능동네트워크 상의 능동센서 언어 설계 및 인터프리터 구현

양윤심[†] · 배철성[††] · 정민수[†††] · 이영석[††††]

요 약

네트워크에 대한 요구는 점차적으로 복잡해지고 그 수가 증가하고 있는 상태이다. 이를 극복하기 위해 현존하는 네트워크 노드의 구조를 프로그래밍이 가능하도록 하는 능동 네트워크 기술이 등장하게 되었다. 이것은 사용자 요구 기능을 수행할 수 있는 프로그램 코드를 전송 및 실행함으로써 통신망에 새로운 서비스를 신속하고 경제적으로 도입하여 망 자원들을 보다 적절하게 활용할 수 있도록 하는 기술 분야이다. 본 논문에서는 능동 네트워크 상에서 능동센서의 기능 및 동작 방식을 기술하기 위한 프로그래밍 언어를 제안하고 이러한 언어를 기반으로 능동센서를 용이하게 다룰 수 있는 능동센서 인터프리터를 설계 및 구현한다.

The Design of Active Sensor Language on Active Network and Implementation of Its Interpreter

Yoon-Sim Yang[†], Cheol-Sung Bea[††], Min-Soo Jung[†††] and Young-Seok Lee[††††]

ABSTRACT

Request for network becomes complicated gradually and network traffic is increasing. To overcome this situation, the structure of network node should be changed to accept new service quickly and economically by executing program code in node itself. Active Network's research can use net resources more properly because of executing program within node itself. In this paper, we design a programming language, namely ASL, for Active Sensor to describe function and behavior of active sensor. We also design and implement the interpreter for proposed ASL.

Key words: 자바가상기계, 능동 네트워크, 능동센서, 이동코드

1. 서 론

능동네트워킹에 대한 개념은 네트워킹 시스템의 향후 방향에 대해 1994년과 1995년에 DARPA 연구 협의체에서 논의되면서 시작되었다. 인터넷이 급격하게 확산되고 인터넷을 이용하는 사용자가 증가하면서 네트워크에 대한 요구는 점차 다양화되고 복잡해지고 있다. 현재의 네트워크 시스템은 신기술이나 이와 관련된 표준을 망에 적용하기까지 많은 시일과 비용이 소요되고 경로 설정이나 흐름 제어와 같은 처리를 종단의 단말에서만 처리한다. 따라서 네트워크의 기능을 분산시키고 사용자의 망에 대한 요구 기능을 시기 적절하게 반영하며 네트워크에 유연성을 제공하기 위해 능동 네트워크(Active Network)라는 새로운 네트워크 패러다임이 등장하였다.

능동 네트워크는 네트워크 노드의 구조를 프로그래밍이 가능하도록 구성한다. 이것은 네트워크 노드가 사용자 요구 기능을 수행할 수 있는 프로그램 코드를 전송 및 실행할 수 있게 하는 기술이다. 이 분야는 통신망에 새로운 서비스를 신속하고 경제적으로 도입하여 자원들을 보다 적절하게 활용할 수 있도록

본 연구는 경남대학교 학술 논문게재 연구비 지원으로 이루어졌음
접수일 : 2003년 3월 5일, 완료일 : 2003년 5월 9일
[†] 준회원, 경남대학교 컴퓨터공학과 박사과정
[††] 경남대학교 컴퓨터공학과 박사과정
[†††] 종신회원, 경남대학교 정보통신공학부 교수
[††††] ETRI 정보보호기술본부 책임 연구원

하는 데 목표를 두고 연구되고 있는 분야이다[1,2].

네트워크의 노드들에 대해 프로그래밍이 가능하다는 것은 능동 네트워크가 가지는 가장 대표적인 특징이다. 능동 네트워크 노드는 데이터뿐만 아니라 사용자가 원하는 프로그램을 패킷을 통해 전송하여 실행하거나 중간에 있는 능동 네트워크 노드에 미리 설치된 프로그램 중에서 해당기능을 실행함으로써 사용자가 원하는 네트워크 기능을 이용하게 된다. 이것은 능동 네트워크 노드(이하 능동노드)에서 애플리케이션 프로그램이라는 새로운 프로토콜을 네트워크에 주입할 수 있게 하는 기술을 제공함으로써 변화하는 요구 사항들에 대해 네트워크가 유연하게 받아들일 수 있도록 하는 데 사용될 수 있다. 중간에 있는 능동노드에 전송되는 패킷은 이동코드(mobile code)의 특성을 갖는다. 이동코드는 상이한 네트워크로 이동이 가능하고 목적지에 도착하면 자동적으로 실행되는 소프트웨어로 정의한다. 능동 네트워크 프로그래밍 언어는 실행 가능한 이동코드를 기술하는 것이다[3,4].

최근 제안된 능동 네트워크 언어들 중에서 자바 기반의 ANTS(Active Network Transfer System)는 자바가 가지는 여러 가지 특징으로 표준화가 가장 유력하다. 본 논문은 ANTS를 효과적으로 사용할 수 있도록 새로운 언어를 제안하고 제안한 언어를 위한 인터프리터를 구현하였다. 그리고 다른 능동네트워크 언어들과 비교하였다.

본 논문의 구성은 다음과 같다. 2장에서는 능동 네트워크 구조와 특징을 기술한다. 3장에서는 능동 센서 프로그래밍 언어인 ASL을 설계하고 4장에서는 ASL 인터프리터를 구현한다. 마지막으로 5장과 6장은 성능평가 및 결론을 기술한다.

2. 관련 연구

능동 네트워크를 구현하는 방법은 프로그램 코드의 전송 여부에 따라 분리(Discrete)방식과 통합(Integrated)방식으로 분류할 수 있다. 분리방식은 능동 네트워크에 사용되는 프로그램이 중간에 있는 능동노드에 미리 설치되어 있고 사용자가 설치된 프로그램 중 어느 하나를 실행하기를 원할 경우 해당 기능을 실행하는 방법이다. 펜실바니아 대학의 Switchware와 콜럼비아 대학의 Netscript 등이 분리방식의 대표적인 예이다. 이 방법은 실행 기능이 복잡하고 프로그램의 크기가 크기 때문에 프로그램 자체를 전송하여 사용자의 요구에 시기 적절하게 반영하기 어려울 경우에 유리하다.

