액티브네트워크

서울대학교 하석재*·최양희**

1. 서 론

액티브네트워크(Active Network)란 기존의 네트워크에서 라우터나 스위치와 같은 중간노드들이 단순히 패킷의 헤더만 처리하던 데서 한 걸음 더 나아가 패킷에 프로그램코드와 데이타를 함께 넣어 전송하고 스위치나 라우터에는 이 패킷에 들어 있는 프로그램코드를 처리할 수 있는 환경을 가진 망을 말한다. 액티브네트워크에서는 기존의 네트워크가 중간노드에서는 단순히 패킷을 경로 설정하고 전달하는 기능을 담당하고 에러처리 및 흐름제어와 같은 패킷의 복잡한 처리는 종단의 단말장치에서만 처리하던 것과는 달리 중간노드에서 여러 가지 처리를 가능하게 함으로 인해 기존의 네트워크가 제공하지 못했던 유연성과 다양한 장점을 제공할 수 있다. 액티브네트워크는 망에 일종의 지능을 부여한다는 측면에서 전화망에서의 AIN (Advanced Intelligent Network)과 유사한 개념이라고 할 수 있다[1].

액티브네트워크의 기본 개념은 1994년과 1995년에 broad DARPA research community에서 미래의 네트워킹 시스템의 방향에 대한 논의에서 처음 출발했다. 여기에서 지적된 현재의 네트워크시스템의 문제점은 다음과 같다.

- 신기술 및 이에 관련된 표준을 통합하는데 걸리는 시간이 길고,
- 여러 개의 프로토콜 계층(Layer)간의 중복되는 기능으로 인해 성능을 낼 수 없고,
- 현재 네트워크 구조에 새로운 서비스를 제공하는 경우의 문제점 등이 논의되었고 이러한 문제들에 대한 대안으로 제시된 것이 바로 액티브네트워크이다.

이 글의 구성은 다음과 같다. 2장에서는 액티브네트워크의 구성요소와 같은 아키텍처와 구현방식에 관해 3장에서는 액티브네트워크의 연구동향을 알아보고 구체적인 연구사례에 대해 알아본다. 4장에서는 액티브네트워크를 이용한 응용을 알아보고 5장에서는 결론과 향후 연구과제로 이루어져 있다.

2. 액티브 네트워크 아키텍처

2.1 구성요소

액티브네트워크는 기존의 네트워크의 패킷에 해당하는 캡슐(액티브패킷이나 스마트패킷으로도 불린다)과 이를 수행할 수 있는 중간노드의 수행환경으로 이루어져 있다(그림 1). 캡슐은 기존의 패킷이 중간노드가 실제로 보고 처리하

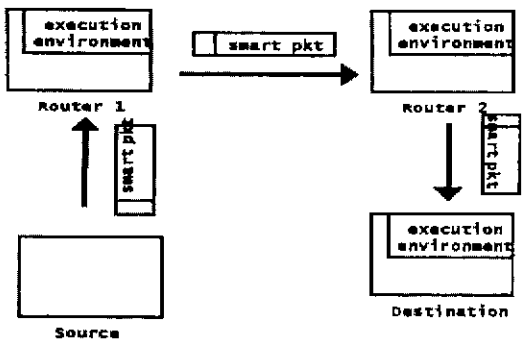

그림 1 액티브네트워크의 아키텍처

*학생회원
**종신회원

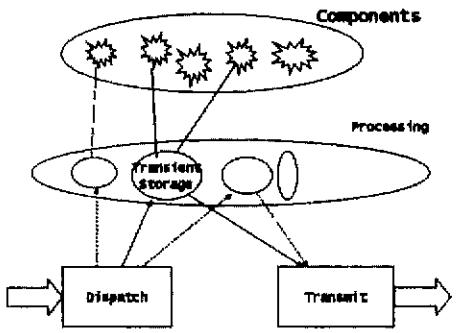

그림 2 액티브노드의 구조[3]

는 부분인 헤더와 내용을 나타내는 페이로드(Payload)로 이루어져 있는 것과는 달리 실제 수행될 수 있는 프로그램코드와 데이타로 구성되어 있다. 라우터나 스위치와 같은 중간노드에서는 각 캡슐들을 읽어들여 프로그램과 데이타를 수행(Execute), 처리(Process)할 수 있는 실행환경(Execution environment)을 제공한다(그림 2). 이러한 실행환경을 정의하는 데는 여러 가지 사항들을 고려해야만 한다. 이러한 고려사항 중 중요한 것으로는 프로그램언어, 프로그램의 인코딩방식, 프리미티브 제공문제, 그리고 자원의 할당 및 공유문제 등이 있다. 이와 같이 액티브네트워크를 설계하는 데는 네트워크, 프로그램언어, 운영체제, 보안과 같은 많은 분야의 연구가 필요하다. 이에 대해서는 2.3절에서 자세히 다루도록 하겠다.

