액티브 네트워크를 위한 액티브 노드 구조 설계

이병기○ 조국현
광운대학교 컴퓨터과학과

Design of Active Node Architecture for Active Networks

Byoungki Lee Kukhyun Cho
Dept. of Computer Science, Kwangwoon Univ.

요 약

액티브 네트워크는 사용자에 의해 커스터마이즈된 응용이나 프로토콜을 망 내의 노드들에 동적으로 삽입하여 망 노드들이 새로운 사용자 요구나 서비스에 신속하게 대응할 수 있도록 한다. 액티브 네트워크는 패킷내에 새로운 응용이나 프로토콜에 대한 코드와 데이터를 캡슐화시켜 전송하는 액티브 패킷과 이러한 액티브패킷을 수신하여 처리하는 액티브 노드로 이루어진다. 본 논문에서는 액티브 네트워크 구조와 구성 요소들에 대해 기술하고, 이를 통해 액티브 패킷의 구조와 수신된 액티브 패킷을 처리할 수 있는 액티브 노드의 구조를 제안한다.

1. 서 론

현재의 망에서 중간의 노드들은 종단의 노드들에 비해 폐쇄적이며, 수직적으로 통합된 시스템이다. 그들의 기능은 표준화 과정을 통해 결정되며, 장비 판매자에 의해 이미 구성되어 내장된 소프트웨어에 의해 구현된다. 새로운 망 프로토콜이나 서비스의 개발은 표준화 위원회에 의해 만들어지지만, 이들 위원회의 모든 구성원이 만족하는 솔루션에 도달하는 데에는 상당히 오랜 시간이 걸린다. 또한 새로운 프로토콜이나 서비스가 만들어졌다 할지라도, 장비 판매자들이 이러한 새로운 기술을 자사 장비에 적용시키는 것은 별개의 부수 작업이며, 또 하나의 문제는 기존의 장비들이 새로운 프로토콜을 지원하지 않기 때문에 발생되는 호환성 문제이다.

액티브 네트워크(Active Network)는 이러한 문제점을 해결하고자 시도하고 있다. 액티브 네트워크란 기존 망에서의 라우터나 스위치와 같은 중간 노드들이 단순히 패킷의 헤더만을 처리하던 데서 한 걸음 더 나아가 사용자가 패킷에 프로그램 코드와 데이터를 함께 넣어 전송하고 중간 노드에서는 이를 처리할 수 있는 환경을 말한다. 또한 중간 노드에서는 단순히 패킷의 경로를 설정하고 전달하는 기능을 담당하고, 에러처리 및 흐름제어와 같은 패킷의 복잡한 처리는 종단의 단말장치에서만 처리하던 것과는 달리 중간 노드에서 여러 가지 처리를 가능하게 함으로서 기존의 망에서 제공하지 못했던 유연성과 다양한 장점을 제공할 수 있다. [1]

본 논문에서는 이러한 액티브 네트워크의 장점을 수용하여 하위계층의 하드웨어로부터 망 서비스들을 분리시키며, 요구에 따라 망 기반구조에 새로운 서비스를 신속히 적용할 수 있도록 하는 액티브 패킷과 액티브 노드의 구조를 제안한다.

2. 액티브 네트워크 구조

2.1 액티브 네트워크 구성 요소

액티브 네트워크는 망 구조를 하드웨어와 소프트웨어를 함께 묶는 메인프레임 마인드에서, 하드웨어와 소프트웨어의 기술 혁신을 분리시키는 가상화 된 접근으로 변화시키는 기회를 제공한다.

액티브 네트워크는 그림 1의 (a)에서 보는 바와 같이, 기존 망에서의 패킷에 해당하는 액티브 패킷과 이를 수행할 수 있는 중간 노드의 수행 환경(Execution Environment)으로 구성된다. 기존의 전통적인 패킷과 달리 액티브 패킷은 실제 수행될 수 있는 프로그램 코드와 데이터로 구성되어 있다. 액티브 네트워크에서 라우터나 스위치 등의 중간 노드, 즉 액티브 노드는 액티브 패킷에 있는 코드가 추출되고, 실행되는 수행 환경을 제공한다는 점에서 프로그램 가능하다고 말하며, 그림 1의 (b)에서와 같이 동작한다. 이러한 수행 환경을 정의하는 데는 여러 가지 사항을 고려해야만 한다. 이러한 고려사항 중 중요한 것으로는 프로그램 언어, 프로그램의 인코딩 방식, 프리미티브 제공문제, 보안 문제 그리고 자원의 할당 및 공유 문제 등이 있다.

