액티브 네트워크의 망관리 메카니즘

오 행 석* 남 택 용** 채 기 준***

◈ 목 차 ◈

1. 서 론

인터넷의 급격한 확산과 인터넷을 이용하는 응용의 수가 급격히 증가함에 따라 인터넷을 이용하는 응용 및 사용자의 네트워크에 대한 요구는 점차적으로 복잡해지고 그 수가 증가하고 있는 상태이다. 이를 수용하기 위한 여러 기술들이 연구되고 있지만, 이들 기술이 표준화 단계를 거쳐 실제 망에 채택되어 사용되기까지는 많은 시일과 비용이 소요된다. 또한 사용자 요구 조건의 변화 속도와 이를 지원하기 위한 네트워크 시스템의 변화 속도간에는 차이가 발생하게 되어 사용자의 망에 대한 요구 기능을 시기 적절하게 반영하는 하는 것은 현재의 네트워크구조 하에서는 불가능하다. 이를 극복하기 위해 네트워크 노드의 구조를 프로그래밍이 가능하도록 하고, 사용자 요구 기능을 수행할 수 있는 프로그램 코드를 전송/실행함으로써 통신망에 새로운 서비스를 보다 신속하고 경제적으로 도입하고 망 자원들을 보다 적절하게 활용할 수 있도록 하는 데 목표를 두고 연구되고 있는 분야가 액티브 네트워크 분야이다.

기존의 네트워크 노드가 단순히 패킷을 저장한 후 포워딩(store and forward) 하는 식의 단순한 네트워킹 기능을 하는 것과는 달리 액티브 네트워크는 사용자가 원하는 프로그램을 패킷을 통하여 전송하여 실행하거나 네트워크 노드에 미리 설치된 프로그램 중에서 해당 기능을 실행함으로써 사용자가 원하는 네트워크 기능을 이용하게 된다. 이처럼 네트워크 노드에서 라우팅과 같은 단순한 기능에서 벗어나 네트워크 종단간에서만 이루어지던 여러 가지 에러 처리 및 흐름 제어와 같은 복잡한 기능 혹은 그 외 사용자가 원하는 기능을 네트워크 노드에서 수행할 수 있다는 것은 사용자나 네트워크 망 자체에 유연성 뿐만 아니라 여러 많은 장점들을 제공할 수 있다.

본 논문은 BcN 에서의 액티브 네트워크 기술을 이용한 망 관리 구조와 망 관리 요소의 기능에 대해 기술한다. 관리목적 상 네트워크는 동종망으로 구성된 도메인으로 구분될 수 있다. 또한 네트워크는 여러 개의 도메인으로 나누어지며, 각 도메인 내에는 한 개 이상의 능동 노드가 존재하게 된다. 각각의 도메인에 대한 자체적인 관리는 각 도메인에 포함되어 있는 하나의 능동 노드가 담당한다. 위에서 기술한 두 가지의 관리, 즉 도메인 내의 관리와 능동 노드 자체에 대한 관리는 기 표준화된 SNMP와 같은 기존의 방식을 활용하는 것으로 정의한다. 따라서 이 문서에서는 액티브 네트워크의 특징 및 장점을 살릴 수 있는 관리 대상을 선정하여, 도메인을 관리하는 능동 노드들 간의 협업적 관리 구조를 제안하며, 망 관리의 목적은 능동 노드들 간의 협업적 망 관리를 통한 고장관리, 보안관리, DDoS 공격 및 대응관리 및 서비스 품질관리를 수행하는 구조를 정의하는데 그 목적이 있다.

본 논문은 액티브 네트워크의 망관리 메카니즘구조에 대하여 알아보고자 한다. 이를 위하여 2장에서는

* 한국전자통신연구원 정보보호연구단 개인정보보호연구팀책임연구원
** 한국전자통신연구원 정보보호연구단 개인정보보호연구팀팀장, 책임연구원
*** 이화여자대학교 컴퓨터공학과 교수

액티브 네트워크의 망관리 관련 연구, 3장에서는 액티브 네트워크의 노드 구조에 대하여 알아본다. 4장에서는 액티브 네트워크의 패킷 구조에 대하여 기술하고, 마지막으로 결론을 기술하고자 한다.

2. 액티브 네트워크의 개념

액티브 네트워크를 구현하는 방법은 프로그램 코드의 전송 여부에 따라 크게 두 가지로 분류할 수 있다. 첫번째는 액티브 네트워크에 사용되는 프로그램이 중간 노드에 미리 설치되어 있고, 사용자가 설치된 프로그램 중 어느 하나를 실행하기를 원할 경우 해당 프로그램을 식별하는 지시자와 해당 프로그램이 실행하는 데 필요한 데이터를 전송하여 해당 기능을 실행하는 방법이 있다. 이 경우 네트워크 상에서 실행 프로그램의 전송과 실제 사용자 요구 기능(메시지)의 실행은 분리되어 이어지게 된다. 프로그램은 인증된 관리자에 의해 네트워크에 배포되며 이런 접근방법을 프로그래머블 스위치 (Programmable Switch) 방식 또는 분리(Discrete) 방식이라 한다. 이 방법에 따른 대표적인 액티브 네트워크 프로젝트는 펜실바니아 대학의 Switchware와 콜럼비아 대학의 Netscript 등이 있다. 이 방법은 실행 기능이 복잡해서 프로그램의 크기가 커서 프로그램 자체를 전송하기에는 사용자의 요구에 시기 적절하게 대응하는 것이 어려울 경우에 유리하다.

