情報保護學會論文誌
第13券 第6號, 2003. 12

암호화된 체크포인트를 이용한 결함 허용성을 가지는 이동 에이전트의 이주 기법 설계*

김 구 수**†, 엄 영 익***‡

Design of Fault-tolerant MA Migration Scheme based on Encrypted Checkpoints

Gu Su Kim**†, Young Ik Eom***‡

요 약

이동 에이전트는 네트워크에서 사용자를 대신하여 특정 작업을 수행하면서, 자율적으로 한 노드에서 다른 노드로 이주할 수 있는 프로그램이다. 본 연구에서는 이동 에이전트가 여러 노드들을 이주하면서 실행하다가 비정상적인 종료 상황이 발생했을 때 이전 사이트에 저장한 이동 에이전트의 체크포인트를 이용해서 이동 에이전트를 안전하게 복원하고 실행을 재개할 수 있는 기법을 제안한다. 체크포인트의 보안을 위해 이동 에이전트의 공개키를 이용하여 체크포인트를 암호화하여 저장하고, 복원 시에는 홈 플랫폼에서 이동 에이전트의 비밀키를 이용해서 이동 에이전트를 복원한다. 홈 플랫폼이 이동 에이전트를 복원하기 위해서 체크포인트를 수신하였을 때 이 체크포인트가 올바른 체크포인트인지 확인하기 위해서 메시지 다이제스트를 이용하며, 이동 에이전트가 체크포인트를 만들 때 생성한 메시지 다이제스트와 복원 시에 만든 메시지 다이제스트를 비교함으로써 홈 플랫폼은 수신한 이동 에이전트의 체크포인트의 정확성을 확인할 수 있게 된다.

ABSTRACT

A mobile agent is a program which represents a user in a network and is capable of migrating from one node to another node, performing computations on behalf of the user. In this paper, we suggest a scheme that can safely recover mobile agent using the checkpoint that is saved at the platform that it visited previously and restart its execution from the abnormal termination point of the mobile agent. For security, mobile agent uses its public key to encrypt the checkpoint and the home platform uses the private key of the mobile agent to decrypt the encrypted checkpoints at the recovery stage. When home platform receives the checkpoint of the mobile agent, home platform verifies the checkpoint using message digest. Home platform verifies the correctness of the checkpoint by comparing the message digest generated at checkpoint creation time with the message digest generated at mobile agent recovery time.

keyword : *이동 에이전트, 이주, 결함 허용성, 체크 포인트*

Ⅰ. 서 론

이동 에이전트란 컴퓨터 네트워크 상에서 사용자를 대신하여 특정 작업을 수행하는 프로그램이 독자적으로 여러 노드들을 이동하면서 필요한 작업을 수행하고 그 결과를 사용자에게 전달하도록 구성된 프

* 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2002-042-A00020).
** 성균관대학교 정보통신공학부(gusukim@ece.skku.ac.kr)
*** 성균관대학교 정보통신공학부(yieom@ece.skku.ac.kr)
† 주저자, ‡ 교신저자, 논문접수일 : 2003년 6월 20일, 심사완료일 : 2003년 11월 13일

로그램을 말한다.[1] 이동 에이전트는 비동기적이고 자율적으로 작업을 수행하며, 환경 변화에 대해 동적이고 유연성있게 적응한다. 또한, 자원이 있는 노드로 이동하여 작업을 수행하므로 통신비용의 절감을 가능하게 하며, 결함 허용(Fault-tolerance) 기능을 제공한다.[2] 이러한 장점 때문에 이동 에이전트 시스템은 전자상거래, 분산 정보 수집, 워크플로우 및 그룹웨어 등의 다양한 분야에 응용될 수 있다.

지금까지 이동 에이전트를 지원하는 여러 시스템들이 개발되어왔으며 대표적으로 Aglets,[3] Voyager,[4] Ajanta,[5] Concordia[6] 등을 들 수 있다. 이들 이동 에이전트 시스템은 이동 에이전트의 필수 기능인 이주 기능을 지원하고 있으며 이동 에이전트의 이주 경로 제어를 위해 Itinerary라는 객체를 사용하고 있다.[7] Intinerary는 이동 에이전트가 방문할 목적지들의 정보를 나열한 것으로서, 이동 에이전트는 Itineary의 각 항목에 지정된 목적지를 차례대로 방문하며 사용자의 작업을 수행한다. 이동 에이전트의 이주 기법에서 Itinerary를 이용하면 이동 에이전트의 개발자가 편리하게 목적지들을 선정하여 이동 에이전트를 이주시킬 수 있지만, 대신 이동 에이전트의 특성인 자율성은 낮아지게 된다.

이동 에이전트가 Itinerary에 기술된 여러 목적지를 방문하면서 사용자의 작업을 수행하는 과정에서 방문한 컴퓨터의 고장, 이동 에이전트의 수행 환경인 플랫폼의 고장 또는 이동 에이전트의 자체 결함으로 인해 이동 에이전트의 비정상적인 종료가 발생할 수 있다.[8] 이동 에이전트 시스템은 이러한 이동 에이전트의 종료 상황에 대처할 수 있는 결함 허용 기능을 가져야 한다.

