액티브 네트워크에서의 SNMP 기반 망 관리 구조

이 병 기[†] · 조 국 현[††]

요 약

기존의 망 관리 구조는 정적이며, 중앙집중적인 클라이언트/서버 기술들을 토대로 한다. 이러한 망 관리 구조에서 망의 모든 요소들은 전체 데이타를 처리하고, 망 관리자에 대한 인터페이스를 제공하는 중앙의 관리 스테이션에 모든 데이타를 전송한다. 이러한 이유로 인해, 중앙집중적인 관리 구조는 관리되는 노드들의 수와 복잡도가 증가함에 따라 한계에 부딪히고 있다. 또한 이러한 수동적인 망 관리 방법은 확장이 불가능하고, 트래픽 증가로 인해 많은 비용이 들기 때문에 더욱 신속한 접근 및 확장성을 가진 기술을 적용하는 것이 필수적이다.

액티브 네트워크는 사용자가 메시지 내에 포함된 프로그램을 망에 삽입할 수 있도록 함으로서 사용자에 의한 계산 및 관리를 수행할 수 있도록 하는 프레임웍 이다. 본 논문에서는 망 노드들에 부담을 주지 않고 망 관리 및 모니터링을 수행하기 위해 액티브 네트워크 기술을 적용한다. 또한 액티브 네트워크 플랫폼의 구조 설계 및 프로토타입 시스템을 구현하고, 액티브 네트워크와 SNMP 기반 관리 응용간의 통합을 제공하기 위해 사용되는 프레임웍에 대해 기술한다.

An Architecture of SNMP-based Network Management on the Active Network

Byeong-Ki Lee[†] · Kuk-Hyun Cho[††]

ABSTRACT

Traditional network management architecture have been based on static and centralized client/server technologies. In this paradigm, every element of the network sends all the data to a central management station that processes the whole data and provides the interface to the user operator. By this reason, centralized management architecture has been stretched to its limits by the number and complexity of managed nodes. The passive network management solution does not scale and is not cost effective due to traffic increase. Therefore it is essential that network management employ techniques with more rapid access and scalability.

Active Networks is a framework within which users inject programs contained in messages into a network capable of performing computations and manipulations on behalf of the user. In this paper we apply the active networks technology to network management and monitoring without placing undue burden on the nodes in the network. Further, we present a architecture and a prototype implementation of the Active Network platform and describe the framework that we used to provide for integration between Active Network and SNMP-based management applications.

1. 서 론

기존의 중앙집중적인 망 관리 시스템에서는 물리적으로 분산된 관리 대상 에이전트들로부터 데이터를 수집하고, 분석하는 작업이 중앙의 관리자 시스템에 집중됨으로서 관리자 시스템에서는 병목 현상이 발생하고, 중복된 많은 관리 데이터로 인해 망 트래픽이 증가하게 된다. 관리되는 에이전트 또한 수동적이며, 다양한 판매자에 의해 제공됨으로서 관리 기능을 수정하

† 준 회 원 : 광운대학교 컴퓨터과학과 박사과정수료
†† 정 회 원 : 광운대학교 전자계산학과 교수
논문접수 : 2000년 6월 25일, 심사완료 : 2000년 7월 31일

거나 확장하기가 상당히 복잡하다. 또한 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.

액티브 네트워크(Active Network)는 이러한 문제점을 해결하고자 시도하고 있다. 액티브 네트워크란 기존 망에서의 라우터나 스위치와 같은 중간 노드들이 단순히 패킷의 헤더만을 처리하는 것에 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].

망 관리는 새로운 도전에 직면해 있는 새로운 기능들의 효율적인 사용을 통해 효과적으로 관리 기능을 수행할 수 있을 것이다. 따라서 본 논문에서는 기본적으로 오늘날 가장 널리 사용되고 있는 SNMP(Simple Network Management Protocol) 기반의 망 관리 구조에 새로운 액티브 네트워크 기술을 통합함으로서 망 관리 작업을 효율적으로 수행할 수 있도록 하는 데에 초점을 둔다.

액티브 네트워크에서의 주요 성분은 패킷 내에 새로운 응용이나 프로토콜에 대한 코드와 데이터를 캡슐화시켜 전송하는 액티브 패킷(Active Packet)과 이러한 액티브 패킷을 수신하여 해석하고 수행하는 액티브 노드(Active Node)로 이루어진다. 본 논문에서 제안하는 망 관리를 위한 액티브 관리 패킷 구조는 액티브 네트워크 위원회에서 제안한 액티브 네트워크 캡슐화 프로토콜(Active Network Encapsulation Protocol : ANEP)을 기반으로 한다[2, 4].

본 논문에서는 이러한 액티브 네트워크의 각 노드에서 SNMP 기반의 관리 기능을 수행할 수 있는 액티브 관리 패킷(Active Management Packet : AMP)과 이러한 AMP를 손쉽게 생성할 수 있도록 하는 액티브 관리 패킷 생성기(Active Management Packet Generator : AMPG)에 대해 기술하며, 망 환경 변화에 신속하게 대응할 수 있도록 하는 액티브 관리자 시스템에 대해서 기술한다. 또한 액티브 관리자에 의해 구성된 액티브 관리 패킷이 자신의 기능을 효율적으로 수행할 수 있도록 액티브 관리 패킷의 수신, 스케쥴링, 실행, 모니터링 및 전송 등의 다양한 기능을 처리하는 액티브 노드 관리 엔진을 설계하고, 구현한다. 또한 이렇게 설계된 액티브 네트워크 관리 플랫폼을 토대로 효율적인 구성 관리 및 장애 관리 응용을 설계한다.

본 논문에서 기술하는 액티브 관리 에이전트의 프로토타입 시스템은 오늘날의 IP 망에서 쉽게 적용될 수 있는 표준(Java, SNMP, ANEP over UDP)들을 적용한다[4, 23, 24]. 액티브 관리 패킷은 사용자의 요구에 따라 망 관리 및 모니터링 프로그램과 다양한 파라메타들이 자바 클래스 형태로 ANEP 내에 캡슐화되며, 또한 액티브 관리 에이전트의 각 구성 요소들은 자바 언어를 이용한 클래스들의 집합으로 구성된다. 자바를 이용하여 액티브 관리 시스템의 기반 구조를 구현하는 이유는 자바 가상 머쉰 환경이 시스템에 의존하지 않는 프로그램들을 지원하며, 또한 안전한 코드 분산을 위해 필요한 호환성 및 보안과 같은 기본적인 기능들을 제공하기 때문이다. 본 논문에서는 2장에서 액티브 네트워크의 일반적인 구조 및 구성 요소들에 대해 기술하고, 3장에서 액티브 관리 시스템의 설계 요구사항 및 관리 시스템의 구조에 대해 기술하며, 4장에서 관리 에이전트의 구성 요소 및 프로토타입 시스템에 대해 기술한다. 그리고 5장에서는 액티브 관리 시스템의 구현 및 관리 응용에 대해 기술하며, 마지막 6장에서는 결론과 향후 연구 방향에 대해 살펴본다.

