특집 10

ns-2와 네트워크 시뮬레이션 방법론

목 차

최선웅 · 장영민
(국민대학교)

1. 서 론

본고에서는 네트워크 시뮬레이션 방법에 대하여 고찰한다. 네트워크 시뮬레이션이란 실제 네트워크 환경을 컴퓨터 프로그램으로 가상으로 구현하고, 이를 이용하여 모의실험을 수행하는 것을 말한다. 즉, 링크, 노드, 장비 모델들을 소프트웨어로 구현하고, 실제 장비를 구성하듯이 프로그램 상에서 구현하고 실험한다. 이와 같은 네트워크 시뮬레이션을 활용하면, 적은 비용으로 많은 이점을 얻을 수 있다. 실제 시스템을 구현하여 성능 평가를 할 경우에는 많은 경비와 시간이 소모되나 시뮬레이션을 사용함으로써 경비 절감 효과를 얻을 수 있다. 실제로는 수행하기 어려운 신규 서비스의 테스트나, 네트워크 구성 변경과 같은 관리적인 작업들, 더 나아가서는 새로운 아이디어를 추가한 프로토콜의 성능 테스트와 같은 작업들을 실제로 구현하여야 하는 어려움 없이 미리 수행해 볼 수 있도록 도움을 준다.

네트워크 시뮬레이션에 사용되는 대표적인 시뮬레이터로는 ns-2[1]가 있다. ns-2는 네트워크 시뮬레이션을 목적으로 개발된 이벤트 기반(event-driven)의 시뮬레이터이다. ns-2를 사용하면 다양한 유무선 네트워크를 시뮬레이션 할 수 있다. TCP, UDP, FTP, HTTP, 라우팅 프로토콜, 멀티캐스팅 프로토콜, RTP, SRN 등과 같은 다양한 인터넷 프로토콜을 비롯하여, 애드 혹(ad-hoc) 네트워크, 센서 네트워크, WLAN, Mobile IP, 위성 네트워크 등의 무선 네트워크을 시뮬레이션 할 수 있다. ns-2 시뮬레이터는 현재도 계속해서 기능이 추가되고 있으며, 성능 개선 작업도 진행되고 있다. 소스 코드가 공개되어 있기 때문에, 사용자들이 개발하여 공유하는 모델들이 상당히 많고 최신 기술이 반영되어 유포되는 속도가 매우 빠르다. ns-2는 네트워크 분야의 연구 논문 작성 시에 가장 널리 사용되고 있다.

1.1 ns-2 역사 및 현황

ns는 1989년 'REAL'이라는 네트워크 시뮬레이터에 기반을 두고 개발이 시작되었다. REAL은 1988년 UC Berkeley에서 패킷 스위칭 방식의 네트워크에서의 데이터의 흐름(flow)과 혼잡제어(congestion control) 등이 어떻게 일어나는지에 대한 연구를 위해 만들어졌다. TCP 등의 흐

름 제어 프로토콜(flow control protocol)을 에뮬레이션 할 수 있는 몇 개의 모듈들과 스케쥴링 기법들이 구현되어 있다. 여기에 기반으로 두고 개발되기 시작한 ns는 그 동안 많은 발전을 거쳐 지금은 컴퓨터 네트워크의 다양한 영역에 대한 시뮬레이션 모듈을 제공하고 있다.

1995년 DARPA의 지원을 받아 USC/ISI, Xerox PARC, LBNL, UCB가 함께 참여한 VINT 프로젝트를 통해 ns의 개발이 진행되었고, 1996년 오브젝트 개념이 도입된 ns-2가 발표되었다. 현재는 DARPA의 SAMAN 프로젝트와 NSF의 CONSER 프로젝트에 의해 개발이 진행되고 있으며, ACIRI를 포함한 여러 연구 단체들과 함께 개발되고 있다. ns-2는 구성이 추가되거나 변경될 때마다 새로운 버전을 홈페이지를 통하여 발표하고 있다. 현재는 2007년 9월에 발표된 ns-2.32 버전까지 나와 있다.