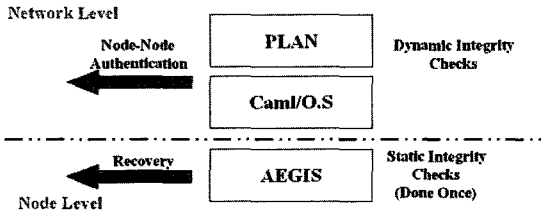
또 다른 방법은 통합(Integrated) 방식 또는 캡슐(Capsule) 방식이라고도 불리며 이는 실행에 필요한 데이터뿐만 아니라 실행 프로그램 자체를 일반 패킷과 마찬가지로 캡슐이라 불리는 능동 패킷을 통하여 중간 노드에 전달하여 실행하는 방법이 있을 수 있다. 이 경우 액티브 네트워크 상에서 전송되는 모든 패킷은 네트워크 노드에서 실행 가능한(되어야 할) 프로그램으로 볼 수 있으며 프로그램은 네트워크 상의 어느 사용자에 의해서도 네트워크에 전송이 가능하다. 이 방법에 따른 대표적인 액티브 네트워크 프로젝트는 MIT 공대의 ANTS, 펜실바니아 대학의 PLANet 등이 있다[1~3].

2.1 SANE(Secure Active Network Environment)[4]

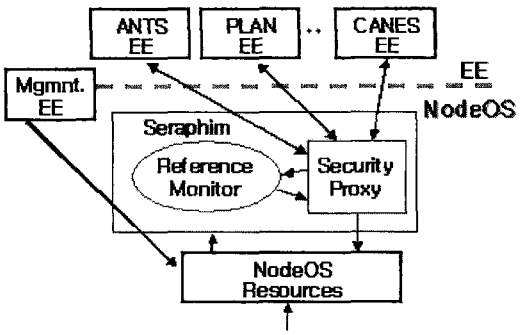
펜실베니아 대학에서 주체가 되어 수행한 SANE 은 안전한 액티브 네트워크를 구현하기 위하여 SwitchWare 환경에서 구현되었으며 계층화된 보안 구조로 인증, 무결성, 암호 등의 보안 서비스를 제공한다. SANE에서 제공하는 보안은 정적인(static) 보안과 동적인(dynamic) 보안 두 레벨로 구분된다. 정적인 보안은 자주 검사할 필요가 없는 대상에 대해 수행하는 것으로 액티브 네트워크가 휴지(idle) 상태에서 동작 상태로 전이하는 부트스트랩 같은 것을 의미한다. 이런 정적인 보안은 비용이 많이 들지만 안전하다. 반면 동적인 보안은 계속해서 검사가 수행이 되어야 하는 보안요소에 대해 수행하는 것으로, 비용이 적게 들지만 시스템의 성능을 저하시킬 수도 있다.

SANE에서 제공하는 이런 보안 서비스의 주된 두 가지 기본 개념은 무결성(integrity)과 신뢰성(trust)이다. 무결성이라 함은, 시스템이 수정되지 않았다고 말할 수 있는 것이다. 신뢰성은 좀 더 복잡한 개념으로, 시스템이 수정되지 않았다고 해서 신뢰할 수 있다고 말할 수는 없지만, 반대로 시스템을 신뢰할 수 있으려면, 시스템이 수정되지 않은 상태로 유지되어야만 한다. 계층화된 구조에서, 시스템의 각 계층은 자신보다 하위 계층을 신뢰할 수 있어야 한다. 또한 패킷에 의해 전달되는 프로그램은 하나의 네트워크 노드에서 다른 노드로 전송되기 때문에 관련된 네트워크 노드들 사이에는 복잡한 신뢰관계가 필요하다. 신뢰관계가 형성되었더라도 네트워크 노드들로부터 프로그램이 전송되는 동안 손상될 수도 있으므로 액티브 네트워크에서 보안을 보장하려면 동적인 보안과 정적인 보안을 결합해서 검사해주어야 한다.

SANE의 시스템 계층 구조는 (그림 1)에 나타나 있다. 여기에서 우리는 SANE의 전체적인 구조와 그 구조가 나타내는 주된 목적을 살펴볼 수 있다. 이 구조의 하위 계층들은 시스템이 요구한 상태에서 시작할 수 있도록 보장한다. 그러기 위해서 SANE의 구조는 AEGIS라고 불리는 안전한 부트스트랩을 사용한다. AEGIS는 펌웨어의 초기화



(그림 1) Sane System 기능 구조



(그림 2) Seraphim 구조

단계에서의 무결성을 제공하기 위해서 액티브 네트워크의 구성요소가 동작할 때까지 부트스트랩에서 반복적으로 무결성을 검사한다.

2.2 Seraphim[5]

일리노이 대학에서 연구된 Seraphim은 다양한 보안 정책과 메커니즘을 수용할 수 있게 확장 가능하고, 재구성 가능한 보안 구조이다. Seraphim은 응용 프로그램과 사용자에게 동적으로 상황에 맞는 정책을 생성하고 수용할 수 있는 능력을 제공한다.

(그림 2)는 Seraphim의 주요 구성 요소에 대한 구조를 보이고, 액티브 네트워크가 실제로 어떻게 상호 동작하는지를 나타내고 있다. Seraphim 구조에서 가장 중요한 부분은 참조 모니터(reference monitor)이다. 모든 노드는 참조 모니터를 가지고 있으며, 노드의 자원으로 접근하려면 반드시 이 참조 모니터를 거쳐야 한다. 참조 모니터는 노드의 자원을 요구하는 모든 요청들을 가로채서 정책에 따라 자원을 허락할지 여부를 결정한다. Seraphim에서는 접근 제어 정책으로임의 접근 제어(discretionary access control), 강제 접근 제어(mandatory access control), 이중 임의 접근 제어(double discretionary access control), 역할 기반 접근 제어(role-based access control) 정책을 제공한다.

