미래인터넷 테스트베드를 위한 OpenFlow 프로토콜과 제어 방안

김병철 | 이재용
충남대학교

요 약

본고에서는 현재 인터넷이 가지는 한계를 극복하기 위해 전세계적으로 연구가 되고 있는 미래인터넷의 테스트베드 분야에서 가장 활발히 개발되고 있는 제어 프레임워크인 OpenFlow 프로토콜과 이를 통한 미래인터넷 망의 제어 방안에 관해 살펴보기로 한다. 또한 OpenFlow 프로토콜을 활용하여 테스트베드 상에서 다양한 응용 실험과 새로운 프로토콜 실험을 수행한 내용 중 주요한 몇 가지만 선택하여 살펴봄으로써 최근 연구 동향을 가늠하도록 한다.

I. 서 론

지금 사용되고 있는 인터넷은 1960년대 실험 목적의 소규모 망으로 출발하여 TCP/IP라는 프로토콜에 기반을 둔 채 현재까지 운영되고 있다. 그러나 현재의 인터넷은 초기의 인터넷 설계 시 고려되지 못했던 다양한 환경을 고려하지 않으면 향후 등장할 새로운 응용 및 서비스를 수용하기 어려운 상황에 처해 있다. 이러한 문제점들은 보안, 관리, 이질성, 이동성, 확장성 및 다양한 무선 유비쿼터스 통신 환경 등에 대한 대처 능력의 부족들을 들 수 있다.

이런 이유로 수 년 전부터 전 세계적으로 새로운 형태의 미래인터넷 구조를 설계하고 핵심 요소들을 개발하며 테스트베드 구축을 하려는 노력이 활발히 진행되고 있으며 이 중 대표적인 것인 미국의 FIND[1], 유럽의 FP7[2] 및 일본의 NwGN[3] 등이다. 또한 미래인터넷 테스트베드 구축 현황 관련해서는 이미 잘 알려져 있듯이 미국의 경우는 GENI (Global Environment for Network Innovation)[4], 유럽 연합은 FIRE (Future Internet Research and Experimentation)[5]가 잘 알려져 있다.

이 중 GENI는 이미 선정된 특정 기술을 바탕으로 구축하는 시험 망과는 전혀 다른 개념으로써 응용계층 등 상위계층은 물론 물리계층 등 하위계층까지 완전한 가상 망을 구축하여 어느 누구의 어떠한 새로운 기술이라도 자유롭게 설치, 시험, 공동 활용할 수 있게 하는 것을 목표로 한다. 이를 통해 수많은 미래의 모험적 기술들이 경쟁적으로 동시에 개발되고 GENI 위에서 운영되어, 범세계적으로 확산, 활용되는 경쟁력 있는 기술들이 살아남아 미래의 기술로 정착이 되게 된다. 즉, GENI는 소위 Seamless Migration/Deployment를 추구하는 공용 가상 시험 망이다. 이러한 가상 망에서의 각 트래픽 플로우 별 제어를 위해서는 새로운 컨트롤 방안이 연구되어야 한다. OpenFlow 기술은 Cluster B 내의 Enterprise GENI에서 사용되는 제어 프레임워크로 스탠포드 대학이 주관이 되어 개발되고 있다.

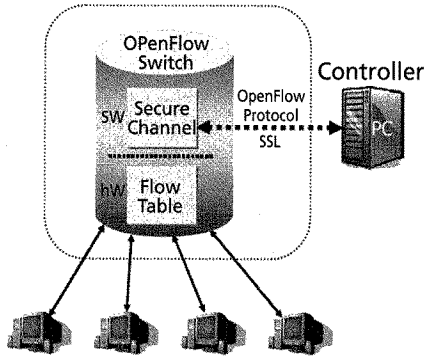
본 고에서는 미래인터넷 연구의 중요한 부분을 담당하고 있는 OpenFlow 프로토콜과 이를 통한 미래인터넷 망의 플로우 제어 방안에 관해 살펴보기로 한다. 또한 OpenFlow 프로토콜을 활용하여 테스트베드 상에서 다양한 응용 실험과 새로운 프로토콜 실험을 수행한 내용 중 주요한 몇 가지만 선택하여 살펴봄으로써 최근 연구 동향을 가늠하도록 한다.

