프로토콜과 서비스의 동적 조합을 위한 액티브 네트워크 실행환경

이화영, 강보영, 임경식
경북대학교 컴퓨터과학과
e-mail : {bykang,hylee,kslim}@ccmc.knu.ac.kr

An Active Network Execution Environment for Dynamic Composition of Protocols and Services

Hwa Young Lee, Bo-young Kang, Kyungshik Lim
Dept. of Computer Science, Kyungpook National University

요 약

기존 액티브 네트워크의 대표적인 실행환경에는 Active Network Transfer System(ANTS)와 Active Signaling Protocol(ASP)가 있다. ANTS 실행환경은 중간 노드에서 수행될 코드를 패킷에 포함시켜 전달하는 코드 분배 방식을 통하여 프로토콜을 제공한다. 따라서 패킷으로 전달할 수 있는 코드의 크기에 한계가 있으므로 복잡한 형태의 서비스를 제공 할 수 없으며, 프로토콜 및 응용들은 완전히 분리되어 동작함으로써 상호간의 연동 및 조합이 불가능한 단점이 있다. 반면 ASP 실행환경은 프로토콜 및 응용의 상호 연동이 가능하나 시스템 컨피규레이션과 응용들 사이의 정보교환 방식이 복잡하여 새로운 서비스의 개발 및 제공이 어렵다. 본 논문에서는 이러한 기존 실행환경의 단점을 개선하고 프로토콜을 기능 및 알고리즘별로 구현한 마이크로 프로토콜을 동적으로 조합함으로써 사용자가 원하는 형태의 프로토콜 및 서비스를 제공하는 새로운 실행환경인 Customizable Architecture for Flexible Execution environment(CAFE) 실행환경을 설계 및 구현하였다. 또한 CAFE 실행환경을 기반으로 무선 웹 컨텐츠 서비스를 수행으로써 CAFE 실행환경의 실용성을 확인하였다.

1. 서론

인터넷 사용의 폭발적인 증가로 인하여 네트워크의 노드가 많아지고 복잡해지면서 네트워크를 관리하는데 소요되는 비용과 시간이 급속히 증가하고 있으며 새로운 프로토콜의 신속한 설치 요구 또한 증가하고 있다. 그러나 폐쇄적이고 유연하지 못한 현재의 네트워크 구조는 이러한 다양한 요구를 충족시키는데 한계를 가지는 문제점이 있다. 또한 프로토콜의 표준화 및 호환성의 문제는 프로토콜을 실제 네트워크에 설치하기까지 많은 시간을 요구함으로써 문제를 더욱 가속화 시킨다[1].

네트워크 노드 구조상의 폐쇄성, 경직성, 설치의 복잡성을 개선하기 위하여 현재 많은 연구가 진행되고 있다. 그 중 DARPA 에서 제안한 액티브 네트워크는 노드 구조를 개방형으로 설계하여 프로토콜의 표준화에서 개발, 설치 및 서비스 단계까지의 기간을 단축시킴으로써 네트워크 서비스 실현에 새로운 개념을 정립하였다. 액티브 네트워크의 노드 구조는 노드 운영체제, 실행환경과 액티브 응용으로 구성되어 있다. 노드 운영체제는 액티브 노드의 자원을 할당 및 보호하는 역할을 하며, 실행환경은 액티브 응용이 정상적으로 수행되기 위한 런타임 환경을 제공한다. 그리고 액티브 응용은 액티브 네트워크에서 필요한 프로토콜을 제공한다[2]. 하지만 기존의 실행환경은 프로토콜의 계층화 및 조합 그리고 응용 계층 서비스와의 연동에 있어 한계를 보이고 있다. 먼저 액티브 응용은 독립적으로 수행됨으로 계층화를 통한 프로토콜을 개발할 수 없으며, 응용계층에 액티브 네트워크를 사용하기 위한 표준화된 인터페이스를 정의하고 있지 않기 때문에

* 본 연구는 한국과학재단 목적기초연구(R01-2003-000-10562-0) 지원으로 수행되었음.

액티브 네트워크 채널로의 접근이 어렵다 [3-4].