ns-2는 인터넷을 통해 무료로 다운받아 사용할 수 있으며, 현재 50여 개국의 사용자들과 600여 개의 기관에서 연구와 교육의 목적으로 사용하고 있다. 많은 사용자들과 연구자들이 ns-2를 사용함과 동시에 개발에 참여할 수 있으며 이들에 의해 만들어진 다양한 contribution 코드들이 존재한다. 이런 작업들이 ns-2 패키지에 반영될 수 있으며 이렇게 많은 사람들이 개발에 함께 참여할 수 있다는 점이 ns-2를 발전시키는 밑거름이 된다.

ns-2는 다양한 플랫폼에서 설치 및 사용이 가능하다. FreeBSD, Linux, SunOS, Solaris 등의 대부분의 UNIX 계열 운영체제에서 작동이 가능하며, Windows 9x/2000/XP 에서도 Cygwin 에뮬레이터 등을 통하여 사용이 가능하다. ns-2는 실행에 반드시 필요한 구성요소들과 부가적인 기능을 하는 구성요소들을 모아 all-in-one 패키지로 제공하고 있는데, 이를 사용하기 위해서는 200MB 정도의 저장 공간이 필요하다.

1.2 ns-3 프로젝트

ns-2를 향상시키기 위해 핵심 아키텍처부터 다시 작성하는 것을 목표로 하는 ns-3 프로젝트가 진행 중이다[2]. ns-3 프로젝트는 높은 모듈화, 병렬 처리 기술 등을 통한 대규모 시뮬레이션 기능 제공, 융통성 있는 트레이싱 및 통계 기능 제공 등을 목표로 하고 있다. 2006년 7월 1일 공식적으로 시작되었으며 주로 연구와 교육용을 목표로 하고 있다. ns-3도 공개 무료 소프트웨어이며 2007년 10월 15일 현재 ns-3 3.0.7 버전이 나와 있고 추후에 지속적으로 802.11, 무선 라우팅 프로토콜, TCP 등에 대한 추가적인 버전이 나올 예정에 있다. 아직은 초기 단계라서 심도있는 연구는 ns-2를 주로 사용해야 한다.

1.3 다른 네트워크 시뮬레이터들

1.3.1 QualNet[3]

Qualnet은 90년대 중반 UCLA에서 개발한 시뮬레이션 언어인 parsec으로 개발된 시뮬레이터인 glomosim에 기반을 두고 있다. glomosim은 99년 2.0버전이 나온 이후 개발인 멈추었고, 이를 기반으로 한 상용 시뮬레이터인 QualNet이 출시되어 발전해 오고 있다.

QualNet은 시뮬레이션과 에뮬레이션을 통해 네트워크의 성능을 예측하는 네트워크 모델링 소프트웨어이며, 맞춤 네트워크 모델링과 시뮬레이션을 위한 다양한 기능을 제공하는 종합적인 툴이다. 사용자들이 클릭과 드래깅을 통하여 네트워크 노드의 지리적 분배, 물리적 연결과 매개변수의 구성, 네트워크 계층 프로토콜과 트래픽의 특징을 정의할 수 있도록 하는 모델 구성 툴을 제공하고, 시뮬레이션 과정을 보여 주는 시각화 기능 등을 제공한다.

1.3.2 OPNET[4]

1986년 미 국방성의 의뢰를 받아 MIT에서 개

발한 네트워크 시뮬레이터를 상용화하여 지금까지 꾸준히 발전시켜온 제품이 바로 OPNET Technologies, Inc의 OPNET Modeler이다. OPNET Modeler의 노드와 프로토콜들은 계층적 구조의 클래스들로 객체 지향적으로 모델링되어 있다. 그리고 광범위한 라이브러리로서 Multi-tier Application, Voice, HTTP, TCP, IP, OSPF, BGP, EIGRP, RIP, RSVP, SNA, Token Ring, Frame Relay, FDDI, Ethernet, ATM, IEEE 802.11 무선랜, MPLS, PNNI, DOCSIS, UMTS, IEEE 802.16e 등이 제공된다. 또한 OPNET Modeler는 라우터, 스위치, 워크스테이션, 패킷 발생기 등을 포함하여 다양한 벤더들의 장비 모델들에 대한 표준 모델 라이브러리를 제공한다.

