http://dx.doi.org/10.7236/JIWIT.2012.12.6.33
JIWIT 2012-6-4

모바일 에이전트 시스템의 구성적 임베딩을 위한 컴포넌트 기반의 프레임워크

A Component-Based Framework for Structural Embedding of Mobile Agent System

정원호*, 강남희**

Wonho Chung, Namhi Kang

요 약 유무선 기술의 발달로 다양한 형태의 임베디드 시스템들이 출현하였으며, 이들로 임베딩되는 소프트웨어도, 이제는 경직성(fixedness)보다 오히려 유연성(flexibility)을 더 요구하고 있다. 즉, 기능 및 자원 제약 특성을 가지는 다양한 장치들로 유연하게 임베딩 될 수 있는 특성을 요구하고 있는 것이다. 모바일 에이전트는, 비연결 상태에서의 동작 능력과 높은 비동기성으로 인해 네트워크의 부하와 대기시간을 줄일 수 있는 유용한 분산 기술 중 하나이다. 본 논문에서는, 서로 다른 기능과 자원 제약 특성을 가지는 다양한 장치들로의 구성적 임베딩을 위한 컴포넌트 기반의 모바일 에이전트 프레임워크인 EmHUMAN이 설계, 구현된다. 그것은 3계층의 컴포넌트들로 구성되어 있으며, 그들을 기반으로 임베디드 시스템이 요구하는 기능, 저장 공간, 컴퓨팅 능력 그리고 네트워크 대역폭 등의 자원 특성에 맞춰 구성적 임베딩이 가능한 것이 특징이다. 각 계층의 컴포넌트들은 필요에 따라 추가, 제거, 변경 등의 확장이 가능하다. EmHUMAN은 모바일 에이전트 기반의 분산 시스템 개발을 위한 프레임워크 역할도 하면서, 그 자체가 에이전트 서버로서의 기능도 수행할 수 있으며 유용한 유틸리티를 내장 API로 제공하고 있어 모바일 에이전트 기반의 분산 응용을 하는데 있어 효율성을 제공할 수 있다.

Abstract Rapid evolution of wired and wireless technologies results in various types of embedded systems, and the software to be embedded into those devices now needs the flexibility rather than the fixedness which was well-known property for the embedded software in the past. Mobile agent is one of the useful distributed technologies of reducing network load and latency because of its disconnected operations and high asynchrony. In this paper, a component-based mobile agent framework, called EmHUMAN, is designed and implemented for structural embedding into the devices showing different functions and resource constraints. It consists of 3 layers of components. Based on those components, a structural embedding, considering resource constraints of required functions, amount of storage space, computing power, network bandwidth, ... etc can be performed. The components in each layer can be extended with addition of new components, removing some components and modifying components. EmHUMAN plays the role of a framework for developing mobile agent based distributed systems. It is also a mobile agent system by itself. EmHUMAN provides several utilities as built-in API's, and thus high effectiveness in programming mobile agents can be achieved.

Key Words : Component, Distributed system, Framework, Mobile agent, Structural embedding

*정회원, 덕성여자대학교 디지털미디어학과
**정회원, 덕성여자대학교 디지털미디어학과
접수일자 : 2012년 10월 19일, 수정완료 : 2012년 11월 20일
게재확정일자 : 2012년 12월 14일

Received: 19 October 2012 / Revised: 20 November 2012 / Accepted: 14 December 2012
**Corresponding Author: kang@duksung.ac.kr
Dept. of Digital Media, Duksung Women's University , Korea

Ⅰ. 서 론

과거의 임베디드 소프트웨어 기술은 실시간 수행과 자원의 효과적 사용을 위한 경직성(fixedness)이 주요한 특징이었다. 그것은 로봇제어, 공장자동화 같이 단일의 지정된 목적에 국한되어 응용되어 지던 과거에는 경직성이 고속의 실시간 처리를 위한 주요 설계 개념이었기 때문이었다. 그러나 서로 다른 자원 제약 특성을 가지는 다양한 유형의 임베디드 시스템들이 속속 출현하고 있는 지금, 그들을 위한 임베디드 소프트웨어는 그러한 장치들로의 유연한 임베딩을 위해 경직성이 아닌 유연성과 적응성을 요구받고 있다[2]. 이러한 경향과 더불어, Java 가상 기계의 경우에도, 유연한 임베딩을 위한 연구가 이루어져 새로운 개념의 가상기계들이 개발되었으며[1], 임베디드 운영체제는 Linux처럼, 공개 소스 기반의 임베디드 시스템을 지향하고 있다[2]. 다른 소프트웨어도 예외일 수 없다.