1.1 관련 연구

관리 기능을 분산시킴으로서 관리 시스템의 확장성 및 유연성을 향상시키기 위한 다양한 방법들이 제안되어졌다. 1991년, Yemini와 Goldszmidt는 그들의 MbD 모델에서 이러한 개념을 기술하였다[13]. 이 모델에서, 관리 기능은 호출되기 전에 동적으로 설정될 수 있다. 이러한 개념을 실현하는 다양한 관리 기능들에 대한 여러가지 접근 방법들이 존재한다[14].

새로운 관리 프레임웍을 위해 자주 명명되는 기본 기술은 이동 에이전트이다. 통신 시스템을 관리하는데 있어서 이러한 이동 에이전트 기술을 이용하는 작업들이 최근에 활발히 연구되고 있다[15-17]. 이러한 노력들은 이동 에이전트가 분산 관리 시스템을 구성하기

위해 사용될 수 있음을 보여준다. 특히, 상호 협력하는 시스템들을 위해, 이러한 프레임웍은 바람직한 것 같다. 이동 에이전트의 장점은 소프트웨어 공학과 관련해 여러가지 프로젝트에 의해 보고되어지고 있다. 여기에서는 망의 각 구성 요소들이 명시된 작업을 수행하거나 특정한 목적을 달성하기 위한 에이전트들로서 설계되는 경우 복잡한 관리 시스템을 더욱 구조화된 방법으로 구현될 수 있음을 보여준다. 이동 에이전트를 기반으로 분산 관리를 위해 제안된 프레임웍들은 기존의 망 구성요소들이 상호간에 동작하는 방법과는 다르다. 기본적인 접근 방법은 관리 객체를 포함해 처음부터 전체 관리 시스템을 구성하는 것으로 시작하며[18], 기존 망 요소들과의 상호동작을 포함하는 또 다른 프레임웍은 이동 에이전트 런타임 환경에서 SNMP 프로토콜 스택을 제공한다[19].

최근에는 또한 이동 에이전트의 기본 기술을 액티브 네트워크와 같은 새로운 망 구조에 적용하는 연구가 활발히 진행되고 있다. 액티브 네트워크를 기반으로 하는 여러가지 관리 프레임웍이 제안되어지고 있다. 그들 중 몇 가지는 기본 기술로서 이동 에이전트를 사용[20]하지만, 그들 중 대부분은 전용 소프트웨어[21]나 하드웨어 기술[22]을 사용한다. 액티브 네트워크에서의 망 관리와 관련한 연구로서 관리 응용들을 위한 소규모의 단일 패킷 스크립트들을 사용하는 방법은 BBN의 스마트 패킷 프로젝트에서 찾을 수 있다[10]. 그러나 여기에서 구현된 방법은 사용가능한 최대 프로그램 크기가 매우 작으며, 지속적인 상태 유지가 어려운 단점이 있다.

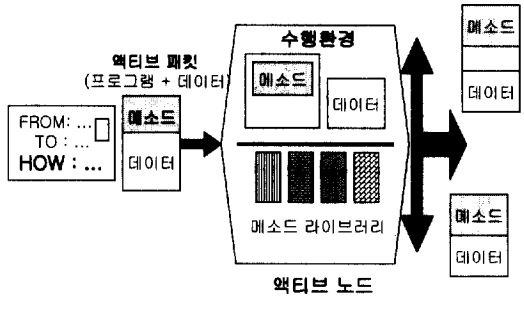
통신 시스템의 발달 그리고 인공지능, 소프트웨어 공학 기술들의 발달로 인해 망 관리 분야에서도 더욱 새롭고 향상된 많은 관리 방법들이 제시되고 있다. 기존의 중앙집중적인 망 관리 방법의 단점들을 해결하기 위해 제시되고 있는 이러한 분산 망 관리에 대한 연구들은 앞서 기술한 다양한 형태로 진행되고 있으며, 어떠한 접근 방법이 더 유용하고, 앞으로의 망 관리 분야에서 궁극적으로 선택되어질 것인가를 예측하기란 현재로선 상당히 어렵다. 따라서 본 논문에서도 또한 기존의 IP 망의 한계를 극복하고자 연구되고 있는 액티브 네트워크 기술을 이용하여 지금까지 사용되고 있는 중앙집중적인 망 관리 방법의 문제점들을 해결하기 위한 하나의 대안으로서 액티브 네트워크 기반의 망 관리 방법을 제시하고 있다.

2. 액티브 네트워크 구조

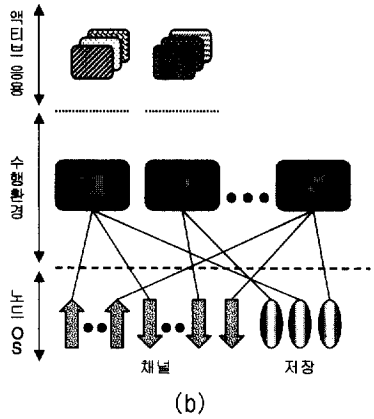
액티브 네트워크에 관한 연구는 현재 여러 곳에서 개별적으로 진행되고 있으며, 현재 연구 진행중인 주제로는 액티브 네트워크 구조, 노드 OS, 구현 기술, 명세, 종단 시스템 문제 등과 보안, 이동성, 혼잡 제어를 포함하는 여러 응용 등이 있다[5-9, 11, 12].

2.1 액티브 네트워크 구성 요소

액티브 네트워크는 (그림 1)의 (a)에서 보는 바와 같이, 기존 망에서의 패킷에 해당하는 액티브 패킷과 이를 수행할 수 있는 중간 노드의 수행 환경(Execution Environment : EE)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 말할 수 있다.



(a)



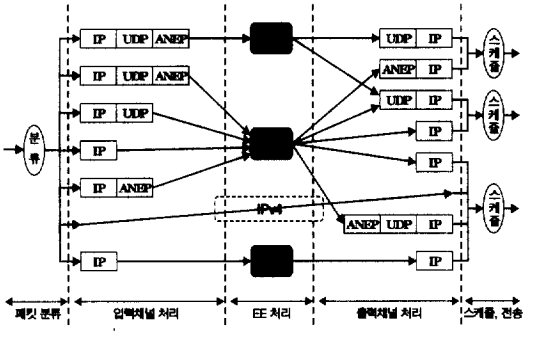
(b)

(그림 1) 액티브 패킷 및 노드의 구조

액티브 노드의 기능은 노드 OS, 수행 환경, 액티브 응용 등의 다양한 구성 요소들에 의해 분리되어 처리된다. 이러한 구성 요소들의 일반적인 구조는 (그림 1)의 (b)와 같다. 수행 환경은 종단간 망 서비스들이 접근될 수 있도록 인터페이스를 제공하는 범용 컴퓨팅 시스템의 '쉘' 프로그램과 같이 동작한다. 이러한 구조는 복수의 수행 환경이 하나의 액티브 노드 상에 존재할 수 있도록 한다. 노드 OS는 수행 환경이 액티브 응용들에 존재하는 추상들을 구성하도록 하는 기본 기능을 제공한다. 이를 위해 노드 OS는 액티브 노드의 자원을 관리하고, 수행 환경으로부터의 전송, 연산, 저장을 포함한 다양한 노드 자원들에 대한 요구를 중재한다[2].