2.2 구현방식

액티브네트워크의 개념을 구체적으로 구현하는 데는 크게 두 가지 방식이 있다.

하나는 기존의 라우터나 스위치를 프로그램 가능하게 만드는 것(프로그래머블 스위치:Programmable switch)으로서 이는 라우터나 스위치가 제공하는 표준 프로토콜이나 서비스를 필요에 따라 이를 동적으로 적재(Dynamically loading)하고 패킷에는 제공되는 프리미티브를 사용하는 프로그램과 데이타를 적재해 전송하는 방식이다. 이 경우에는 프로그램의 적재와 패킷의 처리가 분리되게 되는데 이를 분리된 접근방법(Discrete approach)이라고 한다. 이러한 방식을 따르는 것은 펜실바니아 대학과 벨코어에서 제안한 스위치웨어(SwitchWare)가 있다[8].

다른 방식은 프로그램 가능한 스위치의 방식과는 다르게 캡슐(Capsule)이라고 불리는 패킷에 실제 수행될 프로그램과 데이타를 동시에 가지고(In-band방식으로) 전송된다. 네트워크의 중간노드들은 이를 수행할 수 있는 환경을 제공하며 실제 캡슐들이 이런 중간노드들에서 수행되며 전송되는 방식이다. 이 방식은 프로그램가능한 스위치의 분리된 접근방식에 대해서 통합적인 접근 방식(Integrated approach)이라고 부른다. 캡슐기반의 액티브네트워크는 MIT에서 주로 연구하고 있다[3, 11].

2.3 액티브네트워크 설계시 고려사항

액티브네트워크에서는 공통 프로그래밍 모델과 노드 자원의 효율적인 할당과 감시(Monitoring)를 할 수 있는 방법을 정의해야 한다.

2.3.1 공통 프로그래밍 모델

액티브패킷(캡슐)의 프로그램을 기술하는 언어는 프로그램 인코딩방식, 프리미티브 제공문제, 보안성 등을 고려해야 한다. 게다가 액티브 패킷이 여러 종류의 플랫폼에서 수행되어야 하기 때문에 공통적인 프로그래밍 모델을 설계하고 정의하는 것이 필수적이다.

공통 프로그래밍 모델을 정의할 때 고려해야 할 사항은 다음과 같다.

—프로그래밍 인코딩 방식

액티브 패킷은 프로그램이 패킷에 들어가서 전송, 수행되는 형태이기 때문에 이동성(Mobility), 안전성(Safety), 효율성(Efficiency)이 보장되도록 인코딩을 결정해야 한다. 이동성이란 프로그램을 전송하고 여러 플랫폼에서 수행할 수 있는 능력을 말하는 것이고, 안전성이란 프로그램 수행시 액세스 할 수 있는 자원을 제한할 수 있는 능력을 말한다. 효율성이란 안전성과 이동성을 처리하면서도 충분한 성능을 보장해야 한다는 것이다.

이와 같은 사항들을 고려하면 프로그램의 인코딩방식은 크게 네 가지 종류가 있다. SafeTCL과 같이 소스코드형태로 전송하는 경우와, 자바와 같이 중간언어형태로 전송하는 방식,

그리고 제한된 수행환경을 제공하는 sand-box 기반의 목적코드 형태, 그리고 인증기반 목적코드형태로 나눌 수 있다. 각각 경우에는 장단점이 존재하며 실제 액티브네트워크를 구현하는 경우에 여러 가지 추가적인 기법들을 사용하고 사정에 따라 절충형을 취하기도 한다. 그 중 소스코드나 중간 코드형태로 되어 있는 코드의 수행효율을 높이기 위해 많이 사용되는 기법 중에는 "On-the-fly"컴파일 기법이 있다. 이 기법은 프로그램의 전송은 중간코드나 소스코드의 형태로 전송되고 수행환경에서 수행시(런타임 시에) 컴파일함으로서 코드의 수행효율을 높일 수 있다.

