액티브 네트워크의 구조

김 희 경*

◈ 목 차 ◈

1. 서 론

액티브 네트워크는 1994년 DARPA(Defense Advanced Research Projects Agency)주관으로 차세대 인터넷에서 보다 동적인 라우팅 모델을 개발하기 위해 시작된 프로그램이다. 이 프로그램에는 Network associates, Inc. 와 GTE-BBN Internetworking 외에 MIT, 펜실바니아 대학 등 다수의 대학들이 참여하였다. 액티브 네트워크는 인터넷의 발전에 따라 사용자들은 더욱 복합적인 서비스를 요구하게 된데 반하여 컴퓨팅 환경의 발전은 이에 미치지 못하게 된 것이 등장 배경이 되었다. 네트워크 상에서 수행되는 프로그램들을 하나의 통일된 프로그램 언어로 구현하기는 매우 어려우며 또한 테스트하는데 긴 시간이 소요되어 네트워크 서비스를 수행하는 프로그램을 만들기 어려운 조건을 가지고 있다. 액티브 네트워크의 목적은 새로운 네트워크 서비스를 개발하는 시간과 배포하는데 걸리는 시간을 단축하고자 하는데 있다.

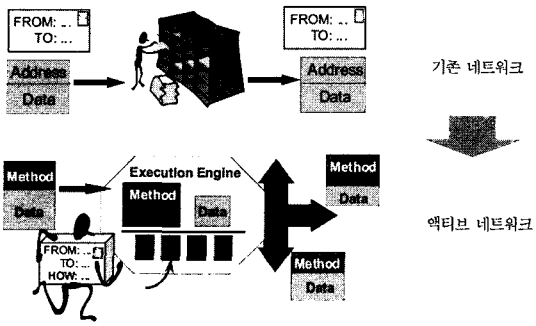
액티브 네트워크를 기존의 네트워크와 비교하여 설명하자면 기존 네트워크의 스위치나 라우터와 같이 패킷이 전송되는 중간 노드들은 IP 패킷의 헤더를 보고 경로 배정 테이블에 따라 패킷을 저장하고 전달하는(store-and-forward) 단순한 기능을 제공하였으나 액티브 네트워크는 데이터의 전달뿐만 아니라 사용자의 프로그램을 패킷이 전송되는 중간 노드에서 실행시키거나 중간 노드에 기 설치되어 있는 프로그램을 선택적으로 실행할 수도 있는(store-compute-forward) 네트워크를 말한다. 네트워크의 노드들은 이제 단순한 라우팅 기능을 수행하는 것이 아니라 오류 처리나 흐름 제어와 같은 기능은 물론 프로그램을 네트워크 경로의 중간 노드의 메모리에 적재하여 실행환경(Execution Environment)에서 수행할 수 있는 융통성과 확장성 및 유연성을 가지게 되었다.

본 논문에서는 액티브 네트워크의 전반적인 구조에 대하여 알아보고자 한다. 2장에서는 액티브 네트워크의 개념, 3장에서는 액티브 네트워크의 노드 구조에 대하여 알아본다. 4장에서는 액티브 네트워크의 패킷 구조에 대하여 기술하고, 마지막으로 결론을 기술하고자 한다.

2. 액티브 네트워크의 개념

액티브 네트워크의 개념은 정적인 라우터의 기능을 캡슐이라고 불리는 "스마트 패킷"을 처리할 수 있는 액티브 노드로 이동하는 것이다. 이 스마트 패킷에는 자신의 고유한 수행 코드를 포함하고 있다. 액티브 네트워크의 또 다른 정의는 삽입된 계산을 통합하는 구조에 대한 새로운 접근 방법이다. 네트워크가 액티브하다는 것은 다음 두 가지의 경우로 설명된다. 첫 번째 경우는 네트워크 내에 있는 라우터나 스위치들은

* 한국원자력연구소 유체공학연구부 선임연구원

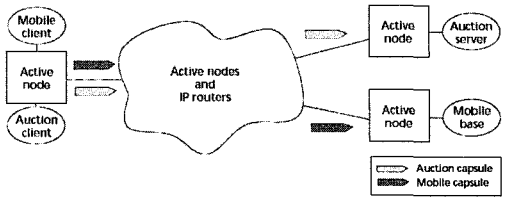


(그림 1) 액티브 네트워크의 개념

자신들에게 유입되는 데이터를 사용하여 계산을 수행할 수 있어야 한다. 두 번째로는 사용자들이 작성한 프로그램도 네트워크의 노드에서 계산을 수행할 수 있어야 한다.

액티브 네트워크의 개념은 정적인 모드에서 수행되는 IPv6 개념을 넘어서고 있다. 네트워크에서 액티브 노드는 캡슐의 내부를 볼 수 있으며 이들 캡슐에 포함되어 있는 실행 코드를 수행할 수 있어야 한다. 또한 역으로 캡슐은 이미 네트워크 상에 위치하고 있는 임의의 코드를 참조할 수 있는 포인터를 가지고 있어야 한다. 액티브 노드는 설치되어 있는 수행 코드를 기반으로 한 행동 양식을 변화시킬 수 있으며 캡슐 내용 자체를 수정할 수도 있다. 캡슐은 네트워크에서 자신의 경로를 지정할 수도 있다. 때로는 네트워크에서 저장되어 있는 정보를 근거로 하여 경로가 변화되기도 한다. 이러한 기능은 고유한 정보로부터 이동되는 패킷을 제어할 수 있기 때문에 네트워크의 혼잡성을 줄이는데 기여할 수 있다. 패킷은 저장된 정보에 대한 지시자로 실행을 요청한 사용자에게 좀 더 가깝게 느껴질 수 있다. 액티브 네트워크의 개념은 새로운 네트워크 기술을 수용하고 이를 운영하는 방법을 변화시키기 위해서 네트워크를 보다 유연하게 만드는데 있다. 위의 그림 1은 일반적인 패킷의 전달 방법과 액티브 패킷의 수행 및 전달 방법에 대한 개념을 보여준다.[1]

응용프로그램들이 액티브 네트워크 상에서 사용되는 예를 경매 프로그램으로 설명하고자 한다. 다음의 그림 2에서 보면 경매 캡슐과 모바일 캡슐이 네트워크 상으로 전송되면 각각의 캡슐 유형에 따라 서로 다른 경로가 설