통합방식은 필요한 데이터와 프로그램 자체를 캡슐이라 불리는 능동 패킷을 통해 능동노드에 전달하여 실행하는 방법이다. 이 경우 능동 네트워크 상에서 전송되는 모든 패킷은 능동노드에서 실행 가능한 프로그램으로 볼 수 있으며 프로그램은 네트워크 상의 어느 사용자에 의해서도 네트워크에 전송이 가능하다. MIT 공대의 ANTS, 펜실바니아 대학의 PLANet 등이 통합방식의 대표적인 예이다[5-7].

본 장에서는 능동네트워크 주요 특징과 구조를 살펴보고 새로운 프로토콜 개발을 목적으로 하는 MIT의 ANTS 능동노드 애플리케이션 프로그래밍 모델을 제시한다.

2.1 능동 네트워크

능동 네트워크는 기존의 수동 네트워크(Passive Network)가 가지는 프로그램 언어, 성능, 보안, 안전성에 대한 새로운 네트워크 구조이다. 서비스 특성이나 사용자의 요구에 따라 적절한 연산을 통해 패킷을 처리하는 차세대 네트워크라고 표현할 수 있다. 기존의 네트워크가 저장-전송의 개념이라면 능동 네트워크는 저장-연산-전송이다. 단순히 전송 기능만을 가지고 있는 기존의 노드(라우터 등)들로 구성된 네트워크와는 다르게 네트워크 전체를 하나의 거대한 컴퓨터로 생각하여, 각 노드가 프로그램을 처리할 수 있는 능력을 가진 네트워크라 할 수 있다. 능동 네트워크의 패킷은 프로그램과 데이터를 같이 포함하여 전송되며 이러한 패킷을 캡슐(capsule)이라 부른다[8,9].

기존 네트워크 노드는 IP패킷의 경로설정과 전송을 담당하는 OSI 계층 3의 네트워크 층까지만 처리한다. 그림1과 같이 능동노드는 이러한 기존 네트워크 노드에 어플리케이션 계층까지 처리하는 네트워크로 사용자의 프로그램을 노드에 실행시켜 네트워크를 프로그램 할 수 있다.

애플리케이션 계층까지 처리한다는 것은 네트워크 노드에서 경로설정과 같은 단순한 기능에서 벗어나 네트워크 종단간에서만 이루어지던 에러 처리 및 흐름제어와 같은 복잡한 기능과 사용자가 원하는 기

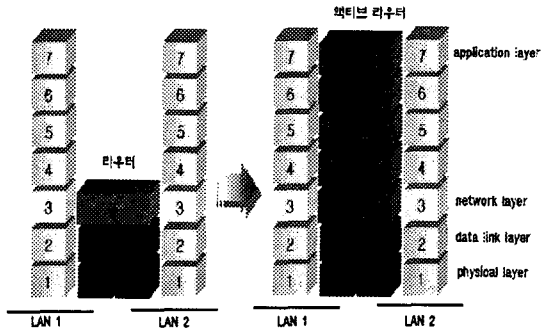

그림 1. 노드와 능동센서 OSI 계층

능을 수행할 수 있음을 의미한다. 그러므로 능동 네트워크는 사용자나 네트워크 망 자체에 유연성뿐만 아니라 여러 가지 많은 장점들을 제공할 수 있게 된다.

예를 들어 인터넷경매에서 매수주문은 여러 개 있지만 유효한 것은 최고가뿐이므로 다른 주문은 서버에 보낼 필요가 없다. 이에 따라 중간에 있는 능동노드로 필터링을 하게 되면 서버에 모든 주문이 집중되는 현상을 피할 수 있다. 또한 멀티캐스트 방식에서 중간의 능동노드가 전송패킷을 캐시하여 각 단말에 보낸다면 망 자원의 낭비를 막고 서버 집중현상 또한 피할 수 있다.

2.2 능동노드

능동네트워크 기능을 수행하기 위한 능동노드 구조는 일반적으로 그림2와 같이 노드운영체제, 실행환경, 애플리케이션으로 구성되어 있다. 노드운영체제(NodeOS)는 다중 실행환경을 지원하고 실행환경과 노드의 자원사이에서 패킷 처리시 요구되는 자원에 대한 중재역할을 담당한다. 실행환경은 사용자 제

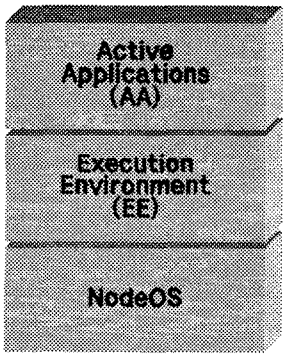

그림 2. 능동노드 구조

공 파라미터나 안전한 프로그래밍 언어를 위한 인터프리터 구현이 포함되고 인증된 관리자에게 제어 인터페이스를 제공한다.

능동 네트워크의 가장 중요한 특징이라 할 수 있는 애플리케이션은 캡슐 내에 들어갈 네트워크 서비스인 프로토콜 정의들을 구현한 것으로 능동 네트워크는 능동노드에 사용자 프로그램을 주입할 수 있는 애플리케이션 기반 구조를 제공한다[10-12].

2.3 ANTS

현재 연구되고 있는 능동네트워크 기술 중 하나인 ANTS는 MIT 대학에서 연구 개발한 것으로 능동노드의 실행환경과 네트워크 애플리케이션 프로그래밍 모델을 제공해 주는 툴킷이다. ANTS 툴킷은 사용자가 ANTS 환경에서 제공하는 수 많은 API들을 이용하여 새로운 프로토콜 형태로 다양한 애플리케이션을 쉽게 구현할 수 있도록 해주는 도구이다.

