主題

에이전트 기반 액티브 네트워킹 기술

대구가톨릭대학교 컴퓨터정보통신공학부 교수 전 용 희
대구가톨릭대학교 대학원 박사과정 장 정 숙
한국전자통신연구원 네트워크보안연구부 팀장 장 종 수
한국전자통신연구원 네트워크보안연구부 부장 손 승 원

차 례

1. 서론

네트워킹과 컴퓨팅의 폭발적인 증가로 복잡한 네트워크 서비스의 도입에 대한 필요성이 빠르게 발생하고 있다. 차세대 네트워크에서는 웹, 멀티미디어, 텔넷 같은 다양한 응용과 상용의 이질 유무선 네트워크 환경, 그리고 각각 다른 서비스 품질(QoS: quality of service)을 요구하는 작업 부하를 지원하는 것이 요구되며, 사이버 공격의 발생 빈도 또한 점점 증가할 것으로 예상된다.

현재의 네트워크는 통합 네트워크 보안에 대한 문제점으로 만족스러운 서비스를 받을 수 없으므로, 기존의 네트워크의 한계점을 극복하기 위하여 새로운 대안으로 나타난 것이 액티브 네트워크(AN: Active Network)이다. 이것은 동적이며 프로그램 가능한 네트워크구조에서 컴퓨팅과 텔레커뮤니케이션과의 관점의 조정이 요구되며, 네트워크 인식(aware) 응용과 응용 인식 네트워크의 결합을 통하여 신속한 서비스 생성을 지원하게 된다.

현재 네트워크 보안 하부구조에서는 지역적인 보안 관리 구조 체제, 보안 시스템 운용에 있어 상호 연동 및 협력 작업에 대한 기능의 결여 그리고 보안 환경 변화에 대한 감지 및 대응 기능이 부족하다. 액티브 네트워크 보안 기술을 통하여 보안 관리 영역간의 상호 결합적 협력 기능과 새로운 보안 메커니즘을 동적으로 수용할 수 있는 구조 그리고 침입자 파악 및 고립화 기술을 통하여 확장성, 가용성 그리고 유연성이 뛰어난 전역적인 네트워크 보안 관리가 가능 할 것이다. 액티브 네트워크 기술은 프로그래밍 된 보안 메커니즘이 네트워크를 통해 액티브 노드 간을 이동하며 실행되고 동적으로 복제, 소멸되는 기능을 제공함으로써 기존의 네트워크 상에서 자동적

이고 지능화된 보안 응용 서비스를 가능하게 하는 기술이다.

액티브 네트워크와 관련된 연구는 미국방부 DARPA(DoD Advanced Research Projects Agency)와 유럽의 IST(Information Society Technologies Program)에서 추진하고 있으며 액티브 네트워크를 응용한 네트워크 보안 기술도 해당 두 기관에 의해 그 연구가 주도되고 있다고 볼 수 있다. 두 기관에서 수행중인 연구는 공격자의 트래픽을 차단만하는 것이 아니라 공격자의 실제 위치를 역추적하고 공격자의 네트워크에 대한 접근을 차단함으로써 고립화시키는 능동적인 대응을 수행할 수 있는 네트워크 보안 프레임워크를 구축하는 것을 목적으로 하고 있다. 미국 DARPA의 IDIP(Intrusion Detection & Isolation Protocol), IDIP 기반 구조에 보안 정책을 포함시킨 CITRA(Cooperative Intrusion Traceback and Response Architecture), 그리고 프로토콜로 구현될 경우의 IDIP와 CITRA을 보완한 AN-IDR (Active Network Intrusion Detection and Response) 과제가 그러하다. 유럽의 경우 자신들이 개발 중인 액티브 네트워크 프레임워크 상에 이동 에이전트를 상정한 플랫폼을 기반으로 하여 DDoS(Distributed Denial of Service) 공격을 탐지하고 해당 트래픽을 차단하기 위한 연구가 수행 중에 있다.

사용자 요구 기능을 수행할 수 있는 액티브 네트워크는 네트워크에 새로운 서비스를 보다 신속하고 경제적으로, 자원들을 보다 적절하게 활용 할 수 있도록 하는 차세대 네트워크 플랫폼이다. 현재 액티브 네트워크는 분산 시스템에서의 처리가 효율적인 소프트웨어 에이전트 기반 액티브 네트워크 연구가 활발히 이루어지고 있다. 따라서 본 논문에서는 에이전트 기반 액티브 네트워킹 기술에 대하여 고찰 및 분석하고자 한다.

2. 관련 연구

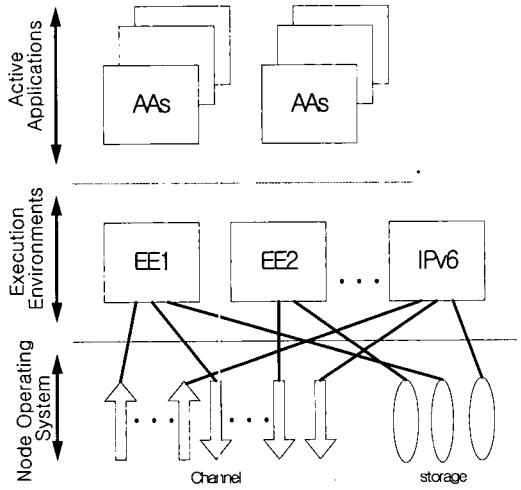
2.1 액티브 네트워크 기술

액티브 네트워크는 패킷에 프로그램 코드와 데이터를 함께 넣어 전송하고 네트워크 노드인 스위치나 라우터에서 이 패킷에 들어 있는 프로그램 코드를 처리 할 수 있는 환경을 가진 네트워크를 말한다. 현재 액티브 네트워크의 구현된 응용 적용과 테스트 실행 환경으로는 ABONE (Active Network Backbone)이 있다. 액티브 네트워크는 중간노드 즉 네트워크 노드에서 경로설정, 전달기능, 그리고 에러처리 및 흐름 처리뿐만 아니라 프로그래밍을 가능하게 함으로써 유연성과 확장성을 제공하는 일종의 지능망이라고 할 수 있다. 네트워크 노드로 프로그램을 주사하여 응용을 허락하므로 유연하고 새로운 네트워크 서비스 개발을 쉽게 그리고 빠르게 지원하여 많은 기능과 성능을 제공한다. 기존의 네트워크는 패킷의 지연과 처리량을 측정하여 네트워크의 성능을 평가하였다. 액티브 네트워크의 성능은 네트워크의 성능이라기보다는 응용의 성능과 관련된다. 액티브 네트워크는 더 적은 수의 패킷을 보내고 혹은 전달하며, 더 긴 지연을 경험하는 운용을 수행하여, 처리력과 개별 패킷 지연 측면에서는 성능 저하처럼 나타난다. 그러나 종단에서 대역폭에 대한 요구가 감소하고 네트워크 폭주가 감소하여 응용에 대한 성능을 사실상 증가시킨다. 그러므로 성능은 특정 응용 척도에 의하여 평가되어야 한다. 액티브 네트워크는 네트워크 노드에서 컴퓨터 작업의 증가와 저장 자원의 추가적인 비용이 부가된다[1-4]. 액티브 네트워크를 가능하게 만들기 위해서는 구조와 구현, 인터페이스 그리고 보안을 고려해야 한다.