본 논문에서는 Itinerary를 이용한 두 가지 이동 에이전트의 이주 기법과 암호화된 체크포인트를 이용한 이동 에이전트의 결함 허용 기능을 제안한다. 이를 통해 이동 에이전트가 임의의 중간 목적지에서 종료하더라도 체크포인트를 이용해 이동 에이전트를 복원하고 실행을 재개시킬 수 있게 된다.

본 논문의 구성은 다음과 같다. 2장에서 체크포인트에 관한 기존 연구를 설명한다. 3장에서는 Itinerary를 이용한 이주기법과 체크포인트를 이용한 이동 에이전트 복원 기법을 설명한다. 4장에서는 체크포인트에 대한 가능한 공격과 제안 기법의 안전성을 설명한다. 5장에서 결론을 맺는다.

II. 관련 연구

이동 에이전트는 방문한 컴퓨터의 고장, 이동 에이전트의 실행 환경을 제공하는 플랫폼의 고장, 이동 에이전트 자체의 고장으로 인해 비정상적으로 종료할 수 있다. 이러한 비정상적 종료 상황에서 이동 에이전트를 복원하고 실행을 재개하는 결함 허용을 제공하는 기법으로 이동 에이전트의 복제 기능을 이용하는 기법이 있다.[9] 이동 에이전트를 복제하여 다중으로 작업을 수행시키면, 어느 한 에이전트가 종료하더라도 살아있는 다른 이동 에이전트를 통해 사용자의 작업을 계속 수행할 수 있다. 하지만 이 방법은 여러 복사본이 동일한 작업을 수행하므로 시스템의 부하가 증가하고, 여러 이동 에이전트가 동일 작업에 대해 다른 결과를 냈을 때 어느 결과를 채택할지 결정해야 하는 문제점을 가진다.

이동 에이전트의 결함 허용 기능을 제공하는 또 다른 방법으로 체크포인트를 이용하는 기법이 있으며 Voyager, Tacoma, Concordia 등의 시스템에서 제공하고 있다.[10] 체크포인트는 이동 에이전트의 실행에 필요한 상태 정보를 파일 시스템에 저장한 것으로, 이동 에이전트가 비정상적으로 종료했을 때 가장 최근의 체크포인트에서 이동 에이전트를 복원하여 실행을 재개시킬 수 있다.

체크포인트를 이용한 결함 허용 기법으로 에이전트 기반의 기법과 플랫폼 기반의 기법이 있다. 에이전트 기반의 기법은 체크포인트만을 관리하는 모니터 에이전트를 생성한다. TACOMA[11]는 방문한 플랫폼에 이동 에이전트의 실행을 감시하는 rear guard 에이전트를 생성한다. rear guard 에이전트가 이동 에이전트의 비정상적 종료를 발견하면 가장 최근의 체크포인트로부터 이동 에이전트를 복원한다. 만약 rear guard가 종료하면 이동 에이전트는 단순히 rear guard를 다시 생성시킨다. 모니터 에이전트의 운영은 이동 에이전트 별로 서로 다른 복원 정책을 둘 수 있는 장점이 있지만 모니터 에이전트와 이동 에이전트간의 잦은 메시지 교환이 발생하는 단점이 있다. 두 번째 방법인 플랫폼 기반의 기법은 Voyager, Concordia 등의 플랫폼에서 사용하는 방법으로 체크포인트의 관리 및 복원을 이동 에이전트가 방문한 플랫폼이 수행한다. 이 기법에서 이동 에이전트는 정기적으로 자신의 체크포인트를 파일 시스템에 저장을 하는데, 플랫폼이 이동 에이전트의 종료 사실을 발견했을 때 플랫폼이 가지고 있는 가장 최근의 체크포인트를 이

용하여 이동 에이전트를 복원한다. 이 기법은 메시지 교환은 없지만 모든 이동 에이전트에게 한 가지 복원 정책만을 제공한다. 그리고 플랫폼마다 서로 다른 복원 정책을 사용할 수 있으므로 일관된 복원 정책 수립이 어렵다.

체크포인트를 이용하는 경우 체크포인트의 보안 문제가 발생할 수 있다. 체크포인트는 파일시스템에 저장되므로 다른 이동 에이전트나 프로그램이 이동 에이전트의 내부 데이터를 훔쳐보거나 변조할 수 있다. 이러한 훔쳐 보기와 변조를 막기 위해 체크포인트의 암호화가 필요하며 Concordia가 이러한 기능을 제공하고 있다.[6] Concordia에서 이동 에이전트가 플랫폼에 도착하면, 해당 플랫폼은 도착한 이동 에이전트를 위한 대칭키를 생성하여 이동 에이전트에게 전달한다. 이동 에이전트는 플랫폼에게서 받은 대칭키를 이용하여 체크포인트를 암호화한다. 그리고 체크포인트를 암호화하는데 사용한 대칭키는 해당 플랫폼의 공개키로 암호화되어 이동 에이전트의 체크포인트를 저장할 때 같이 저장된다. 시스템이 고장으로 인해 다운되었다가 다시 복원되어 기동될 때 체크포인트를 이용하여 이동 에이전트들을 복원한다. 이동 에이전트를 체크포인트로부터 복원할 때는 암호화할 때 사용한 대칭키를 알아야만 한다. 이 대칭키는 플랫폼의 공개키로 암호화되어 있으므로 플랫폼의 비밀키를 알아야만 체크포인트를 복원하는데 필요한 대칭키를 얻을 수 있다. Concordia의 결함 허용 기법은 체크포인트의 비밀성을 보장하지만 앞에서 설명한 플랫폼 기반의 체크포인트 운영 기법이 가지는 단점을 가진다.

