ACL 기반 이동 에이전트 프레임워크

조정은, 김원국, 김상욱
경북대학교 컴퓨터과학과 컴퓨터언어연구실

ACL based Mobile Agent Framework

Jungeun Cho, Wonkook Kim, Sangwook Kim
Department of Computer Science, Kyungpook National University

요 약

본 논문에서는 ACL(Agent Communication Language)를 기반으로 한 에이전트 프레임워크 Mollet을 제안하고 코드와 데이터의 이동성을 중점을 두어 이를 OS 버그 패치 시스템에 적용하여 본다 Mollet의 메시지 전송과 에이전트 이동은 FIPA에서 표준안으로 정의한 ACL을 사용함으로 다른 에이전트와의 호환성을 높이고 있으며 전송 과정에서 노출을 막기 위하여 암호화 기법을 이용한다. 이동성, 협상, 보안, 인증, 등록 등의 에이전트가 가지는 공통적인 속성과 각 에이전트만이 가지는 속성을 모듈화함으로 확장성을 높일 수 있으며 JDBC 사용으로 데이타베이스와의 투명성을 제공하고 있다. 또, Mollet은 하나의 호스트가 서버와 클라이언트의 속성을 동시에 가질 수 있다.

Mollet을 OS 버그 패치 시스템에 적용하여 사용자의 공식 패치 서버 검색을 자동화하고 신속하고 간편한 패치를 제공함으로 시스템의 보안과 신뢰도를 유지할 수 있고 시스템 관리자의 부담과 관리 비용을 줄인다

1. 서 론

이동 에이전트는 분산 컴퓨팅 기술에 대한 새로운 접근으로 일정 식별자를 가지고 사용자의 의도에 따라 자율적이고 융통성 있게 네트워크 상을 이동한다. 에이전트의 이동은 동적인 사용자 환경에 적합한 어플리케이션을 제공한다. 기존의 이동 에이전트 프레임워크로는 aglet, Java-to-go, Voyager 등이 있다. 이들은 에이전트의 메시지 전송 및 에이전트 전송에 있어서 표준화된 통신 언어를 사용하고 있지 않다.

본 논문에서는 FIPA97[4]에서 정의한 ACL을 바탕으로 이동 에이전트 프레임워크인 Mollet을 설계 및 구현 하였다. Mollet은 클라이언트-서버, ANS(Agent Name Service)로 구성되며 이동 에이전트의 공통된 속성과 차별화된 속성을 구분하여 모듈화하고 있다. JDBC를 이용하여 데이터베이스와의 투명성을 제공하고 FIPA 97에 따른 ACL을 기반으로 에이전트간에 통신이 이루어짐에 따라 호환성을 높일 수 있다.

Mollet에서는 ACL로 사용함으로써 다른 에이전트와의 호환성을 높이고 에이전트 사이의 프로토콜을 쉽게 정의하고 사용할 수 있어 에이전트 통신을 쉽게 구현 할 수 있다.

Mollet을 OS 버그 패치 시스템에 적용함으로 시스템 보안의 신뢰성은 유지하기 위해 패치를 신속하게 시스템에 적용시킬 수 있고 패치 검색의 번거로움을 해결하고 지속적인 업그레이드를 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 ACL의 구조를 살펴보고 3장에서는 Mollet의 구조에 대하여 설명한 다음 4장에서는 이를 OS 버그 패치 시스템에 적용하고 5장 결론으로 맺는다.

2. ACL

FIPA97에 따른 ACL은 ACL과 SL로 이루어진다. ACL은 에이전트가 보낼 메시지의 형태, 메세지를 받는 에이전트, 통신 프로토콜을 정의한다. SL은 에이전트가 보내는 메세지의 내용을 전달한다.

ACL API는 ACL과 SL 각각의 파서와 생성기로 구성된다 에이전트가 ACL메세지를 ACL 생성기와 SL 생성기를 통해 생성한다. 데몬이 ACL 해석기를 통해서 메시지를 받을 에이전트에게 메세지를 전송한다. 메세지를 전송 받은 에이전트는 ACL 해석기를 통해 메세지를 보낸 에이전트과 메세지 프로토콜등의 정보를 획득한다. 에이전트가 보낸 내용은 SL 파서를 통해 해석된다.

ACL과 SL는 넌터미널을 하나의 클래스로 정의하여 각 터

미널들을 변수로 설정한다.

ACL 파서는 각 메세지 변수에 해당하는 API를 제공한다. SL 파서는 SL 메세지를 명제와 액션을 구분하여 처리한다. 액션는 다른 액션를 포함하는 문법을 만든다. 이 문법을 처리하기 위해 하나의 액션을 하나의 트리로 구성한다.

ACL 생성기는 각 메세지 변수를 설정하는 API로 구성된다. SL 생성기는 에이전트가 새로운 액션에 대해서 트리를 구성한 후 bottom-up 방식으로 메시지를 구성하는 API를 제공한다.

FIPA ACL의 구조는 그림 1과 같다.