2.3 PLAN(A Packet Language for Active Networks)[6]

펜실베니아 대학에 의해 수행된 PLAN(Programming Language for Active Networks)은 액티브 네트워크 환경에서 능동 패킷을 생성하기 위하여 개발된 언어로서 프로그래밍 관점에서 안전성 및 보안의 문제를 다루었다. PLAN의 기본 설계 방법은 경량화하고 제한된 기능을 가진 프로그램을 제공하는 것이다. PLAN 코드는 다른 프로그래밍 루틴 즉, 노드에 상주하며 좀더 강력한 다른 언어로 작성된 프로그래밍 루틴을 호출할 수 있도록 한다. PLAN 프로그램은 작기 때문에 인증이 필요하지 않지만 노드에 상주하는 서비스 루틴들이 필요로 할 경우 인증을 제공한다. PLAN은 안전성과 보안, 성능, 융통성의 문제를 다음과 같이 다룬다.

2.3.1 안전성 및 보안 (Safety and Security)

PLAN은 완전히 기능적(functional)이며 강한 타입의 언어이고, 네트워크 상에 전달되기 전에 패킷의 타입이 정적(static)으로 검사되므로 타입 오류가 발생하지 않는다. 또한, PLAN 프로그램은 포인터에 안전하고 현재 수행중인 프로그램이 다른 프로그램에 영향을 줄 수 없으며 기본적인 오류 처리는 서비스 루틴에 의해 처리된다.

2.3.2 성능 (Performance)

PLAN을 단순하게 유지함으로써 번역(interpretation)이 경량화 되고, 작업을 빠르고 쉽게 수행할 수 있다. 즉, 간단한 데이터와 제어구조를 지원하므로 컴파일과 인터프리트(interpret)하기 쉽다.

2.3.3 융통성 (Flexibility)

PLAN은 일반적이지는 않지만 네트워크의 구성이나 진단을 위한 프로그램을 작성할 수 있고, 라우터에 상주하는 서비스 루틴을 더 큰 프로토콜로 연결하는 분산 컴퓨팅의 연결성을 제공할 수 있다.

2.4 FAIN(Future Active IP Network)[7]

FAIN은 2000년부터 UCL(University College London)이 주축이 되어 진행 중인 프로젝트로 능동 노드를 기반으로 개방적이고 프로그램 가능하며 신뢰성 있는 액티브 네트워크 구조를 개발하는데 목적을 두고 있다. 이는 액티브 네트워크, 능동 노드, 정책 기반의 네트워크 관리(policy-

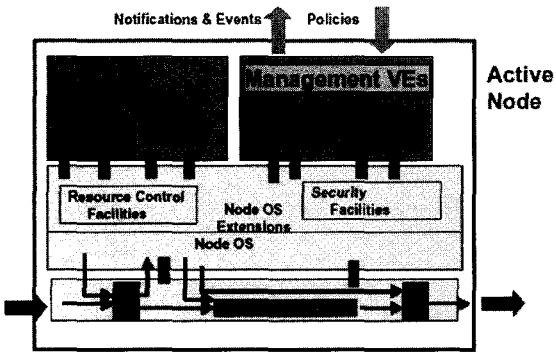
based network management)와 동적인 프로토콜 지원을 통해서 액티브 네트워크 서비스 제공을 목적으로 진행되고 있다.

FAIN 프로젝트는 사용자(customers/consumers), 서비스 제공자(service provider), 응용 개발자(application developers)들의 다양한 요구사항을 반영하여 설계되었다. 사용자 측면에서는 운영자(Operator)의 개입 없이 네트워크를 동적으로 재구성(reconfiguration)하고, 네트워크와 서비스를 통제할 수 있어야 한다. 서비스 제공자 측면에서는 네트워크의 자원과 서비스를 구분할 수 있고 자신의 요구에 맞는 관리 시스템과 부가 서비스를 제공하며 기존의 네트워크 기반 구조와 공존할 수 있어야 한다. 응용 개발자 측면에서는 네트워크에서 제공되는 서비스의 최상위에서 쉽게 응용 프로그램을 개발하고 표준화되지 않는 구성 요소도 사용할 수 있어야 한다.

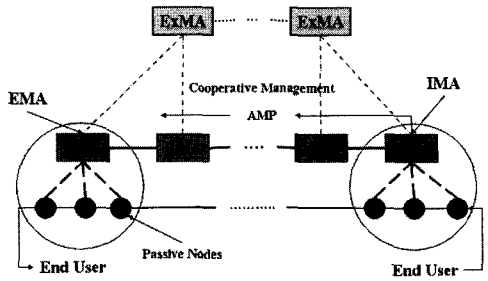
FAIN에서는 보안을 위해 (그림 3)과 같은 구조를 개발했다. 라우터에 들어온 패킷은 기존의 패킷인 경우 다음 노드로 전송이 되고, 능동 패킷인 경우 노드 운영 시스템을 거쳐 실행환경으로 가서 프로그램을 실행하게 된다. 노드 운영시스템에는 보안 모듈과 자원 접근 모듈들이 포함되어 있어 들어온 패킷에 대해 보안 검사를 실시하고 통과된 패킷에 관해 할당된 자원을 분배하는 역할을 하고 있다.