본 논문에서 제안하는 기법은 이동 에이전트의 여러 비정상적 종료 원인들 중 이동 에이전트의 자체 고장에 의한 종료 상황만을 고려한다. 본 제안 기법은 앞에서 설명한 에이전트 기반의 체크포인트 운영 기법을 플랫폼 기반의 체크포인트 운영 기법에 적용한 것으로서, 모니터 에이전트와의 메시지 교환수를 최소화시키면서 홈 플랫폼에 있는 모니터 에이전트를 통해 이동 에이전트별로 서로 다른 복원 정책 수립이 가능하도록 하였다.

Ⅲ. 보안을 고려한 결함 허용성을 가지는 이동 에이전트 이주 기법

3.1 Itinerary 객체 관리 및 키 관리

이동 에이전트가 방문할 각 목적지들의 정보는 Itinerary 객체 I에 저장된다. I는 하나의 방문지에 대한 정보를 Itinerary_Item 객체에 구성하고 이들의 리스트를 가진다. Itineary_Item 객체는 I'으로 나타낸다. 또한 I는 이동 에이전트가 현재 어느 위치에서 수행중인지를 나타내는 c_{MA}를 가지는데, 이 값은 I에 있는 I' 리스트의 인덱스 값으로 표현한다. 임의의 플랫폼 k에 대한 Itinerary_Item은 I'_k로 표기하며, 다음과 같은 정보를 가진다.

$$I'_k(A_k, F_k, V_k, R_k, MD_k)$$

여기서 A_k는 플랫폼 k의 주소, F_k는 플랫폼 k에서 수행할 함수, V_k는 방문여부, R_k는 실행 성공여부, MD_k는 플랫폼 k에서 이동 에이전트의 체크포인트를 입력받아 생성한 메시지 다이제스트이다.

I에 I'를 채우는 방법은 다음 세가지 방법이 가능하다.

1) 이동 에이전트 개발자가 프로그램 작성시에 직접 I'를 기술하거나,
2) 홈 플랫폼에게 I'를 만들어 줄 것을 요구하거나,
3) 다른 플랫폼을 방문하였을 때 해당 플랫폼에게 I'을 추가해 줄 것을 요구할 수 있다.

첫번째 방법은 이동 에이전트 개발자가 자신이 알고 있는 플랫폼의 정보를 이용해서 프로그램 작성시에 직접 I'들을 채우는 방법이다. 이 방법은 개발자의 플랫폼에 대한 정보에 의존하므로 한계를 가진다. 두 번째 방법은 이동 에이전트가 실행 중에 홈 플랫폼에게 I'를 만들어 I를 채워줄 것을 요청할 수 있다. 이 방법은 fillItinerary(Itinerary it, int max) API를 호출하여 I를 구성한다. fillItinerary API는 플랫폼이 관리하는 이동 에이전트 플랫폼의 리스트에서 max개 만큼 I'를 만들어 I를 채운다. 세 번째 방법은 이동 에이전트가 다른 플랫폼을 방문하였을 때 이동 에이전트 자신이 가지고 있는 I'의 리스트를 보여주고 이들 정보 외의 추가 I'를 요청할 수 있다. 만약 새로운 I'이 추가되었다면, 추가된 I'를 홈 플랫폼으로 전송하여 홈 플랫폼에서 관리하고 있는 I를 갱신하고, 이들 정보 중 홈 플랫폼 자신이 가지고 있지 않은 정보가 있다면 플랫폼 정보에 등록한다.

홈 플랫폼은 I를 관리하고 이동 에이전트의 위치 추적 및 결함 허용 기능을 제공하기위해 AHP(Agent-HomeProxy)를 운영한다. AHP는 에이전트가 생성될 때 같이 생성되며 이동 기능이 없는 정적인 에이전트 프록시이다. AHP는 이동 에이전트가 다른 플랫폼으로 이주하더라도 홈 플랫폼에 계속 남아서 I를 관리한고, 이동 에이전트는 이주할 때 AHP가 가지는 I의 복사본을 가지고 이주한다. AHP가 관리하는 I를 I_{AHP}로, 이동 에이전트가 관리하는 I를 I_{MA}로 표기한다. AHP는 이동 에이전트가 이주할 때마다 통신하여 해당 방문지의 I' 정보를 갱신하고, 이동 에이전트의 현재 위치 c_{MA}를 조정한다. 또한 이동 에이전트가 다른 플랫폼에서 비정상적으로 종료했을 때 해당 이동 에이전트를 복원하여 재기동 시키는 책임을 가진다.