프로토콜 개발자의 편의를 위해 이동코드를 위한 애플리케이션의 체계적인 프로그래밍 단계가 정의되어 있어 ANTS 프로그래밍 단계대로 코딩하면 사용자가 원하는 애플리케이션을 용이하게 작성할 수 있다. 이것은 능동노드의 사용자 특성화를 가능하게 하여 애플리케이션 특정 서비스가 네트워크에 다운로드 될 수 있게 해준다.

ANTS의 모든 구현은 이동코드기술을 이용하여 프로토콜 정의들을 네트워크 상에서 전송한다. 프로그래밍 언어로는 자바를 사용하고 런타임 환경으로는 자바가상기계를 기반으로 하는 대표적인 통합방식이다[13-15].

2.4 ANGLE

ANGLE(Active Networks for GLobal Environments)은 국내 ICU(Information & Communications Univ.)에서 연구되고 있는 능동 네트워크 상의 캡슐 내에 운반되는 이동코드 언어이다. ANGLE 이동코드 언어는 스크립트 형태로 미리 라이브러리로 구현된 네트워크 서비스를 호출하여 사용하도록 설계되었다. 프로그램 구조는 그림 3과 같이 *entry*부, *bind*부, *func*부로 나누어진다.

entry부는 기술된 함수 중에서 실행하고자 하는 함수명을 지정하고, bind부는 프로그램을 실행하기 위해 필요한 값을 기술한다. 마지막으로 func부는 프

```
entry: <function name>

bind:
    <name 1>: <value 1>:
    <name 2>: <value 2>:
              :     :
              :     :
    <name n>: <value n>:
endbind

func <function name>:
    <statement 1>:
    <statement 2>:
              :
              :
    <statement n>:
endfunc
```

그림 3. ANGLE 프로그램 구조

로그램은 실행 가능한 단위인 함수로 구성되며 이를 정의한다.

func 부분의 <statement>는 프로그램 구문을 의미하며 라이브러리 함수를 호출하는 구문과 상황에 따른 구문을 실행하는 부분, 그리고 예약된 명령어를 처리하는 구문으로 이루어진다. 만약 ANGLE 언어 구문 중에서 이미 정의된 기능을 수행하는 예약된 명령어 함수들과 라이브러리 함수들 이 외의 기능을 사용하고자 한다면 요구하는 기능에 해당하는 함수를 시스템에 맞게 직접 구현하여 실행 가능하게 해야 한다[16].

3. ASL 설계

본 장에서는 2장에서 제시했던 프로토콜 개발용 ANTS 환경에 새로운 프로토콜을 용이하게 적용시키기 위하여 ASL(Active Sensor Language) 능동센서용 패킷을 설계하였다.

ASL은 능동 네트워크 상의 이동코드에 대한 적재 및 실행을 위해 스크립트 형태로 노드들 사이에 운반되는 일종의 패킷 언어이다. 능동 네트워크에서 패킷은 새로운 프로토콜을 정의한 능동노드의 능동센서(애플리케이션 프로그램) 수행을 요구한다. 능동노드에서 패킷이 요구하는 애플리케이션을 수행하려면 수신 받은 패킷을 분석하여 원하는 애플리케이션을 호출하여야 한다.

3.1 ASL 처리과정

ASL 패킷에는 설치와 실행을 위한 두 가지의 구문으로 분류된다. 능동노드에 수신된 패킷은 패킷분류기를 통해 능동노드에서 수행될 수 있는지의 여부를 판별하여 능동노드에서 수행될 패킷이면 ASL 인터프리터를 호출한다.

이러한 ASL 캡슐 분석을 통해 설치 패킷이면 패킷내의 명시된 URL로부터 애플리케이션을 설치한다. 그리고 수신된 패킷이 실행 패킷이면 자신의 능동노드에 있는 애플리케이션을 실행한다. 그림4는 이러한 처리 과정을 보여 주고 있다.

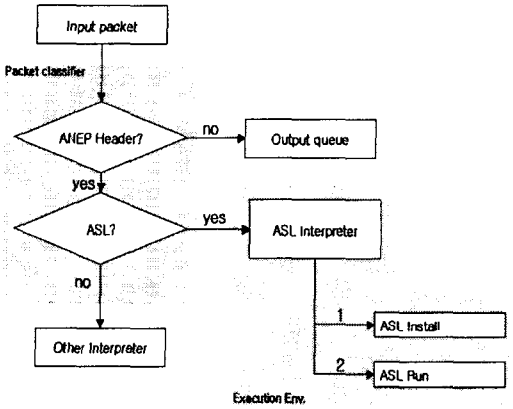

그림 4. ASL 패킷 처리 흐름도

3.2 ASL 해시테이블

능동노드는 수신된 패킷을 분석하여 실행할 애플리케이션의 존재 유무를 확인한다. 해당 애플리케이션이 능동노드에 존재하면 애플리케이션을 호출하여 실행시킨다. 애플리케이션의 존재 유무를 확인하기 위해 각각의 능동노드는 능동센서 관리 테이블을 사용한다. 그림5는 능동센서 관리 테이블의 구성을 보여준다.

Id	클래스 명	버전

필드명	개요 및 설명
Id	자바 클래스 고유번호 (0 - 65535)
클래스 명	현재 노드에 저장된 자바 클래스 이름
버전	현재 노드에 저장된 자바 클래스의 버전 (미사용)

그림 5. 해시테이블 구성도

3.3 ASL 패킷 구조

ANTS 능동 패킷은 크게 IP 헤더 부분, AN 헤더 부분 및 Payload로 구성되어 있다. ASL 패킷은 Payload 부분을 다시 분할하여 그림 6과 같이 OP code 부, Application Number부 및 Optional로 구성된다.

*OP code*는 이 패킷이 수행하는 명령어 코드를 가지고 있다. 여기에 사용되는 코드는 능동센서 설치를 위한 '1' 또는 능동센서 실행을 위한 '2' 가운데 하나의 값을 가진다. OP code를 위한 길이는 1바이트이다. *App.Num*은 능동노드에서 실행되거나 능동노드에 설치할 능동센서의 번호를 가지며, 길이는 2바이트이다. *optional*은 능동노드에서 실행되는 능동센서 수행에 필요한 파라미터들을 기술하는 부분이다. 만약 OP code가 '1' 값은 갖는 설치 명령어라면 이 부분에는 능동노드에 설치할 능동센서가 저장되어 있는 위치정보(URL)가 기술된다. ASL의 패킷은 패킷 구조 설명에서 언급했듯이 OP code에 따라 설치(install)와 실행(run)을 위한 두 가지로 분류된다.