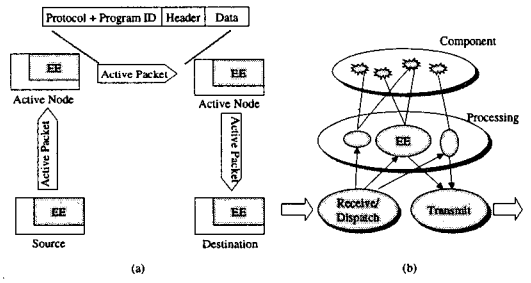

그림 1 액티브 네트워크 및 노드 구조

2.2 수행 환경 (EE)

수행 환경은 액티브 네트웍의 사용자들이 활용할 수 있는 가상 머쉰과 프로그래밍 인터페이스를 정의한다. 수행 환경에 의해 제공된 가상 머쉰의 기능은 아키텍쳐에 의해 정의되지 않는다. 노드 운영체제는 가상 머쉰을 구현하기 위해 수행 환경들에 의해 사용될 수 있는 기본 기능들을 제공한다.[2] 따라서 대부분의 네트웍 아키텍쳐 측면들은 수행 환경에 의해 정의된다. 액티브 네트워크는 망 기반구조에 손쉽게 프로그램들을 추가할 수 있도록 하는데 목적을 둔다. 따라서 수행 환경의 프로그래밍 언어와 런타임 환경의 선택은 중요한 요소가 되며, 이러한 연구 중에 대표적인 스마트 패킷(Smart Packets), ANTS, 스위치웨어(SwitchWare), 그리고 넷스크립트(NetScript)에 대해 살펴본다.

2.2.1 스마트 패킷

BBN 테크놀러지에서 개발된 스마트 패킷은 대규모 복잡한 망들의 관리를 향상시키기 위해 액티브 네트워크 기술을 망 관리에 접목시켜 프로그램 가능한 관리 노드를 만들기 위해 설계되었다. 스마트 패킷에서는 망 관리를 지원하기 위한 기능이 내장된 고수준의 언어인 Sprocket과 이것이 컴파일되어 수행가능한 어셈블리 언어로 변환되는 Spanner라는 두가지 프로그래밍 언어를 제공한다. 다시말해 Sprocket 프로그램들은 Spanner 코드로 컴파일 되고, 이 코드는 다시 프로그램 패킷에 포함되는 기계에 종속하지 않는 형태의 이진 코드로 조합되어, IP 데이터 패킷에 캡슐화되어 전달된다.[6]

2.2.2 ANTS (Active Network Transfer System)

MIT에서는 캡슐을 기반으로 액티브 네트워크를 구축하고, 이를 이용하여 액티브 스토리지, 멀티캐스트 및 트래픽 필터링 등과 같은 여러 응용들을 연구하고 있다. 여기에서 발표한 ANTS는 새로운 네트워크 프로토콜과 서비스를 구성하고 이를 노드에 동적으로 배포할 수 있도록 하기 위한 자바 기반의 액티브 네트워크 툴킷으로서, 액티브 네트워크에 참여할 수 있는 노드의 수행환경과 사용자가 자신의 패킷을 특성화하여 동적으로 노드에 배포할 수 있도록 하는 프로토콜 프로그래밍 모델을 제공한다. ANTS 구조에서는 액티브 네트워크를 구현하기 위해 기존의 패킷을 대체하는 개념으로 패킷과 전송 루틴을 포함하는 캡슐(Capsule)과 라우터나 스위치 등의 중간 노드를 대체하는 개념으로 캡슐에 포함되는 처리 루틴을 수행하고, 관련 상태를 유지하는 액티브 노드 그리고 해당 캡슐을 필요로 하는 노드에 자동적이고 동적으로 전송하는 코드 분배 메카니즘의 3가지 핵심 콤포넌트들로 구성되었다.[7]