이동 에이전트는 결함 허용 기능을 위해 방문한 플랫폼을 떠나기 전에 자신의 체크포인트를 파일 시스템에 저장한다. 이동 에이전트의 체크포인트는 C_{MA}로 표기한다. 이동 에이전트가 C_{MA}를 저장할 때 C_{MA}의 암호화를 위한 자신의 공개키가 필요하다. 이를 위해 홈 플랫폼은 이동 에이전트를 생성할 때 이동 에이전트의 공개키와 비밀키의 쌍을 생성한다. 이동 에이전트의 공개키와 비밀키는 각각 K^{+}_{MA}, K_{MA}로 표기한다. 이동 에이전트의 내용은 다른 프로그램에게 노출될 수 있으므로 이동 에이전트는 공개키 K^{+}_{MA}만을 가지고 이주하며, 이동 에이전트의 비밀키 K_{MA}는 이동 에이전트의 홈 플랫폼이 관리하도록 한다. 따라서 홈 플랫폼은 이동 에이전트의 K^{+}_{MA}, K_{MA}를 안전하게 저장할 수 있는 객체가 필요하다. 홈 플랫폼은 이동 에이전트의 K^{+}_{MA}, K_{MA}쌍을 관리하기 위해 KeyDictionary 객체를 가진다. KeyDictionary 객체는 $KDic$으로 표기한다. $KDic$의 각 항목은 이동 에이전트의 ID_{MA}와 K^{+}_{MA}, K_{MA}를 쌍으로 가진다. 여기서 이동 에이전트의 ID_{MA}는 이동 에이전트 식별을 위해 홈 플랫폼이 부여한 단순 순차 번호이다. 이동 에이전트의 K^{+}_{MA}, K_{MA}는 보안을 위해 플랫폼의 공개키 K^{+}_{P}로 암호화된 $\{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}}$로 저장된다. $KDic$의 한 항목의 내용은 다음과 같이 구성된다.

$$KDic\ (ID_{MA},\ \{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}})$$

결국 이동 에이전트의 비밀키 K_{MA}를 얻는 것은 홈 플랫폼의 비밀키 K_P를 알고 있는 경우에만 가능하다.

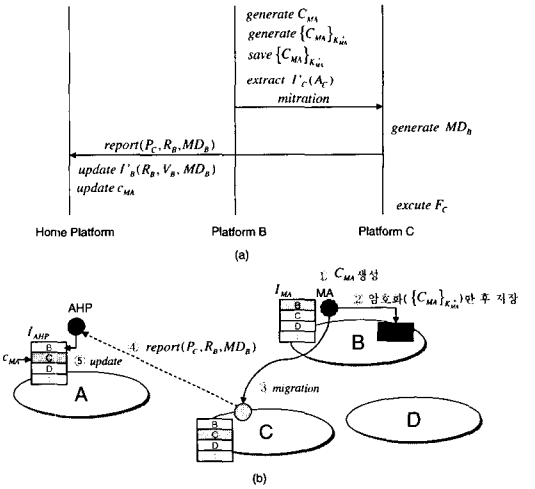
$$\{K^{+}_{MA},\ K_{MA}\}_{K^{+}_{P}} \xrightarrow{K_P} K_{MA}$$

$KDic$에 저장된 이동 에이전트의 K_{MA}은 K^{+}_{MA}로 암호화된 체크포인트 $\{C_{MA}\}^{+}_{K_{MA}}$를 복호할 때 이용된다.

3.2 사전 이주 경로 설정에 의한 이주 기법

사전 이주 경로 설정에 의한 이주에서는 이동 에이전트가 이주를 시작하기 전에 이주 경로가 앞에서 설명한 I의 구성 방법에 의해서 미리 구성되고, 이동 에이전트는 I'에 기술된 각 방문지를 차례로 방문한다. 이동 에이전트가 홈 플랫폼을 떠나 다른 플랫폼으로 이주할 때, 이동 에이전트는 AHP가 관리하고 있는 I_{AHP}의 복사본 I_{MA}을 가지고 이주한다. 이동 에이전트가 홈 플랫폼이 아닌 다른 플랫폼에서 이주를 하는 경우 처리 절차는 [그림 1]과 같다.

[그림 1]에서 이동 에이전트의 홈 플랫폼은 A이고, 이동 에이전트가 플랫폼 B, C, D를 차례로 방문하는 것으로 가정한다. 이동 에이전트는 이주 코드를 실행할 때, 먼저 자신의 실행 이미지를 체크포인트 C_{MA}로 저장한다. 이동 에이전트가 C_{MA}를 저장할 때 그냥 파일 시스템에 저장하면, 다른 프로그램이



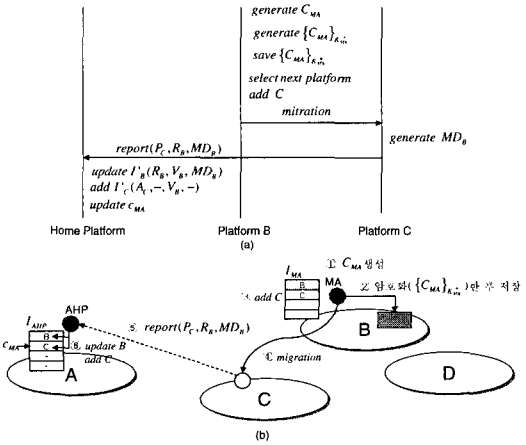
〔그림 1〕 Itinerary의 사전 경로에 의한 이동 에이전트의 이주 절차