2.2 수행 환경 (EE)

수행 환경은 액티브 네트워크의 사용자들이 활용할 수 있는 가상 머쉰과 프로그래밍 인터페이스를 정의한다. 사용자는 패킷 내에서 적절히 코드화된 명령들을 수행 환경에 전송하므로서 가상 머쉰을 제어한다. 이러한 명령들의 수행은 일반적으로 수행 환경과 액티브 노드의 상태를 변경할 수 있으며, 즉시, 또는 임의의 시간 후에 수행 환경이 패킷을 전송할 수 있도록 한다[2].

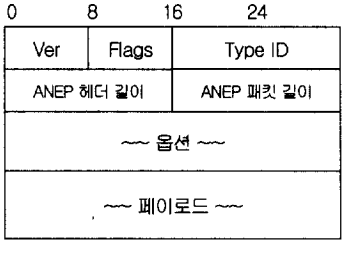


(그림 2) 액티브 노드에서의 패킷 흐름

액티브 노드를 통한 패킷의 일반적인 흐름은 (그림 2)와 같다. 패킷이 수신되면, 먼저 패킷 헤더내의 정보를 토대로 패킷을 분류한다. 이러한 분류 작업을 통해 패킷이 전달되어야 하는 입력 채널을 결정하며, 입력 패킷의 분류는 해당 EE가 가르키는 양식에 의해 조정된다. 일반적인 경우, EE는 이더넷 타입이나 IP 프로토콜의 조합, TCP 포트 번호와 같은 속성을 가진 패킷들에 대한 채널 생성을 요청한다. 이러한 요구는 EE 자체에 의해 또는 액티브 응용에 의해 발생된다. 입력 채널 처리 후, 패킷은 채널의 생성을 요청한 EE에 전달된다. 그림에서의 EE1은 UDP 데이터그램에 캡슐화된 ANEP 패킷을 수신한다. 또한 EE2는 ANEP 패킷을 포함하는 UDP 데이터그램, 일반 UDP 데이터그램, IP 데이터그램내의 ANEP 패킷, 그리고 여러 양식(특정 프로토콜 번호나 특정 소오스 및 목적지 쌍)에 맞는 IP 데이터 그램을 수신한다. 출력 부분에서, EE는 출력 스케줄링 뿐 만 아니라 프로토콜 처리를 포함해 출력 채널에 패킷을 제공함으로서 패킷들을 전송한다. 따라서 일반적인 패킷 처리 방법은 링크 입력, 분류, 입력 프로토콜 처리, EE/AA 처리, 출력 프로토콜 처리, 스케줄링 그리고 링크 전송이다.

2.3 액티브 네트워크 캡슐화 프로토콜 (ANEP)

사용자가 특정 EE에 패킷들을 전달할 수 있기 위해서는 여러 가지 수단이 필요하다. 액티브 네트워크 캡슐화 프로토콜은 이러한 기능을 수행한다[4].



(그림 3) ANEP 구조

(그림 3)에서와 같이, ANEP 헤더는 특정 수행 환경에 할당되는 "타입 식별자" 필드를 포함한다. 만약 하나의 EE가 어느 노드에 존재할 경우, 해당 EE와 같은 타입 식별자를 가진 유효한 ANEP 헤더를 포함한 패킷은 표시된 EE에 연결된 채널들로 라우팅 될 것이다. 이러한 기본 채널들은 EE가 시작할 때 생성된다. 프로토콜 헤더들이 어떻게 구성되어 있는지가 중요하며, 패킷내 일련의 프로토콜 헤더들의 다양한 위치에 ANEP 헤더가 존재할 수 있다.

어떠한 EE에 의해 처리되는 패킷이 ANEP 헤더를 꼭 포함할 필요는 없다. EE들은 액티브 노드를 인식하지 못하는 기존 종단 시스템들에 의해 발생된 패킷을 적절한 채널을 설정하여 처리할 수도 있다. 또한 인증, 기밀성, 무결성 등과 같은 다양한 옵션들이 ANEP 헤더 내에 명시될 수도 있으며, 타입-길이-값(TLV)의

형태로 표현된다. 이러한 옵션들을 위한 필드는 플래그, 옵션 타입, 옵션 길이, 페이로드로 구성된다.

2.4 노드 운영체제

노드 운영체제는 수행 환경들과 하위의 물리 자원들(전송 대역폭, 프로세서 사이클, 스토리지 등) 사이에서 동작하는 계층이다. 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 같은 노드 자원들에 대한 엑세스를 제어하며, 라우팅과 같은 공통의 서비스를 지원하는 것을 포함한다[3].

노드 운영체제 인터페이스는 노드 자원들에 대한 4가지의 기본 추상을 정의한다. 스레드, 메모리, 채널의 세 가지 추상은 각각 계산, 스토리지, 통신을 위한 시스템 자원들에 대한 상이다. 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 관리 시스템 구조

3.1 설계 요구 사항

본 논문은 액티브 네트워크 기술을 이용하는 SNMP 기반의 망 관리 기술에 초점을 두기 때문에 다음의 요구 사항들이 본 논문의 설계 지표로 사용된다.

3.1.1 기존 시스템의 지원

현재 운용중인 망 관리 시스템에 투자된 비용과 노력이 상당할 것으로 예상된다. 따라서 기존 관리 솔루션에 대한 지원이 필요할 것이다.

3.1.2 호환성

호환성은 코드 이동성을 위한 전제조건이다. 오늘날의 망은 상당히 이질적이며, 판매자의 특정한 플랫폼을 토대로 하는 다양한 망 요소들로 더욱 복잡해지고 있다. 자바는 이질적인 환경에서 동작하는 많은 플랫폼에서 동일하게 수행되는 응용들을 구현하는데 적합하다. 이러한 이유로 인해 본 논문에서 구성되는 기반 구조는 자바 기술을 이용한다.

3.1.3 지속적인 상태 유지

대부분의 망 관리 응용에서 응용이 오랜 시간동안 어느 한 노드에 거주하는 것은 자연스러운 요구이다. 액티브 네트워크에서 전송되는 패킷은 수행가능한 프로그램이고, 계속적으로 이동하기 때문에 이동 경로에 있는 노드에 자신의 상태를 지속적으로 유지함으로서, 다음에 오는 패킷이 원할하게 자신의 기능을 수행할 수 있도록 해야만 한다.

3.1.4 보안

보안은 분산 컴퓨팅 분야에서 항상 중요한 문제이며, 특히 액티브 노드는 프로그램가능한 실체이며, 새로운 서비스나 프로토콜들이 전송되어 실행되기 때문에 노드의 안전과 보안에 관한 중요한 문제를 야기한다. 신뢰없는 프로그램들은 그들이 노드에서 수행되기 전에 인증되고 유효성을 검증 받아야만 한다.

3.1.5 노드 자원들에 대한 인터페이스

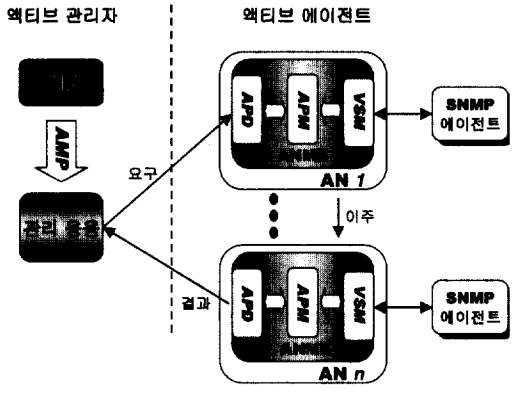
관리되는 노드의 자원들에 접근하기 위한 액티브 관리 패킷의 기능은 망 관리를 위해 필수적이다. 본 논문에서는 이러한 관리 자원들에 대한 정보 수집을 위해 SNMP 프로토콜을 이용하며, 따라서 노드내에 구성된 SNMP 에이전트에 대한 인터페이스를 제공해야만 한다.