본 논문에서는 프로토콜의 조합을 통하여 소프트웨어의 재사용성을 높이고 서비스 인터페이스를 통하여 쉽게 액티브 네트워크 채널에 접근할 수 있는 새로운 실행 환경인 Customizable Architecture for Flexible Execution environment(CAFE)를 제안한다.

2. 관련연구

2.1 ANTS 실행환경 분석

ANTS 실행환경은 Java oriented Active Operating System (JANOS)를 기반으로 동작한다. 탑재되는 응용은 캡슐 프로그래밍 모델을 바탕으로 개발되며 코드 분배 메커니즘을 통하여 액티브 노드에 프로토콜을 설치한다. ANTS 실행환경은 분산 방식의 프로토콜 설치 모델을 바탕으로 간편한 프로토콜 관리 방안을 제시하고 있으며 네트워크 정보 및 가상 네트워크 채널에 접근이 쉽다. 또한 네트워크 트래픽 폭주를 제어 할 수 있는 방법을 제공한다. 반면 인증되지 않은 액티브 응용이 네트워크에 설치될 위험성이 높고 노드에 설치할 수 있는 코드의 크기가 제한됨에 따라 복잡한 형태의 프로토콜의 개발 및 분배가 어려우며, 설치된 액티브 응용들은 완전히 분리되어 동작함으로써 상호간의 연동 및 조합이 불가능한 단점이 있다[5].

2.2 ASP 실행환경 분석

ASP 실행환경에 탑재되는 응용은 프로토콜 프로그래밍 인터페이스에서 제공하는 채널을 통하여 실행환경 및 다른 액티브 응용과 데이터를 교환할 수 있다. 보안 관리자는 접근제어정책 파일과 액티브 응용이 로딩될 때 설정된 보안등급을 기반으로 액티브 응용의 동작을 제한한다. ASP 실행환경은 가상 네트워크 채널이 하나의 기존 프로토콜 계층에 종속적이지 않으므로 다양한 계층을 기반으로 확장할 수 있다. 또한 액티브 응용에서 사용하는 코드의 공유를 통하여 필요한 메모리 양을 줄일 수 있다. 하지만 시스템 컨피규레이션이 복잡하여 접근제어정책이 경직된 구조로 작성되어 확장이 불가능하고 설치된 액티브 응용들 사이의 정보교환 방식이 복잡하여 새로운 프로토콜의 개발이 어렵다[6].

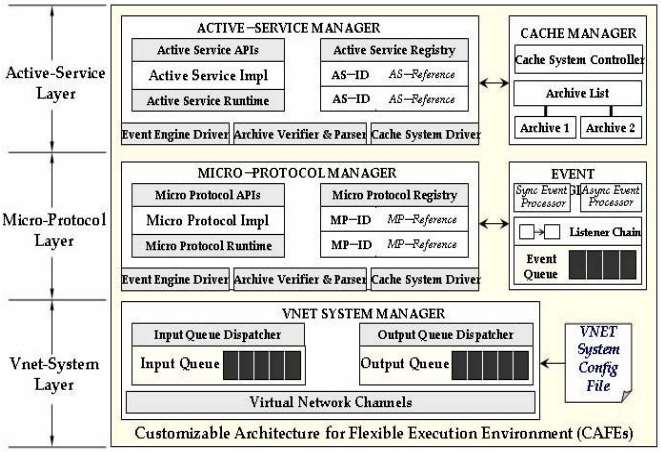
2.3 CAFE 실행환경 요구사항 분석

기존 실행환경의 분석을 통하여 도출한 요구사항은 크게 네 가지로 요약할 수 있다. 첫째, 실행환경은 런타임에 다양한 액티브 응용이 필요에 따라 설치, 삭제, 및 실행되는 동적인 환경이므로 사용하기 쉬워야 한다. 그리고 액티브 응용의 개발부터 탑재 및 실행에 이르기까지 단순한 절차와 인터페이스가 요구된다. 둘째, 실행환경에 설치되어 있는 액티브 응용을 조합 및 참조함으로써 새로운 서비스를 제공할 수 있는 유연한 구조를 지녀야 한다. 셋째, 플랫폼 독립적인 실행환경을 개발함으로써 높은 확장성을 제공해야 한다. 마지막으로 인증, 접근제어 및 암호화를 통하여 액티브 응용의 탑재, 실행 및 관리의 안정성을 확보할 수 있어야 하며 시스템 자원을 보호할 수 있어야 한다.