(그림 2) Auction 의 처리 과정

정된다. 캡슐은 네트워크에서 목적지에 도달하기까지 액티브 노드를 거치기도 하고 일반적인 IP 라우터를 거치기도 한다. 일반 라우터에서는 정상적인 방법으로 전달되며 액티브 노드에서는 관련된 부분을 수행한 후 전달된다. 예를 들면 어떤 모바일 캡슐은 액티브 노드에서 전달 포인터를 저장해 둘 수도 있으며 다른 타입의 캡슐은 저장되어 있는 포인터를 사용할 수도 있다. 이러한 방법으로 새로운 네트워크 서비스가 적용된다.[2]

- 액티브 노드 방식

액티브 노드 방식은 액티브 네트워크의 초기 모델로서 기존 패킷의 형식을 최대한 유지하거나 최소한의 수정을 하여 사용하는 방식을 말한다. 이 방법은 프로그램이 Active Network의 중간 노드에 미리 설치되어 있어서 사용자는 자신이 실행시키기를 원하는 프로그램을 지정할 수 있는 포인터와 해당 프로그램을 수행시키는데 필요한 데이터를 전송하여 프로그램을 수행시킬 수 있다. 이러한 방법을 스위치 접근 방법(Programmable Switch Approach) 방법이라 부르며 이 방법은 중간 노드에서 실행되는 프로그램의 전송과 사용자의 실행 요구 전송이 분리(Discrete)되며 프로그램은 인증된 관리자에 의해서 네트워크에 배포된다. 프로그램의 크기가 커서 사용자들의 요구에 따라 프로그램 자체를 전송하기가 부담스러운 경우에 유리한 방법이며 프로그램이 네트워크의 중간 노드에 기 설치되어 있는 구조를 말한다. 이러한 구조를 적용한 예로는 워싱턴 대학의 DAN[3], MIT의 ANTS[4]을 들 수 있다.

- 액티브 패킷 방식

액티브 패킷 방식은 중간 노드에서 실행될 프로그램과 데이터를 포함하는 패킷(캡슐)을 전송하는 방법

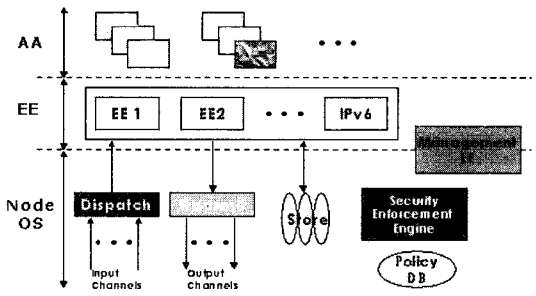
으로 통합 방법(Integrated Approach)이 있다. 실행 노드는 자신의 실행 환경 내에서 데이터를 사용하여 프로그램을 수행한 후 결과를 다음 노드로 전송하거나 송신 측으로 회신한다. 이러한 방법은 전달해야 하는 프로그램의 크기가 큰 경우에 불리하며 또한 네트워크에 부하를 주어 전체적인 네트워크의 효율을 떨어뜨릴 수 있다. 이 방법은 사용자의 프로그램들이 네트워크에 전송되므로 첫 번째 방법인 액티브 노드 방법에 비해 더 많은 보안 문제를 유발할 수도 있다. 액티브 패킷 방식을 적용한 예로는 미국 BBN 연구소의 Smart Packets[5], Active IP Option[6], M0[7] 등을 들 수 있다.

- 혼합 방식

혼합 방식은 위의 두 가지 방식의 장점을 수용한 방식으로 패킷 전달 과정에서 발생 가능한 전달 지연이나 패킷 분실에 대한 해결책을 제시하고 있다. 이 방법에서는 크기가 큰 프로그램의 경우에는 네트워크의 중간 노드에 미리 적재하여 사용하고 비교적 작은 규모의 프로그램은 캡슐 형태로 액티브 패킷에 포함하여 전송하는 방법을 사용하고 있다. 이 방식을 적용한 예로는 미국 펜실바니아 대학의 Switchware 프로젝트[8,9]와 콜럼비아 대학의 Netscript 프로젝트[10]이다.

3. 액티브 네트워크 노드 구조

액티브 네트워크는 처리 속도 면에서 취약한 특성이 있으므로 이를 보완하기 위해서 하드웨어와 소프트웨어 구조를 최적화하여 속도 감소를 보완해야 한다. 빈번히 수행되거나 빠른 처리 속도를 요구하는 기능들은 액티브 네트워크 노드의 시스템 커널에 구현하고 이외의 기능들은 사용자 커널에 구현한다. 액티브 네트워크 노드는 일반적으로 세 개의 계층으로 구성된다. 가장 아래 계층에는 노드 운영체제가 위치하며 패킷 스케쥴링, 자원 관리, 패킷 분류와 같은 기능을 처리하게 된다. 그 다음 계층은 실행환경(EE: Execution Environment)으로 캡슐에 포함되어 있는 프로그램과 데이터를 처리하기 위한 환경을 제공한다. 가장 상위 계층을 구성하는 액티브 응용계층(AA: Active Application)은 특정한 실행환경의 가상 머신(Virtual Machine)에서 실행되는 네트워크 계층 이상의 응용 프로그램으로 구성된다.



(그림 3) 액티브 노드의 구조(DARPA AN WG. 1999)

위의 그림 3은 DARPA의 액티브 네트워크 워킹 그룹(Working Group)에서 제안하고 있는 액티브 네트워크의 일반적인 구조이다.[11]. 입력 채널을 통해 도착한 패킷은 패킷 분류기를 거쳐서 자신이 수행될 실행환경으로 전달된다. 실행환경에서 처리된 패킷은 출력 채널을 통해 전송된다. 실행환경과 노드 운영체제의 경계에 위치하는 관리 실행환경(Management EE)는 노드에 대한 제어 패킷에 대한 처리를 하며 정책 데이터베이스(Policy DB)를 참조하여 전체 액티브 네트워크 노드에 대한 관리를 한다. 액티브 네트워크를 구성 요소를 각각 설명하자면 먼저 가장 상위 계층을 구성하는 AA 는 네트워크 계층 이상의 응용 프로그램이며 사용자 프로그램이나 네트워크 프로토콜, 네트워크 서비스로 구성된다. EE 는 액티브 패킷을 수행하는 기능으로 유닉스의 쉘과 유사한 기능으로 캡슐의 처리 및 실행을 수행할 수 있는 환경을 제공한다. 노드 운영체제는 EE를 지원하는 기능으로 액티브 노드의 자원을 관리하며 패킷 스케쥴링, 패킷 분류를 담당한다. 노드 자신의 보안을 담당하는 Security Enforcement Engine과 노드의 정책이 기술된 정책 DB들도 노드 운영체제에 위치한다.