비연결 상태에서의 동작 가능성과 높은 비동기성으로 인한 네트워크 대역폭의 절감 특성 때문에 모바일 에이전트에 기반을 둔 분산 처리 기술이 분산 응용 분야에서 폭넓게 주목 받고 있다. 모바일 에이전트는 네트워크 사용자를 대신하는 사용자 프로그램이며, 네트워크상의 노드들 사이를 자율적으로 이동하면서 사용자를 대신해서 해당 작업을 수행한다. 또한 클라이언트-서버, 애플릿, 서블릿 그리고 맞춤형 코드(Code-On-Demand)를 모두 포함할 수 있는 단일 패러다임을 제공하고 있으며[3,4], 정보 탐색, 무선 센서 네트워크 응용 그리고 전자 모니터링 등에 폭넓게 응용되고 있다[5-10]. 자바의 출현으로 AGLETS, 등과 같은 자바 기반의 모바일 에이전트 시스템들이 개발되어 등장하였으며[3], 분산 응용에 사용되는 미들웨어를 지원하는 하나의 응용 프레임워크로서의 중요한 역할을 담당하고 있다[11,12]. 특히 Java 기반의 모바일 에이전트 시스템은 메쉬(mesh), Ad-hoc 혹은 Peer-to-Peer 등과 같은 유무선 네트워크상에서의 분산 응용을 위한 좋은 프레임워크가 될 수 있다[13,14]. 이러한 응용 프레임워크는 다양한 형태의 자원 제약 특성을 가지는 서로 다른 모바일 장치로 Java와 같은 미들웨어와 더불어 유연성 있게 임베딩 될 수 있어야 한다.

본 논문에서는, 기능과 자원 제약 특성이 다른 여러 유형의 장치로의 유연한 임베딩을 위해 컴포넌트 기반의 모바일 에이전트 시스템인 EmHUMAN이 설계, 구현된다. 그것은 세 계층의 컴포넌트들로 구성되어 있으며 그들로부터 기능, 저장 공간, 컴퓨팅 능력 그리고 네트워크 대역폭 등이 서로 다른 다양한 수준의 모바일 에이전트 시스템을 구성할 수 있어, 여러 형태의 유무선 장치의 자원 제약 특성에 맞춰 구성적 임베딩이 가능한 것이 특징이다. 즉, 지니고 있는 자원의 수용 능력에 따라 그 기능 및 코드 크기를 조정할 수 있어 유연하게 적응이 가능하다. 그리고 각 계층의 컴포넌트들은 필요에 따라 추가, 제거, 변경 등의 확장이 가능하다. EmHUMAN은 모바일 에이전트 기반의 분산 응용 시스템 개발을 위한 프레임워크 역할도 하지만 그 자체가 에이전트 서버로서의 기능도 수행할 수 있다. 다양한 모바일 에이전트 시스템들이 개발되었지만, 모바일 에이전트 기반의 분산 응용을 위한 프로그래밍은 용이하지 않다. 그것은 첫째, 분산 응용을 위해 손쉽게 사용할 수 있는 응용 레벨에서의 고급 유틸리티를 제공하고 있지 않으며, 둘째, 네트워크상에서의 이동성과 네트워크 상황을 고려한 기능들을 제공하고 있지 않아, 각 응용마다 새롭고, 서로 다른 프로그래밍이 요구되고 있어, 그 어려움과 복잡성을 증가시키고 있기 때문이다. EmHUMAN은 모바일 에이전트 기반의 응용 시스템 개발을 위해 파일 탐색, 그룹 어드레싱, 그리고 탐색 대상 정보의 입력 등의 고급 유틸리티를 내장 API로 지원하고 있으며, 주어진 네트워크 환경에 유용하게 적응하기 위한 이동 및 응답 방식들도 제공하고 있다.