1.3.3 OMNeT++[5]

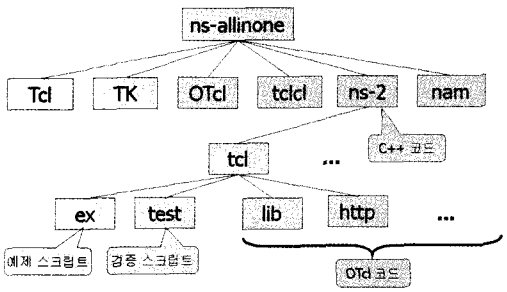
OMNeT++은 강력한 GUI 환경을 제공하는 컴포넌트 기반의 시뮬레이터이다. IT 시스템, 큐잉 네트워크, 하드웨어 구조, 비즈니스 프로세스 등의 시뮬레이션에 많이 사용되어 왔고, 최근 들어 학문적인 용도로도 사용자층이 점차 넓어지고 있는 추세이다. 비상업적인 용도로는 무료로 사용할 수 있다.

2. ns-2의 구조

2.1 디렉토리 구조

ns-2 홈페이지에서 ns-2 관련 소프트웨어를 다운로드 받을 수 있다. 빠른 설치를 돕기 위해서 관련 프로그램들을 하나의 패키지로 묶은 all-in-one 패키지를 제공하고 있다. ns-2 all-in-one 패키지를 다운받아 설치하면, ns-2를 실행시킬 수 있다.

(그림 1)은 ns-2 all-in-one 패키지의 디렉토리 구조를 나타내고 있다. 패키지를 설치하면 아래와 같이 tcl, tk, tclcl, ns-2, otcl, nam, xgraph 등의 디렉토리가 생성되고 각 디렉토리마다 구성요소에 필요한 파일들을 포함하고 있다. ns-2의 핵심적인 소스코드들은 ns-2 디렉토리 아래에 존재한다. 다수의 디렉토리와 파일이 존재하는데, 각각은 여러 가지 프로토콜 및 알고리즘을 구현한 C++ 코드들이다. ns-2 디렉토리의 하위 디렉토리 중 tcl만은 Tcl 스크립트들이 존재하는데, 아래에는 OTcl 라이브러리와 여러 예제 프로그램 등이 OTcl 코드로 존재한다. nam은 네트워크 애니메이션 도구로 시뮬레이션의 결과를 그래픽 사용자 인터페이스를 사용하여 보여줄 수 있는 도구이고, xgraph는 X-Window 응용프로그램으로 시뮬레이션 결과를 분석해서 그래프 형태로 그려볼 수 있는 도구이다.



(그림 1) ns-2 디렉토리 구조

2.2 ns-2 내부 구조

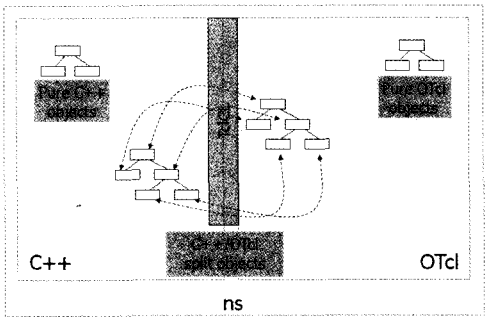
ns-2 시뮬레이터는 C++과 OTcl이라는 두 개의 언어를 사용하여 구현되어 있다. 두 가지 언어를 함께 사용하는 이유는 기본적으로 네트워크 시뮬레이션의 효율성을 높이기 위해서이다. C++은 소스코드를 컴파일 한 후에 실행하는 과정을 거치는 컴파일러형 언어(compiled language)이다. 따라서 실행 시간이 빠르다는 장점과, 소스코드에 변경되는 사항이 있을 때마다 컴파일을 다시 해야 한다는 단점이 있다. 반면에, OTcl은 컴파일이 필요 없는 인터프리터형 언어(interpreted language)라서, C++에 비해 실행

시간은 느리지만 변경되는 사항이 있을 경우에도 부가적인 과정 없이 바로 실행할 수 있다는 장점을 지닌다.