II. OpenFlow 프로토콜

OpenFlow 프로토콜은 제어기로 하여금 망을 통해 스위치 및 라우터의 datapath에 대한 접근을 제공하는 통신 프로토콜로서 Software Defined Networking 구현의 예라고 볼 수 있다. 현재 버전 1.0이 나와있으며 모든 표준에 대한 작성은 스탠포드 대학의 Openflow Switching Consortium에서 담당하고 있다[6].

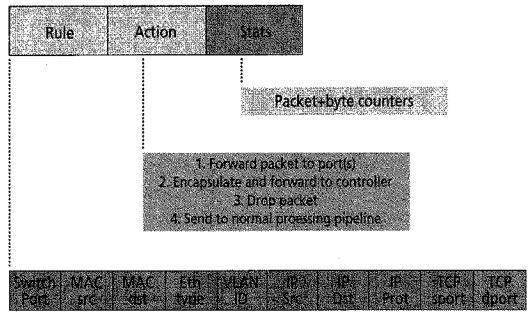
OpenFlow 프로토콜에 의해 제어되는 스위치를 OpenFlow 스위치라고 부르며 이 스위치 내에는 외부 제어기(OpenFlow controller)에 의해 만들어지고 변경되는 flow table이 있어 연구자가 자신만의 처리 로직에 의해 플로우별로 패킷의 경로를 제어할 수 있게 된다. 현재 OpenFlow 프로토콜이 탑재된 스위치는 Cisco, Juniper, Hewlett-Packard 및 NEC 등에서 생산되고 있다.

기본적인 OpenFlow 스위치의 구조는 (그림 1)에서 보는 바와 같다. OpenFlow 스위치는 flow table, 외부 제어기와의 통신을 위한 Secure Channel 및 OpenFlow 프로토콜로 구성된다. OpenFlow 스위치에서 일어나는 플로우 별 action은 주어진 포트로의 전송, 캡슐화 후 컨트롤러로의 전달, 패킷 drop 및 보통의 라우팅 처리 방법이 있으며 이를 위한 flow table 구성과 이러한 플로우 별 구분을 위해 사용되는10-tuple은 (그림 2)와 같다.



(그림 1) OpenFlow 스위치

OpenFlow 프로토콜이 동작하는 네트워크에서 모든 패킷의 경로는 컨트롤러의 제어를 받게 되는데, 예를 들어 A노드에서 B노드로 패킷을 송신하는 경우, OpenFlow 스위치에 패킷이 인입 되며 OpenFlow 스위치는 자신의 플로우 테이블을 검색하여 해당 패킷에 대한 action에 따라 drop, forwarding, redirection을 결정하여 처리하게 된다. 이와 같이 OpenFlow 프로토콜을 이용하게 되면 연구자는 패킷 하나하나에 대해서 원하는 대로 처리할 수 있는 방법을 얻게 된다.



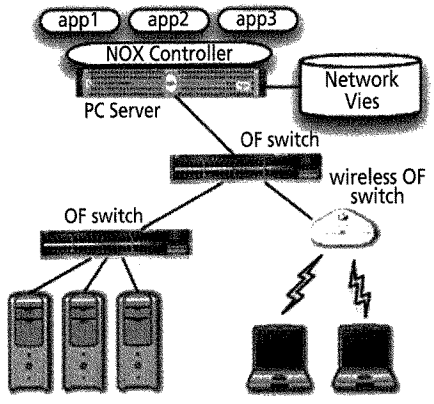
(그림 2) Flow table 구성

III. OpenFlow controller 및 제어 방안

OpenFlow 프로토콜을 통해 다양한 망의 제어 및 운영을 위해 가장 중요한 망 요소 중 하나가 컨트롤러이다. 컨트롤러는 주어진 망을 제어하기 위한 운영체제의 일종을 구현한 것이라고 할 수 있으며 중앙 집중적인 프로그래머블 인터페이스를 제공한다. NOX[7]는 공개 소스 형태의 OpenFlow 컨트롤러로서 http://www.noxrepo.org 사이트에서 에서 최신 정보를 얻을 수 있다. 본 절에서는 이러한 NOX 망 운영체제의 개념 및 동작과정을 살펴보기로 한다.