2.2.3 스위치웨어

스위치웨어는 펜실바니아 대학에서 제안한 액티브 네트워크 아키텍쳐로 프로그램가능한 네트워크 기반구조의 유연성에 이러한 구조를 공유하면서 발생되는 안정성 및 보안성과의 균형을 맞추는데 목적을 둔다. 스위치웨어웨 아키텍쳐는 PLAN(Packet Language for Active Network)이라 불리는 언어로 작성된 이동가능한 프로그램을 포함하는 액티브 패킷, 라우터의 기본 기능이나 동적으로 적재가능한 추가 확장 기능을 제공하는 중간 계층의 스위치렛(switchlet), 그리고 액티브 패킷과 스위치렛에 고수준의 무결성과 보안성을 제공하는 기반 계층인 액티브 네트워크 기반 구조의 3 계층으로 구성되어 있다.[9]

2.2.4 넷스크립트

콜럼비아 대학에서 개발된 넷스크립트는 프로토콜이나 서비스를 구현하기 위한 언어이자 환경으로서, 이 언어로 개발된 프로그램들은 새로운 프로토콜이나 서비스들을 가지고 동적으로 망을 확장하기 위해 망 노드들에 삽입될 수 있다. 넷스크립트 망은 개별적으로 하나 이상의 넷스크립트 엔진들을 수행하는 망 노드들의 집합으로 구성되며, 여기에서 넷스크립트 엔진은 프로그램가능한 패킷 처리 장치를 위한 소프트웨어 형태의 추상이다. 각 넷스크립트 엔진은 상자(box)라 불리는 데이터플로우(dataflow) 콤포넌트들로 이루어져 있으며, 이러한 콤포넌트들은 자신들을 경유하는 패킷 스트림들을 처리한다. 기본적으로 넷스크립트 상자들은 패킷 헤더 분석, 패킷 역다중화 또는 기타 프로토콜 기능들을 수행한다.[8]

2.3 노드 운영체제

노드 운영체제는 수행 환경들과 하위의 물리 자원들(전송 대역폭, 프로세서 사이클, 스토리지 등) 사이에서 동작하는 계층이다.[3] 이러한 노드 운영체제는 복수의 수행 환경들을 동시에 지원하고, 모든 액티브 노드에 공통인 기본 수준의 기능을 제공하기 위함이다. 이러한 공통 기능은 각 수행 환경이 패킷을 수신 및 전송하기 위한 하위 채널을 구현하고, 복수 환경들에 의한 전송 및 대역폭 계산과 같이 노드 자원들에 대한 엑세스를 제어하며, 라우팅과 같은 공통의 서비스를 지원하는 것을 포함한다.

노드 운영체제 인터페이스는 노드 자원들에 대한 4가지의 기본 추상을 정의한다. 스레드, 메모리, 채널의 세 가지 추상은 각각 계산, 스토리지, 통신을 위한 시스템 자원들에 대한 추상이다. 플로우는 계정, 수락 제어를 위한 기본 추상이며, 시스템 내 자원의 스케쥴링을 담당한다. CPU 사이클, 메모리, 망 대역폭과 같은 시스템 자원들은 특정 플로우에 할당된다. 플로우는 기본적으로 입력 및 출력 채널, 메모리 풀 그리고 스레드 풀을 포함한다. 입력 채널에 도착한 액티브 패킷은 해당 플로우에 할당된 스레드와 메모리를 이용하는 수행 환경에 의해 처리되고 나서 출력 채널로 전송된다.

3. 액티브 패킷

액티브 네트워크에서 액티브 패킷(또는 스마트패킷)이라 부르는 데이터 패킷은 정보의 실체이다. 이러한 패킷들은 목적지 주소, 사용자 데이터와 액티브 노드에서 지역적으로 수행되는 메소드를 포함한다. 액티브 패킷에 대한 일반적인 구조는 액티브 네트워킹 위원회에서 제안한 액티브 네트워크 캡슐화 프로토콜(ANEP)[12]을 포함하며, 이를 본 논문에서 제안한 액티브 패킷에 캡슐화한 구조는 그림 2와 같다.