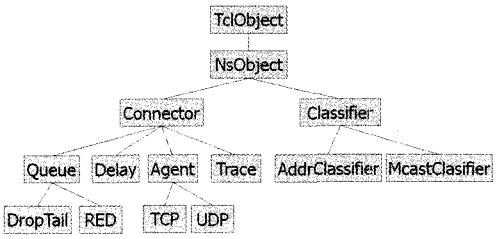
ns-2를 사용하여 네트워크 시뮬레이션을 하기 위해서는 패킷, 헤더, 프로토콜, 알고리즘 등의 자세하고 복잡한 데이터의 처리 방법에 대한 구현이 필요함과 동시에, 네트워크를 구성하는 시나리오를 변화시키면서 시뮬레이션 하여야 하는 작업도 필요한데, 이러한 요구 조건이 두 언어를 함께 사용함으로써 충족될 수 있다. 프로토콜 구현 등의 복잡하고 빠른 속도가 중요한 과정은 C++로 작성하여 컴파일해두고, OTcl로는 네트워크를 구성하고 이벤트를 설정하는 역할을 수행하여 C++로 구현된 프로토콜과 알고리즘 등을 제어함으로써 다시 컴파일하는 과정 없이 시뮬레이션 시나리오를 변경할 수 있게 된다. 이러한 과정을 통하여 전체적인 시뮬레이션 시간이 줄어들어 실행효율이 높아진다.

한가지 더 주목할 점은 C++과 OTcl 모두 객체 지향 언어라는 점이다. 객체 지향 언어를 사용하여 작성함으로써 구성 요소들이 각각의 모듈로 존재하게 되어 코드의 재사용성과 유지보수가 좋아진다는 장점을 지니고 있다. 그러므로 OTcl은 C++ 오브젝트를 사용하기 위한 용도나 변수들의 값을 변경하여 시뮬레이션을 수행시키는 부분을 구현하는데 사용하고, C++은 패킷을 처리하는 알고리즘을 구현하는데 주로 사용한다.

이렇게 두 언어가 상호 보완적으로 작동하기 위해서는 두 언어를 연결해주는 작업이 필요한데 그 부분을 tclcl 라이브러리가 담당하고 있다. ns-2는 C++ 클래스 계층 구조와 OTcl 클래스 계층 구조를 모두 가지는데, 이는 tclcl에 의해 연결되며 두 계층 구조는 유사한 모습을 보인다. 하지만 두 계층구조가 완전히 대응되는 것은 아니고 둘 사이에 연결이 필요 없는 부분에 대해서는 각자의 영역에서만 존재하는 클래스들이 있



(그림 2) OTcl과 C++ 클래스의 상호관계



(그림 3) ns-2 클래스 계층 구조

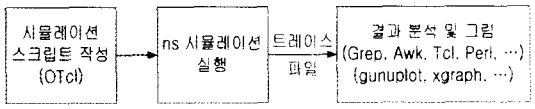
게 된다. (그림 2)는 OTcl과 C++의 객체들이 대응되는 모습을 보이고 있는데 두 개의 공간에서 연결되어 공유되는 객체가 있는 반면 각자의 영역에서만 사용되는 객체도 있음을 나타내고 있다.

(그림 3)은 ns-2 내부의 클래스 계층 구조 중 일부를 보여준다. 우선, TclObject 클래스는 모든 클래스의 기초가 된다. 네트워크 구성요소들은 NsObject 클래스에 의해 구현되며, NsObject 클래스의 하위 클래스에는 Connector 클래스와 Classifier 클래스가 있다. Connector 클래스는 오직 하나의 데이터 출력 경로를 가지며, 그 하위 클래스로는 Agent, Delay, Queue, Trace 등의 클래스가 존재한다. 이들 또한 각각의 하위 클래스들을 가지고 있다. 반면에, Classifier 클래스는 다수의 데이터 출력 경로를 가지며, 그 하위 클래스로는 AddressClassifier 클래스와 MCast Classifier 클래스가 있다. 이들은 라우팅 구현 시 유니캐스트, 멀티캐스트 방법에 사용된다.