NOX 기반의 망 구조 및 주요 요소들을 살펴보면 (그림 3)과 같다. NOX에서는 망의 전체적인 구성 정보인 network view를 기본 정보로 사용하여 트래픽의 제어를 위한 다양한 응용을 가지게 된다. OpenFlow 스위치에 새로운 플로우 트래픽이 들어오게 되면 이를 NOX 컨트롤러에 보내게 되는데 이 때 컨트롤러는 사용자가 원하는 망의 제어를 위한 구성된 다양한 응용에 의해 플로우 테이블을 만들고 이를 OpenFlow 스위치에 적용하게 된다. OpenFlow 스위치의 경

우 switch 가 새로 연결되거나 단절되는 경우, 새로운 플로우 패킷이 들어오거나 새로운 통계 데이터가 수신되는 경우 컨트롤러와 통신하게 되고 이러한 event에 따르는 event handling이 일어나게 되는 것이다.



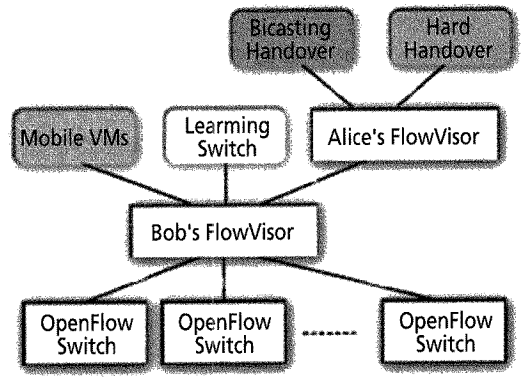
(그림 3) NOX 기반 망의 구조 및 요소

위에서 본 바와 같이 NOX의 목적은 C++ 및 Python으로 망 제어 소프트웨어를 작성하는 단순화된 플랫폼을 제공하는 것으로 현재 버전 0.6.0이 release되어 있다. NOX의 설치 및 설정은 아래와 같은 표준화된 자동 설정 절차를 거쳐 이루어지며 현재 버전은 망 토폴로지 재 구축을 위한 'discovery' 와 'topology', 망 라우팅을 위한 'routing' 및 망에서의 호스트 이동을 추적하기 위한 'authenticator' 라는 응용을 포함하고 있다.

```
git clone git://noxrepo.org/noxcore
cd noxcore/
./boot.sh
mkdir build/
cd build/
../configure –with-python=yes
make
make check
```

한편 OpenFlow 망 자원을 다수의 컨트롤러 및 연구자들과 공유하며 네크워크 가상화를 사용하기 위한 방법을 제공하기 위해 FlowVisor[8]가 도입된다. 이를 통해 OpenFlow 망에 서로 독립적인 망 슬라이스를 생성할 수 있게 되고 각 슬라이스 별 제어가 가능하게 된다. 즉 망의 가상화를 위한 구성 요소로서 OpenFlow 스위치와 컨트롤러 사이에 위치하여 슬라이스 별로 서로 독립적인 별도의 컨트롤러가 동작하게끔 하는 것이다. FlowVisor는 슬라이스를 구성하는 플로우 집합 별로 최소의 데이터 전송률을 할당하여 링크 대역폭을 나누며 각 스위치의 플로우 테이블도 각 컨트롤러에 속하는 플로우 항목 별로 구분 관리하게 된다.

(그림 4)는 이러한 개념을 보여주는 망 구성을 나타낸다. FlowVisor는 망 디바이스, OpenFlow 컨트롤러 및 다른 FlowVisor들 사이에서 transparent한 프록시로 동작하며 다수의 연구자들이 슬라이스 상에 독립적인 운영이 가능하게 한다. Sigcomm 2009[9]에서 OpenFlow 연구그룹은 실제 망 토폴로지 상에서 bicasting 핸드오버, 하드 핸드오버 및 MobileVM 등의 운영이 컨트롤러에 구현되어 슬라이스 별로 서로 독립적으로 동작함을 데모하였다.