2장에서는 본 논문에서 설계된 모바일 에이전트 시스템 프레임워크인 EmHUMAN의 동작 구조와 그 특징들이 기술된다. 3장과 4장에서는 에서는 EmHUMAN을 구성하고 있는 3 계층 컴포넌트가 정의되고, 그들의 조합으로 이루어지는 구성적 임베딩에 관한 개념이 기술된다. 5장에서는 EmHUMAN을 사용하는데 있어서 그 응용 분야와 장단점 등이 기술되고 6장에서는 EmHUMAN기반의 간단한 응용인 파일 탐색 시스템의 프로토타입이 구현된다.

Ⅱ. EmHUMAN의 동작

1. 동작 구조

EmHUMAN의 전체적인 동작 구조가 그림 1에 나타나 있다. 기본 구조는 HumanServer, HumanPlace(또는 HumanMetaPlace), 그리고 HoServer 라고 하는 3개의

주요 서버 부분과 에이전트를 위한 유틸리티를 포함하고 있는 HumanAgent로 구성되어 있다. HumanServer는 사용자 인터페이스 역할을 담당하고 있으며, 에이전트가 방문해야 할 노드의 주소의 지정, 방문을 위한 라우팅 그리고 작업 후 결과 응답 등을 위해 필요한 모든 정보, 예를 들면 노드의 주소, 라우팅 모드, 결과의 응답 모드 등에 관한 정보 설정이 이를 통해 이루어진다. 또 하나의 중요한 기능은 응용 에이전트들의 수행 환경이 되는 Place의 생성이다. HoServer는 네트워크상에서 생성되어 활동 중인 응용 에이전트의 등록 및 제거와 관리가 중요한 특별한 서버이다. 그라허여 네트워크상에서 활동 중인 응용 에이전트에 관한 정보를 필요로 하는 에이전트에게 제공해 줄 수 있다.

EmHUMAN은 프록시 기능도 제공하고 있는데, 이 기능을 담당하고 있는 것이 HumanPlace 혹은 Human-MetaPlace라고 하는 Place이다. 에이전트들의 전송을 포함하는 모든 메시지 통신은 Place를 통해 이루어지는데, 그 과정을 간략히 보여주고 있는 것이 그림 2이다. 더 나아가서 파일 보호, 에이전트 보호 그리고 호스트 보호와 같은 보안 기능을 담당하는 곳도 Place이다. 이러한 Place는 HumanServer에 의해 생성되고 그 이후로 에이전트들의 생성과 그들 간의 통신은 모두 Place에 의해 책임지고 수행된다.