첫째, ASL의 설치 패킷은 지정된 서버로부터 애플리케이션을 설치한다. *OP code*에는 능동센서 설치를 의미하는 '1'을 기입하고 *App.Num*에는 애플리케이션 번호를 기입한다. *optional* 부분에는 능동노드에 설치할 능동센서가 저장되어 있는 URL이 기록된다. 능동센서는 애플리케이션 프로그램으로 자바 클래스 파일 형태로 존재한다. 만약 그림 7와 같이 설정된 패킷이 현재 능동노드에 도달하게 되면 ASL 패킷의 *optional* 부분에 명시된 URL로부터 해당 능동센서를 다운로드하여 자신의 능동노드에 설치하게 되며, 능동센서 관리 테이블에 능동센서 번호(3)와 능동센서 이름을 저장한다. 그림 7은 ASL 설치 패킷 구조를 나타낸다.

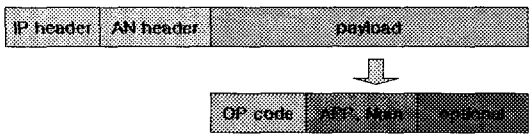

그림 6. ASL 패킷 구조

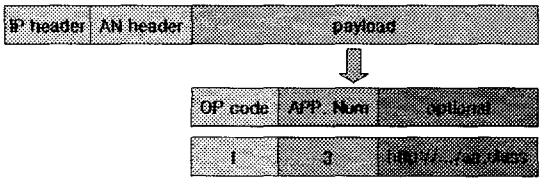

그림 7. 설치 명령문 패킷 예

둘째, ASL의 실행 패킷은 능동노드에 저장된 특정 애플리케이션을 실행하도록 한다. *OP code*는 능동센서 실행을 의미하는 '2'를 기입하고 *App.Num*에는 능동노드에서 실행하고자 하는 애플리케이션 번호가 기록된다. 이 번호는 자신의 능동센서 관리 테이블로부터 참조된다. 인터프리터는 이 애플리케이션 번호를 이용하여 능동센서 관리 테이블에서 해당 애플리케이션을 찾아 애플리케이션을 수행시킨다. 인터프리터에 대한 자세한 기능은 ASL 인터프리터 구현에서 언급될 것이다.

optional 부분에는 애플리케이션 실행에 필요한 파라미터들이 기록되며 각각의 값들은 세미콜론(;)으로 분리되어 있다. 만약 그림 10과 같이 설정된 패킷이 능동노드에 도착하면 인터프리터는 능동센서 관리 테이블에서 능동센서 번호(3)에 대한 애플리케이션을 찾아서 수행한다. 그러나 해당 애플리케이션이 존재하지 않으면 디폴트 서버에서 해당 능동노드로 다운로드하여 실행한다. 그림8은 ASL 실행 패킷 구조를 나타낸다.

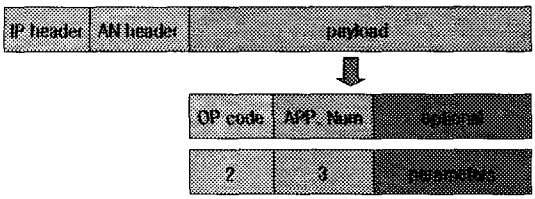

그림 8. 실행 명령문 패킷 예

3.4 ASL의 BNF

그림 9는 BNF 표기법을 이용하여 ASL을 표현한 것을 나타낸다.

```
< asl_ program >     :   < run_statement >          |
                         < install_statement >      |
                         < case_statement >
< run_statement >    :   RUN  < appNum >  < argList >
< install_statement >:   INSTALL  < appNum > { Url }
< case_statement >   :   IF <cond> then < run_statement >   |
                         IF <cond> then < install_statement >
< appNum >           :   0| .. |255
< argList >          :   < argument >+
< argument >         :   < string_value >
```

그림 9. ASL의 BNF

4. ASL 인터프리터 구현

본 연구에서 구현된 ASL 능동센서 언어와 ASL 인터프리터는 StrongARM프로세서에 탑재된 리눅스 운영체제상에 ANTS를 탑재하여 노드운영체제의 일부분으로 이식하였다. 그림 10은 ASL시스템의 전반적인 구조를 나타내고 있다.

ASL 인터프리터는 능동노드가 동작을 시작하게 되면 캡슐의 ASL 스크립트를 해석하여 필요로 하는 애플리케이션을 설치하거나 실행시킨다. ASL 인터프리터는 새로운 능동센서 언어를 설계하고 구현하기보다는 기존의 발전된 모델인 ANTS를 수용하면서 능동센서를 자유롭게 이동 및 실행시키는 기능을 추가하였다.

따라서 ANTS가 제공하는 API기반 ANTS 애플리케이션의 무리 없는 수행을 위해 구현된 ASL 인터프리터는 자바로 구현되었고 그림 11과 같이 ANTS 환경의 노드운영체제에 이식된다. 즉, ASL 인터프리터는 ANTS의 노드운영체제의 한 모듈로서 동작하며 그 주된 임무는 수신된 패킷을 분류하고 필요한 기능(능동센서 설치 또는 실행)을 담당한다.