가. 구조

액티브 네트워크의 구조는 새로운 서비스와

성능 목표 그리고 속성의 결합으로 재구조화하고 동시에 다른 네트워크 계층 프로토콜을 취급할 수 있어야 하며 새로운 프로토콜을 구성하고 온라인상의 새로운 프로토콜의 동적인 개발과 네트워크의 확장성을 향상시켜야 한다. 그림 1은 액티브 네트워크의 구성요소를 보여준다.

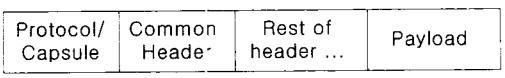


<그림 1> 액티브 네트워크의 구성요소

액티브 네트워크 구현에서는 캡슐과 액티브 노드 그리고 코드 분산 메커니즘을 통하여 구현된다[5].

1) 캡슐(capsule)

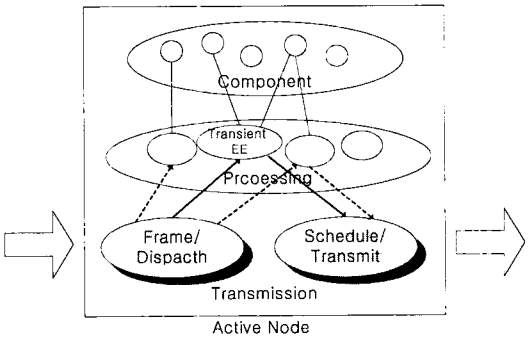
액티브 네트워크에서 캡슐은 기존 네트워크의 패킷을 말하며 데이지를 전달한다. 네트워크에서 정보를 공유하는 캡슐들의 모임을 프로토콜이라 한다. 프로토콜은 서비스 형태와 요구 특성화 그리고 보호 장치를 제공한다. 그림 2는 캡슐 형태를 보여준다.



<그림 2> 캡슐 형태

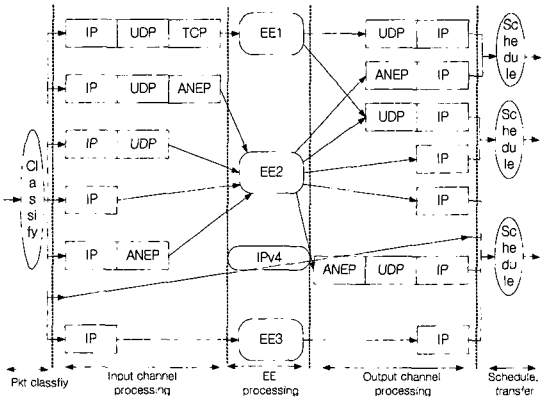
2) 액티브 노드

기존 인터넷에서 라우터들은 액티브 네트워크에서 액티브 노드로 대치되며, 종단 노드에서 들어오는 프로토콜의 캡슐을 실행하고 프로토콜의 상태를 유지한다. 일반적인 라우터와는 달리 액티브 노드는 캡슐 처리 루틴을 위한 API를 제공하고 이들 루틴은 운영체제와 언어기술을 사용하여 안전하게 실행 한다. 액티브 노드는 액티브 네트워크의 사용자들을 위한 가상머신과 프로그램 인터페이스를 제공한다(그림 3참조).



<그림 3> 액티브 노드

각 액티브 노드들은 네트워크 운영체제(NOS: Network Operating System)와 하나 이상의 실행환경(EE: Execution Environment)을 수행하며 액티브 노드가 하는 일은 프로토콜 호스트와 소프트 저장소 그리고 캡슐을 제공하고 자원의 소비를 제한하며 코드를 분산한다. NOS는 노드의 자원을 할당하고 스케줄링하며 실행환경은 노드에 액티브 패킷(캡슐)을 번역하여 노드의 가상머신을 구현한다.



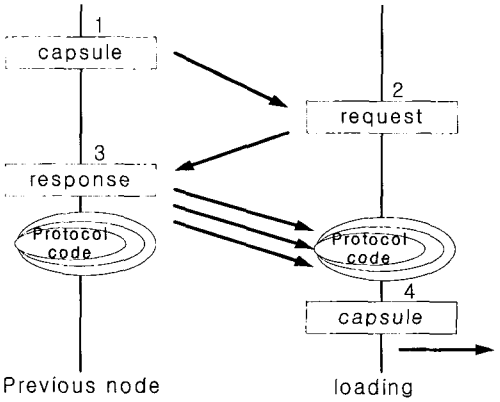
<그림 4> 액티브 노드의 패킷 처리 흐름도

액티브 노드에서 실행환경은 캡슐 수신 후 헤더를 검사하여 패킷을 분류하고 입력 채널을 결정한다. 입력 캡슐의 분류는 해당 실행환경(EE)이 표현하는 양식에 의해 조정된다. 캡슐 처리 단계는 캡슐 입력, 분류, 입력 프로토콜 처리, 실행환경/액티브 응용 처리, 출력 프로토콜 처리, 스케쥴링 그리고 전송단계이다(그림 4참조).

액티브 네트워크 캡슐화 프로토콜(ANEP: Active Network Encapsulation Protocol)은 사용자가 특정한 실행 환경(EE)에 캡슐을 전달하기 위해서 여러 가지 전달 수단이 필요할 때 사용한다. ANEP 헤더는 특정한 실행환경에 할당되는 타입 식별자 필드를 포함하여 특정환경으로 전달 가능하도록 한다. 전달과정에서는 인증과 기밀성 그리고 무결성을 지원하는 옵션을 사용 할 수 있다[6].

3). 코드 분산(Code Distribution)

캡슐은 프로그램(program)을 운반하는 것이 아니고 프로그램의 기술적인 묘사를 운반한다. 코드 분산 메커니즘은 요구 적재 프로토콜(Demand Loading Protocol)을 사용하여 코드를 분산한다[7].



<그림 5> 요구 적재 프로토콜

요구 적재 프로토콜은 캡슐이 노드에 도착하면 프로토콜 코드를 검사하여 저장한다 (그림 5 참조). 만약 요구하는 코드가 실행환경에 존재하지 않는다면 전 노드에게 필요한 코드 프로토콜을 요청하고 정해진 시간동안 프로토콜을 기다려 수신한다. 그리고 실행환경에서 필요로 하는 프로토콜이 통합되면 캡슐을 마지막 액티브 노드로 보내며, 이 처리는 즉시 수행된다. 로드 대응을 수신할 때 캡슐 실행은 중지한 후 유한 시간(TTL)을 기다린다. 시간을 초과하는 경우에 캡슐은 버려진다[8].

나. 구현 방식

프로그램을 로딩 및 처리하는 방식으로 현재 세 가지의 접근방식이 사용되고 있다. 프로그램 로딩과 프로세싱을 분리하여 접근하는 분리형 접근과 캡슐로 통합하는 통합형 접근 그리고 분리형과 통합형의 절충 형태인 복합형으로의 접근 방식이 있다.

1) 분리 접근 방법

기존의 라우터나 스위치인 네트워크 노드를 프로그램 가능하게 만드는 것이다. 프로그램들이 이미 네트워크 노드에 상주되어 있는 경우로써

노드에서 수행될 프로그램 식별자와 데이터를 저장한 캡슐을 전송한다. 네트워크 노드는 프로그램 식별자에 해당하는 프로그램을 적재하여 데이터를 처리한다. 이 방법은 이식된 프로그램에 대하여는 적응이 가능하지만 새로운 프로그램 추가는 불가능하다 이 접근 방법의 개발 사례로는 ANTS[9], CANES[10], DAN[11] 등이 있다.