3. CAFE 실행환경 설계

3.1 CAFE 구조

CAFE 의 실행환경은 분리방식으로 설계되었다. 실행환경을 설계하는 대표적인 방식으로는 통합 방식과 분리 방식이 있다. 통합 방식은 중간 노드에서 수행될 코드를 패킷에 포함시켜 전달하는 방식으로 실행환경은 이 코드를 실행시켜 패킷을 처리한다. 분리방식은 액티브 노드의 로컬 영역에 패킷의 처리에 필요한 코드가 이미 설치되어 있고 각각의 패킷은 중간 노드에서 자신을 처리할 수 있는 코드의 식별자를 전달하는 방식이다. 통합 방식은 패킷과 함께 전달할 수 있는 코드의 크기에 한계가 있으므로 복잡한 형태의 서비스를 제공 할 수 없다. 반면 분리방식은 코드의 크기에 영향을 받지 않고 패킷을 구성하는 방법이 간단하여 다양한 서비스를 제공할 수 있으므로 설치된 프로토콜 및 서비스를 재사용하여 새로운 네트워크 서비스를 제공하기 위한 CAFE 실행환경에 적합한 방식이다.



(그림 1) CAFE 실행환경 구조

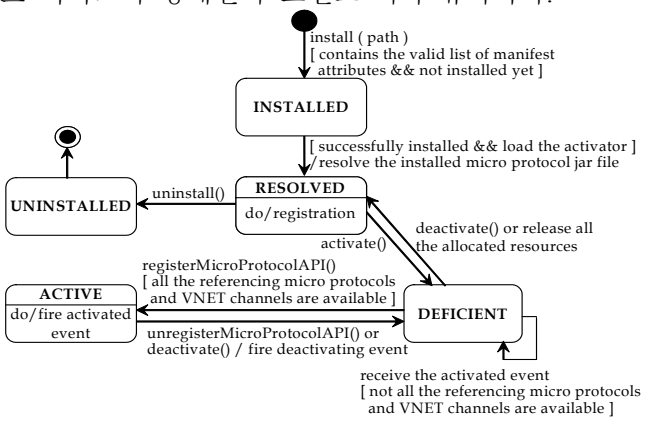
(그림 1)은 CAFE 실행환경의 구조를 나타낸 것으로 VNET 시스템 계층, 마이크로 프로토콜 계층 그리고 액티브 서비스 계층으로 구분된다. **VNET 시스템 계층**은 다양한 네트워크 프로토콜을 기반으로 가상 네트워크 채널을 구성 및 관리하고 시스템 공통 입·출력 큐와 연계하여 액티브 프레임의 송·수신 및 네트워크 정보를 제공하는 역할을 담당한다. **마이크로 프로토콜 계층**은 마이크로 프로토콜을 등록 및 해지하고 등록된 마이크로 프로토콜들을 조합하기 위한 참조자 와 API 를 관리함으로써 마이크로 프로토콜의 조합을 바탕으로 새로운 마이크로 프로토콜을 액티브 서비스 계층에 제공하기 위한 기반을 마련한다. 마지막으로 **액티브 서비스 계층**은 기존의 응용과 액티브 네트워크 사이의 연동을 위한 계층으로서 응용에서 마이크로 프로토콜 계층에 쉽게 접근할 수 있는 통로 역할을 수행한다. 또한 응용 개발에 필요한 라이브러리의 관리를 담당한다. 그 외 실행 환경을 구성하는 컴포넌트는 액티브 서비스와 마이크로 프로토콜을 파일 시스템에 설치 및 삭제하는 **캐쉬 관리자**와 실행환경에서 정의한 다양한 이벤트를 처리하는 **이벤트 엔진**이 존재한다.