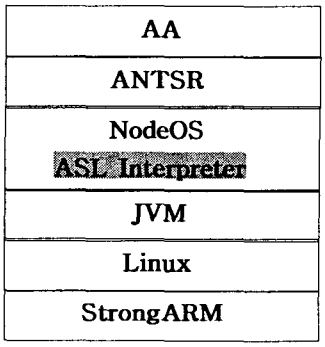

그림 10. ASL 플랫폼

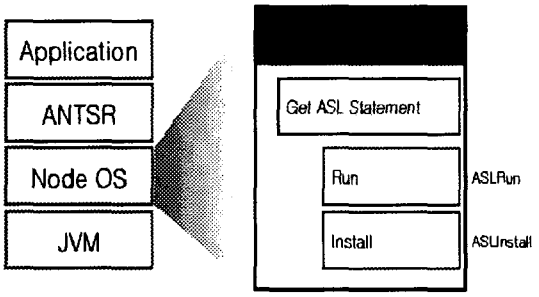

그림 11. ANTS 환경에 ASL 인터프리터 구현

자바로 구현된 ASL 인터프리터는 수신된 패킷을 해석하여 패킷에 따라 능동센서를 설치하거나 실행한다. 인터프리터의 원활한 기능 수행을 위해 ASLMain, ASLInstall, ASLRun 컴포넌트로 구성되어 있다.

ASLMain은 수신된 패킷을 해석하여 설치(Install) 또는 실행(Run) 클래스를 호출하는 ASL 메인 프로그램이다. ASLInstall은 능동센서를 노드에 적재 후, 능동센서 관리 테이블에 정보를 저장하는 ASL 설치 프로그램이고 ASLRun은 해당 능동센서를 찾아서 실행하는 ASL 실행 프로그램이다. 만약 센서가 없으면 디폴트 서버로부터 설치한 후에 실행한다.

4.1 ASLMain

ASLMain 컴포넌트는 능동노드에 수신된 패킷의 명령어 코드부가 '설치' 명령어이면 능동센서의 설치를 위한 ASLInstall 클래스를 호출한다. 명령어 코드부가 '실행' 명령어이면 능동센서의 실행을 위한 ASLRun 클래스를 호출한다. ASLinstall과 ASLRun 클래스는 쓰레드 모델 기반으로 구현하였으며 서로 독립적으로 실행한다. 그림 12는 ASLMain 컴포넌트인 ASLInterpreter.java 소스 코드의 주요코드이다.

ASLMain 컴포넌트의 *ASLInstall()* 메소드는 능동센서를 설치하기 위해 ASLInstall 클래스를 호출한다. 이때 능동센서의 설치에 필요한 정보는 클래스 번호(능동센서 번호)와 능동센서가 위치하는 위치정보(URL)이다. 그리고 독립적인 작업이 가능하도록 쓰레드 모델로 구현되어 있어 쓰레드로 호출한다.

ASLRun() 메소드는 현재 능동노드에 설치된 능동센서를 실행하기 위하여 ASLRun 클래스를 호출한다. 이때 능동센서 실행에 필요한 정보는 클래스 번호(센서 번호)와 해당 능동센서가 동작하는데 필

```
class ASLInterpreter {
private Class invoke:class;
  public static void main(String args[]) throws Exception {
     ...
   if (opcode == 1) {  //Call Install Program
     Thread InstallThread = new Thread(new ASLInstall(classid,parameter));
     InstallThread.start();
    } else if (opcode == 2) { //Call Run Program
     Thread RunThread = new Thread(new ASLRun(classid,parameter));
     RunThread.start();
    }
   ...
  }
}
```

그림 12. ASLMain 컴포넌트의 주요코드

요한 파라미터들이 전달된다. ASLInstall과 마찬가지로 쓰레드 모델로 구현되어 있어 쓰레드로 호출한다.

4.2 ASLInstall

ASLInstall 컴포넌트는 능동노드의 수신된 캡슐에 명시된 특정 능동센서를 URL로부터 다운로드하여 현재 능동노드에 설치한다. 능동센서의 적재가 완료되면 능동노드의 능동센서 관리 테이블에 적재한 능동센서의 정보를 기록한다.

그림 13은 ASLInstall 컴포넌트가 수행하는 처리과정을 그림으로 나타내고 있다.

먼저, 송신측 능동노드로부터 설치 패킷을 수신하면 설치 프로그램은 인터프리터 프로그램으로부터 설치할 능동센서의 위치정보(URL)와 설치할 능동센서 정보(능동센서번호)를 입력받는다. 그리고 나서 설치 프로그램은 명기된 URL로부터 능동센서를 자신의 능동노드에 설치한다. 능동센서를 설치하고 나면, 설치된 능동센서에 대한 정보를 자신의 능동센서 관리 테이블에 추가한다. 그림14는 ASLInstall 컴포넌트의 주요코드 이다.

ASLInstall 컴포넌트의 *DownLoad()* 메소드는 URL에 명시된 송신측 능동노드로부터 URLConnection 객체를 이용하여 접속한 후, 현재 능동노드에 능동센서를 저장한다. *RegistClass()* 메소드는 송신측 능동노드로부터 적재한 능동센서정보를 능동센서 관리 테이블에 저장한다. 능동센서 관리 테이블에 저장된 정보는 현재 능동센서에 설치되어 있는 능동센서 리스트를 보관하고 있다.

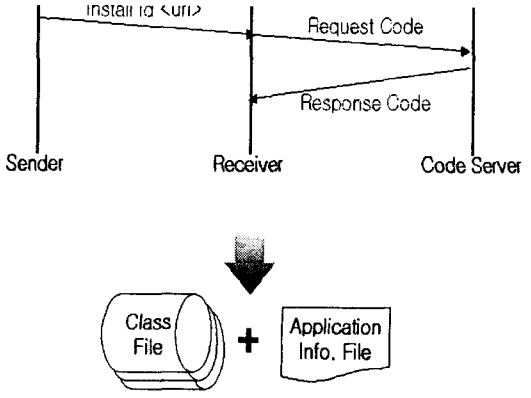

그림 13. ASLInstall 컴포넌트의 능동센서 설치과정

```
class ASLInstall implements Runnable{
    private void DownLoad(String strURL)throws Exception  {
        //URL로부터 능동센서를 설치
        try {
            URL u = new URL(strURL);
            ...
            int int_Doc_Length = Document_Length;
        if (Document_Length > 0){
                InputStream inStrm = uConn.getInputStream(); //InputStream 생성
                while(((input = inStrm.read()) != -1) && (--int_Doc_Length >= 0)){
                    TargetFile.write(input);  }
            }
    }
    private void RegistClass(String CTable,int AppNum,String CName){
        //설치된 센서정보를 센서 관리 테이블에 저장
        try {
            RandomAccessFile randf = new RandomAccessFile(CTable, "rw");
            long lBytes = randf.length();
            randf.seek(lBytes);
            randf.writeBytes(Record);
            randf.close();
          } catch (Exception e) {}
    }
}
```