이동 에이전트의 개인 정보를 쉽게 볼 수 있고, 변조도 할 수 있다. 이를 막기 위해 이동 에이전트의 C_{MA}를 저장할 때 자신의 공개키 K^{+}_{MA}를 이용하여 암호화한 $\{C_{MA}\}_{K^{+}_{MA}}$를 파일 시스템에 저장한다. 체크포인트를 저장한 후 I_{MA}에 지정된 다음 방문지의 플랫폼 주소를 추출하여 그 곳으로 이주한다. 새로운 방문지 플랫폼으로 이주가 완료되면 이동 에이전트는 먼저 자신의 이주 사실을 AHP에게 보고하는데, 이때 새로 방문한 플랫폼 C의 주소 A_C와 이전 방문 플랫폼 B에서의 수행 성공 여부 R_B와 메시지 다이제스트 MD_B를 보고한다. 이 보고를 받은 AHP는 I_{AHP}의 이동 에이전트의 현재 위치 c_{MA}를 I'_C로 조정하고, I'_B 항목 중 R_B, MD_B 항목을 보고 받은 내용으로 수정한다. 특히 MD_B는 추후에 이동 에이전트를 복원할 때 수신한 C_{MA}의 인증을 위해 사용된다.

이동 에이전트는 위의 과정을 I에 기술된 모든 I'의 플랫폼으로 이주할 때 마다 수행하고, 더 이상 I' 항목이 없을 때 홈 플랫폼으로 되돌아온다. 정상적으로 모든 작업이 완료되면 AHP는 I'에 기술된 모든 플랫폼에게 $DeleteCheckpoint(ID_{MA})$ 메시지를 보내어 해당 이동 에이전트의 체크포인트를 모두 삭제하도록 한다.

3.3 이동 에이전트의 자율적 목적지 결정에 의한 이주 기법

이동 에이전트의 이주 경로를 항상 미리 설정하여 이주할 필요는 없다. 때에 따라서는 이동 에이전트가 자율적이고, 능동적으로 목적지를 선정하여 이주할 경우도 있다. 이동 에이전트가 자율적으로 이주 목적지를 선정하는 이주 기법의 절차는 [그림 2]와 같다.



〔그림 2〕 이동 에이전트의 자율적 목적지 결정에 의한 이주

초기에 이동 에이전트와 AHP가 홈 플랫폼에서 생성되었을 때 AHP는 비어있는 I_{AHP}를 가진다. 이동 에이전트가 홈 플랫폼을 떠나 다른 플랫폼으로 이주할 때 최초의 I'가 I_{AHP}에 하나 추가 되고 이동 에이전트는 I_{AHP}를 복사한 I_{MA}를 가지고 이주한다. 이동 에이전트가 방문한 플랫폼에서의 작업을 끝내고 다른 플랫폼으로 이주할 것을 결정하면, 이동 에이전트는 체크포인트를 K^{+}_{MA}로 암호화하여 $\{C_{MA}\}_{K^{+}_{MA}}$를 파일 시스템에 저장한다. 이동 에이전트의 다음 목적지가 결정되지 않은 상태이므로 현재 플랫폼에게 I_{MA}에 기술된 목적지들 즉, 이전에 방문한 적이 있는 목적지들을 제외한 새로운 목적지를 선정해줄 것을 요청한다. 이주 목적지 선정 요청을 받은 플랫폼은 새로운 목적지를 선정하여 이동 에이전트에게 알려주고, 이동 에이전트는 새로운 목적지에 대한 I'를 I_{MA}에 추가한 후 이주한다. 새로운 목적지로 이주한 이동 에이전트는 먼저 이전 플랫폼에서의 메시지 다이제스트 MD_B를 만들고, AHP에게 이주 완료 사실을 $Report(P_C, R_B, MD_B)$ 메시지로 보고한다. 이때 보고의 내용으로 새로운 방문지 P_C와 이전 방문지에 대한 R_B와 MD_B를 보고한다. 보고를 받은 AHP는 I_{AHP}의 I'_B를 수정하고, 새로운 I'_C 항목을 추가한다. 이 과정을 이동 에이전트가 다른 플랫폼으로 이주할 때마다 수행함으로서 I_{AHP}에 이동 에이전트가 이주한 현재까지의 이주 경로 정보들과 각 방문지에서의 수행 성공 여부에 대한 정보, 메시지 다이제스트 정보들이 저장된다.

이동 에이전트는 위의 과정을 이주할 때 마다 수행하고, 이주 종결 조건이 만족되면 홈 플랫폼으로 되돌아온다. 정상적으로 모든 이주가 완료되면 AHP는 I_{AHP}에 기술된 모든 플랫폼에게 $DeleteCheckpoint$ (ID_{MA}) 메시지를 보내어 해당 이동 에이전트의 체크포인트를 모두 삭제하도록 한다.