—공통 프리미티브

각 노드에 내장되어 있는 형태의 서비스로서 액티브패킷이 제공받을 수 있는 기본적인 서비스이다. 이러한 프리미티브는 다음과 같은 여러 개의 카테고리로 나눌 수 있다.

패킷 자체를 다룰 수 있는 프리미티브: 헤더, 페이로드(Payload), 길이 등

노드의 자원이나 환경을 액세스할 수 있는 프리미티브: 주소, 시간, 링크상태 등

패킷 흐름을 제어하는 프리미티브: 포워딩(Forwarding), 복사(Copy), 폐기(Discard) 등

2.3.2 노드의 자원과 할당

수행환경과 프로그램언어를 설계할 때 고려해야 하는 것으로 노드의 자원할당 문제가 있다. 노드의 공유자원에는 전송용량(Transmission bandwidth), 처리용량(Processing capacity), 저장장치(Storage)와 같은 물리적인 자원과 라우팅테이블과 MIB(Management Information Base)와 같은 논리적인 자원이 있다. 이러한 공유자원을 안전하고 공평하게 할당하는 것은 중요한 문제이다.

액티브네트워크에서는 노드의 자원을 효율적으로 관리하기 위해서 자원을 공개, 인증기반, 검증기반의 세 가지 유형으로 나누고 이들의 접근방식을 각각 제한한다. 공개된 자원에는 라우팅테이블 등이 있으며 아무 제한이 없이 자유롭게 접근이 가능하다. 인증기반 자원은 사용시 사용자인증(Authentication)을 거쳐야만 하는데 예로는 텔넷이나 FTP와 같은 서비스(프로토콜)가 있다. 마지막으로 검증(Verification)기반의 자원은 응용이 노드의 자원을 사용하는 경우에도 올바른 동작을 하는지 모니터링을 하고 잘못된 동작을 하는 경우에는 제약을 가하게 된다. 이를 위해 별도의 검증자(Verifier)를 두거나 검증된 동작만을 수행하는 언어를 따로 정의할 수도 있다.

3. 현재의 연구현황

액티브네트워크는 현재 여러 곳에서 독립적으로 연구가 진행되고 있다(표 1). 현재 연구되고 있는 주제로는 아키텍처, 구현기술, 명세, 네트워크관리, 이동성, 혼잡제어, 응용 등이 있다. 이들 중 MIT의 ANTS, 펜실바니아 대학의 스위치웨어(SwitchWare)과 BBN의 스마트패킷(Smart Packet)에 대해 살펴보도록 하겠다.

표 1 액티브네트워크 연구현황

기 관	프로젝트명
MIT	SpectrumWare, ANTS
U.Penn	SwitchWare, PLAN, Security
Bellcore	OPCV2, Active Router, Protocol Booster
Columbia	NetScript
BBN	Smart Packet
GeorgiaTech	CANES
U.Arizona	Liquid Software
UCLA/LBNL	Adaptive Web Caching

3.1 ANTS-MIT

MIT에서는 캡슐을 기반으로 액티브네트워크를 구축하고 이를 이용한 여러 가지 응용을 연구하고 있다. 여기서 발표한 ANTS(Active Network Transfer System)는 캡슐기반의 액티브네트워크를 인터넷에 구축할 수 있게 해주는 일종의 툴킷(Toolkit)이다[11]. ANTS는 자바로 구현되었고 주로 네트워크 프로토콜을 동적으로 배포(Deployment)하기 위해서 제안된 아키텍처이다.

3.1.1 ANTS 아키텍처

ANTS 기반의 네트워크는 ANTS를 수행할 수 있는 수행환경과 캡슐을 기반으로 하는 구조이다. 캡슐은 계층구조로 이루어져 있으며 각각 캡슐, 코드그룹, 프로토콜로 구성된다.

캡슐은 기존의 데이타 네트워크에서의 패킷을 대체하는 개념이고, 코드그룹(Code group)은 하나의 전송단위로서 관련된 여러 개의 패킷이 모여 이루어진다. 그리고 프로토콜은 서비스 단위로서 관련된 코드그룹들로 구성되어 있다.

3.1.2 액티브노드(Active Node)

액티브네트워크에서는 프로그램을 수행하도록 허용하면서 원하지 않는 동작을 막도록 디자인하는 것이 중요한 이슈가 된다. ANTS에서는 이러한 역할을 액티브노드에서 담당한다.

액티브노드에서는 다음과 같이 각 캡슐이 사용할 수 있는 프리미티브를 정의, 제공한다.