액티브 네트워크를 성능과 가격 면에서 생각해 보면 응용 프로그램의 성능에 융통성을 증가시키기 위해서는 네트워크의 액티브 노드들에 계산과 저장을 위한 장비들에 대한 비용이 증가하게 된다. 보안에 대해서는 액티브 프로그램이 액티브 노드에서 수행 중

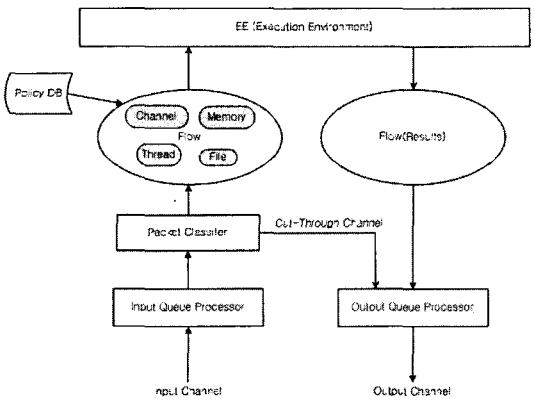
오류를 발생시키더라도 액티브 노드 자체의 안전성에 영향을 미치지 않아야 하며 액티브 노드의 자원을 사용함에 있어서 액티브 프로그램에게 할당된 자원만을 사용하며 그 이상의 공유되는 자원을 사용하지 않도록 구성해야 한다.

- 액티브 네트워크 노드 운영체제

액티브 노드 운영체제(NodeOS)는 다양한 응용 프로그램의 수행을 보장하기 위해서 다중 실행환경을 지원할 수 있어야 한다. 또한 실행환경이 노드의 자원을 적절히 사용할 수 있도록 자원에 대한 관리 기능을 수행한다. 다중 실행환경을 지원함에 따라 특정 실행환경의 장애가 다른 실행환경에 영향을 미치지 않도록 설계되어야 한다. 패킷의 전송에 대한 기본적인 채널을 구현하여야 하며 실행환경에서 응용 프로그램이 요구하는 자원에 대한 관리 기능을 수행하여야 한다. 노드 운영체제는 응용 프로그램들이 안전하게 수행되며 공평한 자원을 할당받을 수 있도록 배려해야 한다. 여기서 자원이란 CPU, Memory, Thread 등과 같은 처리 자원과 Bandwidth, Routing Table 등과 같은 네트워크 자원들을 의미한다. 노드 운영체제는 패킷 분류기(Packet Classifier), 자원 관리기(Resource Manager), 출력 큐 관리기(Output Queue Scheduler) 등으로 구성된다.[그림 4]

- 패킷 분류기

패킷 분류기는 다른 노드로부터 전송된 패킷을 분류한다. 전송되는 패킷에는 실행 코드를 가지고 있는 액티브 패킷과 일반적인 데이터를 전달하는 일반 패킷(IP Packet)이 섞여있게 된다. 따라서 패킷 분류기는 이들을 구별하여 일반 패킷인 경우에는 실행을 요구하지 않으므로 바로 출력 큐로 전달한다. 전송된 패킷이 액티브 패킷인 경우에는 패킷 분류기 내에 있는 Demux Key Table 에 등록되어 있는 분류 규칙과 비교해서 일치된 경로가 존재하면 이미 설정되어 있는 경로로 전달하고 등록되어 있지 않은 경우에는 패킷에 명시된 실행환경으로 전달한다. Demux Key Table 에 등록하고 삭제하는 일은 실행환경과 상호 작용을 통해서 수행된다.



(그림 4) 액티브 노드 운영체제

- 자원 관리기

자원 관리기는 실행환경에서 패킷을 처리하기 위한 자원에 대한 할당, 제어 및 반환에 대한 기능을 수행한다. 실행환경의 논리적인 자원 요청에 대한 물리적인 자원의 공평하고 효율적으로 관리를 위해 DARPA 노드 운영체제 인터페이스에서는 다섯 가지 주요 추상화 개념을 정의하고 있다. 이들은 도메인, 쓰레드 풀, 메모리 풀, 채널 그리고 파일이다.

- 도메인: 도메인은 어카운팅(Accounting)과 접근 제어(Admission Control), 시스템 내부의 스케쥴링을 위한 추상화 개념으로 하나의 패킷 플로우를 처리하기 위해 필요한 노드 운영체제와 실행환경에서 필요한 자원들을 캡슐화한다. 모든 자원의 할당과 프로세싱은 도메인을 기준으로 이루어진다. 이러한 자원은 다음에 설명하고 있는 추상화된 자원인 채널, 메모리 풀, 쓰레드 풀 들로 구성된다. 입력 채널을 통해서 입력된 패킷은 실행환경에서 액티브 코드를 통해서 수행되고 출력 채널을 통해서 출력되는데 이러한 일련의 모든 과정은 하나의 도메인을 통해 처리된다.
- 쓰레드 풀: 쓰레드 풀은 계산 자원인 쓰레드와 CPU에 대한 추상화이다. 쓰레드 풀은 도메인 생성 시 초기화되고 해당 도메인에 할당된다. 각 패킷은 하나의 쓰레드를 배정 받으며 입력 채널, 코드 수행, 출력 채널에 필요한 모든 작업은 이 쓰레드를 통해 이루어진다.

- 메모리 풀: 메모리 풀은 메모리에 대한 추상화이다. 입출력 버퍼와 같은 패킷 버퍼, 실행환경, 상태정보 등을 유지하기 위해 필요한 메모리를 제공한다. 패킷을 저장할 버퍼, 실행환경이나 액티브 노드의 상태를 저장하는데 사용될 메모리를 위한 추상적 개념이다. 메모리 풀은 서로 다른 실행환경들 사이에 정보를 교환하기 위한 목적으로도 사용된다.
- 채널: 채널은 패킷의 송수신을 위한 채널에 대한 추상화이다. 채널은 세 가지 종류가 있다. 입력채널과 출력채널은 실행환경과 하위 통신 기판(substrate)과의 통신을 위해 사용되며, Cut-through 채널은 실행환경의 처리과정을 필요로 하지 않는 패킷을 위한 것이다. 채널은 도메인에 의해서 생성이 되며 각 도메인은 하나의 입력 채널과 하나의 출력 채널을 갖는다. Cut-through 채널은 하나의 입력채널에 하나의 출력채널을 연결해서 구성된다.
- 파일: 파일은 지속적인 저장 장치에 대한 추상화이다. 액티브 노드와 각 실행환경에 대한 영구적인 상태 정보 등을 저장하기 위해 사용된다. 파일 시스템 인터페이스는 POSIX 1003.1을 따르며, 계층적인 이름공간을 사용한다. 각 도메인을 "/"로 구분한다. 예를 들어, 노드운영체제는 "/"로 표현하며, 실행환경은 "/실행환경" (예, /ANTS : ANTS 실행환경의 루트)으로 나타낸다.