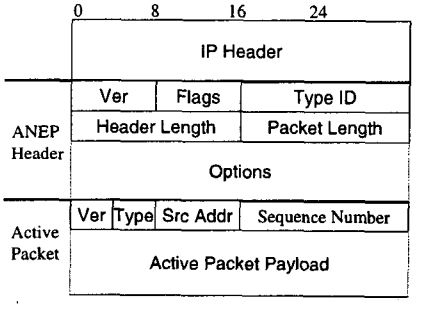

그림 2 IP 및 ANEP를 캡슐화한 액티브 패킷 구조

액티브 패킷은 공통의 액티브 패킷 헤더 내에 포함된 프로그램, 결과 데이터 또는 메시지들을 포함할 수 있으며, ANEP 내에 캡슐화된다. 액티브 패킷 헤더는 버전 번호, 타입, 소오스 주소 및 순서 번호의 4개의 필드로 구성된다. 버전 번호는 구현 언어의 업그레이드 버전이나 패킷 형식 변경을 식별하기 위해 사용된다. 타입 필드는 프로그램 패킷, 데이터 패킷, 에러 패킷 또는 메시지 패킷의 4가지 형태 중 하나를 가르킨다. 프로그램 패킷은 해당 호스트에서 수행하고자 하는 코드를 운반하며, 데이터 패킷은 수행 결과를 소오스 노드에 되돌려 주는 데이터를 포함하며, 메시지 패킷은 수행 가능 코드 이외의 정보 메시지를 전달한다. 마지막으로 에러 패킷은 프로그램 패킷의 전송이나 프로그램 수행시 예외 상황이 발생할 경우의 에러 조건을 반환한다. 소오스 주소는 액티브 패킷의 소오스 주소를 포함한다. 이 값은 각 클라이언트에 대한 ANEP 데몬에 의해 발생된다. 프로그램 패킷이 네트웍을 순회하고, 하나 이상의 응답을 발생시키기 때문에 이 값은 해당 응답이 전송되어야만 하는 클라이언트를 식별하기 위해 사용된다. 순서 번호 필드는 같은 소오스 주소로부터의 메시지들을 구별하기 위한 값을 포함한다. 이 값은 클라이언트로 하여금 응답 패킷과 삽입된 프로그램을 일치시키기 위한 것이다. 소오스 주소와 같이 응답 패킷은 프로그램 패킷의 순서 번호를 되돌려 준다.

4. 액티브 노드

액티브 노드는 액티브 패킷의 수신, 스케쥴링, 실행, 모니터링 및 전송 등의 기능을 수행한다. 본 논문에서 제안된 액티브 노드의 구조는 그림 3과 같다.

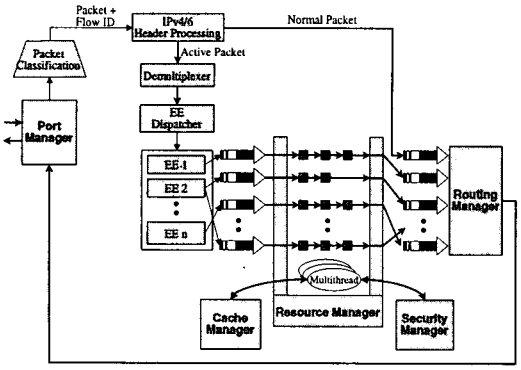

그림 3 제안한 액티브 노드 구조

4.1 포트 관리자

먼저 패킷이 액티브 노드에 도착하면, 포트 관리자가 이를 수신한다. 각 포트 관리자는 자신의 포트를 통해 수신된 액티브 패킷을 다른 액티브 노드로 전송해 주는 역할을 수행한다. 포트 관리자를 통해 수신된 패킷은 패킷 분류 과정을 통해 일반 패킷일 경우 헤더를 해석하여 라우팅 관리자를 통해 다음 노드에 전달되며, 액티브 패킷일 경우 흐름 식별자와 사용자 정의 코드가 추출된다. 흐름 식별자는 적절한 수행 환경으로 액티브 패킷을 역다중화하기 위해 사용된다. 이러한 역다중화 과정은 패킷 흐름간의 원하지 않는 상호작용과 노드 자원으로의 악의적인 접근을 방지하기 위해 각 호출에 대해 서로 다른 수행 환경으로 동작하기 위함이다.

4.2 자원 관리자

자원 관리자는 노드내에 있는 자원 즉, CPU, 캐쉬, 메모리, 전송 대역폭 등에 대한 인터페이스를 제공한다. 액티브 패킷에 있는 코드는 하나의 스레드 내에서 수행되며, 자원들에 대한 모든 요구는 해당 스레드에 의해 이루어진다. 액티브 패킷당 하나의 스레드를 가지는 것은 해당 스레드에 할당된 자원들을 관리하기 쉽기 때문에 자원 관리를 더욱 용이하게 수행할 수 있다. 또한 각 스레드는 노드 자원들의 효율적이고, 안전한 사용을 위해 캐쉬 관리자 및 보안 관리자와 긴밀하게 상호작용한다.