- 환경 접근(Environment access)-노드의 위치, 링크의 상태, 라우팅테이블, 지역시간 등
- 캡슐 처리(Capsule manipulation)-캡슐의 헤더와 페이로드(Payload)에로의 접근 등
- 제어 동작(Control operation)-캡슐의 포워딩, 생성, 복사, 폐기 등의 동작
- 노드 스토리지(Node storage)-일정 기간 동안 응용들이 정의한 객체나 코드들을 저장할 수 있는 저장공간을 제공한다.

또한 액티브노드에서는 수행환경을 정의한다. ANTS에서는 자바를 이용해 구현되었다. 그러므로 액티브노드에서 제공하는 안전성, 보안성, 효율성 등은 자바수행환경에서 제공하는 것을 사용한다. 이는 구현을 간단하게 하기 위해서이다.

마지막으로 액티브노드에서는 코드분배시스템을 정의한다. ANTS에서는 사용되는 코드와 데이타를 In-band로 같이 보내는 방식을 사용한다. ANTS는 필요한 코드를 On-Demand 방식으로 적재시키고 이를 캐쉬(액티브 스토리지라고도 불린다)에 저장해서 흐름(Flow)이 있을 경우, 즉 연속된 캡슐들이 같은 경로로 전송되고 같은 처리가 필요한 경우에 유리하도록 디자인되었다. 또한 ANTS의 특징으로 코드의 요구적재(Demand loading)기능이 있다

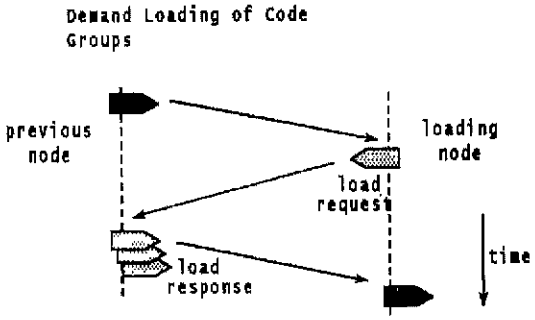

그림 3 코드그룹의 요구적재[11]

(그림 3).

코드그룹의 적재의 동작은 다음의 순서로 이루어진다[11].

1. 각 캡슐은 자신이 속하는 타입이나 프로토콜이 캡슐의 정보에 지정이 되어 있어 이를 보고 노드는 캡슐이 필요한 정보를 알 수 있다.

2. 캡슐이 노드에 도착하면 캡슐의 프로토콜 정보를 읽고 필요한 프로토콜이 캐쉬에 있는지 확인한다. 필요한 프로토콜이 없다면 직전 노드에게 필요한 코드그룹의 적재요구(load request)를 하게 된다. 그리고 일시적으로 캡슐의 수행을 정지시킨다.

3. 적재요구를 받은 노드는 이에 즉시 적재응답(Load response)를 하고 요구받은 프로토콜의 코드를 보내주게 된다.

4. 적재응답을 받은 노드는 이 코드를 캐쉬에 저장하고, 필요한 코드가 모두 적재되면 정지된 캡슐의 수행을 재개한다.

위와 같은 요구적재방식은 첫번째 캡슐이 전송경로를 따라가면서 필요한 프로토콜이 중간 노드의 캐쉬에 적재되면 같은 경로를 따라가는 나머지 캡슐들은 더 이상 코드의 적재나 캡슐의 처리가 일어나지 않고 다만 캡슐이 다른 경로로 전송이 되는 경우에만 코드의 요구적재가 일어나게 되므로 캡슐마다 필요한 오버헤드를 줄일 수 있게 된다.

3.2 스위치웨어(SwitchWare)-펜실바니아대학

스위치웨어는 펜실바니아대학에서 제안한 액

티브네트워크 아키텍쳐로 프로그램가능성(Programmability)과 보안성(Security)과 안전성(Safety)을 유연하게 조정 가능한 액티브네트워크 기반구조(Active network infrastructure)이다. 유연성과 보안성간의 Trade-off는 이 아키텍쳐에서 가장 중요한 특징이다.

스위치웨어는 액티브패킷(Active packet), 스위치렛(Switchlet), 액티브네트워크 기반구조(Active network infrastructure)의 3계층(layer)으로 구성되어 있다(그림 4). 액티브패킷은 기존의 패킷을 대치한 이동 가능한(Mobile) 프로그램을 포함하고 있는 형태이고, 이 프로그램은 PLAN이라고 불리는 언어로 작성되어 있다. 네트워크 요소를 서비스하는 스위치렛은 동적으로 적재가 가능하다.