2) 통합형 접근 방법(캡슐형 접근)

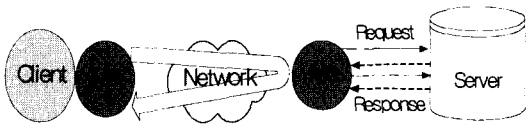
캡슐에 실제 수행될 프로그램과 데이터를 통합하여 함께 전송한다. 스위치나 라우터인 네트워크 노드에서는 프로그램을 소유하고 있는 않으며 각각 캡슐이 프로그램과 데이터를 소유한다. 네트워크 노드는 전송 받은 프로그램과 데이터를 자신의 실행환경에서 수행하여 실행 결과를 전송한다. 프로그램과 데이터의 전달 양이 많은 경우에는 트래픽 문제, 패킷 손실 시 재전송 같은 효율성을 저하시킬 수 있다. 이 접근 방법의 개발 사례로는 IP Option[12], ANEP[13], Smart Packet[14] 등이 있다.

3) 복합형 접근 방법

캡슐 전달에서 발생하는 지연 혹은 분실 같은 비효율성을 제거하기 위해 두 가지 방법을 융합하는 복합형 접근 방법이 있다. 공통적으로 사용하는 프로그램은 네트워크 노드에 이식하고 특정 프로그램은 각 노드가 프로그램과 데이터를 캡슐로 전송하는 것이다. 이 접근 방법의 개발 사례로는 SwitchWare[15], NetScript[16] 등이 있다.

2.2 에이전트 기술

에이전트는 그들의 목표에 따라서 자율적으로 실행하고 사용자를 대신하여 행동한다. 소프트웨어 에이전트는 다음과 같은 특성을 가진다[17]: 자율성(Autonomy), 협동성(Cooperativity), 반응성(Reactivity), 대처성(Proactivity). 이동 에이전트는 사용자 혹은 다른 프로그램을 소유자의 제어 하에 소유자를 대신하여 실행하는 액티브 프로그램이다. 다시 말하면 이동 에이전트는 네트워크에서 이동할 때, 이동 장소를 선택하고 이동 실행 후에도 자율적으로 연속적인 진행 방법을 결정한다. 이동 에이전트는 모든 이동 코드 시스템의 장점을 상속한다. 특별히 이것이 설치되기 전이 아니면 기능이 자동적으로 노드로 전송이 가능하다. 전통적인 이동 코드 시스템과 이동 에이전트 사이의 차이는 데이터 상태와 실행 상태 같은 상태 포함에 의존한다. 이 상태에서 전통적인 이동 코드 개체는 상태의 전송이 가능하지 않기 때문에 이동에이전트 하에서 전송되고 캡슐화 된다. 이런 특성 때문에 자율적인 동작의 속성을 가지는 이동 에이전트를 필요로 한다. 이동 에이전트는 실행하는 동안 하나의 호스트에서 또 다른 호스트까지 이동하여 실행이 가능하다. 이동 에이전트는 그림 6처럼 코드와 함께 실행하는 개체를 대상으로 이동하여 처리한 후 결과를 반환한다[18].



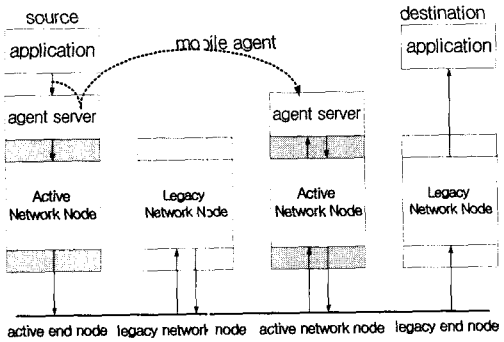
<그림 6> 이동 에이전트 통신 모델

이동 에이전트 사용의 장점으로는 프로토콜 캡슐화, 네트워크 트래픽 감소, 비동기적인 실행과 자율적 실행, 동적인 적응력, 이기종의 시스템 통합 그리고 강건성과 결점 적응을 들 수 있다. 이동 에이전트의 응용은 상업적인 분야에서 뿐만 아니라 교육 분야까지 확장되고 있다.. 에이전트 이동성은 네트워크 내 모든 컴퓨팅 요소의 처리력을 향상시키고 그리고 강력한 컴퓨팅 환경을 생성한다. 에이전트의 이동성은 네트워크 연결성, 보안, 자원 가용성에 대한 몇 가지 문제점을 가

지고 있지만 네트워크 컴퓨팅의 미래에 큰 영향을 미칠 것이다. 이동 에이전트 기술은 확장성 문제를 포함하여, 가난한 성능, 충분하지 않는 대역폭 이용 같은 클라이언트 서버 모델로부터의 단점을 극복했다. 이동 에이전트 내 분산 응용은 특정한 응용의 프로토콜을 필요로 하지 않는다. 이동 에이전트 기술은 플랫폼-독립적이기 때문이다. 이동 에이전트 시스템(MAS: Mobile Agent System) 구성에는 프로그램 언어(PL: Programming Language), 실행시간 환경(RE: Runtime Environment), 이동 에이전트 코드를 실행할 수 있는 인터프리터(MAI: Mobile Agent Interpreter), 이동에이전트 간의 통신을 지원하는 프로토콜(MAP: Mobile Agent Protocol) 그리고 통신 하부구조(CI: Communication Infrastructure)로 구성된다[19].

3. 에이전트 기반 액티브 네트워크

많은 액티브 네트워크 구조들은 현재 코드 이동성 패러다임을 사용한다. 이동 코드 기술은 캡슐화, 전송, 안전성, 효율적인 실행 그리고 프로그램의 중재를 지원한다. 액티브 네트워크 기술과 이동 코드의 기술은 최근 네트워크 통제와 관리 시스템을 향상시킬 뿐 만 아니라 분산 응용까지 현저하게 향상시킨다. 더욱이 고정된 기능과 프로토콜 엔진은 이동 에이전트를 통하여 기본 스테이션과 종단 시스템, 라우터와 스위치에서 동적으로 개발된다. 운영체제하에서 평균 수행시간, 이동성과 안전성 그리고 효율성에 관련된 문제들은 프로그래밍 언어를 통하여 해결되어진다. 이러한 문제들은 액티브 네트워킹에서 새로운 차원으로 개발되고 있다. 그러므로 현재 제공된 많은 액티브 네트워크 하부구조는 이동 코드 기술을 사용한다. 액티브 네트워크의 전개와 유지, 서비스의 주문화, 그리고 프로토콜 캡슐화에 있어 코드의 이동성 적용은 더욱 더 적절하다. 이동 코드 기술은 이동 소프트웨어 에이전트 기술과 매우 밀접하다. 코드 이동성 패러다임은 이동 소프트웨어 에이전트 기술을 수행한다. 그러나 두 기술의 기본적인 차이로서, 액티브 네트워크 기술은 통신을 편리하게 하는 유연성을 제공하며 주문하는 네트워크를 동적으로 재구성하는 네트워크 계층 처리의 개념을 사용하며, 이동 에이전트 기술은 다중 에이전트를 이용하여 사용자와의 친숙한 환경과 사용자의 구체적인 작업을 수행하기위해 장소에서 장소로 이주하는 프로그램의 개체로서 응용 계층에서 응용 프로그램으로서 실행이 가능하다. 잠재적으로 네트워크 울안에서부터 궁극적으로는 유비쿼터스(ubiquitous) 컴퓨팅 환경의 조성까지, 일반적인 이동 에이전트 기술은 에이전트 이주 컴퓨터작업(computation)을 지원하도록 디자인 되었다. 액티브 네트워크의 기본적인 기능이 이동 소프트웨어 에이전트의 응용으로 지원 가능하다. 이동에이전트를 캡슐(액티브 패킷)의 특정형태처럼 볼 수 있기 때문이다. 그림 7은 이동에이전트 기반 액티브 네트워크의 개념도이다. 이동 에이전트 기반 액티브 네트워크 구현방법으로는 분리 접근방법과 통합 접근 방법, 두 접근 방법 모두를 지원하며 자율성과 이동성의 에이전트 기술도 함께 사용한다. 이동 에이전트는 코드 전송을 하는 약한 이동성(weak mobility)과 원격 코드 복제를 하는 강한 이동성(strong mobility) 둘 다를 지원한다. 또한 이동 에이전트 모니터링 메커니즘은 불안전한 상태에서도 생존 가능해야 한다. 액티브 네트워크 구조는 새로운 서비스 도입에 유연하므로 이동 에이전트의 네트워크 응용과 새로운 서비스 도입이 가능하다. 본 절에서는 액티브 네트워크 기술과 이동 에이전트 기술을 통합하여 이동 에이전트를 기반으로 하는 액티브 네트워크에 대하여 분석한다[20].