4.3 캐쉬 관리자

캐쉬 관리자는 자원 관리자와 상호동작하여 액티브 패킷의 수행과 관련한 상태 정보를 유지한다. 이로 인해 다음에 오는 액티브 패킷이 효율적이고 신속하게 동작할 수 있도록 도움을 준다.

4.4 보안 관리자

액티브 노드는 프로그램가능한 실체이며, 새로운 서비스나 프로토콜들이 전송되어 실행되기 때문에, 노드의 안전과 보안에 관한 중요한 문제를 야기한다. 따라서 보안 관리자는 응용 프로그램이 액티브 노드에 있는 내부 데이터 구조나 그 노드에서 수행하는 다른 액티브 패킷의 내부 데이터 구조의 무결성을 손상시키는 것으로부터 보호하기 위한 메카니즘을 제공한다.

4.5 라우팅 관리자

라우팅 관리자는 라우팅 테이블을 유지, 보수하며, 이러한 테이블들을 관리하기 위한 인터페이스를 제공한다. 그러나 라우팅 관리자는 라우팅 테이블 갱신을 위한 데이터를 전송하는 것이 아니라, 목적지에 있는 테이블 항목들을 수정하거나 추가 또는 삭제하기 위한 코드를 운반하는 액티브 패킷을 전송한다. 이를 통해 액티브 패킷은 자신 소유의 라우팅 스킴을 구현할 수 있다.

5. 결론

액티브 네트워크 기술은 전통적인 패킷 교환 망에서의 수동적인 패킷 처리를 넘어서 사용자에 의한 계산 및 수행을 가능하게 한다. 따라서 현재의 망 환경에서 다룰 수 없는 다양하고도 혁신적인 기술들을 연구할 수 있는 기반을 제공한다. 또한 새로운 프로토콜이나 서비스가 출현할 때마다 네트워크 장비에 새로운 프로토콜과 서비스를 내장하지 않고도 사용자나 서비스 제공자가 직접 삽입하므로서 하드웨어 플랫폼의 변경이 필요 없이 신속한 서비스를 가능하게 한다.

본 논문에서는 이러한 액티브 네트워크 기술의 핵심적인 요소라 할 수 있는 액티브 패킷과 액티브 노드의 구조를 제안하였으며, 현재 리눅스 운영체제 상에서 자바 언어를 이용하여 구현중에 있다.

참고 문헌

[1] D.L.Tennenhouse and D.J.Wetherall, "Towards an Active Network Architecture", Multimedia Computing and Networking, January 1996.

[2] Active Networks Working Group, "Architectural Framework for Active Networks", July 1998, http://www.cc.gatech.edu/projects/canes/arch/arch-0-9.ps.

[3] AN Node OS Working Group, "NodeOS Interface Specification", February 1999, http://www.cs.princeton.edu/nsg/papers/nodeos99.ps.

[4] A.B.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, April 1998.

[5] D.J.Wetherall, U.Legedza and J. Gutttag, "Introducing New Internet Services: Why and How", IEEE Network Magazine, July 1998.

[6] J.M.Smith, K.L.Calvert, S.L.Murphy, H.K.Orman and L.L.Perterson, "Activation Networks: A Progress Report", IEEE Computer, April 1999.

[7] B.Schwartz, A.W.Jackson and W.T.Strayer, "Smart Packets for Active Networks", OPENARCH'99, March 1999.

[8] D.Wetherall, J.Guttag and D.Tennenhouse, "ANTS:Network Services Without the Red Tape", IEEE Computer, April 1999.

[9] Sushil da Silva, "Programmig in the NetScript Toolkit", Septem-
ber 1998, http://www.cs.columbia.edu/ dasil.../netscript-0.10/doc/tutorial.html.

[10] D.S.Alexander, W.A.Arbaugh, M.Hicks, P.Kakkar, A.D.Keromytis, J.T.Moore, C.A.Guntter, S.M.Nettles and J.M.Smith, "The SwitchWare Active Network Architecture", IEEE Networks Magazine, special issue on Active and Programmable Networks, 12(3):29-36, 1998.

[11] M.HIcks, P.Kakkar, J.T.Moore, C.A.Gunter and Scott Nettles, "PLAN: A Packet Language for Active Networks", http://www.cis.upenn.edu/ switchware/papers/planet.ps.

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