이는 프로토콜이나 네트워크 요소가 필요시 네트워크을 통해 적재가 가능하다는 것이다. 그리고 액티브네트워크 기반구조는 액티브 패킷과 스위치렛에 고수준의 Integrity 및 보안성을 제공한다. 또한 이 계층에서는 Integrity 검사와 인증 이외에도 강력한 타입검사와 같은 검증(Verification)을 통해 높은 수준의 보안성을 제공한다. 스위치웨어에서는 SANE(Secure Active Network Environment)이라는 액티브네트워크 기반구조를 제안하고 이를 통해 프로토콜이나 네트워크요소를 적재시 Integrity를 보장한다.

그러면 스위치웨어를 이루는 구성요소인 액티브패킷, 스위치렛, 액티브네트워크 기반구조에 대해서 자세히 살펴보도록 하겠다.

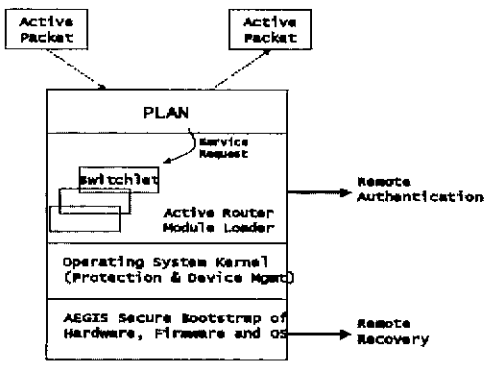

그림 4 SANE의 구조[16]

3.2.1 액티브패킷

스위치웨어에서는 기존 네트워크의 패킷이 액티브패킷이란 이동가능(Mobile) 프로그램으로 대체되고 이는 프로그램코드와 데이타로 이루어져 있다. 액티브패킷의 코드부분은 기존의 패킷헤더의 제어함수(Control function)를 제공하는 동시에 유연성을 제공한다. 이는 기존의 라우터와 같은 환경에서 기존의 헤더가 제공하던 간단한 테이블 참조만이 아닌 좀 더 복잡하고 다양한 작업이 가능하다. 액티브패킷의 데이타부분은 기존의 패킷의 페이로드(payload)부분에 해당하지만 코드부분의 프로그램에서 사용하는 데이타 구조로서의 역할을 겸하고 있다.

여기에 기존의 패킷과는 달리 프로그램을 수행하면서 전송되는 액티브 네트워크의 특성상 네트워크의 성능이 중요한 부분이 된다. 그래서 수행되는 프로그램이 가능한 가벼운 형태로 되어 있어야만 한다. 스위치웨어에서는 PLAN (Programming Language for Active Network)이라는 액티브 패킷용 언어를 정의한다. 이에 대한 자세한 설명은 관련자료를 참고하기 바란다[20].

3.2.2 스위치렛

스위치렛은 중간계층을 구성하며 라우팅프로토콜과 같은 라우터 기능의 기본기능이나 동적으로 적재할 수 있는 추가 확장기능을 제공한다. 이들은 이동가능하지 않다. 스위치렛은 액티브패킷과 같이 필요할 때마다 전송될 필요가 없기 때문에 가벼울 필요가 없다. 스위치렛을 기술하는 언어는 기존의 범용 언어로 가능하나 안전한 수행이 보장되어야만 한다. 스위치웨어에서 네트워크 프로토콜이나 서비스는 PLAN과 스위치렛의 조합으로 구성된다.

3.2.3 액티브네트워크 기반구조

가장 기반 계층인 액티브네트워크 기반구조의 역할은 스위치렛이나 액티브 패킷계층에 안전한 기반을 제공하고 Integrity를 보장한다. SANE(Secure Active Network Environment)은 이러한 기반구조를 구체화한 것이다[16]. SANE은 공개키기반(Public Key In-

frastructure:PKI)이며 다양한 송신자들로부터 보내진 스위치렛에 대한 암호기반의 인증을 제공한다. SANE은 윗 계층의 가벼운 패킷 당 동작을 제공하기 위해서 안전하고 보안성 있게 설계되어 있다(그림 4). 또한 부트스트랩 시 Chained Layered Integrity Checks를 하여 보안성을 유지한다. 그리고 원격 복구 프로토콜(Remote Recovery Protocol)을 제공하여 시스템의 안정성을 높여준다.