3.1.6 이동 코드간 통신

액티브 관리 패킷은 할당된 작업을 수행하기 위하여 다른 노드와 상호동작할 필요가 있다. 이동중인 액티브 관리 패킷간의 통신을 다루기 위해서는 여러 가지 문제가 존재한다. 한가지는 각 패킷의 현재 위치와 되며, 다른 하나는 전송중인 데이터의 순서와 관련된다. 액티브 관리 패킷은 이동하기 때문에 자신의 수행과 관련해 통신해야만 하는 다른 관리 패킷의 현재 위치가 어디인지를 알지 못한다. 따라서 수신자에게 동기적으로 또는 비동기적으로 해당 메시지가 전송되도록 해야 한다.

3.2 액티브 관리 시스템 구조

본 논문에서 기술하는 SNMP 기반의 액티브 네트워크 관리 구조는 (그림 4)에서 보는 바와 같이 액티브 관리자, 액티브 에이전트 및 SNMP 에이전트의 3 부분으로 이루어지며, 다음과 같은 주요 성분들로 구성된다.



AMPG : Active Management Packet Generator
AMP : Active Management Packet
AN : Active Node
ANE : Active Node Management Engine
APD : Active Packet Daemon
APM : Active Packet Manager
VSM : Virtual SNMP Manager

(그림 4) 액티브 관리 시스템 구조

3.2.1 액티브 관리자

액티브 관리자는 SNMP 기반의 다양한 망 관리 응용을 수행할 수 있도록 여러 파라메타를 포함한 서비스 지향의 AMP를 생성하여 액티브 노드에 전송하고 해당 결과를 출력하는 책임을 진다. 이러한 AMP는 미리 정의된 정책을 기반으로 정보를 수집하기 위해 관리 노드들 사이를 이동한다.

3.2.2 액티브 에이전트

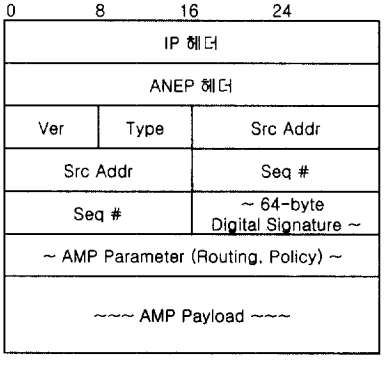
액티브 에이전트는 액티브 관리자로부터 전송된 AMP가 자신의 기능을 효율적으로 수행할 수 있도록 하기 위해 액티브 관리 패킷의 수신, 스케쥴링, 실행, 모니터링 및 전송 등의 다양한 기능을 처리하는 액티브 노드 관리 엔진(ANME) 모듈을 포함한다. ANME는 액티브 관리자로부터 전송된 AMP의 원할한 수행을 위해 로컬의 다양한 물리 자원 및 응용들에 대한 인터페이스를 제공하며, 또한 AMP의 인증 및 보안, 상태 관리, 이주 관리를 위한 기능을 제공한다.

3.2.3 SNMP 에이전트

SNMP 에이전트는 기존 라우터나 스위치 내에 내장된 고유의 망 관리 에이전트이다. 따라서 액티브 노드는 이러한 기존 응용들에 대한 인터페이스를 제공하기 위한 기능이 필요하며, SNMP 에이전트에 대한 인터페이스는 ANME 내의 VSM이 수행한다. VSM은 망 관리와 관련된 AMP의 동작을 수행하고, SNMP 에이전트로부터 해당 결과를 수신하여 이를 액티브 관리자에 전송하는 역할을 한다.

3.3 액티브 관리 패킷 형식

액티브 네트워크에서 액티브 패킷(또는 스마트패킷)이라 부르는 데이터 패킷은 정보의 실체이다. 이러한 패킷들은 사용자에 의해 커스터마이즈된 전송 루틴과 액티브 노드에서 수행하고자 하는 메소드들을 포함한다. 액티브 네트워크에서 관리 기능을 수행하는데 필요한 모든 데이타는 ANEP 페이로드 필드에 포함되며, 페이로드를 인증하기 위한 디지탈 서명을 포함한다. 다음의 (그림 5)는 이러한 AMP의 형식을 보여준다.



(그림 5) 액티브 관리 패킷 형식

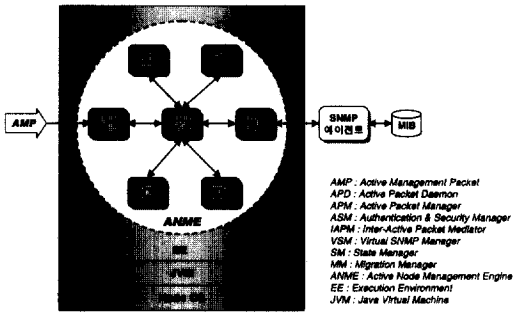
AMP는 공통의 AMP 헤더 내에 싸여진 프로그램, 결과 데이터 또는 메시지들을 포함할 수 있으며, ANEP 내에 캡슐화된다. AMP 헤더는 버전 번호, 타입, 소오스 주소 및 순서 번호의 4개의 필드로 구성된다. 버전 번호는 현재 AMP의 버전으로서 AMP 형식의 변경을 식별하기 위해 사용된다. 타입 필드는 프로그램 패킷, 데이터 패킷, 에러 패킷 또는 메시지 패킷의 4가지 형태 중 하나를 가르킨다. 프로그램 패킷은 라우팅 경로에 있는 노드에서 수행하고자 하는 코드를 운반하며, 데이터 패킷은 AMP의 수행 결과를 관리 노

드에 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 해당 노드에 AMP를 전송한 노드의 주소를 포함한다. 이 값은 각 클라이언트에 대한 ANEP 데몬에 의해 발생된다. 프로그램 패킷이 네트워크를 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 위함이다.

AMP의 페이로드 부분에 포함되어 전달되는 코드와 데이터는 바이트 코드 형식으로 존재하며, 실제로 사용자 데이터를 토대로 관리 노드 상에서 동작하는 메소드들을 기술하는 자바 클래스이다. 사용자 데이터를 포함하는 이러한 자바 클래스의 인스턴스는 클래스 정의를 포함한 바이트 코드들과 함께 직렬화된 객체로서 AMP 내에서 전송된다.

4. 액티브 에이전트 구조 설계

제안되는 액티브 에이전트 구조는 액티브 노드의 운영체제와 자바 수행 환경인 자바 가상 머쉰 (JVM), AMP의 수신, 스케쥴링, 실행, 모니터링 및 전송 등의 기능을 수행하는 ANME 그리고 SNMP 기반의 망 관리 에이전트로 이루어져 있으며 다음 (그림 6)과 같다.



(그림 6) 액티브 에이전트 구조

노드 운영 체제는 현재의 시스템에서 보편적으로 사용되는 운영 체제(Windows, Linux, Solaris)를 이용하며, ANME의 구현을 위해 분산된 이질적인 환경에서 플랫폼에 독립적인 자바를 사용한다.