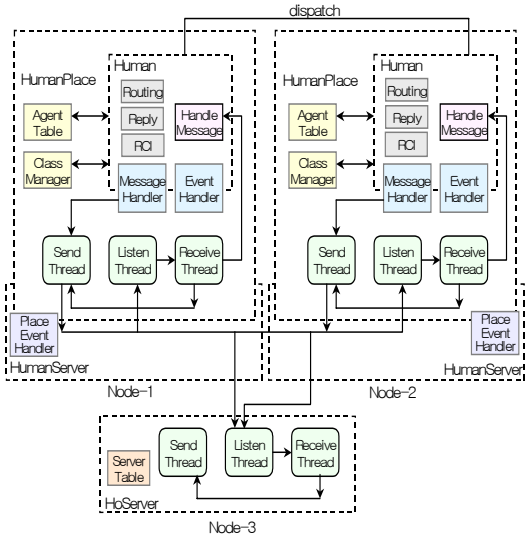

그림 1. EmHUMAN의 동작 구조
Fig 1. Behavioral Architecture of EmHUMAN

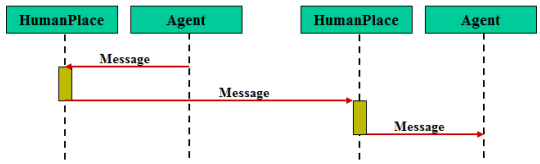

그림 2. 에이전트 간의 메시지 통신
Fig 2. Message communication between agents

2. 유틸리티 및 에이전트 보호

가. 유틸리티

EmHUMAN은 에이전트 프로그래밍에 있어서 유용한 유틸리티 기능을 API로써 제공하고 있다. 그리하여 모바일 에이전트 응용을 위해 필요하지만 구현하기 용이하지 않은 그러한 기능들을 시스템이 제공함으로써 응용의 수월함을 높여주고 있다. 분산 파일 탐색이 그 좋은 예라 할 수 있다. 이는 네트워크상의 어떤 탐색 대상 노드들의 집단에 대한 주소 지정과 원하는 파일의 탐색과 같은 동작들을 필요로 한다. 그리하여 탐색 대상 파일들의 리스트와 탐색 대상 노드들의 주소 세팅을 위한 효율적인 인터페이스도 제공해야 한다. 예를 들면, IP 주소 123.456.789.120부터 123.456.789.129까지 10개의 노드들이 하나의 IP 그룹 주소 123.456.789.12X로 명시될 수 있다. 같은 방법으로 123.456.789.100부터 123.456.789.199까지 100개의 노드는 123.456.789.1XX로 이와 같은 방법으로 특정 집단의 노드들이 하나의 IP 주소로 세팅될 수 있는 것이다. 이는 네트워크상에서 멀티캐스팅처럼 광범위하게 노드들을 처리할 수 있는 유용한 주소 지정 기술이다. 또한, 특정 파일 혹은 파일들의 리스트를 탐색하기를 원한다면, 시스템이 제공하는 인터페이스를 통해 파일 혹은 파일 리스트를 세팅하고, 그 다음 간단하게 시스템에서 제공하는 API인 예를 들면, “CompreFile( )”을 호출하기만 하면 된다.

모바일 에이전트는 주어진 작업을 위해 여러 노드들을 거치게 되는데, 그 거치는 노드들의 리스트를 에이전트의 여정(itinerary)이라고 한다. 이 여정에 관한 정보를 기반으로 에이전트는 자율적으로 네트워크상을 이리저리 이동하며 작업을 수행한다. 그러나 그 여정은 네트워크의 조건 예를 들면, 네트워크 분할, 연결 실패, 노드의 고장 등에 의해 영향을 받을 수 있다. 이처럼, 이동 시점에서 고려될 수 있는 네트워크 환경에 대해 유연하게 대처할 수 있도록, EmHUMAN은 몇 가지 유용한 여정 방식을 제공하고 있다[17]. 모바일 에이전트는 자신에게 주

어진 작업을 원격지 노드에서 수행하고 그 결과를 응답하여야 작업을 완료했다고 할 수 있으며, 다음 노드로 이동하여 작업을 시작할 수 있다. 그러나 결과물 응답 시점에서 돌발적인 상황이 발생할 수 있는데, 예를 들면, 1) 부담스러운 결과물의 크기, 2) 작업 유형에 따른 결과물에 대한 다른 방식의 응답 필요, 3) 결과물 응답 대상 노드로의 접속 불가능 상황, 그리고 4) 여정을 맞추기 위한 작업 에이전트의 신속한 이동의 필요 등이 있을 수 있다. 예를 들어, 파일 탐색과 같은 분산 응용의 경우, 그 탐색 결과는 지역적으로 분산되어 있는 여러 노드들 혹은, 신속성을 필요로 한다면, 그들 중 가능한 어느 한 노드로 신속하게 결과물을 전송해 주고 다음 작업을 위하여 다른 노드로 이동하는 것이 효율적일 경우도 있는 것이다. 이처럼, 적시에 정보를 얻어내는 것이 중요하며, 또한 협동 작업의 유형이 많아서, 결과물을 응답해야 할 노드가 서로 다른 분산 응용의 경우 유연하게 대처할 수 있다는 특징을 가지고 있다. EmHUMAN은 결과물 응답 시점에서 고려될 수 있는 여러 상황에 대해서도 필요시 유연하게 대처할 수 있도록 응답 방식들을 지원하고 있다[17].