그림 14. ASLInstall 컴포넌트의 주요코드

4.3 ASLRun

그림 15는 ASLRun 컴포넌트의 처리 과정을 나타내고 있다. ASLRun은 ASLMain으로부터 현재 능동노드에서 실행시킬 능동센서 정보를 수신 받는다. 그리고 현재 능동노드의 능동센서 관리 테이블에서 실행할 능동센서정보를 검색하여 능동센서를 실행시킨다. 이때 해당 능동센서 정보가 현재 능동노드에 존재하지 않으면 디폴트 코드 서버로부터 능동센서를 적재한 후 실행한다.

ASLRun 컴포넌트의 주요코드는 그림16과 같다. *ASLRun()* 메소드는 ASLRun 클래스 수행에 필요한 초기값을 설정하는 생성자 메소드이고, *FindAp-*

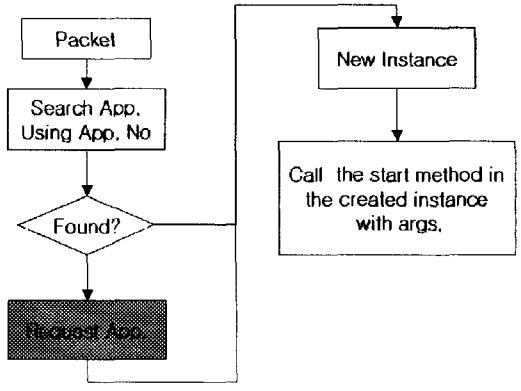

그림 15. ASLRun 컴포넌트의 처리과정

plication() 메소드는 파라미터로 전달된 능동센서가 현재 능동노드에 존재하는지를 확인한다. 이 작업은 능동센서 관리 테이블에 존재하는 리스트를 검색하는 것이 아니라 실행하고자 하는 능동센서의 물리적 존재유무를 검사한다. 이러한 처리를 위해 자바의 Reflection API를 이용하여 구현하였다.

GetCommand() 메소드는 인라인 명령어를 처리하는 메소드로서 ASLRun 클래스의 *run()* 메소드로부터 능동센서를 현재 능동노드에 적재하는 명령어와 적재된 능동센서의 시작 메소드를 호출하는 명령어 두개를 접수하여 처리한다. *run()* 메소드는 현재 능동노드에 설치된 능동센서를 실행하기 위한 ASLRun 쓰레드의 시작 메소드이다. 먼저 실행할 능동센서를 능동센서 관리 테이블에서 검색하고, 능동센서가 존재하지 않으면 다시 디폴트 코드 서버에서 능동센서를 적재할 수 있도록 ASLRemote 클래스를 호출한다. 만일 실행할 능동센서가 존재한다면, 능동센서의 시작 메소드를 호출함으로써 능동센서를 실행하게 된다.

ASLRun은 자신의 능동노드에 존재하는 능동센서 뿐만 아니라 현재 존재하지 않는 능동센서에 대한 실행 요청도 기꺼이 수행할 수 있어야 한다. 따라서 ASLRun은 수신된 패킷이 현재의 능동노드에 존재하지 않는 능동센서를 수행하도록 요청하게되면 디폴트 코드서버에서 해당 능동센서를 다운로드한 후, 능동센서정보를 추가하고 적재한 능동센서를 실행할 수 있도록 ASL 원격 설치 프로그램인 ASLRemoteInstall 클래스를 포함한다.

```
class ASLRun implements Runnable {
  ASLRun(int Cid, String para) //파라메타를 초기화
  public static void FindApplication(String FName, int AppNum, String AppVer)
        // 실행할 클래스가있는지 확인
  public void GetCommand(String line)
  public void run() {
    FindApplication("ClassInfo",ClassId,Version);
    if (ClassName.equals("")) {
       Thread RT = new Thread(new ASLRemoteInstall(ClassId));
       RT.start(); }
    LoadAndExecute Le = new LoadAndExecute();
    Le.GetCommand("new "+ClassName);
    Le.GetCommand(CallStatement);
  }
}
```

그림 16. ASLRun 컴포넌트의 주요코드

4.4 ASL 인터프리터 수행 결과

그림 17은 ASL 인터프리터의 실행결과를 간단한 예제를 통해 보여주고 있다. 인터프리터의 수행을 위해 가상의 패킷을 임의로 생성하여 사용하였고 각 능동노드에서 수행되는 능동센서는 자바 언어로 구현한 프로그램을 사용하였다. ASL 인터프리터는 수신된 캡슐을 분석하여 OP code가 1이라는 설치캡슐이면, 설치할 애플리케이션을 기재된 URL로 가서 다운받아 설치한다. 그리고 OP code가 2인 실행캡슐이 들어오면 파라메타를 입력받아서 지정된 애플리케이션을 실행하는 화면이다.

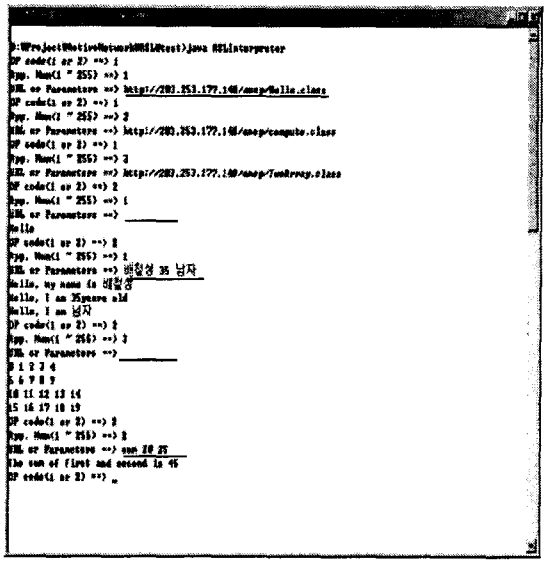

그림 17. ASL 인터프리터 실행화면

5. 성능 평가

능동 네트워크를 위한 프로그래밍 언어는 스트롱 타입, 가비지 콜렉션, 모듈, 동적 로딩, 플랫폼 독립적인 표현(Platform independent representation), 쓰레드(Thread), 정적 타입 등을 제공해야 한다. 현재 국제적으로 연구진행중인 능동 네트워크 언어들은 일반적으로 C/C++, ML, JAVA 계열로 나누어진다.