4.1 액티브 에이전트 구성 요소

4.1.1 액티브 관리 패킷 데몬(APD)

ADD(Active Packet Dameon)는 액티브 관리자로부터 전송된 AMP를 해당 TCP, UDP 포트를 통해 수신하고, 이를 해석하는 역할을 수행한다. 먼저 AMP가 액티브 에이전트 노드에 도착하면, 역다중화기에 대기 행렬화되며, APD는 수신된 패킷이 ANEP에 따르는가를 검사하며 이를 만족할 경우 헤더의 타입 ID 필드를 토대로 AMP의 수행을 관리하는 APM으로 역다중화하 하는 역할을 수행한다.

4.1.2 액티브 관리 패킷 관리자(APM)

APM(Active Packet Manager)은 액티브 에이전트로 수신된 AMP가 자신이 작업을 원할히 수행할 수 있도록 AMP를 관리하는 역할을 수행한다. APM은 AMP가 해당 노드에 도착하면, AMP의 수행을 지원하는 ASM, SSM, IAPM, VSM, MM을 포함한 모든 관리 객체들을 초기화하는 책임을 진다. AMP에 있는 코드는 하나의 스레드 내에서 수행되며, 자원들에 대한 모든 요구는 해당 스레드에 의해 이루어진다. AMP당 하나의 스레드를 가지는 것은 해당 스레드에 할당된 자원들을 관리하기 쉽기 때문에 유용하다. 수신된 AMP가 자신의 관리 작업을 완료하면, APM은 MM을 통해 다음에 방문할 노드를 결정한다. 그리고 나서 수행 결과를 다음 목적지로 이동하는 AMP에 추가하여 전송하고, 해당 AMP와 관련된 모든 자원을 해제한다.

4.1.3 인증 및 보안 관리자(ASM)

액티브 노드는 프로그램 가능한 실체이며, 새로운 서비스나 프로토콜들이 전송되어 실행되기 때문에, 노드의 안전과 보안에 관한 문제를 야기한다. 따라서 ASM(Authentication & Security Manager)은 AMP 내의 응용이 액티브 노드에 있는 내부 데이터 구조나 그 노드에서 수행하는 다른 AMP의 내부 데이터 구조의 무결성을 손상시키는 것으로부터 보호하기 위한 메카니즘을 제공하며, 자바의 보안 클래스 패키지를 이용하여 구현된다. 본 논문에서는 인증 및 보안 관리를 위해 자바에서 제공하는 디지털 서명 알고리즘이나 MD5 알고리즘 등의 보안 패키지를 이용한다. 인증은 공중 키와 사설 키의 쌍을 제공하는 DSA를 사용한다.

4.1.4 AMP간 통신 관리자(IAPM)

AMP는 보통 자신의 활동을 조정할 필요가 있으며, 이러한 조정은 자신들간에 메시지를 전달함으로서 수행될 수 있다. AMP간 통신은 그들의 위치에 투명한 방법으로 이루어지며, 이러한 작업은 IAPM(Inter-Active Packet Mediator)을 통해 관리되어진다. IAPM은 이러한 AMP간 통신을 위해 메시지 저장소(Message Pool : MP)를 이용하며, Serializable 인터페이스를 이용하여 구현된다. AMP는 같은 액티브 노드에서 수행하는 스레드들간의 동기화 객체로서 두 AMP간의 동기적인 메시지 교환을 위해 내용에 무간한 메시지 저장소를 제공하며, 메시지의 내용은 TLV(Tag-Length-Value) 형식으로 표현된다.

4.1.5 상태 관리자(SSM)

SM(State Manager)은 APM과 상호동작하여 AMP의 수행과 관련한 상태 정보를 유지한다. 이로 인해 다음에 오는 AMP가 효율적이고 신속하게 동작할 수 있도록 도움을 준다. SP는 AMP들이 자신과 관련된 상태 정보를 저장하기 위한 저장소로서 AMP들간의 메시지 전달은 비동기적으로 수행되며, 이동중인 AMP간의 상호 통신을 위한 서비스 제공을 하는 별도의 중개 서버를 필요로 하지 않는다. SM은 차후의 사용을 위해 재구성 될 수 있는 지속적인 객체를 생성할 수 있도록 하기 위해 자바의 Serializable 인터페이스를 이용한다.

4.1.6 이주 관리자(MM)

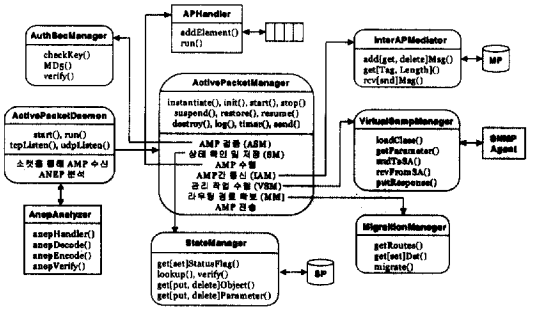
MM(Migration Manager)은 AMP의 다음 목적지를 결정하기 위한 기능을 수행한다. AMP의 이주 전략은 액티브 관리자 시스템에서의 AMPG로 부터 설정되며, 액티브 노드에 수신된 AMP는 이렇게 설정된 이주 전략에 따라 migrate() 메소드를 통해 다음 액티브 노드로 전송되거나, 스케줄된 액티브 노드 순회가 끝난 경우 동작 결과를 액티브 관리자에게 반환할 수 있도록 한다.

4.1.7 가상 SNMP 관리자(VSM)

VSM(Virtual SNMP Manager)은 액티브 노드로 삽입된 AMP가 SNMP 에이전트와 상호작용할 수 있도록 하는 인터페이스를 제공한다. VSM은 AMP가 고유의 액티브 네트워크 관련 기능을 수행한 후, AMP내의 SNMP 관련 요청 서비스 타입과 OID 그리고 접근 제어 등의 정보를 추출하여 SNMP 에이전트에 전달하고, AMP가 이주를 계속할 경우 해당 요구의 결과를 다음 노드로 진행하는 AMP 내에 추가하거나 또는 해당 AMP가 지속 상태를 가질 경우 이를 SM에 전달하는 역할을 수행한다.

4.2 액티브 에이전트 프로토타입 구현

SNMP 관리 기능을 지원하는 액티브 에이전트 프로토타입 시스템의 주요 클래스 및 메소드는 (그림 7)과 같이 구성된다.