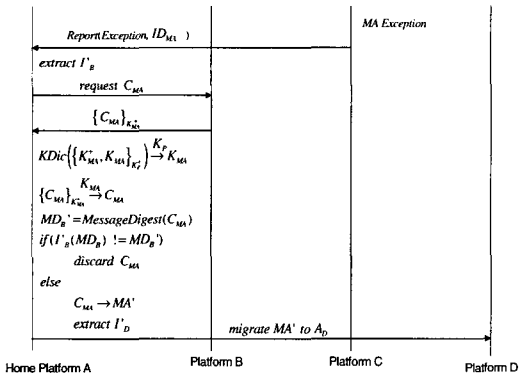
앞에서 설명한 사전 이주 경로 설정은 프로그래머가 이동 에이전트의 이주 경로를 미리 결정할 수 있는 경우에 사용할 수 있는 기법이다. 이에 반해 자율적 목적지 결정은 이동 에이전트의 이주 경로를 사전에 설정할 수 없는 경우, 즉 방문한 플랫폼에서의 처리 결과에 따라 이주 목적지가 달라져야하는 상황에서 사용할 수 있는 기법이다. 개발하려는 이동 에

이전트의 작업 성격에 따라 위의 두가지 방법을 선택적으로 사용하도록 하여 이동 에이전트의 이주 정책 수립에 유연성을 가지도록 하였다.

3.4 체크포인트를 이용한 이동 에이전트의 복원

임의의 플랫폼에서 이동 에이전트가 비정상적으로 종료하면, 이 사실은 이동 에이전트의 홈 플랫폼으로 통보가 된다. 이동 에이전트의 비정상적 종료 메시지를 받은 AHP는 이동 에이전트가 플랫폼을 방문했을 때 저장해놓은 체크포인트를 수신하고 메시지 다이제스트를 이용한 인증을 거친 후 이동 에이전트를 복원하여 실행을 재개시킨다. 이동 에이전트의 복원 절차는 [그림 3]과 같다.

[그림 3]에서 이동 에이전트의 이주 경로가 I에 I'_B, I'_C, I'_D순으로 들어있다고 가정하고, 플랫폼 C에서 이동 에이전트가 비정상적으로 종료했다고 가정한다. 플랫폼 C에서 이동 에이전트가 비정상적으로 종료한 경우, 플랫폼 C는 이동 에이전트의 홈 플랫폼 A에게 *Report*(*Exception*, ID_{MA})를 전송한다. 플랫폼 A가 종료한 이동 에이전트의 AHP에게 *Report*(*Exception*, ID_{MA})을 알리면, AHP는 이동 에이전트의 복원 작업을 시작한다. 먼저 AHP는 I_{AHP}에서 이동 에이전트가 종료한 플랫폼 I'_C을 찾아내어 에러 발생 사실을 설정하고, 이전에 방문한 플랫폼 중 성공적으로 실행을 마친 마지막 플랫폼인 I'_B을 찾는다. 찾아낸 I'_B의 플랫폼에게 ID_{MA}의 체크 포인트를 전송해 줄 것을 요청한다(*Request* C_{MA}). 플랫폼 B는 암호화된 체크포인트 $\{C_{MA}\}^{+}_{K_{MA}}$를 AHP에게 전송한다. AHP는 수신한 $\{C_{MA}\}^{+}_{K_{MA}}$을 복원해 줄 것을 플랫폼에게 요청한다. 플랫폼은 *KDic*에서 이동 에이전트의 비밀키 K_{MA}를 찾아내고, 이 키를 사용하여 C_{MA}를 복호한다. 플랫폼은 수신된 C_{MA}의 내용이 올바른지 인증하기위해서 복호된 C_{MA}를 입력으로 한 메시지 다이제스트 MD_B'를 생성한다. 만약 MD_B'의 내용이 I'_B에 저장되어있는 MD_B의 내용과 일치하지 않다면 잘못된 체크포인트로 간주하고 C_{MA}를 파기한다. 만약 메시지 다이제스트가 일치한다면 이동 에이전트를 복원하고 다음 방문지 I'_D로 이주시킴으로서 복원 과정을 마친다.



[그림 3] 체크포인트를 이용한 이동 에이전트 복원 절차

이동 에이전트가 모든 플랫폼을 방문하여 작업를 완료하고 홈 플랫폼으로 돌아왔다면 AHP는 I_{AHP}에 기술된 모든 플랫폼에게 *DeleteCheckpoint*(ID_{MA}) 메시지를 보내어 해당 이동 에이전트의 체크포인트를 모두 삭제하도록 한다.

Ⅳ. 체크포인트의 안전성 증명

이동 에이전트가 각 방문지의 파일 시스템에 저장하는 체크포인트 C_{MA}을 암호화하지 않고 단순히 저장한다면, C_{MA}의 내용이 다른 프로그램에 노출되어 이동 에이전트의 개인 정보가 노출되거나 변경될 수 있다. 이번 장에서는 C_{MA}에 대한 가능한 공격들과 이들 공격에 대한 안전성을 증명한다.

이동 에이전트에 대한 공격 중에서 공격자가 현재 실행 중인 이동 에이전트의 공개키 K^{+}_{MA}를 공격자의 공개키 $K^{+}_{attacker}$로 대치하는 공격이 가능하다. 하지만 이 문제는 실행 중인 이동 에이전트의 보호 문제로서 본 논문의 범위에서 벗어나는 주제이므로 고려하지 않는다.