노드의 물리적인 자원은 크게 메모리, 쓰레드, 채널, 파일로 구분한다. 메모리는 실행환경에서 실행코드를 적재하고, 입출력 채널에 패킷을 저장하며, soft-state 데이터를 저장하기 위해 필요한 메모리를 제공한다. 쓰레드는 코드의 실행과 채널상의 패킷 처리 등을 위해 필요한 CPU 자원을 제공한다. 채널은 각 플로우에 대한 논리적인 채널을 제공한다. 액티브 패킷을 처리하기 위해 설정된 플로우는 각각 하나의 입력채널(Input Channel)과 출력채널(Output Channel)을 가지며, 일반 IP 패킷은 컷쓰로우 채널(Cut-Through Channel)을 통해 처리된다. 파일은 한 플로우에 대한 지속적인 데이터나 플로우들 간의 공유 데이터를 위한 저장 장치를 제공한다. 같은 특성을 가지는 한 패킷 플로우를 처리하기 위해 필요한 모든 자원은 '플로우(flow)'로 추상화 된다. 하나의 패킷 플로우가 노드운영체제에 들어왔을 때 필요한 채널, 메모리, 쓰레드, 파일은 한 플로우로 구별되며, 노드운영체제는 각 플로우 단위로 자원을 할당, 제어 및 반환한다.

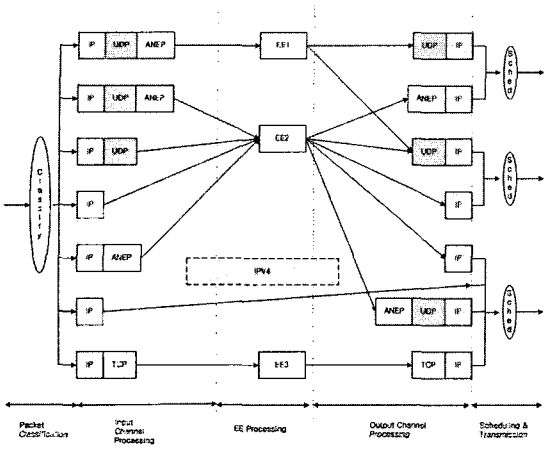
- 출력 큐 처리기

출력 큐 처리기는 패킷 분류기나 플로우의 출력채널로부터 들어온 출력 패킷을 다른 노드로 전달한다. 이때 패킷 분류기로부터 받은 일반 IP 패킷들과 각 플로우의 출력채널로부터의 패킷에 대한 네트워크 대역폭을 공정하게 분배해야 한다.

이외에도 자원 관리기에서 플로우에 대한 자원을 할당할 때 적용할 정책을 갖고 있는 정책 데이터베이스(policy database)와 패킷 분류기와 출력 큐 처리기에서 필요한 액티브 패킷의 인증, 암호화, 권한부여 등과 같은 액티브 패킷의 보안과 노드 자신의 보안을 보장하는 보안 강제 엔진(security enforcement engine)이 있다.

- 액티브 네트워크 노드 실행환경(EE: Execution Environment)

AA(Active Application) 가 실행되는 기반 구조 제공, 자바 가상 머신과 유사한 개념이다. 실행환경은 프로그램 코드를 적절한 실행환경으로 보낼 수 있도록 제어하는 프로그래밍 인터페이스(programming interface)와 가상기계(virtual machine)를 정의한다. 실행환경에서 처리되어질 프로그램은 패킷 자체에 실려서 인밴드(in-band)로 운반되거나 아웃오브밴드 (out-of-band)로 설치(install)된다. 아웃오브밴드 다운로딩은 패킷이 도달했을때 온-디멘드(on-demand)로 일어나거나 별도의 시그널링 단계에서 발생할 수도 있다. 실행환경은 유닉스시스템의 쉘(shell)처럼 동작하며, 액티브 어플리케이션이나 코드가 실행되는 기반 구조를 제공한다. 노드 운영체제와 액티브 어플리케이션 사이에 존재하며 두 계층의 중계 역할을 담당한다. 실행환경은 자바(Java) 가상 머신(Virtual Machine)과 유사한 개념으로 설명될 수 있으며, 액티브 어플리케이션의 실행을 위한 제한된 프로그래밍 환경을 제공한다[12]. 또한, 노드 운영체제를 통해 액티브 코드의 주요 자원에 대한 접근을 제어한다. 특히, 실행환경 중에서 관리를 위한 실행환경은 노드에 대한 보안 정책 설정, 새로운 실행



(그림 5) 액티브 노드를 통한 패킷 처리 단계

환경의 적재, 기존의 실행환경에 대한 설정 변경, 액티브 네트워크 관리 등의 특별한 기능을 수행한다. 위의 그림 5는ANEP(Active Network Encapsulation Protocol) 표준을 사용하여 액티브 노드를 통한 일반적인 패킷 처리를 설명하는 기본적인 동작모델을 나타낸다. 패킷이 노드에 도착하면 패킷 분류 단계와 입력 채널 처리 단계를 거쳐 적절한 실행환경으로 전송되며, 실행환경의 처리가 끝나면 출력채널 처리 단계를 거쳐 적절한 인터페이스로 패킷이 전송된다.

- 액티브 코드(AC: Active Code)

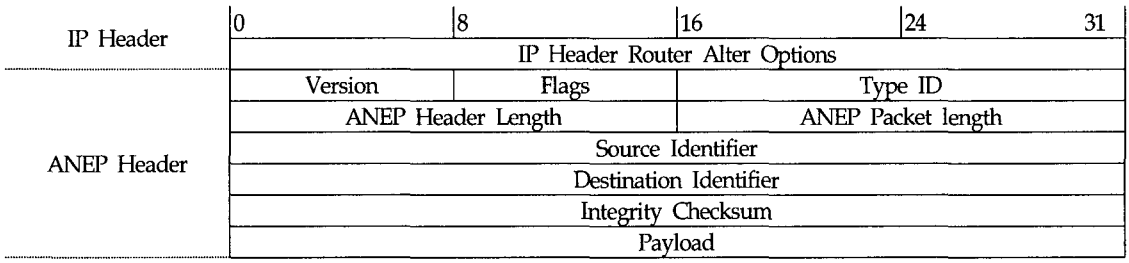
액티브 코드는 실제 실행환경 내에서 동작하는 코드를 의미하며, 전송되는 패킷 내에 있는 사용자의 프로그램 코드나 프로그램에 대한 참조 식별자 모두를 지칭한다. 액티브 어플리케이션은 액티브 네트워크 기반의 사용자 서비스로서, 액티브 코드들을 접근하거나 응용할 수 있는 방법들을 제공한다. 액티브 코드 및 어플리케이션은 액티브 네트워크 구현 방식에 따라 다양한 형태로 구현된다. 액티브 코드는 액티브 네트워크의 구현에 있어서 직면하게 되는 두 가지 문제점인 코드 수행시 액티브 노드의 안전성을 보장하는 것과 액티브 노드의 침입에 대한 안전성을 담당하는 역할을 한다.