3. 협업적 망 관리 구조

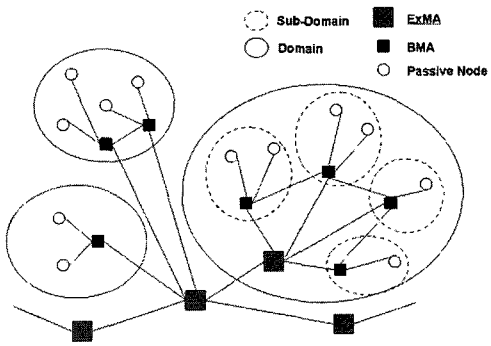
망 관리 구조는 계층(hierarchical)구조를 갖는다. 본 논문에서 나오는 Agent는 능동 패킷을 받았을 경우, 패킷 내



(그림 3) FAIN 능동 노드 구조



(그림 4) 액티브 네트워크 관리 구조 1



(그림 5) 액티브 네트워크 관리 구조 2

에 탑재하고 있는 코드를 실행할 수 있는 실행 환경(EE)을 가지고 있다. 전체 망 관리 수행을 위한 구조와 이런 실행 환경을 가진 능동 노드의 구조 및 역할에 대해서는 3-2절에서 기술된다.

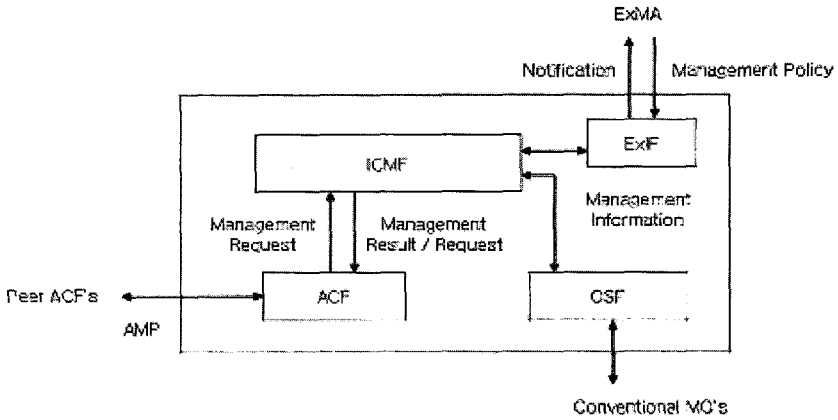
3.1 망 관리 구성 요소

3.1.1 BMA

능동 망 관리의 가장 기본이 되는 노드이다. 능동 패킷을 실행시킬 수 있는 환경(EE)을 갖추고 있으며, 망 관리 함수로 능동 코드를 사용한다. 기존의 망 관리 방법들과의 호환성을 유지하기 위해, BMA는 SNMP, CMIP 등의 다른 망 관리 함수들을 동시에 가지고 있을 수 있으며, 이를 통해 BMA가 속하는 도메인의 망 관리를 수행할 수 있다. BMA는 상호AA를 주고 받을 수 있으며, 이를 이용해 통신할 수 있다.

(가) IMA/EMA

IMA는 사용자가 서비스를 요구할 때, 경로 상에서 가



OSF : Operations System Function
ACF : AMP Communication Function
ICMF : Inter-domain Cooperative Management Function
ExIF : ExMA Interface Function

(그림 6) BMA 기능 블록

장 먼저 만나게 되는 BMA 를 말한다. IMA는 사용자의 보안, 관리 요구 정보를 가지고 EMA와 경로 상의 다른 BMA 혹은 ExMA들과 능동 패킷을 이용해 통신한다. EMA는 사용자가 서비스를 요구할 때, 경로 상에서 가장 마지막에 만나게 되는 BMA를 말한다.

(나) BMA 기능 구조

BMA의 논리적 기능 블록 구성은 (그림 6)과 같다.

- OSF : SNMP와 같은 일반적인 Manager-Agent 구조로 해당 도메인내의 망 관리 기능을 수행한다.
- ACF / ICF : 같은 도메인 내에 존재하는 BMA간에 AMP 교환을 통하여 Inter-domain Cooperative Management 응용을 수행하는 기능을 담당한다.
- ExIF(ExMA Interface Function) : BMA를 관리하는 상위 Management Agent인 ExMA와의 인터페이스를 통해 Event Notification을 발생시키고, ExMA로부터 관리 정책을 받아 해당 BMA의 관리 정책을 설정하는 기능을 수행한다.

(2) ExMA

BMA의 상위 관리 Agent 개념으로서 sub-BMA 들을 관리하는 기능을 수행한다. ExMA의 망 관리는 능동 패킷을 송수신함으로써 망 관리 기능을 수행한다. 기본적으로 ExMA는 BMA와 유사한 구조를 가지며, 이러한 ExMA들의 집합을 관리하는ExMA 가 존재할 수도 있다. 만약, 한 도메인 내에 여러 개의 BMA가 존재한다면, 이를 관리하는 하나 이상의 ExMA 가 존재할 수 있으며, 이 ExMA를 관리하는 ExMA가 별도로 존재할 수 있다. 즉, ExMA는 동종 혹은 이종망 간의 관리를 수행할 수 있다.

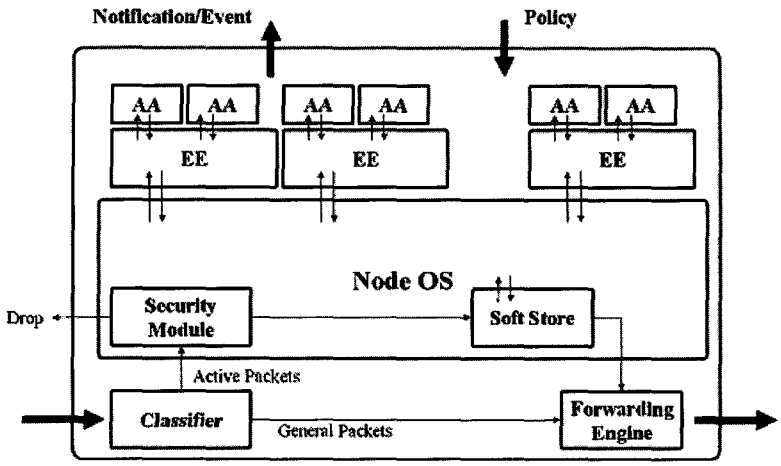
3.2. 능동 노드(BMA) 구성 요소

3.2.1 BMA 구조

BMA 구조는 (그림 7)과 같다. 능동 노드는 일반적인 라우터에서의 스위칭 역할이외에, 능동 패킷을 실행할 수 있는 환경을 지닌 노드를 말한다.