(그림 7) 액티브 에이전트의 주요 클래스 및 메소드

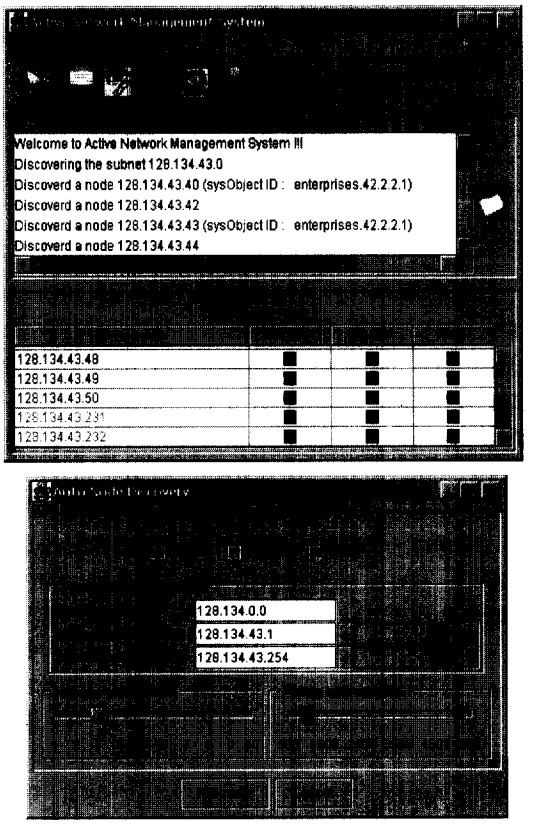
(그림 7)에서와 같이 ActivePacketDaemon(APD) 클래스는 해당 TCP, UDP 포트로 입력되는 AMP를 수신하는 데몬 프로세스이다. udpListen() 메소드를 통해 UDP 포트 3322에서 AMP를 수신하여, ANEP 헤더를 분석(class AnepAnalyzer)하고, ANEP 타입 식별자를 토대로 AMP 패킷을 역다중화하여, APM에 전달한다. APD로 수신된 AMP의 수행을 관리하는 역할은 APM이 담당한다. APM을 구현하는 ActivePacketManager 클래스는 액티브 에이전트의 핵심 부분이며, AMP와 ANME의 여러 다른 서비스와의 인터페이스를 제공한다. APM은 APD로부터 전달된 AMP를 큐에 저장(class APHandler)하고, 인스턴스화 시킨다. 그리고 나서 AMP의 AuthSecManager 클래스를 통해 사용자와 코드에 대한 인증을 수행하고, AMP의 현재 상태를 저장한다. APM은 인스턴스화된 AMP의 망 관리 작업 수행을 위해 해당 객체와 파라메타를 VSM(class VirtualSnmpManager)에 전달한다. VSM은 해당 객체와 파라메타(요구 타입, 커뮤니티, OID)를 분석하여 SNMP 에이전트에 대해 sndToSA() 메소드를 통해 관리동작을 수행하고, rcvToSA() 메소드를 통해 결과를

받아 이를 다시 APM에 전달한다. APM은 VSM으로부터 결과를 받아 AMP를 구성하여 이를 액티브 관리자에 전송하거나 만약 해당 AMP가 다른 노드로의 라우팅이 예정되어 있을 경우, MM(class MigrationManager)를 통해 다음 목적지를 결정하여, AMP를 전송한다.

5. 액티브 관리 응용

5.1 액티브 관리자 시스템 구현

액티브 관리자는 액티브 에이전트의 기능을 수행하는 액티브 노드와의 상호작용을 통해 관리 및 제어 동작을 수행한다. 관리대상 노드들인 액티브 에이전트들은 (그림 8)에서와 같이 관리자에 의해 브로드캐스트 폴링을 통해 탐색된다.



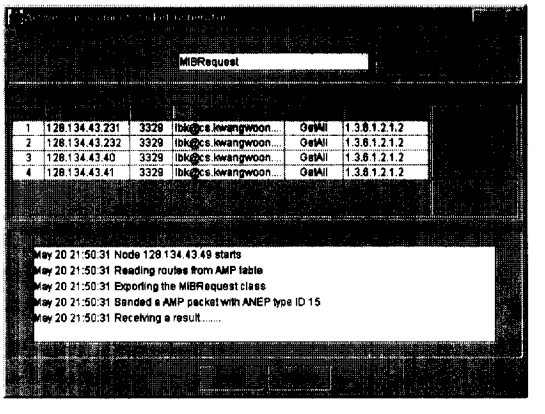
(그림 8) 액티브 관리자 시스템 사용자 인터페이스

자동 노드 탐색에 의해 발견된 액티브 에이전트는 액티브 관리자의 탐색 노드 리스트에 해당 노드 이름이 추가된다. 발견된 액티브 노드 에이전트들의 상태는 실시간으로 ICMP, SNMP, ANEP의 동작 상태에 따라 액티브 관리자 시스템의 GUI 상에 디스플레이 된다. 또한 액티브 관리자는 AMPG를 통해 생성된 AMP를 해당하는 액티브 노드에 전송하고, 그 결과를 수신한다.

5.2 액티브 관리 패킷 생성기(AMPG) 구현

AMPG는 기본적으로 사용자에 의해 커스터마이즈된 AMP를 생성 및 구성하기 위한 도구이다. AMPG는 먼저 구성 관리 AMP에 의한 노드 탐색 과정을 통해 발견된 액티브 에이전트들에 대한 관리 요구를 발생시키기 위해 순회되어야 할 노드들을 선택한다.

(그림 9)에서와 같이 선택된 노드들이 순회노드 리스트에 추가되면, 관리자는 액티브 에이전트에서 수행해야할 수행가능 파일의 클래스를 할당하고, 해당 노드들에 대한 요청 MIB 값과 서비스를 선택한다. 마지막으로 관리자는 순회노드 리스트에 있는 노드들에 대한 라우팅 순서를 결정하여 AMP를 구성하여, 이를 해당 액티브 노드 에이전트들에 UDP 연결을 통해 전송한다.

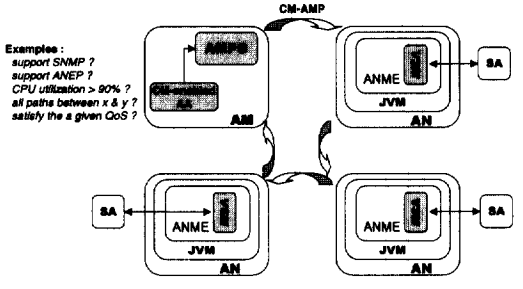


(그림 9) AMP 생성기 사용자 인터페이스

5.3 구성 관리 응용

망 관리에서 노드 자동 탐색 및 구성은 관리 시스템의 기본 기능 중의 하나이다. 단순한 구현에서는 단지 관심있는 망 노드들을 찾고자 시도한다. 이러한 탐색 및 구성 기능을 확장하면 해당 노드에서 활용할 수 있는 서비스나 어떠한 조건을 만족하는 장비들을 포함한 더욱 상세한 뷰를 구성할 수 있다. 이러한 탐색 및 구성 기능의 복잡도가 증가하게 되면, 일반적인 클라이

언트/서버 접근 방법을 이용하기가 상당히 어렵게 된다.