3.2.4 PLAN

스위치웨어에서는 언어의 기술과 강력한 타입체크를 통해 동작의 검증(verification)이 가능한 언어인 PLAN을 정의하고 이을 통한 보안성 및 안전성을 보장한다. PLAN에서 제공하는 언어와 노드자원의 할당문제 및 검증문제는 2.3.2절을 참고하기 바란다.

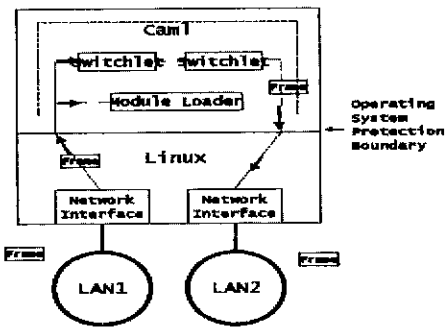

그림 5 Active Bridging[19]

3.2.5 Active Bridging

펜실바니아대학에서는 스위치웨어의 프로토타입으로서 Active Bridging을 시험적으로 구성하고 이를 테스트하였다. Active Bridging에서는 Caml이라는 동적으로 적재가능하며, 안전한 언어를 사용했다. Caml은 자바와 같이 동적으로 적재가능 하면서 가상머신을 제공해 기계독립 적인 특성을 제공한다(그림 5)[19].

3.3 스마트 패킷-BBN 테크놀러지[5]

BBN의 스마트패킷은 시스템을 간단하고 안전하게 유지하기 위해서 각 노드들은 상태정보를 유지하지 않는다(stateless). 사용하는 프로그램은 C언어와 유사하고 보안성이 제공되는 Sprocket이라는 언어를 이용해 기술하고 이를 컴파일 하면 Spanner라고 불리는 복잡한 CISC-like한 형태의 어셈블리로 컴파일 되며 이를 패킷화하여 전송하게 된다. 대신 컴파일 된 코드를 보내기 때문에 크기를 1K바이트 정도의 작은 크기로 잘라 보내게 된다.

스마트패킷의 전송은 ANEP(Active Network Encapsulation Protocol)로 캡슐화되어 기존의 IPv4 패킷이나 UDP패킷의 형태로 전송된다[5, 14]. 이 방법은 일종의 터널링(tunneling)으로 액티브네트워크 시험망(testbed)을 구축할 때 사용되는 기법이다. ANEP는 기존의 네트워크와의 공존할 수 있는 방안으로 활발히 논의되고 있으며 전달과정과 수행과정은 단순하다. 그림 6과 같이 ANEP헤더를 해석할 수 있는 노드만 스마트패킷을 수행하며 전송하게 되고 기존의 노드들은 단순히 포워딩(전달)하는 역할만을 담당하게 된다.

BBN의 스마트패킷은 이외에도 각 스마트패킷의 수행을 제한하기 위해서 접근제어리스트(Access Control List:ACL)을 패킷에 넣어 선택적인 수행이 가능하도록 하면서 보안성을 제공한다.

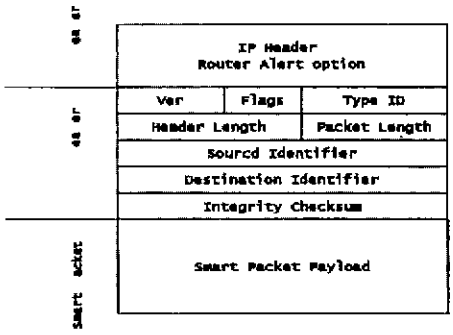

그림 6 스마트패킷의 구조와 ANEP[5]

4. 응 용

액티브네트워크를 구축하고 이를 활용할 수 있는 응용은 다양하다. 네트워크 응용이나 관리부문을 액티브네트워크를 통해 구현하면 여러 가지 장점이 있고 다양하게 연구가 진행되