4. 액티브 네트워크 패킷 구조

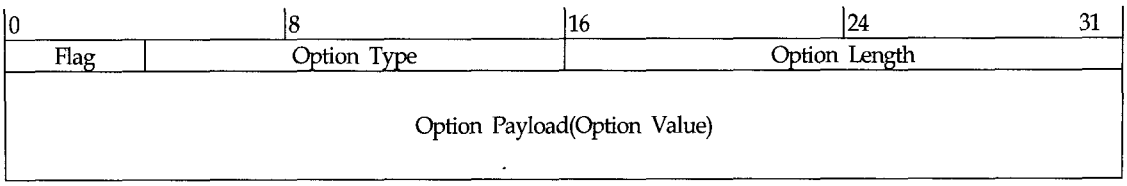
액티브 노드에 액티브 패킷이 도착하게 되면 패킷의 헤더를 보고 적절한 EE 를 선택하여 해당 EE 로 패킷을 보내는데 이때 EE 를 선택하는 방법에는 두 가지가 있다. 첫째는 ANEP(Active Network Encapsulation Protocol)라 불리는 새로운 가변 길이 헤더를 사용하는 방법이며, 둘째는 고정 길이 필드를 기반으로 하는 사용하는 방법이다. 우선 SAPF(Simple Active Packet Format)라 불리는 헤더를 살펴보고 다음으로 특정 EE 를 위해 설계된 패킷 헤더들의 형식에 대해서 살펴본다.

EE 분류 (EE Demultiplexing)

- ANEP : 액티브 네트워크의 액티브 노드에서는 다양한 언어로 작성된 프로그램을 로딩하고 실행시킬 수 있으며, 이러한 프로그램들은 액티브 패킷의 페이로드에 실려 전송된다. ANEP는 그림 6과 같은 헤더 구조를 가지며 프로그램은 ANEP에 의해 명시된 실행환경 안에서 실행되어 처리된다. ANEP 구조는 다음과 같은 필드들을 가진다.



(그림 6) ANEP 헤더의 형식



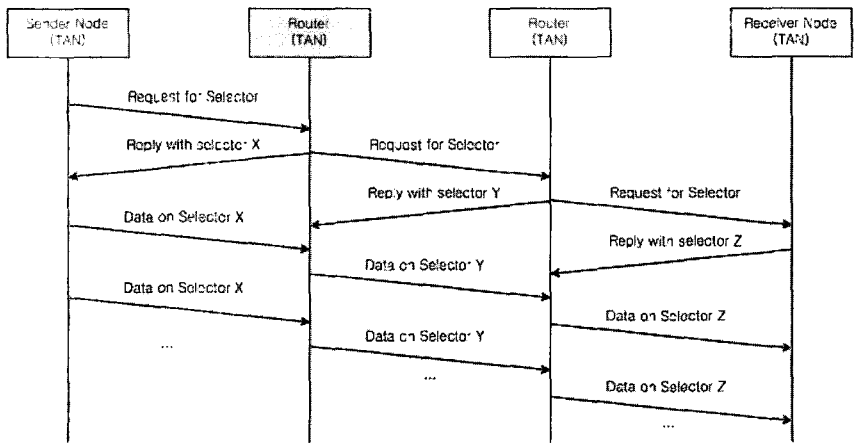
(그림 7) 옵션 필드 형식

- Version : 현재 사용중인 ANEP 헤더의 종류
- Flags : 액티브 노드가 Type ID를 인식하지 못할 때 패킷을 어떻게 처리할 것인지에 대해서 명시. 필드 값이 0이면 디폴트 라우팅 메커니즘을 사용, 1이면 패킷을 폐기
- Type ID : 패킷이 수행될 실행환경 값
- ANEP Header Length : 32비트 워드 단위로 ANEP 헤더의 길이
- ANEP Packet Length : 패킷 전체 길이를 옥텟 단위로 표시
- Option : 소스 식별자, 목적지 식별자, 무결성 체크섬, N/N 인증 표시

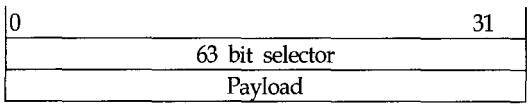
옵션 필드의 형식은 그림 7과 같으며 현재 제공되는 옵션 타입으로는 소스 식별자(1), 목적지 식별자(2), 무결성 체크섬(3), N/N인증(4) 등이 있다. 2비트 플래그 필드의 첫 번째 비트는 옵션 타입이 어떤 특정 Type ID에서만 의미를 갖는지를 명시하고 두 번째 비트는 해당 옵션 타입을 인식하지 못할 때 취해야 할 동작을 명시한다. 0일 경우에는 옵션을 무시하고 헤더를 처리하며, 1일 경우에는 패킷을 폐기한다.

- SAPF(Simple Active Packet Format)

ANEP 헤더가 가변의 옵션 필드를 가지고 있으며 특정 실행환경을 위해 명시된 패킷 헤더들도 가변길이 필드들을 가지고 있으므로 ANEP 헤더를 이용하여 실행환경을 분류하는 방법은 느리고 복잡하다는 단점을 피할 수 없게 된다. 이러한 ANEP의 단점을 보완하기 위해 SAPF는 헤더 정보를 줄여서 패킷 포워딩을 위한 기본 기능과 실행환경 분류를 위한 시간을 최소화하고자 제안되었다. 실제로 ANEP 대신 SAPF를 사용하여 액티브 패킷을 처리하였을 경우 30%정도 처리시간이 빨라진다고 한다. SAPF는 첫 번째 데이터가 전송되기 전에 이웃 TAN(Tags for Active Networking)노드들 사이에 셀렉터(selector)를 설정하게 되는데 그림 8은 그 과정을 보여준다. 송신자는 데이터 전송을 시작하기 전에 상태 레코드(state record)를 생성하고 셀렉터를 위해 다운 스트림 TAN 라우터에게 실렉터 요청을 한다. 셀렉터 요청 패킷을 받은 TAN 라우터는 상