(그림 10) 구성 관리 응용 동작 구조

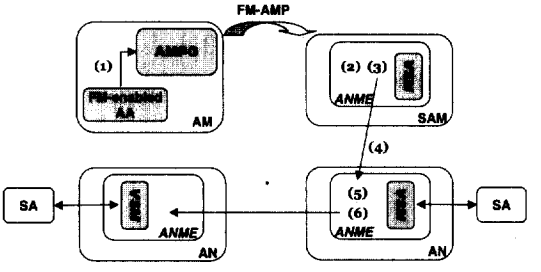
가장 일반적으로 사용되는 탐색 기법중의 하나가 특정 도메인에 있는 장비들의 IP 주소들에 ping 메시지를 전송하는 것이다. 이를 통해 탐색 프로세스는 수신된 응답을 토대로 네트워크의 뷰를 구성한다. 본 논문에서는 이러한 ping 기능을 확장하여, ICMP 메시지뿐 만 아니라 SNMP, ANEP 메시지를 포함해 다양한 조건들을 조합하여 탐색 패킷(CM-AMP)을 구현하였다. 예를 들어, (그림 10)에서 보는 바와 같이 해당 노드가 SNMP나 ANEP 프로토콜을 지원하는가 ? CPU 이용률이 90% 이상인가? 또는 해당 노드가 x,y 노드 사이에 존재하는가? 등과 같이 여러가지 조건을 응용한 CM-AMP를 구성할 수 있다. 이러한 조건들을 조합하면 특정 부분 또는 특정 기능을 수행하는 노드들에 대한 네트워크 토폴로지를 구성할 수 있다. 예를 들어, CM-AMP내에 노드들의 타입과 관련한 조건을 삽입하면, 단지 특정 타입의 노드들로만 이루어진 네트워크 모델을 구성할 수 있다. (그림 8)에서는 특정 도메인 내에 있는 노드들 중 SNMP와 ANEP를 모두 지원하는 노드들의 네트워크 뷰를 구성하고자 할 때 삽입되는 CM-AMP의 구성을 보여주고 있다.

5.4 장애 관리 응용

5.4.1 장애 관리 구조

장애 관리는 망에서 발생되는 문제들을 식별하고, 이를 추적 및 해결하기 위한 방법들과 관계한다. 망 문제들은 망 요소들이 부정확하게 동작함으로서 발생되는 하드웨어 및 소프트웨어 장애로서 분류될 수 있다. 일반적으로 장애 관리 동작의 흐름은 다음과 같다.

1) 망 내의 각 장비들로부터 관리 정보 수집
2) 해당 정보들을 필터링시켜 경보를 발생
3) 발생된 경보들을 상호연관시켜 그 정보를 유발한 장애를 진단
4) 장애의 교정을 위한 계획 수립
5) 장애 교정
6) 관리되는 망으로부터 장애가 제거되었음을 검증



(그림 11) 장애 관리 동작 흐름도

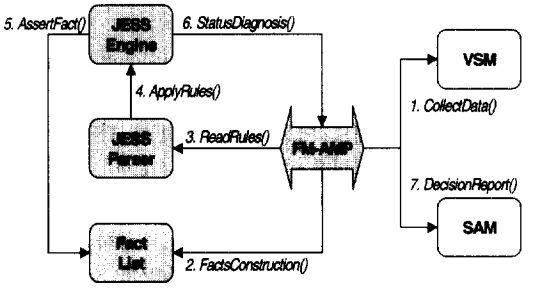
(그림 11)에서는 본 논문에서 설계된 장애 관리 기능의 구조의 동작을 보여준다. 각 노드들은 자바 가상 머쉰내에서 액티브 노드 관리 기능을 수행한다. 앞에서 설명한 바와 같이 VSM은 관리 대상 노드들에 대한 정보를 수집하기 위해 SNMP 에이전트에 대한 가상 인터페이스를 제공한다.

장애 관리 응용이 탑재된 FM-AMP(Fault Management-AMP)에 의해 장애 관리 작업을 수행하기 위한 과정은 여러 단계로 이루어진다. 첫번째 단계는 기능 위임 단계로서 작업을 수행하는데 필요한 장애 관리 응용을 AMP에 탑재하는 과정이다. (그림 8)의 AMPG에서와 유사한 방법으로 장애 관리를 수행하는 클래스와 관련 파라메타가 탑재된 FM-AMP를 SAM(Sub-Active Manager)에 삽입한다(1). FM-AMP는 자신의 통신 특성에 따라 SAM과 상호작용할 것이다(2). 이 상호작용의 목적은 해당 SAM이 수신한 메시지들로부터 관리 노드들의 상태에 관한 국부적인 사실들을 확보하기 위한 것이다. 그러한 사실들로는 메시지 심도, 메시지 수신 시간, 해당 메시지를 발생시킨 액티브 노드들의 식별자 등을 포함한다. FM-AMP는 이때 낮은 심도의 메시지를 폐기하고, 경보들을 평가하는 등의 메시지 심도에 따라 사실들을 분류하기 위해 장애 관리 응용을 적용한다. 다음 단계는 같은 경보가 여러개 발생한 경우 이를 제거하기 위해 메시지를 여과하는 작업이다. 그리고 나서 FM-AMP는 특정 양식에 일치하여 여과된 경보들의 집합에 대해 상관관계를 조사한

다. 이러한 작업이 특정 위치에서 수행된 후, FM-AMP는 경보를 발생시킨 노드들로 이주한다(4). 즉 첫 단계의 자동 이주가 시작된다. 이러한 이주 결정은 SAM에 구성된 정보와 경보 여과 및 상관 조사가 FM-AMP 자체에 의해 수행된 후 결정된다. 특정 노드에 도달한 FM-AMP는 해당 노드를 진단하기 위해 자신의 장애 관리 응용을 적용한다(5). 이 단계는 발생된 경보의 원인과 해당 경보가 거짓인지를 판단하기 위한 과정이다. FM-AMP는 적용된 응용을 바탕으로 장애를 진단하고, 이를 SAM에 보고한다(6).

5.4.2 장애 관리 프로토타입 시스템

(그림 12)에서는 장애 관리 기능의 프로토타입 시스템을 보여준다. 여기에서는 계층적인 망 모델을 가정하고 있으며, SAM이 특정 도메인 내의 노드들로부터 메시지를 수신하는 책임을 진다. 두번째 가정은 물리적이고 논리적인 연결 정보를 표현하는 망 모델이 SAM에 유지되고 있음을 가정한다. 세번째 가정은 상태 메시지가 유일한 식별자, 메시지 수신 시간, 경보 심도, 메시지의 발생지 식별자를 포함하는 공통의 형식을 가지고 있다고 가정한다.



(그림 12) 장애 관리 응용 프로토타입

위의 그림에서 JESS(Java Expert System Shell)는 FM-AMP의 지능적인 동작을 구성하는 자바 기반의 전문가 시스템 쉘이다[26]. FM-AMP는 망 노드에 도달한 후, 해당 노드의 상태 데이터를 수집하기 위해 VSM과 통신한다. 데이타 분석을 통해 망 노드들의 상태에 관한 사실들의 집합을 구성한다. 다음에, 문법 에러를 검사하는 전문가 시스템 규칙 파일을 파스하기 위해 JESS 파서 객체가 생성하고, FM-AMP로부터 읽어들인 규칙들이 JESS 추론 엔진 객체에 연결된다. 그리고 나서 FM-AMP는 추론 엔진의 사실 목록 내에 있는 사실들을 평가한다.