4.1 다른 프로그램이 체크포인트의 내용을 읽으려고 하는 경우

이동 에이전트는 자신의 공개키 K^{+}_{MA}를 사용하여 C_{MA}를 암호화하여 저장하므로 C_{MA}의 내용을 읽으려면 이동 에이전트의 비밀키 K_{MA}가 필요하다. 이동 에이전트의 K_{MA}는 홈 플랫폼의 *KDic*에 저장되는데, *KDic*의 각 항목은 다음의 형태를 가진다.

$$KDic(ID_{MA}, \{K^{+}_{MA}, K_{MA}\}_{K^{+}_{p}})$$

$$KDic(ID_{MA}, \{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}}) \xrightarrow{ID_{MA}} \{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}}$$
$$\{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}} \xrightarrow{K_P} K_{MA}$$
$$\{C_{MA}\}_{K^{+}_{MA}} \xrightarrow{K_{MA}} C_{MA}$$

〔그림 4〕 이동 에이전트의 체크 포인트 복원을 위한 복호 절차

여기서 ID_{MA}는 대상 이동 에이전트의 ID이며, 이동 에이전트의 공개키 K^{+}_{MA}, 비밀키 K_{MA}는 홈 플랫폼의 공개키 K^{+}_{P}를 사용하여 암호화한 후 $\{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}}$로 저장된다. $\{C_{MA}\}_{K^{+}_{MA}}$를 복호하여 C_{MA}를 얻는 과정은 [그림 4]와 같다.

따라서 홈 플랫폼의 비밀키 K_P를 알아야만 이동 에이전트의 비밀키 K_{MA}를 얻을 수 있다. 홈 플랫폼 이외의 다른 프로그램은 홈 플랫폼의 비밀키 K_P를 알지 못하므로 이동 에이전트의 비밀키 K_{MA}를 얻을 수 없고, 이동 에이전트의 체크포인트 C_{MA}의 내용을 읽을 수 없다.

4.2 다른 프로그램이 이동 에이전트의 체크포인트를 변조하려고 하는 경우

다른 프로그램이 이동 에이전트의 체크 포인트 C_{MA}을 얻어 이를 변조하여 $\{C_{MA}'\}^{+}_{K_{MA}}$를 만들려고 한다면 K_{MA}을 알고 있을 경우에 만 가능하다. K_{MA}을 얻기위해서는 $\{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}} \xrightarrow{K_P} K_{MA}$을 수행해야 하는데, (1)의 경우처럼 홈 플랫폼 이외의 다른 프로그램은 K_P를 알지 못하므로 이동 에이전트의 비밀키 K_{MA}를 얻지 못하고, 따라서 C_{MA}의 내용을 변경할 수 도 없다.

4.3 홈 플랫폼으로 가장한 다른 폴랫폼이 이동 에이전트의 체크포인트를 요구하는 경우

홈 플랫폼이외의 다른 플랫폼(P')이 홈 플랫폼으로 가장하고 이동 에이전트의 C_{MA}를 요구할 수 있다. 이 경우 P'가 받게 되는 체크포인트는 $\{C_{MA}\}_{K^{+}_{MA}}$이다. P'가 이동 에이전트의 체크포인트 $\{C_{MA}\}_{K^{+}_{MA}}$를 얻는다 하더라도 홈 플랫폼의 $KDic(ID_{MA}, \{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}})$의 내용을 알지 못하므로 이동 에이전트의 비밀키 K_{MA}를 얻을 수 없고 따라서 C_{MA}을 얻을 수 없다.

4.4 AHP에게 $\{C_{MA}\}_{K^{+}_{MA}}$대신 $\{C_{MA'}\}_{K^{+}_{MA}}$을 전송한 경우

AHP가 이동 에이전트의 체크포인트를 요구했을 때, AHP는 다른 이동 에이전트(예를 들어 악성 에이전트)의 체크포인트 $C_{MA'}$을 K^{+}_{MA}로 암호화한 $\{C_{MA'}\}_{K^{+}_{MA}}$을 수신할 수 있다. 따라서 홈 플랫폼은 수신한 체크포인트로부터 이동 에이전트를 복원하기 전에 이 체크포인트가 올바른 체크포인트인지 검사해야한다. 체크포인트의 인증은 수신한 체크포인트 $C_{MA'}$를 입력으로 하여 생성한 메시지 다이제스트 MD'와 이동 에이전트가 각 방문지에서 C_{MA}을 입력으로 하여 생성한 메시지 다이제스트 MD_K(I_{AHP}의 I'_K에 저장되어 있음)를 비교하여 이루어진다. 이 두 메시지 다이제스트가 다르다면 수신한 체크포인트가 올바르지 않다는 것을 확인할 수 있고 이 경우 수신한 체크포인트를 파기한다. 수신한 체크포인트의 인증 절차는 [그림 5]와 같다.