3. ns-2 시뮬레이션 방법

3.1 시뮬레이션 과정 개요

ns-2를 사용하여 실제로 네트워크 시뮬레이션을 수행하는 과정에 대하여 알아보자. 만일 기존의 프로토콜을 수정하거나 새로운 프로토콜을 제안하고자 하는 경우에는 ns 시뮬레이션을 하기 전에 C++ 코드를 수정하여 재컴파일하는 과정이 필요하지만, 본 고에서는 기본적으로 제공되는 ns-2 기능만을 사용하는 시뮬레이션으로 한정한다.



(그림 4) ns-2 시뮬레이션 과정

(그림 4)는 기본적인 ns-2의 시뮬레이션 과정을 보여 준다. 가장 먼저 시뮬레이션 하고자 하는 시나리오를 기술하는 스크립트 파일을 작성하여야 한다. 스크립트 파일은 OTcl을 사용하여 작성한다. 스크립트 파일을 작성했으면, 해당 스크립트를 사용하여 ns 시뮬레이터를 실행시킨다. 일반적으로 ns 시뮬레이션을 수행하면, 시뮬레이션 과정 중에 발생한 이벤트들을 기록해 놓은 트레이스(trace) 파일이 생성된다. 따라서 시뮬레이션의 결과를 생성하기 위해서는 트레이스 파일을 분석하게 된다. 트레이스 파일을 분석하는 방법은 각자의 취향과 목적에 따라 다양하게 존재한다. 많이 사용되는 방법은 grep, awk, tcl, perl 등을 사용하는 것이다. 그리고 많은 경우에 시뮬레이션의 결과를 그림으로 표현하게 되는데, 이때 많이 사용되는 프로그램으로는 gnuplot과 xgraph 등이 있다.

3.2 간단한 스크립트 파일 예제

본 절에서는 간단한 예제 스크립트 파일을 소개하고, 그 내용을 간략하게 설명한다. 한 노드에서 CBR (Contant Bit Rate) 트래픽을 생성하여, UDP 프로토콜을 사용하여 다른 노드로 전송하도록 하는 예이다.

```
#1 set ns [new Simulator]
#2 set tracefd [open out.tr w]
#3 $ns trace-all $tracefd
#4 set n0 [$ns node]
#5 set n1 [$ns node]
#6 $ns duplex-link $n0 $n1 100Mb 1ms
   DropTail
#7 set udp [new Agent/UDP]
#8 $ns attach-agent $n0 $udp
#9 set null [new Agent/Null]
#10 $ns attach-agent $n1 $null
#11 $ns connect $udp $null
#12 set cbr [new Application/Traffic/CBR]
#13 $cbr set rate_ 5Mb
#14 $cbr set packetSize_ 500
#15 $cbr attach-agent $udp
#16 $ns at 0.0 "$cbr start"
#17 $ns at 10.0 "finish"
#18 proc finish {} {
#19   global ns tracefd
#20   $ns flush-trace
#21   close $tracefd
#22   exit 0
#23 }
#24 $ns run
```

- 시뮬레이션 시작과 종료

1행은 'Simulator' 클래스의 새로운 오브젝트 'ns'로 생성한다. 'ns' 오브젝트를 사용하여 시뮬레이션을 시작시키거나 종료시킬 수 있다. 시뮬레이션을 시작하는 명령어는 24행이다. 시뮬레이션을 종료시키는 기능은 18행부터 23행까지 'finish'라는 이름의 프로시져로 구현되어 있다.

Tcl의 프로시져는 C언어의 함수와 유사하다. 19행은 프로시져에서 사용할 외부의 변수를 가져오는 명령이다. 20행에서 파일 버퍼에 있는 내용을 하드디스크에 저장시키고, 21행에서 트레이스 파일을 닫는다. 22행에서는 'exit' 명령어를 사용하여 프로그램을 종료시킨다. 이 finish 프로시져를 호출하는 명령이 17행이다. 10.0 초에 finish 프로시져를 호출하여 시뮬레이션을 종료시킨다.

- 트레이스 파일 설정

2행에서 'out.tr'이라는 이름의 파일을 열고, 3행에서 해당 파일에 시뮬레이션 이벤트 트레이스를 저장하도록 설정하였다.

- 노드 설정

4행은 'n0'라는 이름의 노드를 생성한다. 5행은 'n1'라는 이름의 노드를 생성한다.