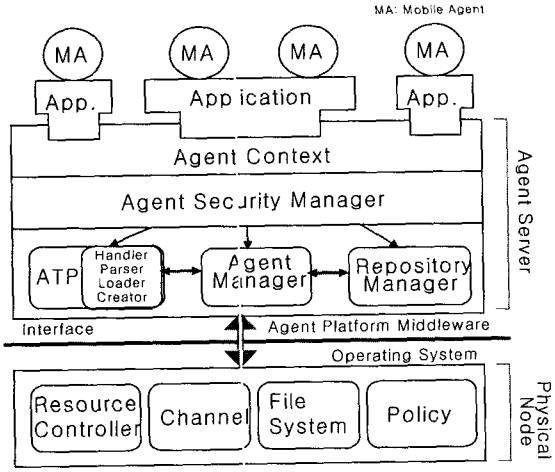


<그림 7> 이동 에이전트 기반 액티브 네트워크

3.1 구성

(1) 액티브 노드

네트워크를 프로그래밍 가능하도록 하는 것이 액티브 네트워크의 궁극적인 목표이다. 그림 8처럼 액티브 노드의 구조는 에이전트 플랫폼(에이전트 서버)과 노드 운영체제(물리적인 노드), 두 가지 논리적인 계층으로 나누어진다.



<그림 8> 에이전트 기반 액티브 노드의 실행환경

노드 운영체제는 자원의 관리와 네트워크 통신으로부터 에이전트 플랫폼을 유지하고 에이전트 플랫폼은 노드 운영체제와의 상호작용으로부터 액티브 에이전트를 보호한다. 에이전트 플랫폼은 액티브 노드의 응용 계층으로 액티브 노드와 종단 시스템에서 제공한다. 에이전트 플랫폼은 액티브 에이전트와의 접촉으로 통신, 평가와 컴퓨터작업 그리고 네트워크 프로토콜과 응용의 동적인 개발 그리고 동적인 연결, 나아가 액티브 노드 맞춤형의 동적인 전개를 편리하게 한다. 에이전트 플랫폼은 악용 에이전트 행위, 서비스 거부 공격에 대비하여 안전과 보안을 위해서 필터링 기능은 필수적 사항이다. 에이전트 플랫폼 자체에서도 다양한 액티브 에이전트를 고려해서 자원 소비, 정책, 그리고 보호(protection) 같은 각 스케줄링 메커니즘을 구현가능하다. 에이전트 플랫폼은 컴퓨터작업, 저장소 그리고 통신이용에서 하위 계층 액티브 노드로 접근하며 에이전트 플랫폼은 소프트웨어 추상 인터페이스를 통하여 지역적인 노드 자원들의 간접적 조작이 가능하다. 그러므로 노드 운영체제는 초기의 스케줄링 메커니즘에 의해 이용 가능한 자원을 안정적으로 통제가능하다[21,22].

- 에이전트 서버

에이전트 서버는 에이전트 문맥(agent context), 에이전트 보안 관리자(agent security manager), ATP(agent transfer protocol), 에이전트 관리자(agent manager) 그리고 저장소 관리자(repository manager)로 구성된다. ATP 핸들러는 ATP 형식으로 액티브 메시지를 이동 에이전트로 포장하여 전송하며 에이전트 관리자는 모든 이동 에이전트들이 이 노드에서 전개되는 프로토콜에 대한 관리를 기록한다. 그림 9는 이동 에이전트의 캡슐화를 보여준다. 그리고 다음 과정은 액티브 노드들이 이동 에이전트를 처리하는 과정이다[20].

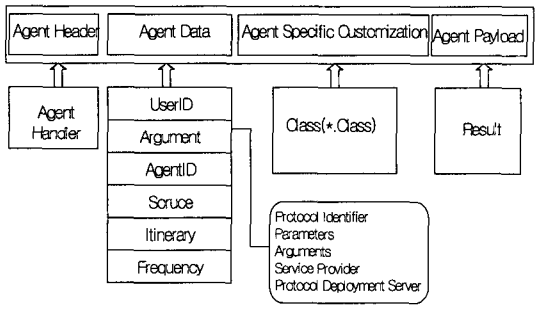
i. ATP 핸들러는 네트워크에 들어오는 메시

지에서 ATP 메시지를 구별한다.

ii. ATP 파서는 ATP 메시지를 분석한다.

iii. ATP 로더는 에이전트 실행 문맥을 초기화하고 이동 에이전트에 적재한다.

iv. ATP 생성기가 명세에 따라 이동 에이전트로 캡슐화한다.



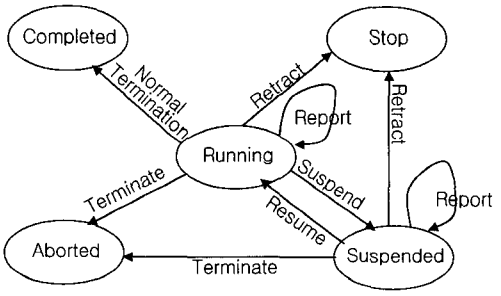
<그림 9> 이동 에이전트 캡슐화

(2) 이동 에이전트

액티브 네트워킹을 위해서 액티브 네트워크 하부구조는 이동 에이전트 기술을 유도하고 에이전트 플랫폼을 구축한다. 액티브 에이전트는 이동 에이전트를 의미하며 액티브 노드 사이에서 액티브 개체 송신에 대한 묘사에 사용된다. "active"의 용어는 전통적인 "passive" 네트워크 패킷과는 현저히 다르게 사용되며 "active agent"는 "capsule" 혹은 "active packet"보다는 더욱 더 적절한 용어이다. 네트워크 행위는 액티브 에이전트에 의해 동적으로 수정되고 맞추어(customized) 진다. 액티브 에이전트는 특정한 응용서비스를 전개하는 액티브 네트워크 요소로서 특정한 사용자 프로그램 즉 내용을 운반하여 주사(inject) 가능하다. 네트워크 요소는 액티브 에이전트가 요구하는 작업을 미리 정의된 명세와 예약에 의해 경로를 따라 순차적으로 수행가능하다. 액티브 에이전트는 이동 에이전트로부터 상속성 그리고 액티브 네트워킹을 위한 캡슐화와 액티브 패킷 같은 전달수단으로 두 가지 특성을 가진다.