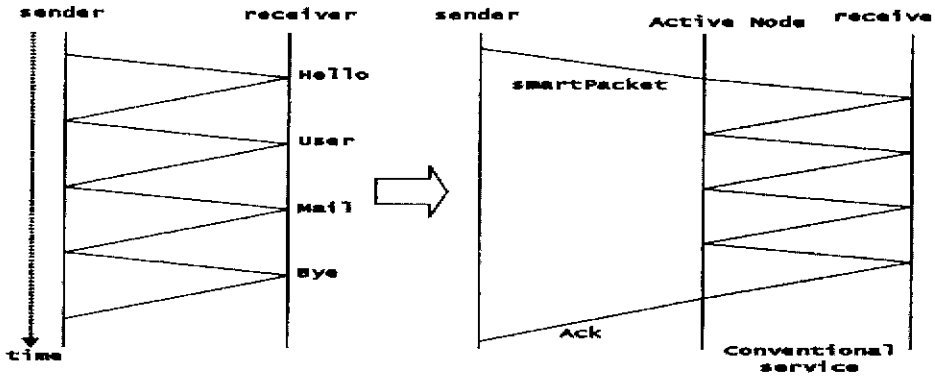

그림 7 mSMTP의 예[4]

고 있다. 메타메일(mSMTP)과 메타HTTP (mHTTP)을 예를 보도록 하자[4].

그림 7에서 보는 것과 같이 기존의 메일은 종단간 통신이었으나 일종의 에이전트프로그램과 같이 중간 액티브노드에서 목적시스템과 통신을 하고 결과만을 송신단말 측에 보냄으로 인해 네트워크의 부하 및 성능을 향상시킬 수 있다.

이와 같은 예로는 ping이나 traceroute 등이 있다[9]. 또한 ATM에서 연결을 설정할 때 필요한 연속된 시그널링 작업을 액티브네트워크로 구현한 경우도 있다. 이 외에 멀티캐스트도 액티브네트워크의 주요한 응용이 될 수 있다. 기존의 NACK/ACK Flooding이나 Reliable Multicast문제를 해결하는 데 액티브네트워크를 활용하면 좋은 솔루션이 될 수 있다[12]. 또한 비디오 게이트웨이를 구축하거나 무선통신에서 스스로 링크상태를 감지해 패킷을 압축함으로 인해서 전송용량(bandwidth)를 효율적으로 활용할 수도 있다.

네트워크관리 부문에서는 SNMP(Simple Network Management Protocol)를 액티브네트워크에서 구현하는 연구가 있다. SNMP가 가지는 여러 가지 기능을 액티브네트워크에서 실현된다면 현재의 SNMP보다 훨씬 강력한 기능을 제공할 수 있다. 또한 네트워크 문제점을 스스로 감지, 복구하고 자동으로 네트워크를 구성하고 관리하는 망인 Self Organizing Network도 연구되고 있다.

5. 결론 및 향후 과제

지금까지 액티브네트워크의 기본 개념, 아키텍쳐, 연구동향, 그리고 액티브네트워크의 응용에 대해 살펴보았다. 기존의 네트워크의 문제점을 개선하고 망에 지능을 부여한다는 측면에서 액티브네트워크는 분명 새로운 접근방식이지만 아직 해결해야 할 많은 이슈가 있다. 다음은 액티브네트워크가 해결해야 할 문제를 각 부문별로 기술한 것이다.

-보안문제(Security) : 액티브네트워크가 패킷의 프로그램코드를 수행한다는 측면에서 중요하게 연구되고 있는 분야이다. 이는 주로 아키텍쳐와 관련이 있으며 안전하고도 효율적인 수행이 가능해야 한다.

-성능(Performance) : 액티브네트워크에서 가장 문제점으로 지적되고 있는 부분이다. 중간의 스위치나 라우터가 단순히 패킷의 헤더만을 처리하지 않고 패킷의 프로그램을 수행하고 전달하는 방식이다. 이 때문에 성능향상에 많은 연구가 필요하다. 성능은 보안문제와 함께 액티브네트워크를 설계할 때 가장 중요하게 여기는 부분이다.

-관리 : 망을 관리를 자동화하는 부분이나 효율적인 유지, 보수가 용이한 아키텍쳐에 대한 연구가 되어야 한다.

-응용 : 액티브네트워크의 장점을 부각할 수 있는 응용을 고안하고 제공해야 한다.

-연동문제 : 기존의 네트워크와의 공존이나

서로 다른 액티브네트워크간의 연동문제를 해결할 수 있는 방법을 고안해야 한다.

현재까지의 단계는 주로 액티브네트워크의 아키텍처를 제안하고 이를 검증했다면 앞으로 해야할 일은 이를 보완하고 실제 활용하는 단계로 나가기 위한 제반사항들을 정의하고 이를 해결하는 것이라고 할 수 있겠다.

참고문헌

[1] Bell Communications Research Inc., "Advanced Intelligent Network(AIN) Release 1 Switching Systems Generic Requirements", TA-NWT-001123, May 1991.