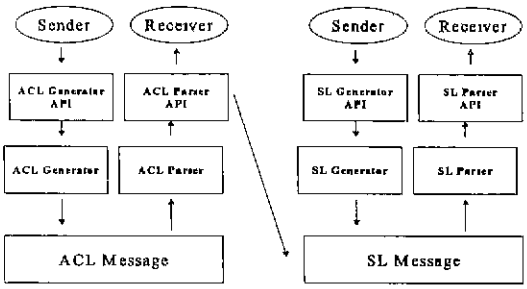

그림 1 FIPA ACL의 구조

3. 이동 에이전트 프레임워크 : Mollet

ACL을 기반으로 한 이동 에이전트 프레임워크 Mollet에서 하나의 호스트는 서버와 클라이언트의 특성을 동시에 가질 수 있다. 클라이언트에서 생성된 에이전트는 자신에게 등록을 하고 다른 서버로 이동하여 작업을 한다. 작업이 완료되면 클라이언트에게 필요한 정보를 보내거나 되돌아와 작업을 마치게 된다. 에이전트의 기본적인 공통 속성들을 API 형태로 정의하여 각 에이전트만이 가지는 속성과 분리하여 모듈화함으로 확장성을 높일 수 있다.

Mollet의 구조는 다음과 같다.

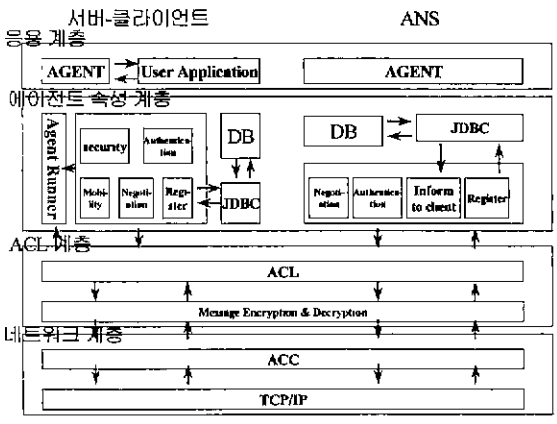

그림 2 Mollet의 구조

3.1 네트워크 계층

에이전트간 통신과 에이전트의 이동을 담당하며 TCP/IP와 ACC(Agent Communication Channel)로 이루어진다. ACC는 OMG의 Mobile Agent Facility(MAF)와 FIPA 표준안에서 제안한 CORBA/IIOP(Internet Inter ORB Protocol)를 사용한다.

3.2 ACL 계층

에이전트와 에이전트의 메시지를 FIPA ACL로 생성하거나 전달받은 메시지를 해석하는 곳으로 에이전트가 전달하고자 하는 메시지나 에이전트 자체의 이동을 ACL 형식으로 변형한 후 이를 암호화하여 전달한다.

ACL에서의 이동 에이전트의 전송은 다음과 같다.

```
(request
    :sender an-agent@iiop://cs kyungpook.ac.kr
    :receiver an-other@iiop'//woorisol.kyungpook ac.kr
    :content
        (action an-ans@iiop://cs.kyungpook.ac.kr
        (move
        (:agent-name an-agent@iiop://cs.kyungpook.ac.kr)
        (:agent-code "..." )
        (:agent-profile "..." ))) ...
)
```

에이전트와 메시지 전송 과정에서 메시지 노출을 방지하기 위해 공개키와 비밀키를 사용하는 공개키 암호화 기법을 사용한다.[8] 따라서, 메시지를 전송하고자 하는 서버 사이에서 발생하는 보안 문제를 해결 할 수 있다.

3.3 에이전트 속성 계층

등록자 인증 과정은 PBE(Password Based Encryption) 방식을 사용한다. 등록 정보로는 에이전트 ID, 에이전트 소유자, 생성날짜, 생존기간 등이 있다. 서버가 등록된 에이전트를 다른 클리이언트 또는 서버에 알릴 때 peer-to-peer방식과 multicast 방식을 제공한다. 에이전트의 이동은 보안 문제를 고려 하여 서버를 통해서 이루어 진다.

에이전트의 상태에 따라 현재 상태를 보존하고 이동하는 경우와 이동 후 재실행시키는 경우를 제공한다. 상태를 보존하고 이동하기 위해서 자바의 serializable과 externalizable 기법을 이용한다.

에이전트를 실행 시키기 위해서 하나의 Thread를 생성하여 에이전트를 실행한다. 에이전트를 동적으로 실행하기 위해서 자바의 클래스로더가 사용된다.

서버를 이동 에이전트로부터 보호하기 위해 리소스 암호화와 자바에서 지원하는 보안 관리자를 사용하고 이동 에이전트를 서버로부터 보호하기 위해 에이전트 checksum을 검사한다.

클라이언트-서버 모델에서의 데이타베이스는 사용자가 등록한 에이전트와 그에 따른 부가적인 정보를 기록한다. ANS의 데이타베이스에는 등록된 에이전트에 관한 정보만을 기록한다. JDBC를 이용한 DB 접근으로 DB의 종류에 관계없이 자료를 저장할 수 있는 투명성을 제공한다.