- 링크 설정

6행은 앞에서 정의한 두 개의 노드 'n0'와 'n1'을 연결하는 링크를 정의하는 명령어이다. 링크의 대역폭은 100Mbps이고, 전파 지연(propagation delay)이 1ms인 양방향 링크를 설정하였다. 링크의 버퍼를 운용하는 알고리즘으로는 'DropTail'이 선택되었다.

- 전송 계층 설정

7행은 UDP 송신 에이전트 'udp'를 생성하고, 8행은 생성된 UDP 송신 에이전트 'udp'를 'n0' 노드에 연결한다. 9행은 UDP 트래픽을 수신하기 위한 Null 에이전트 'null'을 생성하고, 10행은 생성된 Null 에이전트 'null'을 'n1' 노드에 연결한다. 11행에서 'udp'와 'null'을 연결함으로써, 'udp' 에이전트가 전송하는 UDP 트래픽이 'null' 에이전트로 전송되도록 설정하고 있다.

- 데이터 생성

12행은 CBR 트래픽을 발생시키는 'cbr' 에이전트를 생성하였다. 13행과 14행은 'cbr' 에이전트의 트래픽 생성 속도를 5Mbps로, 각 패킷의 크기는 500바이트로 설정하고 있다. 15행은 'cbr' 에이전트를 'udp' 에이전트에 연결하여 'cbr' 에이전트에서 생성한 트래픽을 'udp' 에이전트가 전송하도록 설정한다. 16행은 'cbr' 에이전트가 0.0초부터 트래픽을 생성시키도록 한다.

3.3 ns-2 실행하기

ns-2 시뮬레이션을 수행하기 위해서는 '% ns 〈filename〉'과 같은 명령을 수행하여야 한다. '〈filename〉'은 시뮬레이션 시나리오를 기술하는 OTcl 스크립트 파일명을 말한다.

3.3.1 트레이스 파일

ns-2 시뮬레이션을 수행하면, 시뮬레이션 과정 중에 발생한 각 패킷의 출발과 도착 정보, 링크나 큐에서의 패킷 누락 등과 같은 모든 이벤트 정보를 트레이스 파일에 저장하여 알려준다. ns-2는 시각적으로 시뮬레이션 과정을 볼 수 있는 nam 트레이스 기능도 제공하지만, 본고에서는 텍스트 파일 형식의 트레이스만 설명한다.

다음은 ns-2 시뮬레이션의 결과로 생성되는 트레이스 파일의 예이다.

```
+ 0 0 1 cbr 500 ------- 0 0.0 1.0 0 0
- 0 0 1 cbr 500 ------- 0 0.0 1.0 0 0
+ 0.0008 0 1 cbr 500 ------- 0 0.0 1.0 1 1
- 0.0008 0 1 cbr 500 ------- 0 0.0 1.0 1 1
r 0.00104 0 1 cbr 500 ------- 0 0.0 1.0 0 0
+ 0.0016 0 1 cbr 500 ------- 0 0.0 1.0 2 2
- 0.0016 0 1 cbr 500 ------- 0 0.0 1.0 2 2
r 0.00184 0 1 cbr 500 ------- 0 0.0 1.0 1 1
```

트레이스 파일의 각 행은 다음 필드들로 구성되어 있다.

· 이벤트 종류

r, +, -, d의 4가지 종류가 있다. 각각 패킷이 다른 노드에 도착하는 이벤트, 패킷이 버퍼에 들어가는(enqueue) 이벤트, 패킷이 버퍼에서 빠져나오는(dequeue) 이벤트, 패킷이 폐기되는 이벤트를 의미한다.