[2] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A Survey of Active Network Research," IEEE Communications Magazine, Vol. 35, No. 1, pp. 80-86, January 1997.

[3] D. L. Tennenhouse, and D. Wetherall. "Towards an Active Network Architecture", Computer Communication Review, Vol. 26, No. 2, April 1996.

[4] A. Kulkarni, G. J. Minden, R. Hill, Y. Wijata, A. Gopinath, S. Sheth, F. Wahhab, H. Pindi and A. Nagarajan, "Implementation of a Prototype Active Network", OPENARCH'98, San Francisco, 1998.

[5] B. Schwartz, W. Zhou, A. Jackson, W. T. Strayer, D. Rockwell, and C. Partridge, "Smart Packets for Active Networks", BBN Technologies, Jan 1998.

[6] S. Bhattacharjee, K. Calvert, and E. Zegura. "An Architecture for Active Networking", Proceedings of High Performance Networking 97, White Plains, April 1997.

[7] Y. Yemini and S. da Silva, "Towards Programmable Networks", IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, L'Aquila, Italy, 1996.

[8] D. Alexander, W. Arbaugh, M. Hicks, P. Kakkar, A. Keromytis, J. Moore, C. Gunter, S. Nettles, and J. Smith, "The SwitchWare: Active Network Architecture", IEEE Network Special Issue on Active and Controllable Networks, vol. 12 no. 3, pp. 29-36, 1998.

[9] M. Hicks, P. Kakkar, J. T. Moore, C. A. Gunter, and S. Nettles. "PLAN: A Programming Language for Active Networks", Proceedings of the International Conference on Functional Programming(ICFP), Baltimore, 1998.

[10] U. Legedza, D. Wetherall and J. Guttag, "Improving the Performance of Distributed Applications Using Active Networks", IEEE INFOCOM '98, San Francisco, 1998.

[11] D. Wetherall, J. Guttag, and D. L. Tennenhouse. "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", IEEE OPENARCH'98, San Francisco, April 1998.

[12] Li-wei, H. Lehman, S. Garland, and D. L. Tennenhouse, "Active Reliable Multicast", IEEE INFOCOM'98, San Francisco, 1998.

[13] Battacharjee, K. L. Calvert, and E. W. Zegura, "On Active Networking and Congestion", Technical Report GIT-CC-96-02, College of Computing, Georgia Institute of Technology, 1996.

[14] D. S. Alexander, B. Braden, C. A. Gunter, A. W. Jackson, A. D. Keromytis, G. J. Minden, D. Wetherall, "Active Network Encapsulation Protocol (ANEP)", http://www.cis.upenn.edu / ~ switchware / ANEP / docs / ANEP.txt, July 1997.

[15] H. Orman, DARPA "ActiveNets Security", Active Nets Workshop, Philadel-

phia, 1997.
[16] D. Alexander, W. Arbaugh, A. Keromytis, and J. Smith, "A Secure Active Network Architecture: Realization in SwitchWare", IEEE Network Special Issue on Active and Controllable Networks, vol. 12 no. 3, pp. 37-45, 1998.
[17] C. A. Gunter, "Security Infrastructure for Active Networks", Active Nets Workshop, Philadelphia, June 1997.
[18] S. Bhattacharjee, K. Calvert, and E. W. Zegura, "Implementation of an Active Networking Architecture", presented at Gigabit Switch Technology Workshop, Washington University, St. Louis, July 1996.
[19] D. Alexander, M. Shaw, S. Nettles, and J. Smith, "Active Bridging", ACM SIGCOMM '97, Cannes, France, September 1997.

하 석 재
서울대학교 컴퓨터공학과 졸업
서울대학교 컴퓨터공학과 석사
서울대학교 컴퓨터공학과 박사과정
관심분야: 고속라우터기술, 액티브네트웍, 무선 Ad-Hoc네트웍, 자바
E-mail: sjha@mmlab.snu.ac.kr

최 양 희
서울대학교 전자공학과 학사
한국과학기술원 전기공학과 석사
프랑스 ENST 컴퓨터공학 박사
프랑스 CNET 연구소
미국 IBM 왓슨연구소, 한국전자통신연구소 책임연구원
현재 서울대학교 컴퓨터공학과 재직중
관심분야: 고속통신망, 멀티미디어통신, 인터넷
E-mail: yhchoi@mmlab.snu.ac.kr