(그림 8) TAN 노드의 셀렉터 설정



(그림 9) SAPF 헤더

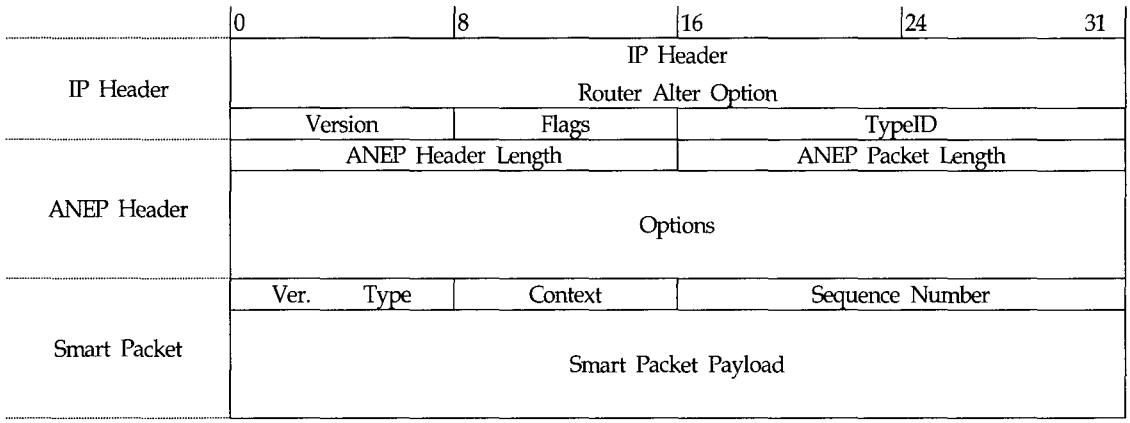
태 레코드를 생성하고 라우터 안에서 유일한 셀렉터를 선택하여 요청에 응답한다. 송신자는 셀렉터에 대한 정보를 수신한 후 상태를 저장하고 셀렉터를 이용하여 데이터를 보내기 시작한다. 셀렉터를 이용한 SAPF 헤더는 그림 9와 같으며 헤더의 길이가 고정되어 있으므로 신속하게 패킷 분류 작업을 수행할 수 있다.

- SMART Packet

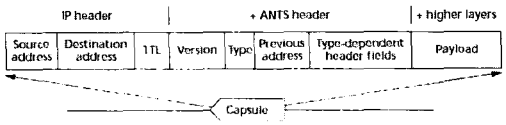
SMART 패킷은 효율적인 망 관리를 위하여 설계되었으며 액티브 노드의 부하를 줄이고 필요에 따라서 관리 정보를 제어할 수 있으며 구현 언어는 Spanner, Sprocket이다. BBN(Bolt, Beranek and Newman, Technologies, http://www.bbn.com)에서 추진하고 있는 Smart Packet은 액티브 네트워크 기술을 이용하여 NMS(Network Management System)의 성능과 유연성을 향상시키려는 목적으로 개발된 네트워크 관리 및 모니터링에 관련된 프로젝트이다. 스마트 패킷(smart packet)의 프로그램과 데이터는 ANEP로 캡슐화 되어 기존의 IPv4 패킷이나 UDP 패킷의 형태로 전송된다. ANEP 헤더를 처리할 수 있는 노드만이 스마트 패킷을 처리하여 전송하고, ANEP 헤더를 처리하는 기능을 가지지 않은 기존의 노드들은 단순히 포워딩하는 역할만을 담당하게 된다. 그림 10 은 ANEP를 이용하여 스마트 패킷을 캡슐화한 액티브 패킷 모습을 보여준다. 스마트 패킷에서의 액티브 패킷 구조는 IP 헤더, ANEP 헤더, 스마트 패킷으로 구성된다. 스마트 패킷의 헤더는 다음과 같은 필드들로 구성된다.

Ver.: 현재 스마트 패킷의 버전
Type: 페이로드에 적재된 메시지 타입 표시
  프로그램 패킷: 적절한 호스트에서 실행되어질 코드를 운반
  데이터 패킷: 네트워크 관리 프로그램을 생성한 곳으로 반환하는 실행 결과를 운반
  에러 패킷: 프로그램 패킷의 전송 또는 Spanner 코드의 수행시 예외사항이 발생할 경우에 반환되는 오류 정보를 운반
  메시지 패킷: 노드에서 실행되어질 코드에 대한 정보 메시지를 운반
Context: 스마트 패킷의 생성자를 나타냄
Sequence Number: 동일한 Context의 메시지를 구분하는데 사용

이러한 스마트 패킷은 ANEP 패킷 내에 캡슐화되며, ANEP 패킷은 다시 IP 패킷 내에 실려서 전송된다. IP 헤더는 기존 네트워크 구조에서 액티브 패킷을 전송하기 위해서 사용되며, ANEP 헤더는 Type ID 필드를 통해 스마트 패킷을 위한 실행환경으로 전달되도록 하기 위해 사용



(그림 10) ANEP를 이용한 스마트 패킷의 액티브 패킷 구조



(그림 11) ANTS 액티브 패킷 구조

된다. 스마트 패킷의 페이로드에는 사용자가 작성한 프로그램이 적재된다.

- ANTS(Active Node Transfer System)

빠른 전송과 배포를 목적으로 하며 동시에 다수의 프로토콜을 지원하도록 설계되었다. 주요 구성요소로는 캡슐과 액티브 코드, 코드 분배 시스템이다. ANTS는 1998년에 MIT에서 개발한 액티브 네트워크 툴킷으로 이동코드(mobile code), 요구시 적재(demand loading), 코드 캐싱(code caching)기술 등을 사용한다. 응용과 액티브 노드의 구현은 자바로 되어 있으며 구체적인 액티브 패킷 구조는 그림 11과 같다. 위의 그림 11 에서는 사용자의 데이터와 수행할 프로그램을 가리키는 식별자를 캡슐화한 IP 패킷을 보여준다. 이 수행 프로그램은 소스 노드에서 응용 프로그램에 의해서 선택되며 패킷이 전달되는 모든 액티브 노드에서 수행된다. 일반적인 노드에서는 단순한 전달만이 이루어지며 이들 프로그램들은 노드 환경에서 질의를 위해 ANTS 가 제공하는 API 에 의존적이다. 또한 복수의 사용자가 (프로그램과 캡슐사이의 명시적인 연관으로 인한 충돌 없이) 동시에 다른 경로를 사용하여 배포할 수 있다.