(3) 에이전트 통신 프로토콜

에이전트 통신 프로토콜(Agent Transfer Protocol)은 액티브 노드 사이 에이전트 전송을 위해서 필수적이다. 이동 에이전트는 다양한 플랫폼에서 적용 가능하고 액티브 노드들의 번역과 실행이 가능한 한 다른 프로그래밍 언어에서도 작성되어 진다. 하지만 이동 에이전트 통신 프로토콜은 에이전트 이동성에 대해서는 균등해야 한다. 에이전트 통신 프로토콜은 유비쿼터스(ubiquitous) 컴퓨팅 능력을 가져야 한다[20].



<그림 10> 이동 에이전트 라이프 사이클

그림 10은 에이전트 통신 프로토콜의 라이프 사이클을 보여준다. 액티브 노드에 이동에이전트가 도달하여 다섯 가지 상태의 라이프 사이클을 나타낼 수 있다: 실행(running), 일시중지(suspended), 정지(stopped), 취소(aborted) 그리고 완결(completed).

4. 프로그래밍 기술

현재까지 개발된 이동 에이전트 플랫폼에서 에이전트 언어는 C, C++, 자바, Tcl/TK, Telescript 등의 모든 스크립트 언어로 개발이 가능하다. 현재는 자바 언어를 기반으로 하는 이동에이전트 플랫폼이 많이 개발되고 있다. 그 이

유는 자바 언어가 다음과 같은 특성을 가지고 있기 때문이다:

- 객체 지향적이다.
- 분산 환경에 적합한 네트워크 언어이다.
- 인터프리터언어로 바이트 코드가 자바환경에서 동작하므로 이식이 양호하다.
- 보안을 고려하여 설계되었다.
- 멀티 쓰레드(Multi-thread)를 지원한다.

4.1 자바기반과 비-자바기반 플랫폼

표 1은 현재 이동 에이전트를 이용하는 응용에 주로 사용하는 이동 에이전트 언어를 자바 언어 기반과 비-자바 언어들로 분류하고, 각각의 이동 에이전트 플랫폼들의 종류를 나타낸다(표 1 참조).

<표 1>. 언어에 따른 이동 에이전트 플랫폼

Java-based Mobile Agent Platform	**Non-Java** Mobile Agent Platform
• Aglets Workbench (IBM Tokyo R. Lab) • Concordia(Mitsubishi) • Grasshopper(IKV++) • Odyssey(General Magic) • Voyager(ObjectSpace) • Mobile Objects & Agents (OSF) • Mole(Univ. of Stuttgart)	• Telescript (General Magic) • Ara(Univ. of Kaiserlautern) • Tacoma (Univ. of Tromso & Cornel Univ.) • Agent Tcl (Dartmouth College) • Obliq(DEC)

(1) 자바 기반 플랫폼 비교

이 절에서는 표 1에서 나타낸 목록 중에서 다섯 가지 자바 기반 이동 에이전트 플랫폼 즉, IBM사의 Aglets[23], 미쓰비시사의 Concordia[24], MASIF 표준[25]에 따라 개발된 IKV++의 Grasshopper[26], General Magic사의 Odyssey[27], 그리고 ObjectSpace의 Voyager[28]에 대한 특성의 개요를 기술하고, 자바 기반 이동 에이전트 플랫폼의 일반적 플랫폼 구조, 성능, 보안, 그리고 통신에 대하여 측정하여 비교한 결과를 제시한다[29].

(2) 일반적인 플랫폼 구조 비교

표 2는 일반적인 플랫폼 구조에 대한 비교로서, 통신 프로토콜, 표준화 지향, 진입 위치(Entry point), 분산 가비지 콜렉션(DGC: Distributed Garbage Collection)에 대한 비교를 나타낸다.

(3) 성능 측정에 대한 비교

표 3은 Aglet, Concordia, Voyager 그리고 Grasshopper 플랫폼들에 대한 몇 가지 성능 측정 비교를 제공한다. 성능은 그들의 내부적인 구조와 그들이 정의한 계층의 수에 의해 영향을 받는다. 성능 측정은 두 호스트 사이 MA(Mobile Agent)의 이동(migration), 다른 호스트에서 주거하는 MA사이에서 MA의 생성, 파괴 그리고 복

<표 2>. 일반적 플랫폼 구조 비교

Product	Communication Protocol	Compliance to standards	Entry points	Persistence	Distributed Garbage Collection
Aglet	Plain TCP/IP and RMI	-	One	Explicit, for a limited period	Not mentioned
Concordia	RMI	-	Multiple	Implicit, only for increased reliability	Not mentioned
Grasshopper	Plain TCP/IP, RMI, CORBA IIOP	MASIF	One	Implicit, explicit for arbitrary period	Not mentioned
Odyssey	Plain TCP/IP, RMI, CORBA, IIOP, D-COM	-	Multiple	No	Not mentioned
Voyager	Proprietary	-	Multiple	Implicit, explicit for arbitrary period	Yes

제에 대한 비교 결과를 나타낸다. 두 대의 기계, Host A와 B를 테스팅 목적으로 사용한다. 호스트 A는 일반적인 펜티엄 급으로 OS는 Window NT를 사용하고, 호스트 B도 펜티엄 급으로 OS는 Window 98을 사용한다. 두 호스트는 이더넷 10BaseT LAN으로 연결되었다.

<표 3>. 성능 측정 비교(single operation)

Product	Migration	Creation	Destruction	Cloning	Remote Communication
Aglet	190	17	7	60	60
Grasshopper	1400	44	12	170	42
Odyssey	320	7	N/A	2	90
Voyager	170	12	0.7	3	8.5

(4) 보안 특성과 통신 특성 비교

표 4는 이동 에이전트 플랫폼들에 대한 보안에 대하여 인증, 접근 통제, 기밀성, 무결성 전송, 그리고 저장소 보호에 대한 비교 결과를 나타낸다. 표 5는 통신 방법에 따른 비교 결과를 나타낸다.

5. 연구 동향

5.1 Netscript

콜롬비아 대학의 NetScript 프로젝트는 프로그래밍 언어로 네트워크 시스템을 개발하기 위한 환경을 제공한다[16]. 이 언어는 패킷 스트림 처리를 하기 위한 수단을 제공하며 라우팅, 패킷 분석, 시그널링과 관리기능의 구현에 적합하다. NetScript 에이전트는 라우터와 스위치 같은 중간 네트워크 노드를 포함하는 원격 시스템을 종단 시스템처럼 쉽고 빠르게 프로그래밍 가능하게

<표 4>. 보안에 대한 비교

Product	Authentication	Access Control	Confidentiality	Transmission Integrity	Storage Protection
Aglet	Domain Authentication through symmetric algorithms	Set through GUI	Symmetric algorithm	Yes	No
Concordia	User and group identity encrypted with symmetric algorithm	Permissions are stored in a preferences file	SSL protocol	Yes	Yes
Grasshopper	Yes, with X.509 certificates	Set through GUI	SSL protocol		No
Odyssey	No		No	No	No
Voyager	No	Security Manager	No	No	No