나. 에이전트의 보호

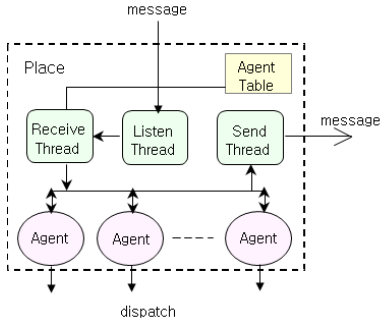

그림 3. Place 보호 모드에서 활동 중인 에이전트
Fig 3. Human agents under the protection PP

EmHUMAN은 Place와 Metaplace에 기반을 둔 2가지 유형의 보호 모드를 제공하고 있다. Place는 그 상에서 활동 중인 원격 혹은 로컬 에이전트들 간의 통신을 책임지고 있으며, 그들과 호스트를 외부의 불분명한 에이전트 혹은 메시지로부터 보호하는 책임도 지고 있다. 이런 경우의 보호 모드를 Place 보호 모드(Protection by Place: PP) 라고 하며 그 관계를 보여주고 있는 것이 그림 3이다. 그리하여 다른 노드들로부터 인입되는 여러 메시지들은 Place에서 필터링되어 걸러진다. 그러므로 EmHUMAN에서 지원하는 프로토콜(보호 필드)을 사용하지 않는 불분명한 에이전트 혹은 메시지들은 모두 해당 Place에서 걸러지게 된다.

Place가 에이전트들의 수행 환경이라면, 그림 4에 보여준 바와 같이, Metalplace는 Place들의 수행 환경이다. 즉 Place들을 위한 Place라고 할 수 있다.

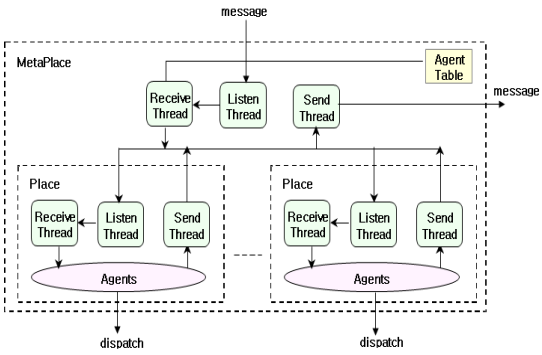

그림 4. Metaplace 보호 모드에서 활동 중인 에이전트
Fig 4. Human agents under the protection PM

동작 중인 Place는 HumanServer 상에 존재할 수도 있고, 아니면 Metaplace상에 존재할 수도 있다. 그런 경우 다른 노드들로부터 인입되는 메시지들은 Metaplace에서 1차로 자체 프로토콜에 의해 필터링 된다. 그러므로 EmHUMAN에서 지원하는 Metaplace 프로토콜 사용하지 않는 불분명한 에이전트 혹은 불분명한 메시지들은 해당 Metaplace에서 걸러지게 된다. 이런 경우의 보호 모드를 Metaplace 보호 모드(Protection by Metaplace: PM) 라고 한다. Metaplace를 통과한 메시지 혹은 에이전트는 목적하는 Place 상으로 탑재되기 위해서는 해당 Place에 의해 이루어지는 또 한번의 보호 모드, 즉 PP를 거쳐야 한다. 그러므로 Metaplace가 존재하는 경우, 에이전트는 2중의 보호 체계를 지원 받음으로 인해 더욱 안전하게 활동할 수 있다.

Ⅲ. EmHUMAN의 컴포넌트 계층

1. 컴포넌트 계층 구조 및 특성

프레임워크 같은 시스템은 어떤 환경에서 어떻게 사용될지 알기 어려우므로 다양한 환경에 효율적으로 대처할 수 있도록 설계되어야 하는데, 이를 위한 바람직한 구

조가 컴포넌트 기반의 구조라 할 수 있다. EmHUMAN은 그러한 프레임워크이며 [그림-5]에 보여준 바와 같이 프리미티브(Primitive) 계층, 매크로(Macro) 계층, 그리고 시스템(System) 계층 이렇게 3 계층 구조로 이루어진 컴포넌트 계층 구조를 가지고 있다.