Mollet의 흐름은 다음과 같다.

```
process mollet server.
begin.
 create_new_agent().
  If ( Authentication_to_server() == SUCCESS )
      register_agent_to_server();
  If ( Authentication_to_ANS() == SUCCESS )
      register_agnet_to_ANS();
end.

process mollet ANS.
begin:
   if ( inform_method() == PEER_TO_PEER ) {
      get_client_list();
      inform_client();
   }
   else if ( inform_method() == MULTICAST ) {
      get_client_lists();
      inform_client_list();
   }
end'

process mollet client'
begin:
   listen_to_ANS();
    if ( vaild_information() == TRUE ) {
      get_agent_to_server().
      run_agent();
   }
end.
```

4. OS 버그 패치 적용 예

시스템 버그는 응용의 특징과 시스템 환경 설정에 따라 여러 가지 환경에서 발생할 수 있다. 기존의 응용은 모든 환경에 적합하게 재구성되어야 하나 Mollet을 OS 버그 패치 시스템에 적용함으로써 자동으로 각 버그에 해당하는 검사 에이전트가 시스템의 버그를 검사한 후, 해당 패치 에이전트가 코드와 데이터를 이동한다. 검색 자동화와 더불어 신속하고 간단한 패치 자동화를 이룸에 따라 시스템 관리자의 부담과 관리 비용을 줄일 수 있다.

OS 버그 패치 시스템은 등록과 패치로 이루어진다.

단계1: 등록

① 초기에 클라이언트는 설치 프로그램을 실행시킨다.
② 설치 프로그램은 지정된 디렉토리에 파일을 실행되면 현재 등록 서버(ANS)에 연결하여 클라이언트의 정보를 등록한다.
③ 등록이 허락되면 특정 포트를 통해 데몬(daemon)을 생성한다.

단계2: 패치

① 패치 프로그램을 작성한 다음 설치 방법과 업그레이드되어야 하는 파일 등을 조사한다.
② 시스템 버그 조사를 위하여 검사 에이전트를 만들고 패치를 위한 에이전트를 작성한다.
③ 서버 관리자는 작성한 두 에이전트를 ANS에 등록한다 ANS에서는 서버로부터 등록된 에이전트와 ACL을 이용하여 통신함으로 해당 클라이언트에 먼저 검사 에이전트를 전송한 다음 클라이언트에서 패치 에이전트의 요구가 있으면 전송한다
④ 클라이언트에 도착한 검사 에이전트는 클라이언트의 데몬을 통해 전달되고 클라이언트와의 협상을 통하여 클라이언트의 검사 허락이 있으면 시스템을 검사하여 그 결과를 클라이언트에게 반환한다.
⑤ 버그가 발견되면 이를 시스템 관리자에게 알린다. 시스템 관리자가 버그 패치를 허락하면 클라이언트 에이전트는 ANS에 패치 에이전트를 요구하고 패치 에이전트를 전송한다.
⑥ 클라이언트와의 협상을 통하여 협상에 성공하면 시스템 버그를 패치한 후, 그 결과를 ANS에 알리고 종료한다.

5. 결론 및 향후 연구 방향

지금까지 FIPA97에 따른 ACL을 구현하고 이를 기반으로 한 이동 에이전트 프레임워크인 Mollet를 설계했다. Mollet은 ACL을 사용함으로써 다른 에이전트와의 호환성을 높힌다. Mollet은 클라이언트-서버, ANS의 분산 서버 환경으로 구성되며 JDBC를 통한 데이터베이스와의 투명성을 제공한다 Mollet을 OS 버그 패치 시스템에 적용시킴으로써 이동에이전의 장점인 코드의 최적화를 통해 정확하고 신속하게 OS의 버그를 패치 할 수 있다.

이동 에이전트 프레임워크 Mollet는 OS 버그 패치 시스템뿐만 아니라 PC 닥터, 소프트웨어 버전업 시스템 등의 다양한 응용 분야에 적용될 수 있다.

향후 연구 방향으로는 이동 에이전트의 가장 큰 약점인 에이전트와 에이전트 시스템의 보안에 대해서 더욱 많은 연구와 에이전트간에 효율적은 정보교환을 위한 에이전트 간의 메시지 프로토콜을 개발이 필요하다.

참고 문헌

[1] Anselm Lingnau, Oswald Drobnik, An Infrastructure For Mobile Agents : Requirements And Architecture, http://www.tm.informatik uni-frankfurt de/ma/papers.html
[2] Colin G Harrison, David M.Chess, Aaron Kershenbaum, Mobile Agents : Are They a good idea?, IBM Research Report.
[3] 유닉스 보안 취약점 자동복구 기본 시스템 개발에 관한 연구 , 한국정보보호센터, 1997.
[4] FIPA97 specification, Version 1.0 Part 1 Agent Management
[5] FIPA97 specification, Version 1.0 Part 2 Agent Commnucation Lanuage
[6] FIPA98 Draft Specification Part 11 Agent Management Support for Mobility
[7] Aglets Specification 1.0
[8] RIVEST, R., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems Commun. ACM 21, 2(Feb 1978), 120-126