앞서 기술된 내용은 모든 망 노드들에 공통이지만, SAM에서의 FM-AMP의 목적은 망 노드들의 목적과는 다르다. 규칙을 토대로 정의하는 SAM의 첫 번째 행위는 메시지 범주를 결정하기 위한 경보 메시지 식별이다. 신뢰성이 없는 메시지들을 제거하기 위해 메시지들이 여과된다. 이러한 여과 과정은 실제 경보의 검출을 유도한다. 장애를 지역화하기 위해 JESS에서는 경보들을 검사하고 이들의 상관관계를 조사하며, 거짓 경보들을 분리한다. 이러한 과정의 결과는 장애를 발생시킨 망 성분들의 주소 집합이다. AMP는 이러한 주소들의 목록을 유지하고, 여기에 포함된 각 노드들을 방문하고자 시도한다. FM-AMP는 먼저 해당 요소가 다운 상태인지를 검사하기 위해 PING 유틸리티를 사용한다. 만약 해당 요소가 다운일 경우, 이 사실이 관리자에게 보고되며, 해당 주소 목록으로부터 해당 요소가 제거된다. 이러한 지능적인 장애 관리 응용을 탑재한 액티브 관리 패킷은 망 장비들에 동적으로 관리 응용을 삽입함으로서 망 관리 능력을 향상시킬 수 있을 것이다.

6. 결론 및 추후연구방향

액티브 네트워크 기술은 전통적인 패킷 교환 망에서의 수동적인 패킷 처리를 넘어서 사용자에 의한 계산 및 수행이 가능하게 함으로서 현재의 망 환경에서 다룰 수 없는 다양하고도 혁신적인 기술들을 연구할 수 있는 기반을 제공한다.

본 논문에서는 이러한 액티브 네트워크 기술을 토대로 기존의 널리 사용되고 되고 있는 SNMP 프로토콜을 이용하여 망 관리를 수행할 수 있는 SNMP 기반의 액티브 네트워크 구조에 대해 기술하였다. 또한 이러한 구조를 기반으로 액티브 관리자 및 액티브 에이전트의 프로토타입 시스템을 구현하였으며, 이를 이용한 구성 관리 및 장애 관리 응용과 같은 망 관리 응용에 대해 기술하였다. 본 논문의 목적은 액티브 네트워크 기반 위에서 새로운 망 관리 응용을 시험할 수 있도록 하는 프로토타입을 구성하는데 있으며, 이러한 경험을 바탕으로 액티브 노드에서의 더욱 용이한 프로그래밍과 신속한 수행을 위해 액티브 노드 관리 엔진의 기능이 계속적으로 향상되어야 할 것이다. 또한 본 논문에서 구현된 액티브 네트워크 관리 메카니즘을 토대로 구성, 장애, 성능 관리 등의 다양한 망 관리 기능을 효

율적으로 수행할 수 있도록 하는 관리 응용을 개발해야 할 것이다. 망 관리를 위해 액티브 네트워크 기술을 이용한 새로운 망 관리 기법은 다가올 미래의 네트워크에서 새롭고 효과적인 방법으로 망 관리를 수행할 수 있을 것이다.

참 고 문 헌

[1] D. Tennenhouse and D. Wetherall, "Towards an Active Network Architecture," Computer Communication Review 26(2), April 1996.

[2] DARPA AN Architecture Working Group, "Architectural Framework for Active Networks," http://www.cc.gatech.edu/projects/canes/papers/arch-1-0.pdf, July 1999.

[3] DARPA AN Node OS Working Group, "NodeOS Interface Specification," http://www.cs.princeton.edu/nsg/papers/nodeos99.ps, January, 2000.

[4] D. Alexander, et al, "Active Network Encapsulation Protocol (ANEP)," RFC Draft, http://www.cis.upenn.edu/~switchware/ANEP/docs/ANEP.txt,July 1997,

[5] D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS : A Toolkit for Buildig and Dynamically Deploying Network Protocols," IEEE OPENARCH'98, April 1998.

[6] D. Alexander et al, "The SwitchWare Active Network Architecture," IEEE Network Magazine, July 1998.

[7] M. Hicks et al, "PLAN ; A Programming Language for Active Networks," International Conference on Functional Programming (ICFP'98), September 1998.

[8] Livio Ricciulli, "An Adaptable Network Control and Reporting System," Sixth IFIP/IEEE International Symposium on Integrated Network Management, May 1999.

[9] M. Molteni, L. Ricciulli and S. Tsui, "User Guide to Anetd 1.4," http://www.csl.sri.com/ancors/anetd, April 2000.

[10] B. Schwartz, A. Jackson and W. Strayer, "Smart Packets for Active Networks," OPENARCH'99, March 1999.

[11] A. B. Kulkarni et al, "Implementation of a Prototype Active Network," OPENARCH'98, April 1998.

[12] Danny Raz and Yuval Shavitt, "An Active Network Approach to Efficient Network Management," Proceedings of the 1st International Working Conference on Active Networks (IWAN'99), July 1999.

[13] Y. Yemini, G. Goldszmidt, and S. Yemini, "Network Management by Delegation," Proceedings of ISINM '91, April 1991.

[14] A. Bieszczad, B. Pagurek and T. White, "Mobile Agents for Network Management," IEEE Communications Surveys, Fourth Quarter 1998.

[15] S. Albayrak, F. J. Garij, Eds., "Intelligent agents for telecommunication applications," IATA'98, July 1998.

[16] Ahmed Karmouch, "Mobile software agents for telecommunications," IEEE Communication Magazine, July 1998.

[17] T. Magedanz, R. Glitho, Eds., "Mobile agent-based network and service management," Journal of Network and System Management, September 1999.

[18] G. Knight, R. Hazemi, "Mobile agent-based management in the INSERT project," Journal of Network and System Management, September 1999.

[19] M. Zapf, K. Herrmann, K. Geihs, "Decentralized SNMP management with mobile agent," INM'99, May 1999.

[20] C. Breugst, T. Magedanz, "Mobile agents-Enabling technology for active intelligent network implementation," IEEE Network Magazine, May 1998.

[21] H. Bos, "Application-specific policies : Beyond the domain boundaries," INM'99, May 1999.

[22] D. S. Decasper, "A scalable high-performance active network node," IEEE Network Magazine, January 1999.

[23] James Gosling, Bill joy, and Guy Steele, The Java Language Specification, Addison Wesley, 1996.

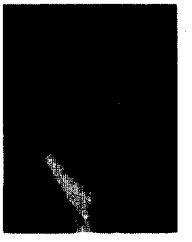
[24] J. Case, M. Fedor, M. Shoffstall, J. Davin, "A Simple Network Management Protocol(SNMP)," RFC1157, May 1990.

[25] Friedman-Hill, E. J. : "JESS : The Java Expert

System Shell.," http://herzberg.ca.sandia/Jess/
[26] AdventNet, AdventNet SNMP API 3.0, URL : http://www.adventnet.com/

이 병 기

e-mail : lbk@cs.kwangwoon.ac.kr
1991년 원광대학교 물리학과 (학사)
1994년 광운대학교 컴퓨터과학과 (석사)
1994년~1996년 ㈜코오롱정보통신 기술연구소 연구원
1998년 광운대학교 컴퓨터과학과 박사과정수료
관심분야 : 망 관리, 액티브 네트워크, 이동에이전트



조 국 현

e-mail : khcho@cs.kwangwoon.ac.kr
1977년 한양대학교 전자공학과 (공학사)
1984년 일본 동북대학교 대학원 (공학박사)
1984년~현재 광운대학교 교수, 광운대학교 이과대학 학장, 전자계산소장 역임
1999년 현재 전산대학원장, 한국정보과학회 이사, 개방형 컴퓨터 통신연구회 부회장 역임, 현재 개방형 컴퓨터 통신연구회 회장
관심분야 : 정보통신망 관리, 분산처리, 멀티 캐스팅 및 정보통신분야의 표준화