3.2 프로토콜 및 서비스 모델

전통적인 프로토콜 개발방법은 비슷한 역할을 수행하는 프로토콜의 그룹화를 기반으로 계층화하고 각각의 프로토콜은 계층간의 인터페이스를 통하여 연동함으로써 서로 다른 시스템간의 통신을 원활히 하여 복잡한 프로토콜의 개발과정을 단순화시킨다. 하지만 이러한 방법은 특정 계층에 해당하는 프로토콜을 개발하기 위해 필요한 모든 기능을 하위계층에서 완벽하게 지원하지 못하고 모든 프로토콜이 하나의 완성된 소프트웨어 패키지로 구성되어 소프트웨어의 재사용성을 떨어뜨리는 문제점을 가지고 있다. 그러므로 기존의 프로토콜을 기능별로 분할하여 개발한 마이크로 프로토콜과 라이브러리를 설치하고 응용과의 연동에 필요한 액티브 서비스를 조합 및 참조함으로써 새로운 프로토콜 및 서비스를 개발할 수 있는 모델이 필요하다.

3.2.1 마이크로 프로토콜 및 액티브 서비스 상태전이 모델

CAFE 실행환경은 런타임에 새로운 마이크로 프로토콜과 액티브 서비스가 설치, 실행 및 삭제되는 동적인 환경을 지향한다. 이러한 환경에서 조합 및 참조를 통하여 새로운 프로토콜과 서비스를 제공하기 위해서는 마이크로 프로토콜과 액티브 서비스가 내외부적인 수행관련 요소를 바탕으로 자신의 수행상태를 결정하기 위한 상태전이 모델이 필요하다. (그림 2)는 마이크로 프로토콜의 상태전이 모델을 기술한 것이며 액티브 서비스의 상태전이 모델도 이와 유사하다.



(그림 2) 마이크로 프로토콜 상태 전이도

3.2.2 상태전이 감지 메커니즘

가상 네트워크 채널, 마이크로 프로토콜 그리고 액티브 서비스는 자신의 내부적인 수행환경과 외부적인 의존 요소를 바탕으로 자신의 상태가 전이된다. 그러므로 마이크로 프로토콜은 자신이 참조하는 가상 네트워크 채널 그리고 조합하고 있는 다른 마이크로 프로토콜의 상태 변화를 동적으로 감지할 수 있어야 한다. 액티브 서비스는 참조하는 마이크로 프로토콜과 액티브 서비스에 대한 상태 변화를 알아낼 수 있어야 한다. 마이크로 프로토콜은 가상 네트워크 채널과 다른 마이크로 프로토콜의 상태변화 이벤트를 수신하기 위하여 VnetChannelListener 와 MicroProtocolListener 를 등록한다. 특히 조합하고자 하는 마이크로 프로토콜의 상태변화를 감지할 경우 필터를 사용하여 원하는 마이크로 프로토콜에 대한 리스너(listener)를 등록할 수 있다. 그리고 가상 네트워크 채널 역시 실행환경에 존재하는 모든 채널의 상태 변화를 수신하는 리스너를 등록할 수 있다. 그 후 시스템 내에서 리스너에서 감지중인 이벤트가 발생할 경우 이벤트 엔진에 의하여 리스너의 콜백 메소드가 호출된다. 리스너를 등록한 마이크로 프로토콜은 메소드의 인자로 전달된 이벤트를 분석하여 참조하는 마이크로 프로토콜 또는 가상 네트워크 채널의 상태변화에 대처한다.

4. CAFE 실행환경 구현

CAFE 실행환경의 구현부는 가상 네트워크, 마이크로 프로토콜 및 액티브 서비스 각각을 구성하고 관리하기 위한 **VNET 시스템 관리자**, **마이크로 프로토콜 관리자** 그리고 **액티브 서비스 관리자**로 구성된다.

4.1 VNET 시스템 관리자

VNET 시스템 관리자는 하부 네트워크에 대한 논리적인 인터페이스를 제공하는 가상 네트워크 채널을 구성하고 관리한다. 또한 동시 다발적인 액티브 패킷의 송·수신을 조정하기 위한 시스템 공통 입·출력 큐를 관리함으로써 마이크로 프로토콜 계층과 가상 네트워크 채널 사이에 액티브 패킷의 송·수신에 대한 중재자의 역할을 수행한다.

4.1.1 가상 네트워크 채널 컨피규레이션

VNET 시스템 관리자는 가상 네트워크 채널 컨피규레이션을 위한 XML 스키마를 정의하고 이것을 바탕으로 기술된 프로파일을 분석하여 액티브 네트워크 채널을 구성한다. XML 스키마는 정형화된 형태로 기술되어 특정 프로토콜 계층에 종속적인 채널을 생성하는 것이 아니라 다양한 계층을 기반으로 오버레이 네트워크를 구축할 수 있으며 XML 의 확장기법을 사용하여 쉽게 확장이 가능하다.