3.2.2 패킷 분류(Packet Classifier) 및 전송(Fast Forwarding)

능동 노드에서는 기존의 라우터와 같이 일반적인 패킷이 들어왔을 경우, 똑같은 역할(Fast Forwarding Engine)을 수행한다. 만약, 능동 패킷을 수신하였을 경우에는 노드 운영체제(혹은 인증 함수를 거쳐)로 패킷을 넘겨주게 된다.



(그림 7) BMA 노드 구조

3.2.3 NodeOS

능동 관리 패킷(AMP)이들어왔을 경우, 이를 실행할 수 있는 환경(EE)이 탑재되어 실행되는 운영체제를 말한다. 현재 능동 패킷에 사용되는 코드들은 Java, C, PLAN 등으로 다양하며, 이들을 수용하기 위해서는 각각의 실행 환경을 모두 수용할 수 있는 운영체제가 필요하다.

(가) Security Module

전송된 AMP가 이를 발생시킬 권한이 있는 노드에서 전송된 것인지 혹은 전송된 코드의 결함 유무를 High Level Security와Low Level Security로 나누어 판단한다. 이 기능 블록에서의 작업을 모두 마치게 되면, 인증된 패킷은 Soft Store로 전달되며 인증되지 않은 패킷은 버려진다.

(나) Soft Store

전송하는 AA의 크기나 데이터의 크기가 패킷 크기보다 클 경우, 분할되어 전송되어 온 코드나 데이터를 한 시적으로 저장하거나 재결합해 주는 기능을 수행한다. AA를 NodeOS 혹은 EE에 넘겨 준 이 후에는 전송되어온 패킷을 잠시 보관하다가 AA로부터 변경된 데이터를 받게 되면, 데이터와 함께Fast Forwarding Engine에 패킷을 넘겨준다.

3.2.4 EE (Execution Environment)

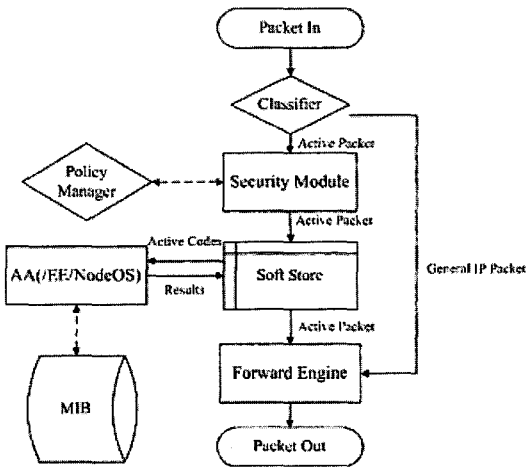
NodeOS 위에서 실행되는 프로세스로서 다양한 언어로 기술된 코드를 포함하는 능동 패킷을 실행할 수 있는 환경을 마련해 준다.

3.2.5 AA (Active Application)

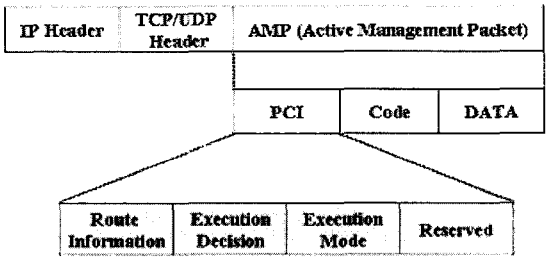
실제 실행되는 응용을 말한다. 이 AA를 사용하여 망관리 기능을 수행하게 된다.

3.3 AMP (Active Management Packet)

AMP은 망 관리를 목적으로 하는 능동 코드를 전달하거나 구동시키는 패킷을 말한다. 실제 능동 코드는 AMP



(그림 8) 능동 노드에서의 패킷 흐름



(그림 9) AMP 포맷

내에 존재할 수도 있고 BMA 내에 위치할 수도 있다. AMP는 패킷제어를 위한 정보와 능동 코드 및 코드 수행 데이터로 구성된다.

3.3.1 패킷 제어 정보 (PCI, Packet Control Information)

한 BMA에 도달한 AMP에 대해서 그 AMP가 지정한 능동 코드를 실행시킬 것인가 아니면 그대로 패스시킬 것인가를 판단한다. 또한 실행 조건과 다음에 보내질 목적지를 결정한다. 일반적인 패킷 제어 정보는 다음과 같다.

(가) 패킷 라우트 정보 : Source, Next, Destination, Condition-to-Return

Source, Destination, Condition-to-Return은 AMP를 생성한 Source가 지정하게 된다. 중간 노드에서는 다음 노드만을 결정하며, Condition to Return은 해당 AMP가 Source로 보내지기 위한 조건을 담고 있다.