자바는 플랫폼에 대해 독립적이고 어떤 플랫폼이든지 실행이 가능한 이식성이 강한 언어이다. 또한 완전한 객체 지향 언어이며 프로그램의 동적인 수행 및 동시에 많은 일을 하는 프로그램을 작성할 수 있도록 멀티쓰레드 프로그래밍을 지원한다. C와 같은

언어는 약한 타입 언어이기 때문에 임의의 메모리 영역 참조에 대해 부적합하다. 이는 자바와 같이 강한 타입 언어를 사용하여 해결할 수 있다. 이러한 능동 네트워크의 언어적 요구조건들을 비교해 본 결과 가장 근접한 언어로서 자바를 ASL 능동센서 언어로 선정하였다[17,18].

표 1은 현재 능동 네트워크의 대표적인 언어들에 대하여 주요 기능별 지원여부 차이를 나타내고있다.

ANGLE시스템은 JVM기반으로써 자바로 작성된 이동코드를 실행시킨다. 그러나 자바 기반 실행환경에서만 이동코드를 수행시킬 수 있고 그 외의 실행환경에서는 이동코드를 수행할 수가 없다.

또한 ANGLE 언어는 이동코드를 위한 애플리케이션을 프로그래밍 함에 있어 비교적 간단한 프로그래밍 기법을 제시하고 있다. 그러나 프로그래밍 기법이 간단한 만큼 복잡한 수행을 요구되는 패킷에 대한 처리를 하기 위해서 기존의 자바에서 제공해 주는 API만으로 복잡한 애플리케이션을 작성하기가 어렵다.

ANGLE시스템은 별도로 제공되는 API가 충분치 않기 때문에 직접 코딩으로 구현하여야 한다. 그렇게 된다면 프로그래밍 함에 있어 점점 복잡해질 수밖에 없다.

반면에 ASL시스템은 ANTS의 노드운영체제에 두 가지가 존재하여 순수 자바로 작성된 이동코드와 자바 이 외의 C나 C++로 작성된 이동코드도 실행시킬 수 있다. 즉, 자바 기반 실행환경 이외의 실행환경에서도 사용될 수 있다.

ASL 이동코드 프로그래밍 기법의 중요한 구성요소인 ANTS Toolkit은 개발자의 편의를 위해 이동코드를 위한 애플리케이션의 체계적인 프로그래밍 단계가 정의되어 있다. ANTS 프로그래밍 단계대로 ANTS 환경에서 제공하는 별도의 수많은 API들을 이용하여 코딩하면 복잡하더라도 사용자가 원하는 애플리케이션을 용이하게 작성할 수 있다. 이것은 능동노드의 사용자 특성화를 가능하게 하여 애플리케이션 특정 서비스가 네트워크에 다운로드 될 수 있게 해준다.

ANTS는 애플리케이션 구현의 용이함과 다양한 애플리케이션 개발 가능성이 돋보이며 새로운 프로토콜의 수행을 위해 자바가상기계를 포함하고 있다. 모든 구현이 자바언어로 되어있기 때문에 능동 네트워크의 언어적 요구조건을 만족하고 있다.

본 논문에서 제안한 ASL시스템은 ANTS에서 제공해 주는 별도의 수많은 API를 이용하여 애플리케이션 구현을 용이하게 함으로써 ANGLE 언어보다도 더욱 효율적인 프로그래밍 기법을 제시하고 있다.

표 2는 ASL시스템과 ANGLE시스템에 대한 차이를 도표로 나타내고있다.

6. 결 론

능동 네트워크는 네트워크에 대해서 사용자가 필요로 하는 기능이나 망 기능의 향상을 위해 네트워크 노드에 프로그램의 실행이 가능한 실행환경을 구축한다. 이것은 사용자에게 보다 유연하고 다양한 네트워크 수준의 기능을 제공하는데 기여할 수 있다. 그리고 네트워크 관리나 새로운 서비스 제공 측면에서도 능동적이고 유연하게 대처할 수 있어 네트워크의 고성능화를 앞당길 수 있을 것이다.

능동 네트워크에 대한 국내 연구는 아직 큰 진전이 없을 뿐만 아니라 활발히 연구가 추진되지 못하고 있다. 국제적으로는 능동 네트워크의 주요 애플리케이션 분야, 핵심 기술, 접근방법 등에 대해 점차 공감대를 이루어 가는 과정에 있으며 학계와 연구기관을 중심으로 표준화를 추진해 나가고 있다.

표 1. 능동 네트워크의 연구 비교

요구조건 / 기반언어	Dynamic Binding	Down-loadable	Platform Independence	Safety	Security	Speed	Memory	Extendable
Switch-ware	○	○	○	○			○	
NetScript	○	○	○	○			○	△
Smart-Packets		○		○		○		
CANE				○		○		
ANGLE	○	○	○	○	○		○	△
ASL(+ANTS)	○	○	○	○	○		○	○

표 2. ASL시스템과 ANGLE시스템과의 비교

	ANGLE	ASL
가상기계	자바 가상기계	자바 가상기계
사용언어	자바 언어	자바 언어
이동코드 처리	자바로 작성된 이동코드 처리	자바와 C/C++로 로 작성된 이동코드 처리
응용 프로그램 확장성	자바의 기본 API를 제공 간단한 서비스의 구현은 용이하다. 그러나 복잡한 서비스의 구현은 기본 API만으로는 불충분하여 필요 기능에 대한 직접 구현이 필요.	자바의 기본 API와 ANTS API를 제공 복잡한 서비스라도 직접 구현할 필요없이 제공되는 API를 간단한 호출로 작성이 용이
보안기능	자바의 보안 메커니즘을 사용	자바의 보안 메커니즘을 기본으로 하고 패킷의 무결성 검증하기위해 MD5 message digest를 사용하여 데이터 보안성에 있어 더 많은 확신을 제공.
패킷 언어	패킷내에 이동코드 처리에 대한 프로그램을 작성	패킷내에 이동코드 처리에 대한 OP code, 능동센서 번호, 능동센서 위치 등을 기술.