· 발생 시간

해당 이벤트가 발생한 시간
· From 노드
이벤트가 발생한 링크에 연결된 입력 노드
· To 노드
이벤트가 발생한 링크에 연결된 출력 노드
· 패킷 종류
예를 들어, CBR, TCP, UDP, ACK 등
· 패킷의 길이
· 플래그
'Congestion experienced', 'Congestion Action', 'TCP fast start' 등 플래그를 표시하기 위해 사용된다. trace/trace.cc에 정의되어 있다.
· 플로우 ID
· 소스의 주소
(소스 노드 ID).(포트 번호)로 구성된다.
· 목적지의 주소
(목적지 노드 ID).(포트 번호)로 구성된다.
· 패킷의 일련 번호 (Sequence Number)
· 패킷 ID
패킷 고유 식별번호

3.4 결과 분석

ns-2는 트레이스 기능을 통하여 시뮬레이션 과정에 발생한 이벤트 정보를 제공한다. 트레이스 파일에는 모든 이벤트 정보가 담겨져 있기 때문에, 시뮬레이션의 결과를 얻기 위해서는 목적에 따라 필요로 하는 정보만을 추출해 내는 것이 필요하다. 이를 수행하는 방법에는 여러 가지가 있다. 일반적으로 그 과정은 데이터 파일을 분석하는 분석 툴과 분석 툴을 통하여 추출된 정보를 그래프로 그리는 두 단계를 거치게 된다.

3.4.1 분석 툴

유닉스 계열에서 제공되는 'grep' 명령어는 파일에서 원하는 데이터만을 필터링 할 수 있게 도와준다. 간단한 경우에는 grep 명령어를 사용하여 트레이스 파일을 분석할 수 있다. 좀 더 복잡한 분석을 위해서는 awk를 사용할 수 있다. awk는 작지만 매우 강력한 텍스트 파일 분석 도구로서, 트레이스 파일을 칼럼(column) 별로 쉽게 구분할 수 있게 하여, 여러 가지 연산을 가능하게 해준다. 그 외에도 Tcl이나 Perl과 같은 언어를 사용하여 분석 스크립트를 작성할 수도 있다.

3.4.2 그래프 그리기

앞에서 설명한 여러 가지 분석 툴에 의하여 추출된 데이터 값들을 그래프로 그리는 방법도 다양하다. 그 중 가장 대표적인 방법은 gnuplot을 사용하는 것이다. 또는 ns-2에서 기본적으로 제공하는 xgraph를 사용하는 것도 가능하다.

4. 결 론

본고에서는 대표적인 네트워크 시뮬레이터인 ns-2에 대하여 소개하고, 이를 사용하여 네트워크 시뮬레이션을 수행하는 방법론에 대하여 고찰하였다. 시뮬레이션의 속도 향상과 편의성을 동시에 제공하기 위하여 C++과 OTcl의 두 가지 언어를 사용하여 구현되어 있는 ns-2의 구조를 설명하였다. 또한, 시뮬레이션 시나리오를 기술하고, ns-2를 사용하여 이를 수행한 후, 시뮬레이션의 결과물로서 얻어지는 트레이스 파일을 분석하고 이를 그림으로 표현하는데 사용되는 여러 가지 프로그램들에 대하여 설명하였다.

참고문헌

[1] [ns2] http://www.isi.edu/nsnam/ns/
[2] [ns3] http://www.nsnam.org/
[3] [qualnet] http://www.qualnet.com/
[4] [opnet] http://www.opnet.com/
[5] [omnetpp] http://www.omnetpp.org/

저자약력

최 선 웅

1998년 서울대학교 전산과학과 학사
2000년 서울대학교 전산과학과 석사
2005년 서울대학교 전기, 컴퓨터 공학부 박사
2005년~2007년 삼성전자 정보통신총괄 책임연구원
2007년~현재 국민대학교 전자공학부 전임강사
관심분야 : 무선 네트워크, 네트워크 자원관리, 시스템 성능 평가
이 메 일 : schoi@kookmin.ac.kr

장 영 민

1985년 경북대학교 전자공학과 학사
1987년 경북대학교 전자공학과 석사
1999년 University of Massachusetts, Dept. of Computer Science 박사
1987년~2000년 ETRI 이동통신연구단 연구원 및 선임연구원
2000년~2002년 덕성여대 컴퓨터과학과 교수
2002년~현재 국민대학교 전자정보통신공학부 교수
2005년~현재 국민대학교 Ubiquitous IT Convergence Research Center 소장
관심분야 : 4G 이동통신, 이종망간 연동, 통신방송인터넷융합
이 메 일 : yjang@kookmin.ac.kr