(나) 실행 여부 판단 정보 (Execution Decision) : Priority, Frequency

모든 BMA(AA)는 동일한 Priority를 부여 받는다. BMA는 자신의 Priority와 같은 Priority를 갖는 AMP을 생성한다. 중간 노드에서는 자신의 Priority보다 낮은 Priority를 갖는 AMP는 무시한다. 실행 후 결과에 Disagree할 경우 해당 AMP의 Priority를 낮게 배정한다. Frequency는 동일한 AMP에 대하여 짧은 시간 내 반복 수행을 방지하기 위한 것이다.

(다) 실행 모드 정보 (Execution Mode) : Synchronous, Asynchronous

능동 코드 실행 결과를 기다릴 것인가 아니면 실행 완료 이전에 다음 노드로 전달될 것인가를 지정한다. 이 정보는 Source에 의해 정해진다.

3.3.2 능동 코드와데이터

능동 코드는 실제 EE에서 실행되는 코드이다. 능동 데이터는 능동 코드의 in/out/in-out 파라메타이다.

4. BMA를 통한 협업적 관리의 수행 예

4.1 보안 관리

4.1.1 목적 및 범위

기존의 패킷 교환 네트워크는 해킹과 같은 보안 공격에 많은 취약점을 가지고 있다. 침입차단시스템(Firewall)과 침입탐지시스템(Intrusion Detection System)같은 보안 시스템이 개발되고 있지만 DoS나 Probe 등을 비롯한 다양한 공격에 대해 적극적으로 대처할 수 없다. 결과 DARPA를 비롯한 여러 기관에서 전송중인 능동 패킷이 라우터에서 관리자의 정책을 담고 있는 코드를 실행할 수 있고, 그 코드의 실행결과에 따라 라우터의 상태를 변경할 수 있는 액티브 네트워크 전반적인 구조를 제안하였다. 하지만 액티브 네트워크에서 중요한 것은 기존 네트워크와 달리 능동 패킷이 능동 노드의 자원에 접근함으로써 발생하게 되는 네트워크 보안이다. 따라서 능동 노드의 NodeOs단에 Crypto Engine, Integrity Engine, Authentication Engine, Authorization Engine 등을 비롯한 능동 노드 인증 및 능동 패킷/코드 인증 보안 모듈을 둠으로써 능동 노드간 서로 안전한 협업적 관리를 할 수 있도록 한다. 즉 보안모듈에서는 능동 노드가 받은 능동 패킷이 올바른 능동 노드로부터 왔는지 인증하고, 능동 패킷과 코드가 오는 도중 변조되지 않았는지 무결성을 제공하며, 능동 패킷에 담고 있는 코드가 올바른지 확인하는 low-level 보안과 문제가 없을 경우 비로소 능동 코드가 실행되는 high-level 보안을 제안하여 능동 노드 간 안전한 통신을 가능케 하고자 한다.

4.1.2 필요성

액티브 네트워크는 기존의 네트워크에서 부족했던 유연성 측면에서는 많은 이점을 갖지만 그 대신 이렇게 패

킷에 담긴 능동 코드가 실행되기 위해 능동 노드의 자원에 접근해야 하기에 정당하지 못한 패킷이 능동 노드의 자원이나 시스템 자체에 악영향을 끼칠 수 있는 다음과 같은 보안상의 위협이 발생한다.

- 능동 코드에 의한 액티브 네트워크의 오용
- 다른 능동 코드에 의한 능동 코드의 오용
- 액티브 네트워크 노드에 의한 능동 코드의 오용
- 일반 네트워크 구조아래 능동 코드 혹은 실행 환경의 오용

따라서 하나의 도메인 내의 능동 노드들은 서로 협업적 보안관계를 통해 여러 가지 공격에 대응하여 능동 노드를 보호하는 능력이 필요하다. 또한 라우터 등의 중간 노드의 시스템에 사용자의 실행 가능한 코드가 직접 접근해서 그 노드의 기능을 무력화시킬 수도 있다는 점에서 액티브 네트워크의 보안은 기존의 네트워크 망보다 더욱 세심하게 고려되어야 한다.

4.1.3 보안 요구사항

(가) 인증

액티브 네트워크에서 인증은 능동 노드가 받은 능동 패킷이 올바른 능동 노드로부터 왔는지 확인하는 과정이다. 보통 인증은 요청을 원하는 개체와 요청을 받는 개체 간에 공개키/개인키 쌍과 공개키 인증서를 가져야 하는 공개키 기반 암호화 방식을 사용한다. 예를 들어, 사용자는 사용자의 개인키를 이용하여 능동 패킷에 디지털 서명을 하고 보낸다. 그러면 능동 패킷을 받은 능동 노드는 그것이 올바른 사용자로부터 왔는지 확인하기 위해, 즉 사용자의 공개키를 입증하기 위해 공개키 인증서를 사용한다. 만약 유효하면, 능동 노드는 패킷의 디지털 서명을 입증하기 위해 사용자의 공개키를 사용한다. 따라서 인증을 위해 공개키 암호화 방식과 PKI 기반 구조를 사용할 것이며, 그것은 Crypto Engine과 Authentication Engine의 상호작용으로 이루어진다.

(나) 안전한 정책을 기반으로 하는 권한부여

권한 부여는 능동 패킷에 대한 인증이 완료된 후 실제로 이 능동 노드에서 능동 패킷을 실행할 것인지 여부에 대한 권한을 점검하는 것이다. 먼저 패킷이 능동 노드에서의 실행을 위해 요구하는 자원에 대한 접근이 정당한 것인지 검증하고, 자원 사용에 대해 제한된 권한을 부여하게 된다. 각 능동 노드는 정책 데이터베이스 (Policy DB)를 갖고 있으며, 정책 데이터베이스는 그 능동 노드에서 실행 가능한 각 서비스 별 정책을 포함하고 있다. 해당 서비스의 정책에는 어떤 호스트와 사용자가 얼마만큼의 권한을 갖고 실행할 수 있는지 명시되어 있기 때문에 권한 부여 모듈은 패킷이 요청한 서비스에 해당하는 정책 데이터베이스를 참고하여 적절한 권한을 부여한다. 또한 능동 노드는 동시에 많은 사용자의 코드가 실행되어야 하므로 우선권 부여를 고려한 정책적 관리가 필요하게 된다.