Version: 현재 ANTS의 버전

Type: 액티브 패킷 타입 구분과 액티브 패킷을 위한 코드가 존재하는 위치 표시

Previous address: 가장 최근에 지나온 액티브 노드의 주소를 나타내며 요구시 적재되는 경우에는 코드 요청 노드로 사용됨.

Type-dependent header fields: 타입 값에 따라 다양한 종류의 헤더 필드가 존재

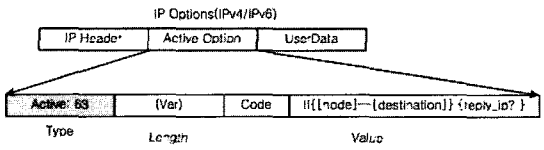
Payload: 각 액티브 노드에서 수행되는 프로그램이 존재

실제 ANTS 헤더의 필드 길이나 헤더의 위치는 구현에 따라 달라질 수 있다. ANTS 버전 3.1에서는 ANEP 페이로드에 ANTS 헤더를 위치시키고 있다. ANTS의 액티브 패킷에는 액티브 패킷을 처리하기 위해 사용될 포워딩 루틴에 대한 참조(reference)를 포함한다. 포워딩 루틴은 두 가지로 분류 될 수 있는데, 첫 번째는 모든 액티브 노드에 항상 탑재되어 있어서 언제나 이용될 수 있는 well-known 루틴이며 두 번째는 어떤 액티브 노드에는 존재하지 않을 수 있으므로, 필요시 코드 분배 메커니즘을 이용하여 노드로 주입시켜야 하는 응용 특정(application-specific) 루틴이 있다.

ANTS를 실제로 기존 IP 네트워크에 구현하고자 제안한 방법으로 액티브 IP 옵션(Active IP option) 방법이 있다. 기존의 IP 패킷의 IP 옵션 필드에 액티브 패킷 구분을 위하여 옵션 형식으로 정의해 놓는 방법이다. 새로운 캡슐화 방식을 위하여 액티브 패킷의 새로운 헤더를 정의하는 것과 달리 이 방식은 기존의 IP와 호환성을 유지한다는 측면에서 장점을 지닌다. 현재 IP 옵션 필드는 일반적 데이터그램이 아닌 데이터그램을 처리하는데 사용되거나 혹은 주로 네트워크 감시와 측정을 위해 사용되고 있다. 따라서 기존의 인터넷 패킷과는 다른 형태와 특성을 지닌 액티브 패킷에 적합할 것이다. IPv6 에서 IP 자체 헤더는 간단해지고 여타 복잡한 처리는 모두 옵션 필드가 할 수 있도록 하였다는 관점에서 IP 옵션 필드를 활용하는데 대한 타당성을 찾아 볼 수 있다. 옵션 타입 필드의 첫 번째 비트는 copy 필드이며 라우터가 패킷 분할(fragmentation) 시에 분할된 모든 분할 패킷에 해당 옵션을 복사해서 넣을 것인지('1') 아니면 첫 번째 패킷에만 넣을 것인지('0')를 명시한다. 다음의 두 비트는 옵션 필드를 나타내며 '00'은 네트워크 제어를 위한 데이터그램으로 '10'은 디버깅과 네트워크 측정을 위해 사용된다. 나머지 '01' 과 '11'은 차후 사용을 위해 예약되어 있다. 그림 12는 액티브 IP 옵션 필드를 나타내고 있으며 옵션 타입의 copy 필드에는 '0', option class 필드에는 '01', 마지막 option number 필드에는 '11111' 비트값이 들어간다. ANTS를 실제 네트워크에 구현시킬 때에 ANEP 헤더를 사용할 것인지 액티브 IP 옵션 방법을 사용할지는 추

0	1 2	3 7
Copy	Option Class	Option Number

(그림 12) 옵션에 포함된 IP Type Field(8 bits)

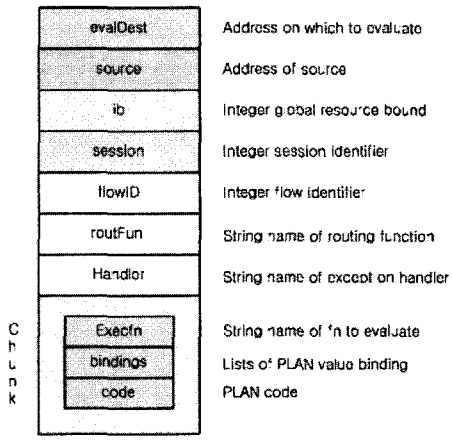


(그림 13) 옵션에 포함된 IP Type Field(8 bits)

후 액티브 네트워크가 어떤 형태로 발전해 갈 것인지에 따라 결정되어질 것이다. 즉, 액티브 네트워크가 기존 IP 네트워크와 별도의 망을 가지는 형태로 발전하게 된다면, ANEP와 같은 형태가 필요할 것이고, 그렇지 않고 기존 네트워크 망에 상당히 국한된 형태의 망으로 발전한다면 액티브 IP 옵션 방법이 좀 더 실용화 가능성을 가지게 될 것이다.[그림 13]

- Switchware

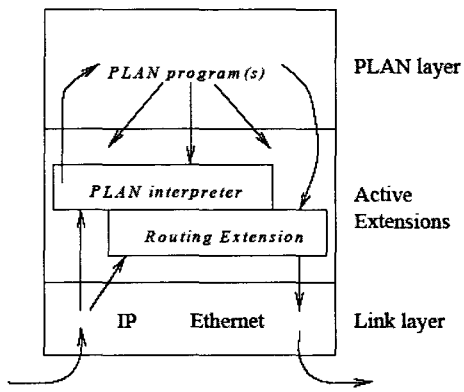
펜실베니아 대학에서 연구한 Switchware는 사용자 요구사항을 PLAN(Packet Language for Active Network)이라는 언어로 작성하여 패킷 안에 넣어 운반하는 캡슐 방식(capsule approach)과 필요한 프로그램들을 미리 어떤 관리자에 의해 노드에 적재시켜 놓는 프로그램 가능한 스위치 방법(programmable switch approach) 모두 사용하고 있다. Switchware는 유연성, 안전성, 성능 향상, 가용성 등의 요소에 균형을 맞추어 설계되었다. 또한 라우터의 기능을 임의적으로 확장시킬 수 있도록 지원하기 위해 OCaml이라는 언어를 제공하며 액티브 네트워크의 유연성과 액티브 노드의 보안과 안전성간의 균형을 잘 유지하고 있다. PLAN 패킷 형식은 그림 14와 같으며 각 필드들은 위에서부터 차례로 살펴보기로 한다. 처음 두 개의 필드는 주소를 나타내는 필드로 IP 주소(32비트)와 포트 넘버(16비트)가 합쳐져 48비트의 길이를 가진다. Evaluation Destination 필드는 패킷에 적재된 프로그램이 실행되어질 목적지 주소를 의미하며, source 필드는 패킷 생성자의 주소를 나타낸다. 그 다음 resource bound 필드는 IPv4의 TTL 필드나 IPv6의 hop count 필드와 유사하며 session 필드는 IP 헤더의 프로토콜 필드와 유사한 호스트의 종단 응용을 위한 식별자를 제공한다. flowID 필드는 라우터를 통과하는 패킷 플로우를 위한 식별자를 제공하고, routing function 필드는 사용되는 경로 배정 함수를 명시한다. handler 필드는 패킷을 처리하는 도중 라우터에서 오류가 발생했을