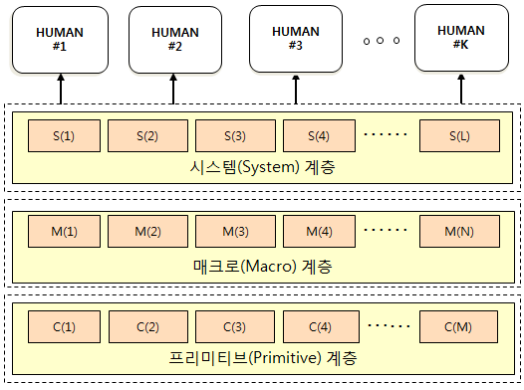

그림 5. EmHuman의 컴포넌트간 계층 구조
Fig 5. Layered architecture of EmHuman Components

그리하여 요구되는 용도 및 기능에 따라 하위 계층의 컴포넌트를 사용하여 상위 계층의 컴포넌트를 구성하는 과정을 단계별로 거치면서 최상위 계층에 도달하면 목적하는 사양의 모바일 에이전트 시스템 HUMAN을 얻을 수 있게 된다.

최하위 계층인 프리미티브 계층을 구성하는 각 컴포넌트는 상위 계층의 컴포넌트의 동작을 위한 서비스 기능을 지원하며, 독자적 수행 능력을 가지고 있지 못한 수동 컴포넌트들이다. 프리미티브 컴포넌트들을 사용하여 구성되는 매크로 컴포넌트 각각은 자체적으로 독립적인 기능을 가지고 있으며 자체로 그 기능을 수행할 수 있는 컴포넌트이며, 하드웨어 플랫폼과 대응하여 가동시킬 수도 있다는 점에서 프리미티브 컴포넌트와는 차별된다. 프리미티브 계층의 컴포넌트들을 기반으로, 구성 가능한 매크로 컴포넌트는 다양할 수 있다. 어떠한 프리미티브들을 사용하여 어떠한 매크로 컴포넌트를 구성하느냐 하는 것은 목표 응용 시스템에 따라 달라질 수 있다. 마찬가지로, 매크로 컴포넌트들을 사용하여 구성되는 시스템 컴포넌트는 능동적으로 독자적인 기능을 수행하는 컴포넌트라는 점과 각 시스템 컴포넌트를 구성하고 있는 매크로 컴포넌트들 간에 상호 작용이 존재하고 있다는 점에서 매크로 컴포넌트와는 다르다. 논리적인 관점에서 보면, 매크로 계층의 컴포넌트들이 다양해지면 시스템 계층에서 다양한 컴포넌트들이 구성될 수 있으며, 시스템 계층의 컴포넌트가 다양하게 존재할수록 목표로 하는 응용 시스템 구축 또한 다양한 형태로 구성될 수 있을 것이다.

2. 계층의 구성

최하위 계층인 프리미티브 계층은 각종 Java 클래스들로 구성된다. 매크로 컴포넌트들은 EmHUMAN에서 구현되어 있는 프리미티브 컴포넌트들 간의 Fan-in 혹은 Fan-out 관계에 따라서 구성되어진다. 그러므로 매크로 계층의 컴포넌트들을 구성하고 있는 각 프리미티브들은 서로 간에 밀접하게 연관되어 있어 커플링이 높은 편이다. EmHuman의 기능을 확장 또는 변경하기 위해 새로운 프리미티브 혹은 매크로 컴포넌트들을 추가 혹은 변경할 수 있으며, 이는 계속 변경 수정되어야 하는 에이전트 시스템의 사양에 대해 효율적으로 대처할 수 있도록 한다. 표 1처럼 EmHUMAN의 최하위 계층인 프리미티브 계층은 30여개의 클래스들로 구성되었고, 그들로부터 18개의 매크로 컴포넌트들이 구현되어 있다. 또한, 필요시 더 많은 프리미티브 컴포넌트 혹은 매크로 컴포넌트들을 만들어 구성할 수 있을 것이다.