권한 부여에서는 패킷의 권한 부여 외에 노드내의 정책 데이터베이스를 관리할 수 있다. 즉 정책 데이터베이스 내의 정책 정보들은 수정 혹은 삭제될 수 있으며, 새로운 서비스나 항목에 대한 정책이 정책 데이터베이스에 추가될 수도 있다.

(다) 능동 패킷 기밀성

기밀성은 전송되는 정보의 불법적인 노출을 방지하는 기술로 능동 패킷을 암호화하여 전송함으로써 제공할 수 있다. 송신지가 자신의 비밀키로 암호화하여 전송하면 수신 측에서 공유하고 있는 비밀키로 복호화 하거나, 송신지가 수신지의 공개키로 메시지를 암호화하여 전송하면 수신지가 자신의 개인키로 메시지를 복호화한다.

(라) 능동 패킷/코드 무결성

능동 패킷과 코드는 네트워크 전송 중에 악의 있는 사용자에 의해 수정 변조되거나 재전송될 수 있다. 따라서 능동 패킷을 받은 능동 노드는 능동 패킷과 코드의 무결성을 검증해야 한다. 일반적으로 능동 패킷은 전송 중 변하지 않는 정적인 부분과 전송 중 합법적으로 변할 수 있는 동적인 부분이 있기 때문에 능동 노드들 간에는 end-to-end 방식보다는 hop-by-hop 방식으로 무결성이 검증되어야 한다. Hop-by-hop 방식으로 검증되기 위해서 사용자는 공개키/개인키 쌍과 유효한 공개키 인증서를 가지고 있어야 하며, 각각의 능동 노드는 그의 이웃과 공유하는 비밀 키를 가지고 있어야 한다. 각각의 패킷은 정적인 부분에 대해 전자서명을 가지고 있어야 하며, 동적인 부분에 대해서는 메시지 인증코드(MAC: Message Authentication Code)를 가지고 있어야 한다, 또한 anti-replay를 위

해 timestamp나 혹은 특정 값이 있어야 한다. 즉, 무결성은 패킷을 받은 능동 노드가 MAC과 전자서명 및 anti-replay를 확인하는 것이다. 만약, 능동 노드에서 패킷 무결성 확인 후 동적인 부분을 수정해야 할 일이 생긴다면 수정 후 결과로 나온 새로운 MAC 값을 다음 홉으로 보내면 된다. 이것은 Integrity Engine과Security Facilities, Authentication Engine, Crypto Engine의 상호작용으로 이루어진다.

(마) 코드 입증

코드 입증이란 추가적인 보안 기능으로, 코드를 수행하기 전 신뢰성 없이 새롭게 도착한 능동 코드의 수행을 막고자 하는 것이다. 만약 코드 입증이 실패면 그것은 코드는 그것에 대한 권한부여 및 우선권도 역시 신뢰할 수 없기에 실행될 필요가 없다는 것을 의미하므로 버려지고, 성공이면 권한 부여 엔진으로 가게 된다.

4.1.4 보안 구조 범위

보안구조의 범위는 하나의 도메인 내에서의 능동 노드들 간의 협업적 관리이며, 2- level security framework을 제공한다.

(가) Low-Level Security Operation

능동 코드가 실행되기 전 거치게 되는 보안 단계로 인증 및 능동 패킷/코드의 무결성을 확인한다.

(나) High-Level Security Operation

Low-Level Security Operation에서 패킷의 무결성이 보장된 이후 실제 능동 코드가 수행되는 보안 단계이다.

4.1.5 Security Module 보안 구조

(가) Crypto Engine

대칭형 암호화/복호화, 비대칭 암호화/복호화, 해쉬 함수와 같은 다양한 암호화 알고리즘을 수행하는 엔진으로 기타 다른 보안 엔진에서 사용자를 인증하거나 능동 패킷/코드의 무결성 검증 시 사용된다.

(나) Integrity Engine

Crypto Engine 상의 암호화 알고리즘을 이용하여 능동 패킷과 능동 코드의 무결성을 확인하는 엔진이다.

(다) Verification Engine

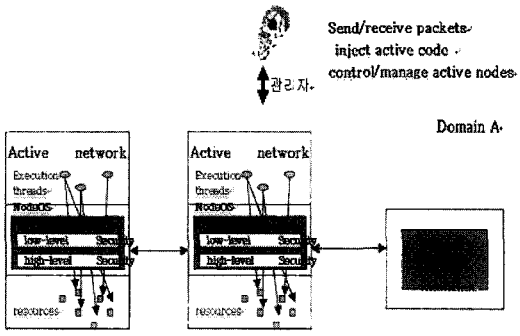
Crypto Engine 상의 암호화 알고리즘을 이용하여 코드를 수행하기 전 신뢰성 없이 새롭게 도착한 능동 코드의 수행을 막는 엔진이다.