4.2 마이크로 프로토콜 및 액티브 서비스 관리자

마이크로 프로토콜 및 액티브 서비스 관리자는 CAFE 실행환경에서 기존의 프로토콜을 세분화하여 모델링한 마이크로 프로토콜 그리고 응용과 프로토콜을 연결하기 위한 액티브 서비스를 관리한다. 각각의 마이크로 프로토콜과 액티브 서비스가 정상적으로 동작하기 위한 독립적인 런타임 환경을 지원하고 설치된 마이크로 프로토콜과 액티브 서비스의 조합 및 참조를 통하여 새로운 마이크로 프로토콜과 액티브 서비스를 개발하기 위한 방법을 제공한다.

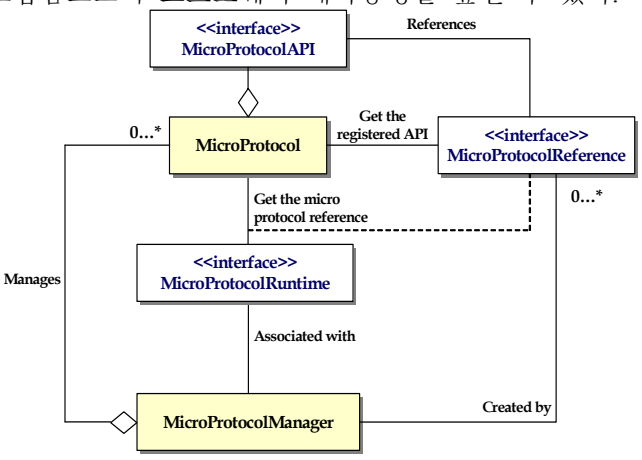
4.2.1 마이크로 프로토콜 및 액티브 서비스 런타임 환경

CAFE 실행환경에 탑재된 마이크로 프로토콜은 런타임 인터페이스를 통하여 관리자와 정보를 교환하며 수행에 필요한 자원을 확보한다. 즉, 런타임 인터페이스는 마이크로 프로토콜과 관리자 사이의 정보교환의

중재자 역할을 수행함과 동시에 정상적인 실행에 필요한 환경적인 여건을 제공함으로써 각각의 마이크로 프로토콜들이 서로 방해를 받지 않고 독립적으로 수행될 수 있도록 한다. 런타임 인터페이스는 마이크로 프로토콜과 관리자 사이에 존재하여 실제 CAFE 실행환경 내부 구현 부분을 마이크로 프로토콜로부터 격리시킴으로써 소프트웨어의 의존성을 낮추고 정의하지 않은 다른 방법을 통한 시스템 접근을 봉쇄한다. 또한 서로 다른 마이크로 프로토콜은 각자의 런타임 인터페이스를 가지게 되므로 조합과 참조를 위하여 등록한 MicroProt-ocolAPI 이외의 다른 방법을 통하여 현재 수행중인 마이크로 프로토콜에 접근할 수 없다. 액티브 서비스 런타임 환경 또한 마이크로 프로토콜의 런타임 환경과 동일한 기능을 한다.

4.2.2 마이크로 프로토콜 조합

프로토콜 조합은 기존의 패키지화된 프로토콜과는 달리 프로토콜의 기능 또는 알고리즘별로 개발하여 실행환경에 탑재한 마이크로 프로토콜을 바탕으로 마이크로 프로토콜을 재조립함으로써 프로토콜의 개발기간을 단축시키고 응용계층에서 원하는 네트워크 서비스를 제공하기 위한 방법이다. 이러한 프로토콜 개발 방법론은 마이크로 프로토콜이 제공하는 API 를 인접 계층의 존재와 상관없이 자유롭게 호출함으로써 응용이 원하는 형태의 프로토콜 계층화가 가능하며, 응용에서 요구하는 네트워크 서비스를 기능 또는 알고리즘에 따라 개발된 마이크로 프로토콜을 동적으로 조합함으로써 소프트웨어 재사용성을 높일 수 있다.