(그림 4) FlowVisor 구성

IV. OpenFlow 프로토콜을 통한 주요 응용 개발 현황

OpenFlow 프로토콜을 사용하여 다양한 망 제어 기능을 구현하고 이를 서로 공유하는 워크샵이 개최되어 연구자들

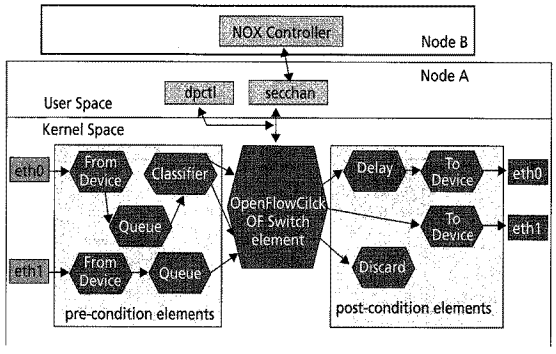
간에 정보를 교류하고 구현에 따른 문제점과 향후 연구 방향을 논의하였는데 이 중 대표적인 것이 2009년의 Sigcomm[9]이고 이외에도 스탠포드 대학에서 2009년 8월에 열린 Campus Trials Workshop[10]과 2010년 Duke 대학에서 열린 GEC7[11]에서의 데모 등을 들 수 있다.

이 장에서는 이 행사들을 통해 구현된 주요 데모 시나리오, 망 제어 알고리즘 및 프로토콜을 설명하여 향후 OpenFlow프로토콜을 통해 다양한 응용 프로그램을 개발하는 연구자들에게 도움이 되고자 한다.

1. Click을 위한 OpenFlow switch

Click[12]이란 유연하고(flexible) 설정 가능한 라우터를 만들기 위한 새로운 소프트웨어 구조로서 MIT 연구실에서 엘리먼트라고 불리는 패킷 처리 모듈들을 연결하여 다양한 기능의 라우터를 구성 가능하도록 한 것이다. 이러한 click 라우터를 통해 주문형 라우터를 쉽게 제작할 수는 있으나 이러한 구조는 패킷 별로 동적인 처리가 가능하지는 않다는 문제점이 있다.

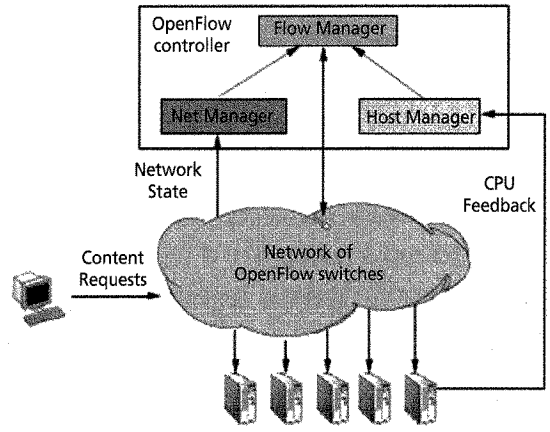
따라서 OpenFlowClick[13]을 Click kernel 모듈에 설치하여 Click 라우터에 openflow 인터페이스 기능을 추가하고자 한 것이 본 개발의 내용이며 이를 통해 컨트롤러는 특정 패킷들이 거쳐가야 하는 엘리먼트들의 경로를 동적으로 결정할 수 있게 된다. 이러한 시나리오 중의 하나를 나타내면 (그림 5)와 같다. 이 그림에서 보면 패킷의 분류에 따라 OpenFlow 컨트롤러의 제어에 따라, 지연시켜 eth0 로 전달하는 패킷, 직접 eth1 으로 전달하는 패킷, 그리고 폐기되는 패킷으로 동적으로 분류되어 처리되는 동작을 볼 수 있다.