(라) Authentication Engine

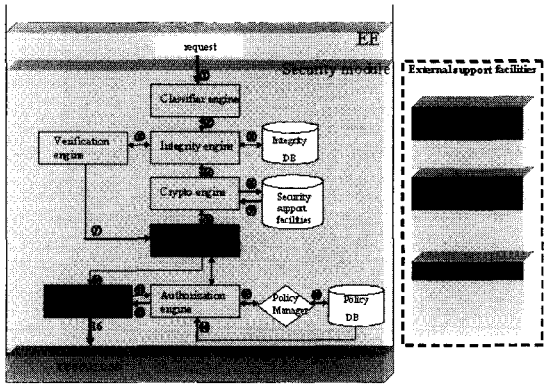
Crypto Engine 상의 암호화 알고리즘을 이용하여 능동 노드가 받은 능동 패킷이 올바른 능동 노드로부터 왔는지 확인하는 엔진이다.

(마) Authorization Engine

권한 부여는 능동 패킷에 대한 인증이 완료된 후 실제로 이 능동 노드에서 능동 패킷에 대한 수행 여부의 권한을 체크하고 부여하는 엔진이다.



(그림 10) 협업적 관리를 위한 보안구조



(그림 11) Security Module 보안구조

(바) Policy DB

능동 노드 안에서 누가 무엇을 할 수 있는지에 관한 보안 정책을 담고 있는 데이터베이스이다.

(사) Policy Manager

Authorization Engine 에 의해 요청되면 요청된 서비스와 관련된 보안 정책들을 정책데이터베이스에서 찾고 그 결과를 돌려주는 매니저다. 또한 노드내의 정책 데이터베이스에 새로운 서비스나 항목에 대한 정책을 추가, 수정 삭제 할 수 있도록 해준다.

5. 결 론

본 논문에서는 액티브 네트워크에서 하나의 도메인을 담당하는 능동 노드들간의 협업적 망 관리 구조와 보안관리, QoS 관리 및 DDoS 공격 대응 관리에 대해서 기술하였다. 향후 BcN과 같은 다양한 서비스가 제공되고, 다양한 도메인들이 혼재하게 되는 상황에서 망 관리를 수행하기 위해서는 기존의 망 관리 구조보다는 유연하고 확장성이 제공되는 망 관리 구조가 필요하며, 액티브 네트워크 기술을 이용한 망 관리 구조가 적절할 것이다. 따라서 본 연구에서는 액티브 네트워크 관리를 위한 계층적인 망 관리 구조와 해당 구조를 구성하는 각각의 요소에 대해서 그 기능과 역할을 기술하였다. BMA가 도메인을 관리하는 주체로서 제시되었으며, BMA를 구성하는 기능 블럭 및 요소들을 제시하였다. 또한 이들 BMA간에 액티브 네트워크 기술을 이용한 보안 관리, QoS 관리 및 DDoS 관리에 방안을 제시하였다. 제안하는 망 관리 구조의 범위는 본 연구에서는 동종망으로 구성되는 하나의 도메인에 대한 협업적 망 관리 구조만이 제시되었으나, 이기종으로 구성되는 다수의 도메인에 대한 협업적 망 관리 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] 액티브 네트워크 기술 동향, 이수형, 남택용, 나중찬, 손승원, 주간기술동향 996호, 한국전자통신연구원, 2001.

[2] 3GPP TS 32.102 v6.3.0 (2004-06) TM Architecture.

[3] IEEE 1520 Standards, Initiative for Programmable Network Interfaces.

[4] D. S. Alexander, W. A. Arbaugh, A. D. Keromytis, J. M. Smith, "A Secure Active NetworkEnvironment Architecture: Realization in SwitchWare," IEEE Network Magazine, special issue on Active and Programmable Networks, 12(3), 1998.

[5] R. H. Campbell, Z. Liu, M. D. Mickunas, P. Naldurg, S. Yi, "Seraphim: An Active Security Architecture for ActiveNetworks," IEEE OPENARCH 2000, Tel-Aviv, Israel, 2000.

[6] M. Hicks, "PLAN System Security," July 1998.

[7] A. Galis, B. Plattner, J. M.Smith, S. Denazis, E. Moeller, H. Guo, C. Klein, J. Serrat, J. Laarhuis, G. T. Karetsos and C. Todd "A Flexible IP Active Networks Architecture," IWAN 2000 Conference, November 2000.

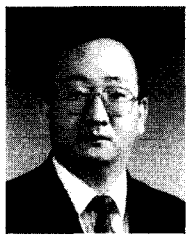
◐ 저 자 소 개 ◑

오 행 석

1981년 : 한양대학교 공과대학 졸업(학사)
1983년 : 한양대학교 대학원 졸업(석사)
1997년 : 충북대학교 대학원 졸업(박사)
1983년~현재 : 한국전자통신연구원 정보보호연구단 개인정보보호연구팀 책임연구원
관심분야 : 컴퓨터 네트워크, 소프트웨어공학 etc.
E-mail : hsohs@etri.re.kr

남 택 용

1987년 : 충남대학교 계산통계학과 졸업(학사)
1990년 : 충남대학교 계산통계학과 졸업(석사)
2005년 : 한국외국어대학교 전자정보공학과 졸업(박사)
1987년~현재 : 한국전자통신연구원 정보보호연구단 개인정보보호연구팀 팀장(책임연구원)
관심분야 : 정보보호, 인터넷, 차세대네트워크 구조 etc.
E-mail : tynam@etri.re.kr



채 기 준

1982년 : 연세대학교 수학과 졸업(학사)
1984년 : 미국 Syracuse University 컴퓨터학과 (이학석사)
1990년 : 미국 North Carolina State University 컴퓨터공학과 (공학박사)
1990년~1992년 : 미국 해군사관학교 컴퓨터학과 조교수
1992년~현재 : 이화여자대학교 컴퓨터학과 교수
관심분야 : 네트워크 보안, 인터넷/무선통신망/고속통신망 프로토콜 설계 및 성능분석
E-mail : kjchae@ewha.ac.kr