본 논문에서 설계한 ASL 능동센서 언어와 구현한 ASL 인터프리터는 새로운 프로토콜을 신속히 디자인하고 전개할 수 있게 해주는 방법론 개발에 도움을 줄 것으로 기대한다. 그리고 장기적인 연구를 통해 능동센서 프로그래밍 언어에 관한 학문적인 기여와 능동 네트워크 관련한 국내 기술 수준 향상을 위한 자료로서 활용이 가능하다.

참 고 문 헌

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

[2] DARAPA, "Composable Services for Active Network", AN Composable Services Working Group, May 1998.

[3] David J. Wetherall, "Service Introduction in an Active Network", Dissertation of MIT, 1999.

[4] D. L. Tennenhouse, S.J. Garland, L. Shrira and M.F. Kaashoek, "From Internet to ActiveNet", Request for Comments, January 1996.

[5] A. Jackson and C. Partridge, "Smart Packets", http://www.net-tech.bbn.com/smtpkts/.

[6] E. Zegura and K. Calvert, "CANES: composable active network elements", http://www.cc.gatech.edu/projects/canes.

[7] M. Hicks et al, "PLANnet: An Active Internetwork", In Conf. on Computer Communications (INFOCOM 99), IEEE, pp. 1124-1133, New York, NY, Mar. 199

[8] Soo-hyung Lee, Jungchan Na "주간기술동향 (Trend of technology week)", ETRI(Electronics and Telecommunications Research Institute), 2001.

[9] Yoon-sim Yang, Jun-young Jun, "The study on programming language for Active Network node", Korea Multimedia Society Vol. 5, pp. 2-4, 2002.

[10] Jonathan M. Smith, Kenneth L. Calvert, Sandra L. Murphy, Hilarie K. Orman, and Larry L. Peterson, "Activating networks: A progress report", IEEE Computer, Vol. 32, No. 4, pp. 32-41, April 1999.

[11] David J. Wetherall, Ulana Legedza and John Guttag, "Introducing New Internet Services: Why and How", IEEE Network Magazine, July/August 1998.

[12] D. Wetherall et al, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", In 1st Conf. on Open Architectures and Network Programming (OPENARCH 98), pp. 117-129, IEEE San Fran-cisco, CA, Apr 1998.

[13] Ulana Legedza, David J. Wetherall, and John

Guttag, "Improving The Performance of Distributed Applications Using Active Networks", IEEE INFOCOM'98, 1998.

[14] C.A. Thekkath, T.D. Nguyen, Evelyn Moy and E.D. Lazowska, "Implementing Network Protocols at User Level", ACM Sigcomm, 1993.

[15] David Murphy, John Guttag, and David L. Tennenhouse, "Building an Active Node on the Internet", Technical Report MIT-LCS-TR-723. Master of Engineering thesis, May 1997.

[16] ANGLE(Active Network for GLobal Environment) http://cnlab.icu.ac.kr/active.html

[17] Gosling, J. and H. McGilton, "The Java Language Environment" (White Paper), Sun Microsystems, 1995.

[18] Gosling, J. "Java Intermediate Bytecodes", SIGPLAN Workshop on Intermediate Representations (IR95), San Francisco, CA, 1995.

[19] M. El-Darieby, A. Bieszezad, Intelligent Mobile Agents:Towards Network Fault Management Automation , In Proc. of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, pp. 611-622, May 1999.

[20] S. Bhattacharjee et al, "Active Networking and the End-to-End Argument", In Intl. Conf. on Network Protocols(ICNP 97) IEEE Atlanta, GA, pp, 220-228, Oct 1997.

[21] Ulana Legedza and John Guttag, "Using Network Level Support to Improve Cache Routing", 3rd International Web Caching Workshop, June 1998.

[22] N. C. Hutchinson and L. L. Peterson, "The x-Kernel: An Architecture for Implementing Network Protocols", IEEE Trans on Software Engineering, Vol. 17, No. 1, pp. 64-76, Jan. 1991.

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

양 윤 심

2001년 8월 경남대학교 컴퓨터공학과 학사
2003년 2월 경남대학교 컴퓨터공학과 전공 공학석사
2003년 3월~현재 경남대학교 컴퓨터공학과 홈네트워크 전공 박사과정
관심분야 : JavaMachine, HomeNetworking, Active-Network

배 철 성

1993년 창원대학교 자연과학대학 전자계산학과 학사
1993년 2월~1996년 1월 신용협동조합중앙회 전산부
1996년 2월~1999년 6월 한국중공업 정보시스템실
2000년 창원대학교 대학원 전자계산학과 공학석사
2002년 3월~현재 경남대학교 컴퓨터공학과 홈네트워크 전공 박사과정
관심분야 : Embedded System, JavaMachine

정 민 수

1986년 서울대학교 컴퓨터공학과 학사
1988년 한국과학기술원 전산학과 공학석사
1994년 한국과학기술원 전산학과 공학박사
1990년~현재 경남대학교 컴퓨터공학과 부교수
관심분야 : Embedded System, JavaMachine, Compiler

이 영 석

1992년 충남대학교 컴퓨터공학과 학사
1994년 충남대학교 컴퓨터공학과 석사
1994년~1997년 LG정보통신(주) 중앙연구소 연구원
2002년 충남대학교 컴퓨터공학과 박사
2002년~현재 한국전자통신연구원 선임연구원
관심분야 : 가상사설망, 네트워크 보안, 이동컴퓨팅, 분산시스템

교 신 저 자

양 윤 심 631-701 경남 마산시 월영동 449 경남대학교 컴퓨터공학과