체크포인트를 이용한 이동 에이전트 복원의 안전성은 결국 홈 플랫폼의 비밀키 K_P와 각 방문지에서 생성된 메시지 다이제스트 MD의 검사에 의해 보장된다.

$$C \rightarrow A : \{C_{MA'}\}_{K^{+}_{MA}}$$
$$A : \{C_{MA'}\}_{K^{+}_{MA}} \xrightarrow{K_{MA}} C_{MA'}$$
$$MessageDigest(C_{MA'}) \rightarrow MD'$$
$$I'_C \rightarrow MD_C$$
$$if\,(MD_C == MD')$$
$$\quad recover\,MA\,from\,C_{MA'}$$
$$else$$
$$\quad discard\ C_{MA'}$$

〔그림 5〕 메시지 다이제스트를 이용한 체크 포인트 인증

V. 결 론

본 논문에서 이동 에이전트의 이주 경로를 제어하기 위해 사전 이주 경로 설정 기법과 이동 에이전트의 자율적 목적지 설정 기법을 제안하였다. 두 기법 모두 이동 에이전트의 이주 경로가 AHP의 I_{AHP}에 저장되며 이동 에이전트의 현재 위치를 추적할 수 있도록 하였다. 각 방문지에 대한 정보 I'는 이동 에

이전트가 특정 플랫폼에서 비정상적 종료를 하였을 때 안전한 복원을 할 수 있도록 메시지 다이제스트를 포함하고 있다. 이동 에이전트가 다른 플랫폼으로 이주하기 전 자신의 체크포인트를 저장할 때 체크포인트의 비밀성을 보장하기 위해 자신의 공개키 K^{+}_{MA}를 사용하여 암호화한 후 저장한다. 그리고 새로운 플랫폼으로 이주가 완료되면 이동 에이전트는 체크포인트에 대한 메시지 다이제스트를 AHP에게 보낸다. 이동 에이전트가 비정상적으로 종료했을 때, AHP는 암호화된 체크포인트를 수신하여 이를 복호하고, 복호한 체크포인트를 입력으로 하여 메시지 다이제스트를 생성한다. 이 메시지 다이제스트를 이전에 받은 메시지 다이제스트와 비교함으로서 올바른 체크포인트인지 인증할 수 있다. 체크포인트의 복호화에는 이동 에이전트의 비밀키 K_{MA}가 필요하며, K_{MA}는 홈 플랫폼의 비밀키 K_P를 알고 있어야만 $KDic(ID_{MA}, \{K^{+}_{MA}, K_{MA}\}_{K^{+}_{P}})$에서 추출할 수 있으므로 체크 포인트의 안전성이 보장된다.

참 고 문 헌

[1] V. A. Pham and A. Karmouch, "Mobile Software Agents: An Overview", IEEE Communications Magazine, Jul. 1998.

[2] W. R. Cockayne and M. Zyda, "Mobile Agents", Manning Publications Co., 1999.

[3] http://www.trl.ibm.co.jp/aglets/

[4] ObjectSpace, Inc. "ObjectSpace Voyager Core Package Technical Overview", Technical report, ObjectSpace, Inc., July 1997.

[5] N. Karnik, "Security in Mobile Agent Systems", Ph. D. dissertation, University of Minnesota, 1998.

[6] J. Peng, B. Li, "Mobile Agent in Concordia", http://www.cs.albany.edu/~mhc/Mobile/Concordia.pdf

[7] N. M. Karnik and A. R. Tripathi, "Design Issues in Mobile-Agent Programming Systems", IEEE Concurrency, Jul.-Sep. 1998.

[8] W. Lin and Q. Yanping, "An Analytic Survey of Fault-Tolerance Mobile Agent Architecture", http://www.cs.concordia.ca/agent/FTMA/FTMAsurvey.pdf

[9] S. Pleisch and A. Schiper, "Modeling Fault-Tolerant Mobile Agent Execution as a Sequence of Agreement Problems", In Proc. of the 19th IEEE Symposium on Reliable Distributed Systems(SRDS), pages 11~20, Nuremberg, Germany, Oct. 2000.

[10] A.R. Tripathi, T. Ahmed, N. M. Karnik, "Experiences and Future Challenges in Mobile Agent Programming", Microprocessors & Microsystems (Elsevier), vol.25 no.2, April 2001

[11] D. Johansen, R. Renesse, and F. B. Schneider, "Operating System Support for Mobile Agents", In Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems(HotOS-V), pp.42~45, May 1995.

〈著者紹介〉

김 구 수 (Gu Su Kim) 학생회원

1994년 2월 : 성균관대학교 정보공학과 졸업(학사)
1996년 2월 : 성균관대학교 정보공학과 졸업(석사)
2002년 3월~현재 : 성균관대학교 정보통신공학부 박사과정
<관심분야> 분산컴퓨팅, 이동 에이전트

엄 영 익 (Young Ik Eom) 정회원

1983년 2월 : 서울대학교 계산통계학과 졸업(학사)
1985년 2월 : 서울대학교 대학원 전산과학과 졸업(석사)
1991년 8월 : 서울대학교 대학원 전산과학과 졸업(박사)
2000년 9월~2001년 8월 : Dept. of Info. and Comm. Science at UCI 방문교수
현재 : 성균관대학교 정보통신공학부 교수
<관심분야> 분산시스템, 이동 컴퓨팅 시스템, 이동 에이전트, 시스템 소프트웨어 등