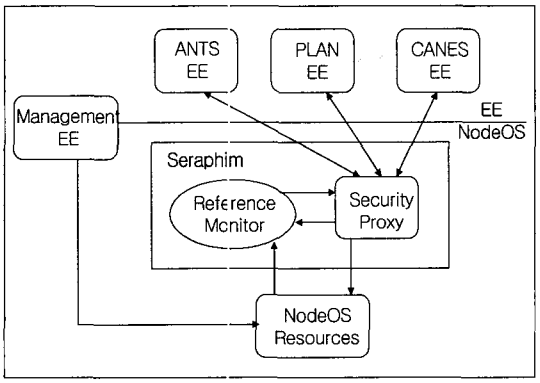
<표 5>. 통신 특성 비교

Product	point to point remote method invocation	Messaging modes	Dynamic Message	Multicast Messaging	Agent Proxies
Aglet	Yes, Generic	Synchronous, future reply	No	Yes	Yes
Concordia	No	An agent group object serve as a synchronization and collaboration point	No	Yes	No
Grasshopper	Yes, arbitrary	Synchronous, notification, further reply	Yes	Yes	Yes
Odyssey	No	Locally, normal invocation semantics, Remotely, through meetings	No	No	No
Voyager	Yes, arbitrary	Synchronous, further reply	Yes	Yes	Yes

하는 이동형 에이전트로 구성되어 지역제어와 원격제어를 수행한다. NetScript는 동적으로 네트워크 노드에 있는 액티브 프로그램을 배포하고 실행 할 수 있다.

5.2 Seraphim

일리노이 대학에서 개발한 Seraphim 프로젝트[30]는 Cherubim 프로젝트[31]에 이어서 진행되었다. Cherubim 프로젝트는 다른 보안 도메인 사이 상호작용을 동적으로 지원하기위해 이동 에이전트, 액티브 능력 그리고 맞춤형 보안 정책을 사용하여 액티브 네트워크 하에서 안전한 노드 구조를 구축하는 것으로, 노드의 보안 문제를 해결하기위한 메커니즘으로 스마트 보안 패킷을 사용하는 이동 에이전트 스킴을 이용하였다. Seraphim 프로젝트에서는 현재 빠른 프로토콜과 서비스를 개발하고 있는 액티브 네트워크에의 악성 사용에 대한 보안 문제를 해결하기 위하여 동적이며, 재구성가능, 확장성 그리고 상호운용이 가능한 액티브 네트워크 보안구조를 개발하였다. 그림 11은 Seraphim의 구조를 보여준다.

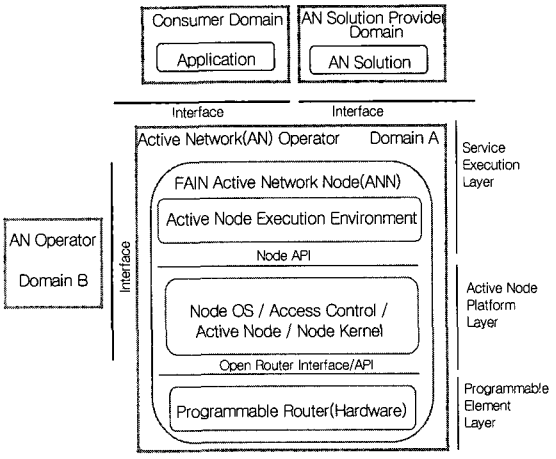


<그림 11> Seraphim의 구조

Seraphim 구조는 알려지지 않는 응용을 위해서 다양한 보안 스킴을 허락함으로서 동적인 보안 정책을 수립 가능하다. 참조 모니터(Reference Monitor)는 노드운영체제(NodeOS)에서 확장되어 구현된다. 모든 노드는 참조 모니터를 가지며 코어 보안 서비스는 액티브 능력으로 서명을 검증한다. 정책 프레임워크는 참조 모니터의 컴포넌트이다. 참조 모니터는 접근 점검으로 액티브 능력을 평가하며 MAC, DAC, DDAC 그리고 RBAC의 4가지 다른 접근 통제 정책을 수행한다. Seraphim의 구조에서 액티브 능력(Active Capability)은 접근 통제 결정 처리에 사용되어 보안정책을 운반하는 이동 에이전트이다. 액티브 캡슐은 증명서, 액티브 코드 그리고 서명을 가지고 있다. 이것은 자바 바이트 코드로 실행 가능하고 디지털 서명에 의해 보안된다.

5.3 FAIN

유럽 연합 FAIN(Future Active IP Network) 프로젝트[32]는 새로운 액티브 노드 개념을 기반으로 개방적이며, 유연하고, 프로그램가능하며, 신뢰적이고, 안전한 그리고 관리 가능한 네트워크 구조를 개발하는 것을 목표로 한다. 네트워크 구조는 미래 정보 사회에서 기대되는 분산 응용의 네트워크 통제와 관리 그리고 광범위한 새로운 비즈니스 모델을 지원하기위해서 전개하고 실행하는 상호운용적인 액티브 IP 네트워크 노드에 의해 검증될 것이다. FAIN 프로젝트는 액티브 네트워킹, 분산된 객체 그리고 이동 에이전트 기술의 혁신적인 통합으로 액티브 네트워크를 위한 포괄적인 구조를 제안하였다(그림 12참조). FAIN은 정책을 기반으로 액티브 서비스를 개발하는 프로젝트로서 FAIN 프레임워크에서 이동 에이전트 기술을 적용한 플랫폼으로 DDoS 공격에 대한 역추적 및 대응에 대한 연구이다. FAIN 구조는 USA와 국제적인 연결로 Pan-European 시험에서 검증되고 평가될 것이며 유럽 정책을 위해 주요한 권고와 지침을 제공할 것이다.



<그림 12> FAIN 노드 구조

5.4 eSMART

국내의 한국전자통신연구원에서 액티브 보안 관리 기술, 이동형 센서 기술 그리고 사이버 공격 역추적 기술을 이용하여 eSMART(ETRI Security Mechanism using Active Response Technology) 시스템을 개발하였다. eSMART 프레임워크는 다양한 유형의 사이버공격에 대한 신속한 대응 및 피해 국지화가 가능한 액티브 감지형 네트워크 보안 엔진 및 이동형 센서로 구성되어있으며, DDOS(Distributed Denial of Service) 등과 같은 우회 루트 공격을 감행하는 사이버 테러공격자의 위치 역추적을 통한 공격자의 근접점에서 공격적으로 대응이 가능하다. eSMART 시스템은 서비스 계층, 엔진 계층 그리고 NodeOS 계층으로 계층적으로 구조되었다[33].

6. 보안 문제

액티브 네트워크에서 특히 중요하게 취급되는 부분이 바로 보안이다. 액티브 네트워크에서 보안의 주요 요점은 안전한 실행환경과 분산코드의 인증을 제공하는 것이다. 액티브 네트워크 구조에서 보안은 다음과 같은 요구로 해석한다.

- 프로그램의 정확한 평가
- 프로그램의 확실한 검사 방법
- 사용자의 인증
- 프로그램 실행동안 수행 방법을 제한
- 프로그램 실행은 인증과 보안 정책 기반

또한 액티브 네트워크의 하부구조는 안전한 보장을 제공해야 한다.