(그림 5) OpenFlow+click 망의 예

2. Plug-n-Serve 응용[14]

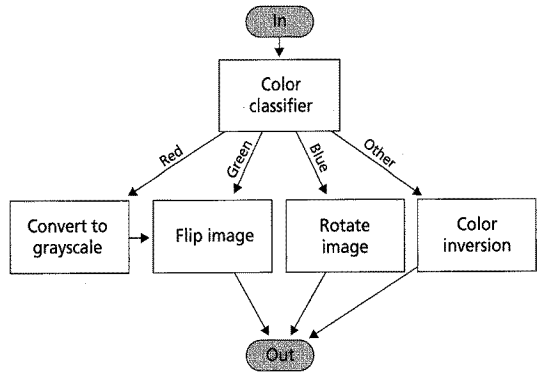
Plug-n-Serve는 OpenFlow 기반의 서버 부하 분산 시스템으로 이를 통해 제어가 되지 않는 망에서의 웹 서버 응답 시간을 효과적으로 줄이게 된다. 이를 위해 망의 현 상태를 측정하고 새로운 HTTP 요청의 경로를 직접 제어할 수 있는 방안을 구현하였는데 추가된 주요 기능 요소는 (그림 6)에서 보는 Flow manager, Net manager 및 Host manager이다. 먼저 Flow manager는 선택된 부하 분산 알고리즘에 의해 플로우를 라우팅하고 관리하는 OpenFlow 컨트롤러이고 Net manager는 망을 프로빙하고 토폴로지를 감시하며 사용 레벨을 모니터링하는 모듈이다. 마지막으로 Host manager는 시스템 내 각 사용자의 상태와 부하를 모니터링하고 이를 Flow manager에 보고하는 역할을 수행한다. Sigcomm 데모에서는 저자가 개발한 LOBUS 알고리즘의 효율성을 이 응용을 통해 보였다.



(그림 6) Plug-n-Serve의 주요 제어 로직

3. OpenPipe 응용[15]

본 응용에서는 모듈러 구조의 pipeline을 만들기 위해 OpenFlow를 사용하였으며 이를 통해 비디오 처리 응용을 구성하여 데모하였다. 구성할 모듈들을 팔레트에서 드래그하고 연결하기 위한 기본적인 GUI를 개발하였으며 이를 통해 (그림 6)과 같은 기본적인 비디오 처리 기능을 OpenPipe를 통해 구현하여 성능을 보였고 동작 중인 시스템에 추가적인 모듈을 삽입하여 새로운 모듈의 동작이 잘 이루어짐도 보였다.



(그림 7) OpenPipe 비디오 처리 데모 시나리오

4. OpenRoad 응용[16]

이 응용은 상용 망에 다수의 무선망 실험이 가능하도록 구성한 무선 테스트베드로서 FlowVisor를 사용하여 슬라이스 간을 분리하였으며 (그림 8)에서 보듯이 30개의 WiFi AP와 WiMAX 기지국에 모두 OpenFlow를 설치하여 구성하였다.

이 실험 망을 통해 확장성이 있는 이동 망에서의 망 관리 및 모니터링에 관한 다양한 실험을 수행하였다.

5. GEC 7 에서의 Clemson OpenFlow 실험

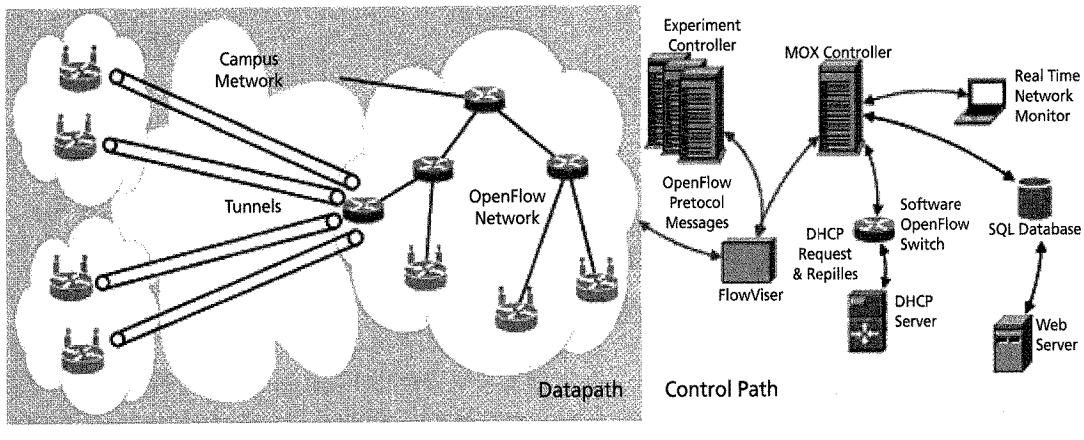
GEC란 GENI Engineering Conference로서 2010년 3월에 미국 노스캐롤라이나 Durham에서 제 7차 회의가 개최되었다. 이 회의에서 GENI 관련되어 개발되고 있는 다양한 결과물들의 데모가 있었는데 그 중 하나인 Clemson OpenFlow 실험은 프로그래머블한 유, 무선 망 연구를 가능하도록 테스트베드를 구축하고 캠퍼스 망에서의 교육적인 사용과 연구에 필요한 운용 툴들을 제공하는 것이다.