(그림 3) 마이크로 프로토콜 조합

(그림 3) 에서 마이크로 프로토콜에 해당하는 Micro-Protocol 클래스는 다른 마이크로 프로토콜과의 조합을 지원하기 위한 MicroProtocolReference 및 MicroProtocolAPI 인터페이스를 가진다. 그리고 인터페이스에 대한 실제 구현부는 감추어져 있다. 참조자는 해당 마이크로 프로토콜의 API 를 레지스트리(registry)에 등록하여 통합적으로 관리하며 식별자를 통하여 레지스트리를 검색할 수 있다. 즉, 특정 마이크로 프로토콜은 자신의 네트워크 서비스를 구성하는데 필요한 다른 마이크로 프로토콜을 식별자를 이용하여 관리자로부터 얻고 이것을 바탕으로 MicroProtocolAPI 에 접근하여 응용에서 요구하는 네트워크 서비스를 구성한다.

5. 결론 및 향후 연구

본 논문에서는 액티브 네트워크 개념을 도입하여 기존의 프로토콜을 기능별로 분할한 마이크로 프로토콜을 동적으로 조합함으로써 사용자가 원하는 형태의 새로운 프로토콜을 구성하고, 액티브 서비스 상호 참조를 통하여 새로운 네트워크 서비스를 신속히 적용할 수 있는 CAFE 실행환경을 설명하였다. CAFE 실행환경의 가상 네트워크 채널은 다양한 프로토콜 계층을 기반으로 동작함으로써 실행환경의 네트워크 이질성을 극복하였다. 이를 바탕으로 프로토콜 및 서비스 모델을 설계하여 새로운 프로토콜 및 서비스의 개발에 대한 용이성을 확보하였고, 가상 네트워크 채널, 마이크고 프로토콜 및 액티브 서비스에서 발생한 이벤트를 대신 처리해주는 이벤트 위임(delegation) 모델을 통하여 실행환경에 탑재되어 동작하는 마이크로 프로토콜 및 액티브 서비스의 상호 연동방안을 정립하였다. 또한 개발된 CAFE 실행환경을 기반으로 무선 환경에서 서비스 제공자의 웹 서버에 단말 환경에 적합한 컨텐츠를 유지하고 있지 않더라도 단말에 최적화된 컨텐츠를 제공하여 주는 무선 웹 컨텐츠 변환 시스템의 실험을 하였다. 이 실험을 통하여 단말에 최적화된 무선 인터넷 서비스를 제공하기 위한 환경으로서 CAFE 실행환경이 적합함을 확인할 수 있었다.

향후에는 실행환경에 존재하는 다양한 마이크로 프로토콜과 액티브 서비스들 사이의 조합 및 참조에 대한 접근제어 기술에 대한 연구가 이루어져야 한다. 또한 동적인 마이크로 프로토콜 및 액티브 서비스 설치 메커니즘과 실행환경이 동작하는 액티브 노드의 원격 관리 방법에 대한 연구가 필요하다.

참고문헌

[1] Stephen F. Bush, Amit B. Kulkarni, Active Networks and Active Network Management, Kluwer Academic/Plenum Publishers, ISBN 0306465604, 2001

[2] Tennenhouse, D.L., Wetherall, D.J, “Towards an Active Network Architecture,” Computer Communication Review, Vol. 26, No. 2, April 1996

[3] S. Bhattacharjee, K. Calvert, Y. Chae, S. Merugu, M. Sanders, E. Zegura, “CANEs: an execution environment for composable services,” Proceedings of DARPA Active Networks Conference and Exposition(DANCE'02), pp. 255-272, May 2002

[4] Sriram Ramabhadran, Joseph Pasquale, “A Framework for Application-Specific Customization of Network Services,” Proceedings of Data Compression Conference (DCC'97), pp. 456, March 1997

[5] D. Wetherall, J. Guttag, D. Tennenhouse, “ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols,” Open Architectures and Network Programming(IEEE OPENARCH'98), pp. 117-129, April 1998

[6] Robert Barden, Bob Lindell, Steven Berson, “The ASP EE: An Active Network Execution Environment,” Proceedings of DARPA Active Networks Conference and Exposition(DANCE'02), pp. 238-254, May 2002