액티브 네트워킹은 종단 노드에서 코드를 다운로드하고 실행하는 능력을 사용자에게 제공한다. 여기에서 액티브 코드와 실행 환경의 공격에 대한 보안은 중요하다. 일반적인 보안의 형태는 다음과 같다[34]:

- 액티브 코드로 인한 실행 환경의 오용
- 다른 액티브 코드로 인한 액티브 코드의 오용
- 실행환경으로 인한 액티브 코드의 오용
- 네트워크 하부구조로 인한 액티브 코드 혹은 실행 환경의 오용

액티브 코드로 인한 실행환경의 오용은 악용 액티브 코드가 실행 동안에 호스트의 보안 약점을 이용하는 것으로 가장, 서비스 거부 공격, 비 권한 접근 그리고 복합 공격이 있다. 다른 액티브 코드로 인한 액티브 코드의 오용은 다른 악용 액티브 코드가 실행동안에 액티브 코드의 약점을 이용하는 것으로 부인, 가장, 서비스 거부 공격 그리고 비 권한 접근이 있다. 실행환경으로 인한 액티브 코드의 오용은 호스트 실행환경이 실행동안에 액티브 코드의 약점을 이용하는 것으로 가장, 서비스 거부공격, 도청, 데이터와 상태 조작 그리고 복제 등이 있다. 네트워크 하부구조로 인한 액티브 코드 혹은 실행 환경의 오용은 네트워크 하부구조가 실행 동안에 액티브 코드 혹은 호스트 실행 환경의 약점을 이용하는 경우이다.

6.1 액티브 네트워크의 일반적 보안 요구안

액티브 네트워크 일반적 보안 요구 안으로는 비밀성/기밀성, 무결성, 계정성(accountability)과 부인 봉쇄, 가용성, 인증, 접근 통제, 권한, 안전한 통신 그리고 QoS 개념을 보증하여 QoS의 보안위반을 제한하는 것 등이 있다.

6.2 액티브 네트워크의 확장 보안 요구안

액티브 네트워크에서 확장 보안 요구 안으로는 이웃 액티브 노드의 신뢰된 확인, 실행환경의 검증, 노드에서 노드로 코드의 안전한 전송/분산, 정책을 기반으로 하는 액티브 노드의 설치와 제거, 정책기반 액티브 코드 취소, 정책을 기반으로 노드의 자원으로 접근, 액티브 코드 실행을 위한 실행시간 동안 접근 통제, 실행환경 사이 비 권한 상호작용을 예방, 광범위한 네트워크 보안 관리, 안전한 감사, 안전한 코드 실행, 동적인 정책 계획, 지속성, 미리 정의된 노드 조작, 계층적으로 구조화된 보안 서비스, 익명의 원칙을 지원 그리고 대상으로 명확한 코드 분산 등이 있다.

6.3 이동에이전트 기반 액티브 네트워크 보안

액티브 네트워크에서 이동 에이전트를 보안하는 방법으로는 실행환경 보호와 액티브 코드를 보호하는 방법으로 나누어진다[35].

(1) 실행 환경을 보호

실행 환경을 보호하는 보안 방법으로는 서명된 코드, 상태평가, 안전한 코드 번역, 결점을 고립, 증명을 수반하는 코드, 경로 기록 그리고 실행 환경의 자원을 보호하는 방법이 있다.

(2) 액티브 코드를 보호하는 방법

액티브 코드를 보호하는 방법으로는 부분적인 결과 캡슐화, 공유된 기밀과 협력, 실행 추적, 환경적인 키 생성, 암호화된 기능/데이터 컴퓨팅, 모호한 코드, 액티브 코드 자체의 암호화, 전용의 하드웨어/스마트 카드 사용 그리고 익명/추적 불가능한 액티브 코드를 지원 등이 있다.

향후의 액티브 보안 기반구조는 다음과 같은 보안 변화를 충족시켜야 할 것으로 보인다.

- 공격에 대한 수동적 대응에서 능동적 대응이 가능한 보안
- 시스템 또는 내부 망 보안에서 전체 네트워크에 대한 보안
- 보안 시스템의 독립적 운용에서 상호 결합적 운용
- 공격 및 시스템의 대응 방법에 대한 서술 및 이의 수용이 가능한 유연한 보안 시스템 구조
- 새로운 보안 정책 및 기술의 수용이 용이한 개방적 실행 구조를 가지는 보안 시스템

위의 보안 요구 조건들을 충족시키기 위해 DARPA를 중심으로 보안 기반 구조 정의, 통신 프로토콜, 서술 언어 설계, 보안 시스템 실행 구조 등이 현재 진행되고 있다[1].

7. 결론

액티브 네트워크는 프로그래밍이 가능한 네트워크로 현재의 네트워크 문제를 해결하기 위한 네트워크의 새로운 패러다임이다. DARPA에서 연구하는 액티브 네트워크의 목표는 멀티캐스트에서 오디오/비디오 동기화와 최대-율 비디오 전송, 종단 응용으로 유용한 데이터 율을 증가시켜 더 적은 패킷을 재전송하도록 네트워크 서비스의

정량적인 향상과 미래 DoD가 필요로 하는 구조를 생성하는 것이다. 그리고 네트워크를 기반으로 결점-적응 메커니즘 개발과 동적인 접근 제어를 위한 인증 형태 그리고 타입과 정책을 기반으로 하여 트래픽을 분리하고 관리적인 기능강화를 통하여 이중으로 견고한 이동 보안을 들 수 있다.

따라서 액티브 네트워크의 개발 경향은 캡슐(스마트카드)의 효율적인 처리와 안전하게 생존 가능한 복합형 프로토콜 개발 그리고 빠르고 안정적인 새로운 서비스의 개발을 들 수 있다. 나아가 새로운 서비스의 표준화 필요성 없이 보급되어 사용 가능하도록 하며 네트워크의 복잡성(크기, 속도, 다양성)과 보조를 맞추어 중요한 네트워크 서비스를 향상시키고 그리고 라우팅을 위한 새로운 전략을 개발하여 이동성 서비스까지 대규모 네트워크에서 향상된 네트워크 서비스를 보급하는 것이다. 이동 에이전트 기반 액티브 네트워크는 미래의 상호작용을 위해 KQML, KIF, COBRA같은 표준 통신 기술 기초를 제공해야하며, 현재 이들의 표준화 동향으로는 OMG, MASIF 그리고 FIPA같은 이동 에이전트 표준화 작업이 진행되고 있다. 완전한 액티브 네트워크를 테스트하고 분석하는 것은 가상적인 컴퓨터 네트워크를 조성하여 프로그래밍 하는 것이다. 액티브 네트워크에서의 해결해야할 중요한 문제는 효율적인 보안과 성능 그리고 기존의 응용 서비스와의 연동 문제이다.

본 논문에서는 액티브 네트워킹을 위하여 에이전트를 이용하는 이동 에이전트 기반 액티브 네트워크에 대하여 고찰 및 분석을 하였다. 먼저 액티브 네트워크 기술과 이동성을 가지고 있는 이동 에이전트 기술에 대하여 기술하였으며, 이동 에이전트 기반 액티브 네트워크의 액티브 노드와 이동 에이전트 그리고 에이전트 통신 프로토콜에 대하여 기술하였다. 나아가 활발히 전개되고 있는 에이전트 기반 액티브 프로그래밍 기술로서 자바와 비자바 기반 언어에 대하여 분석하였으며 이동 에이전트 기반 액티브 네트워크 프로젝트를 중심으로 연구동향을 기술하였다. 마지막으로 이동 에이전트의 보안에 대하여도 분석하였다. 이동 에이전트 기반 액티브 네트워크 기술은 프로그래밍 된 보안 메커니즘이 네트워크를 통해 액티브 노드간을 이동하며 실행되고 동적으로 복제, 소멸되는 기능을 제공함으로써 기존의 네트워크 상에서 자동적이고 지능화된 보안 응용 서비스를 가능하게 할 것이다.

참고문헌

[1] http://www.darpa.mil/ito/research

[2] Tennenhouse, D., et al., "A Survey of Active Network Research", IEEE Communications Magazine, January 1997.

