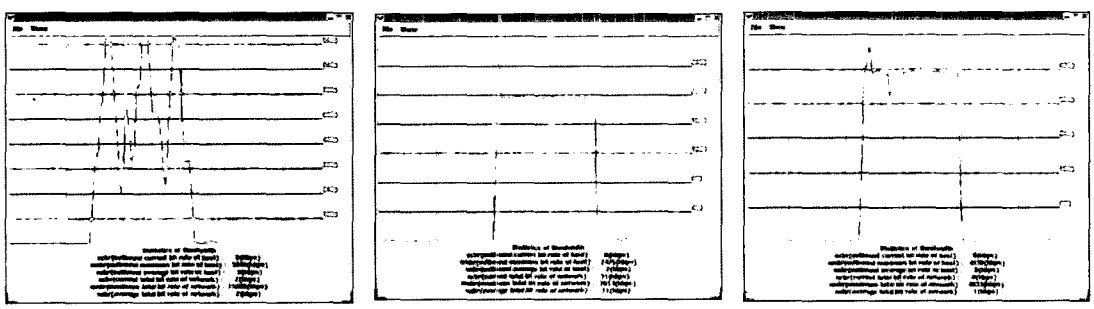
표 3 트래픽 제어를 위한 조건

큐잉 규칙	Token Bucket Filter(TBF) rate(Mbps)=제어를 하지 않은 경우, 2,4,6,8,10 limit(Kbytes)=100 burst(Kbytes)=50
확률 분포 모델	균등 분포 1 tick=500ms a(시작)=1 tick, b(종료)=39 tick

하여 사용될 구간은 라우터 노드 N2와 N3로 이루어진 부분이다.

그림 9는 N2와 N3 구간의 시뮬레이션 결과를 사용하여 PC1에서 발생된 트래픽이 라우터를 거쳐 PC2로 진행되는 과정에서 표 3과 같은 조건으로 라우터에서 트래픽을 제어한 경우를 보여준다. 라우터를 거쳐 PC2로 전달되는 트래픽의 결과를 라우터의 해당 네트워크 장치에 연결된 실시간 트래픽 모니터를 사용하여 확인한 결과이다. 그림 9의 (a)는 라우터의 해당 네트워크 장치에 대역폭 제어를 하지 않은 것으로, 최대 데이타율이 10Mbps이다. (a)의 경우를 제외한 경우에서 정해진 대역폭을 넘는 경우가 있음을 보이지만 대부분은 지정된 대역폭내에서 트래픽이 제어되었다.

그림 10은 앞서 이루어진 테스트를 10회 반복 수행 후 rate 값에 따른 패킷 손실률의 관계를 보인 것으로, PC1에서 트래픽을 발생시켜 라우터에서 트래픽을 제어하도록 하며, 라우터에서 제어된 결과를 모니터링 하여 폐기된 패킷량과 손실률을 로그 파일에서 확인한다. 이때 사용된 내부 큐의 크기는 limit와 burst의 차이로써 50kbyte로 설정한다. 결과를 보면, 적용한 제어대역폭이 작을수록 패킷 손실률은 커지는 것을 알 수 있는데, 이는 제어대역폭이 작을수록 내부 큐에서 외부로 나가는



(a) 제어를 하지 않음 (b) 최대 데이타율 2Mbps (c) 최대 데이타율 4Mbps

그림 9 제어된 트래픽의 모니터링 결과

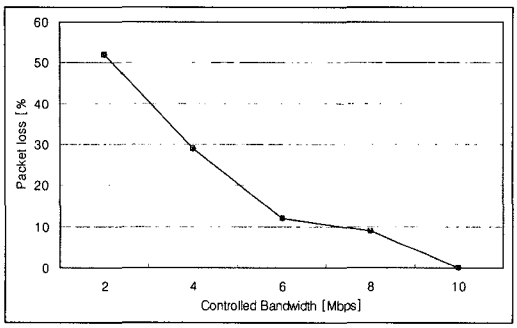


그림 10 대역폭 제어에 따른 패킷 손실률

패킷의 양이 적어지게 되어 큐가 차있는 상태가 많기 때문이다.

4. 결 론

본 논문에서는 네트워크 테스트 환경의 재구축에 의한 번거로움을 피하고 쉽게 네트워크 테스트 환경을 재구성할 수 있는 리눅스 기반의 테스트 베드인 NTCT를 구현하였다. NTCT는 NS2 연동 트래픽 발생기, TC를 적용한 트래픽 제어기, 실시간 트래픽 모니터를 포함하고 있다. NS2 연동 트래픽 발생기는 네트워크 시뮬레이션 도구인 NS2를 사용하여 테스트 환경에서 다룰 수 없는 다양한 구조의 네트워크 환경을 만들어 시뮬레이션하고 그 결과를 테스트 네트워크에 재현함으로써 실제 네트워크에서 발생할 수 있는 트래픽을 발생시킬 수 있다. 그리고 트래픽 제어기는 리눅스의 TC 모듈을 사용하기 때문에 필요에 따라 TC 모듈에서 제공하는 여러 스케줄링, 큐잉 기법들을 적용하며 제어된 트래픽의 프로파일이 사용자 정의가 가능한 형태를 갖도록 한다. 마지막으로 실시간 트래픽 모니터를 포함하여 런타임 동안 발생되거나 제어된 트래픽의 결과를 측정하여 트래픽의 변화를 보다 자세하게 분석할 수 있으며 제어기에 의한 패킷 손실률을 측정하여 트래픽 제어로 인한 서비스질의 보장 정도를 측정할 수 있다.

참 고 문 헌

[1] R. Braden, D. Clark and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, IETF, June, 1994.
[2] S. Black, "An Architecture for differentiated services," IETF RFC 2475, 1998.
[3] B. Braden, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, April 1998.
[4] S. Floyd and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, V.1 N.4, pp. 397-413, August 1993.
[5] B. Forouzan, "Data Communication and Networking," 3rd Ed, McGraw-Hill, 2003.
[6] K. Fall and K. Varadhan, "The ns2 Documentation," A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, 2005.
[7] W. Richard Stevens, "Unix Network Programming," Volume 1, Second Edition, Prentice. Hall PTR, 1998.

김 남 군
2004년 인하대학교 컴퓨터공학부 학사
2006년 인하대학교 정보통신공학부 석사 졸업. 현재 삼성전자 근무. 관심분야는 임베디드 소프트웨어, 컴퓨터네트워크

박 재 현
1986년 서울대학교 제어계측공학과 학사
1988년 서울대학교 제어계측공학과 박사
1994년 제어계측공학과 박사. Univ. of Michigan 연구원, United Technology Corp. 연구원. 현재 인하대학교 정보통신공학부 교수. 관심분야는 임베디드시스템, 실시간시스템, 컴퓨터네트워크