이외에도 OpenFlow 제어를 통해 패킷과 서킷 스위치에 대한 통일된 제어 방안과 그 구조[17]를 제안한 방법과 이의 성능 검증을 위한 데모 등 다양한 연구 결과가 많이 제안되어 있다.

V. 결 론

본고에서는 OpenFlow 프로토콜의 개념 및 스위치-컨트롤러 간의 동작 과정과 기본적인 컨트롤러 기능에 대해 알아보았다. 또한 망 자원을 다수의 컨트롤러 및 연구자들과 공유해 사용하기 위해 도입된 FlowVisor의 동작에 대해서도 알아 보았다. 마지막으로 이러한 OpenFlow 프로토콜을 이용해 테스트베드에 적용해 본 다양한 응용과 실험 내용에 관해 살펴보았다.

미래인터넷은 현재 인터넷이 가지는 한계를 극복하기 위한 하나의 방안으로 전세계적으로 연구가 활발히 진행되고 있는 만큼 국내에서도 GENI, FP7 FIRE 및 일본의 최근 연구 동향을 파악하고 공동으로 연구에 참여하는 것이 필요하다고 볼 수 있다. 이러한 관점에서 본고에서는 현재 GENI에서 가장 활발히 연구되고 있는 OpenFlow에 대한 연구 내용과



(그림 8) OpenRoad 테스트베드

개발 방향을 소개하고 이를 활용한 테스트베드 구축과 응용 현황을 살펴보았다.

참 고 문 헌

[1] FIND: http://find.isi.edu/

[2] FP7: http://cordis.europa.eu/fp7/home_en.html

[3] Shuji Esaki, Akira Kurokawa, and Kimihide Matsumoto, "Overview of the Next Generation Network," NTT Technical Review, Vol.5, No.6, June 2007

[4] GENI: Global Environment for Network Innovations, http://www.geni.net/

[5] FIRE: Future Internet Research and Experimentation, http://cordis.europa.eu/fp7/ict/fire/

[6] http://www.openflowswitch.org

[7] Natasha Gude , et. Al., "NOX: Towards an Operating System for Networks " CCR, July 2008

[8] Rob Sherwood, wt. al., "Carving Research Slices Out of Your Production Networks with OpenFlow", ACM Sigcomm 2009

[9] Sigcomm 2009, Barcelona Spain, http://conferences.sigcomm.org/sigcomm/2009/

[10] http ://www.openflowswitch.org/downloads/Workshop2009/OpenFlowWorkshop-NickMcKeown.pdf

[11] http://groups.geni.net/geni/wiki/DemoInfo

[12] Eddie Kohler, et.al., "The Click Modular Router", ACM SIGOPS Operating Systems Review, Vol. 33, Dec. 1999

[13] Yogesh Mundada, Rob Sherwood, Nick Feamster, "An OpenFlow Switch Element for Click" ,ACM Sigcomm 2009

[14] Nikhil Handigol, et. al., "Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow" ACM Sigcomm 2009

[15] Glen Gibb, et. al., "OpenPipes: Prototyping high-speed networking systems", ACM Sigcomm 2009

[16] Kok-Kiong Yap, et. al., "The Stanford OpenRoads Deployment", *WiNTECH '09*, September 21, 2009, Beijing, China

[17] Vinesh Gudla, et. al., "Experimental Demonstration of OpenFlow Control of Packet and Circuit Switches", OFC 2010 SanDiego USA

약 력

김 병 철

1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1996년 한국과학기술원 전기 및 전자공학과 박사
1993년 ~ 1999년 삼성전자 CDMA 개발팀
1999년 ~ 현재 충남대학교 정보통신공학부 부교수
관심분야 : 이동인터넷, 이동통신 네트워크, 데이터통신

이 재 용

1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1995년 한국과학기술원 전기 및 전자공학과 박사
1990년 ~ 1995년 디지콤 정보통신연구소 선임연구원
1995년 ~ 현재 충남대학교 정보통신공학부 교수
관심분야 : 초고속통신, 인터넷, 네트워크 성능분석