(그림 14) PLAN의 패킷 형식

때, 송신자에서 제공하는 오류 처리 함수의 이름을 명시한다. 마지막 3개의 필드는 액티브 노드에서 실행되어질 프로그램 코드 묶음(chunk: code chunk)을 의미한다.

PLANet 노드의 구조를 그림 15에 도시하였다.[13] PLANet 은 액티브 네트워크를 위한 패킷 언어인 PLAN으로 작성된 모든 프로그램을 포함한 패킷으로 구성된 액티브 인터넷을 말한다. Switchware는 유연성과 안전성, 보안성, 성능 및 가용성 등의 서로 다른 기능들을 제공하는 레이어 구조로 되어있다. 레이어 간의 이러한 기능들은 각 레이어간의 융통성과 보안성의 균형을 맞출 수 있도록 나누어진다. 가장 상위에 위치한 PLAN 레이어는 고수준의 성능과 보안을 담당하며 반면에 가장 하위의 Link 레이어는 과도한 보안 제어에 기인한 소수의 기능과 낮은



(그림 15) PLANet 노드 구조

(표 1) ANTS 와 Switchware 의 특성 비교

Comparison criteria	ANTS	SWITCHWARE
Runtime environment	Java virtual machine	PLAN/JAVA or OCaml interpreter
Requirements	JDK 1.0.2	JDK 1.1.X and PIZZA library
Programming language	Java	PLAN / OCaml or Java
Link layer	UDP	UDP and TCP
Code distribution	Automatic distribution of code in separated channels	Explicit
Routing	Static routing	Static and dynamic routing
Lifetime of Capsule/packet	User-define TTL when creating a new capsule	User-define Resources bounds when creating a new packet
Cache usage	Decrementing TTL from entering data	None
CPU cycle	No control	No control
Host protection	Based on Java security mechanisms	Authentication mechanisms and namespace security
Preventing code spoofing	Via fingerprint authentication	Via namespace security

성능 특징을 가진다. 위의 표 1에는 ANTS와 Switchware의 특성을 비교하였다.[2]

5. 결 론

액티브 네트워크를 구성하는 노드의 구조와 대표적으로 구현된 패킷의 구조에 대하여 알아보았다. 액티브 네트워크는 기존의 네트워크 개념이나 구조의 비효율성과 수동성을 개선해 보고자 하는 생각에서 출발한 새로운 패러다임이다. 네트워크를 구성하고 있는 각각의 노드들은 단순한 패킷 전달 기능에서 벗어나 사용자가 요구하는 프로그램을 수행하여 결과를 전달하며 서비스 요구에 따라 다양한 작업을 수행하여 보다 능동적이고 유연한 네트워크를 구성하는데 그 목적이 있다. 액티브 네트워크 구조는 네트워크 종단 간에서 이루어지던 서비스의 개념을 네트워크 전체 영역으로 확장하였으며 새로운 프로토콜의 자동화된 전달 및 배포를 통해 네트워크 기술과 서비스에 유연성과 확장성을 도입하였다. 액티브 네트워크 구조의 취약성에 대한 연구가 수행되어 다양한 서비스와 기능을 제공하는 네트워크의 구축이 가능할 것으로 생각된다.

참 고 문 헌

[1] ETRI, "Active Network 기술 동향", 주간기술동향, 2001

[2] Nadjib Achir, "Active Networking System Evaluation: A Practical Experience", MoMuC 2000, Tokyo, Japan, Oct. 2000

[3] D. Decasper and Plattner, B., "DAN: Distibuted Code Caching for Active Networks", INFOCOM'99, New York, 1999.

[4] D. Wetherall, et al., "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols" IEEE OPENARCH'98 Proc., San Francisco, Apr. 1998.

[5] B. Schwartz et al.,"Smart Packets for Active Networks", BBN Technologies, Jan. 1998

[6] Dave Wetherall and David Tennenhouse, "The Active IP Option", Proceedings of the 7th ACM SIGOPS European Workshop, Connemara, Ireland, Sept. 1996.

[7] P. A. Queloz and A.Villazon, "Composition of Distributed Services with Mobile Code", Autonomous

Agents and Multi-Agent Systems Journal, Kluwer Academic Publishers, 4 (4) : 331-337, December 2001.

[8] D.S. Alexander, et al., "The SwitchWare Active Network Architecture", IEEE Network Specail Issue on Active and Controllable Networks, vol. 12 no.3, 1998.

[9] Michael Hicks, Pankaj Kakkar, Jonathan T. Moore, Carl A. Gunter and Scott Nettles, "PLAN : A Packet Language for Active Networks", ICFP, 1998.

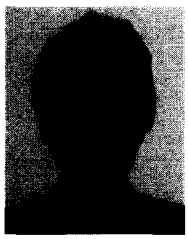
[10] S. da Silva, D.Florissi and Y.Yemini, "Composing Active Services in Netscript", DARPA Active Networks Workshop, Tucson, AZ, Mar. 9-10, 1998.

[11] AN Node OS Working Group, "Architectural Framework for Active Networks", Jul. 1999.

[12] Peter Lee and George Necula, "Research on Proof-Carrying Code for Mobile-Code Security", DARPA Workshop on Foundations for SecureMobile Code, Mar. 26-28, 1997.

[13] D. Scott Alexander et al., "The SwitchWare Active Network Implementation", August 10, 1998, ACM SIGPLAN Workshop

◐ 저 자 소 개 ◑



김 희 경

1987년 동국대학교 전자계산학과 졸업(학사)

1987년~현재 한국원자력연구소 유체공학연구부 선임연구원

관심분야 : 컴퓨터 네트워크, 소프트웨어공학 etc.

E-mail : hkkim@kaeri.re.kr