[3]http://www.sds.lcs.mit.edu/activeware, http://www.netlab.cs.iitm.ernet.in/cs481/ current/cs97187/an.html

[4] Ulana Legedza, David Wetherall and John Guttag, "Improving the performance of distributed applications using active networks", IEEE INFOCOM'98, 1998.

[5] Calvert, K. et al, Architectural Framework for Active Networks, Active Networks Working Group Draft, July 1999.

[6] Alexander D.S., Braden B., Gunter C.A., Jackson W.A., Keromytis A.D., Milden G.A., and Wetherall D.A., Active Network Encapsulation Proto-

col (ANEP), Active Networks Working Group Draft, July 1997.
[7] David Wetherall, Ulana Legedza and John Guttag, "Introducing New Internet Services: Why and How", IEEE Network Magazine, July/ August 1998
[8] http://www.darpa.mil/ato/programs/active-networks/actnet.htm
[9] D. J.Wetherall, J. V. Guttag, and D. L.Tennenhouse, "ANTS: Architecture for Building and Dynamically Deploying Network Protocols", Proc. IEEE OPENARCH 98, April 1998.
[10] K. Calvert, E. Zegura, and J. Sterbenz. "CANEs: A Modest Approach to Active Networking", In Proceeding of IEEE Computer Communications Workshop, September 1997.
[11] Dan Decasper and Bernhard Plattner, "DAN: Distributed Code Cashing for Active Networks", Proc. IEEE INFOCOM 98, San Francisco, CA, March/April 1998.
[12] D. J. Wetherall and D. L. Tennenhouse, "The Active IP Option", In Proceeding of 7th ACM SIGOPS European Workshop, September 1996.
[13] D.S. Alexander, et al. Active Network Encapsulation Protocol (ANEP). In RFC Draft, July 1997.
[14] B. Schwartz, et al., "Smart Packets for Active Networks". In Proceeding of IEEE Communication Surveys, First Quarter 1999.
[15] Alexander, D.S., Arbaugh, W.A., Hicks, M.A., Kakkar P., Keromytis A., Moore J.T., Nettles S.M., and Smith J.M., "The SwitchWare Active Network Architecture", IEEE Network-Special Issue on Active and Controllable Networks, vol. 12 no. 3, 1998.
[16] Da Silva, S., Florissi, D. and Yemini, Y., NetScript: A Language-Based Approach to Active Networks, Technical Report, Computer Science Dept, Columbia University, January 27, 1998.
[17] M. Wooldrige and N. Jennings. Intelligent Agents : Theory and Practice. 1995.
[18] Konstantinos Psounis, "Active Networks: Applications, Security, Safety, and Architectures", in IEEE communications survey, First Quarter 1999.
[19] Jonathan Dale, "A Mobile Agent Architecture for distributed Information Management", 1997.
[20] Chih-Lin hu, Wen-Shyen E. Chen, "A Mobile Agent-Based Active Network Architecture", Department of Electrical Engineering National Taiwan University, IEEE 0-7695-0568-6/00, 2000.
[21] Stamatis Karnouskos, "Agent Populated Active Networks", 2nd International Conference on Advanced Communication Technology (ICACT 2000), February 16-18, 2000,

Muju, Korea.
[22] Stamatis Karnouskos. "Dealing with Denial-of-Service Attacks in Agent-enabled Active and Programmable Infrastructures", 25th IEEE International Computer Software and Applications Conference (COMPSAC 2001), October 8-12, 2001, Chicago, Illinois, U.S.A (ISBN 0-7695-1372-7).
[23] http://www.tri.ibm.co.jp/aglets
[24] http://www.meitca.com/HSL/projects/Concordia
[25] http://www.gent.org/pub/satp/palers/maf5.html
[26] http://www.ikv.de/prducts/grasshopper
[27]http://www.genmaigc.com/technology/odyssey .html
[28] http://www.objectspace.com/voyager
[29] Menelaos K. Perdikeas, Fotis G. Chatzipapadopoulos, "An Evaluation Study of Mobile Agent Technology: Standardization, Implementation and Evolution," IEEE, 0-7695-0253- 9/99, pp.287-291, 1999.
[30]http://choices.cs.uiuc.edu/Security/seraphim/
[31]http://choices.cs.uiuc.edu/Security/cherubim/
[32] http://www.ist-fain.org/
[33] http://www.etri.re.kr
[34] Yang Kun, Guo Xin, Liu Dayou, Security in Mobile System: Problems and Approaches, Department of Computer Science, JiLin University, Changchun, 130023, National Natural Science Fund and 863 high-tech Project, pp21-28.
[35] Stamatis Karnouskos, "Security Implications of Implementing Active Network Infrastructures using Agent Technology", Computer Networks36 (2001) 87-100.

전 용 희

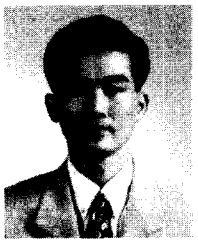
1978년 고려대학교 전기공학과 졸업(공학사)
1985~1987년 미국 플로리다공대 대학원 컴퓨터공학과
1989년 미국 노스캐롤라이나주립대 대학원 Elec. and Comp. Eng. 졸업(공학석사)
1992년 미국 노스캐롤라이나주립대 대학원 Elec. and Comp. Eng. 졸업(공학박사)
1978~1978년 삼성중공업(주) 근무
1978~1985년 한국전력기술(주) 근무
1989~1989년 미국 노스캐롤라이나주립대 Dept of Elec. and Comp. Eng. TA
1989~1992년 미국 노스캐롤라이나주립대 부설 CCSP(Center For Comm. & Signal Processing) RA
1992~1994년 한국전자통신연구원 광대역통신망연구부 선임 연구원
1994~현재 대구가톨릭대학교 컴퓨터·정보통신공학부 교수
2000~현재 한국통신학회 학회지 편집위원
2001.3~2003.2 동 공과대학장 역임

<주관심분야> 네트워크 보안, Active Network, 통신망 성능분석, QoS 보장 기술

장 정 숙

1991년 경일대학교 공과대학 컴퓨터공학과 졸업(학사)
1992년 ~ 1995년 대구가톨릭대학교 교육대학원 전자계산교육전공 (석사)
1998년 ~ 현재 대구가톨릭대학교 대학원 컴퓨터·정보통신공학 전공 박사 수료

<주관심분야> 네트워크 보안, Active Network, 통신망 성능분석, 고속 통신망 응용 서비스



장 종 수

984년 경북대학교 전자공학과 학사,
1986년 경북대학교 대학원 전자공학과 석사,
2000년 충북대학교 대학원 컴퓨터공학과 박사,
1989년 7월 ~ 현재 한국전자통신연구원 정보보호연구본부 네트워크보안연구부 보안게이트웨이연구팀 팀장/책임연구원,

<주관심분야> Network Security, Active Network, Biometry

손 승 원

1984년 경북대학교 전자공학과 졸업 (공학사),
1994년 연세대학교 전자공학과 졸업 (공학석사),
1999년 충북대학교 컴퓨터공학과 졸업 (공학박사),
- 1983년 ~ 1986년 삼성전자 연구원
1986년 ~ 1991년 LG 전자(주) 중앙연구소 H18mm 캠코더 팀장
1991년 ~ 현재 한국전자통신연구원 정보보호연구본부 네트워크보안연구부 부장/책임연구원,

<주관심분야> 네트워크보안, 차세대인터넷, Active Internet