예를 들어, 매크로 컴포넌트 OLS는 2개의 프리미티브 컴포넌트 HoServer와 HoModel로 구성되어 있으며 서버 시스템 컴포넌트를 구성하는데 사용된다. 그리고 프리미티브 컴포넌트 HoServer 클래스는 HoModel 클래스 내의 메소드들을 호출하여 사용하고 있어 둘 사이에 Fan-in/Fan-out 관계를 보여주고 있다. 매크로 컴포넌트 COM은 2개의 프리미티브 컴포넌트인 HumanAddress와 HumanMessage 클래스로 구성되는데, 이들은 에이전트들의 주소 지정과 에이전트들 간의 통신 등의 기능을 담당하며 서버 시스템 컴포넌트 구성에 공통으로 사용된다. 매크로 컴포넌트 IPF는 MenuPil과 RadioIcon 이라는 2개의 프리미티브 컴포넌트로 구성어 있으며, 에이전트의 방문 경로인 여정리스트 정보의 세팅을 위한 인터페이스 기능을 담당하고 있는데, 이는 유틸리티 시스템 컴포넌트를 위해 사용된다.

매크로 계층의 컴포넌트들로 구성되는 시스템 계층의 컴포넌트는 2가지 유형인 서버 시스템 컴포넌트와 유틸리티 시스템 컴포넌트로 나뉘어진다. 표 1을 보면 EmHUMAN은 18개의 매크로 컴포넌트를 가지고 있으

며, 서버 시스템 컴포넌트 구성을 위해서 11개의 매크로 컴포넌트들이 사용되고 있으며, 유틸리티 시스템 컴포넌트 구성을 위해 7개의 매크로 컴포넌트들이 사용되고 있음을 알 수 있다.

표 1. EmHUMAN의 프리미티브/매크로 계층 컴포넌트
Table 1. Primitive/Macro components in EmHUMAN

Macros	Primitives(class)	Descriptions	
COM	HumanAddress, HumanMessage	COMmon	S
UTL	HumanFile, HumanList, HumanTime	UTiLity	U
OLS	HoServer, HoModel	OnLine-Server	S
BUI	HumanServer	Basic User Intf.	S
IPF	MenuPil, RadioIcon	Itinerary Path intF.	U
RPF	MenuPrl, RadioIcon	Reply Path intF.	U
FPF	MenuFile	File Path intF	U
PEH	HumanPlaceEvent, HumanPlaceListener	Place Event Handler	S
BPF	HumanPlace	Basic Place Ftn	S
BMF	HumanMetaPlace	Basic Metaplace Ftn	S
BCM	HumanClassLoader, HumanClassManager, HumanInputStream, HumanOutputStream	Basic Class Manager	S
EVH	HumanEvent, HumanListener	EVent Handler	S
MSH	HumanMessageEvent, HumanMessageListener	MeSsage Handler	S
BAM	HumanTable	Basic Agent Mngr	S
BAF	Human	Basic Agent Ftn	S
ROM	HumanRouting	ROuting Mode Ftn	U
RPM	HumanReply	RePly Mode ftn	U
RCI	HumanRCI	Remote Class Invoc.	U

IV. 구성적 임베딩

1. 시스템 구성 테이블

매크로 컴포넌트들의 선택적 조합을 기반으로 하여, 시스템 컴포넌트의 생성을 보여주는 테이블을 시스템 구성 테이블이라고 하며, 서버 시스템 구성 테이블(SSCT)과 유틸리티 시스템 구성 테이블(USCT)로 나뉘어진다. 표 1에서 나타냈듯 서버 시스템 컴포넌트를 구성하는데 사용될 수 있는 매크로 컴포넌트는 COM을 포함하여 11개가 있다. 이 11개의 매크로 컴포넌트들을 기반으로, 현재 EmHUMAN에서 구현되어 있는 5개의 서버 시스템 컴포넌트 SS(1) 부터 SS(5)를 구성하여 보여주고 있는 것이 표 2이다.

표 2, 서버 시스템 구성 테이블
Table 2. Server System Configuration Table (SSCT)

	Macros	서버 시스템 컴포넌트 구성 테이블: SS(k)				
		1	2	3	4	5
1	COM	●	●	●	●	●
2	OLS	●				
3	BUI			●		
4	PEH		●	●		●
5	BPF		●			
6	BMF					●
7	BCM		●			●
8	EVH		●		●	●
9	MSH		●		●	●
10	BAM		●			
11	BAF				●	●

여기서 SS(1)은 2개의 매크로 컴포넌트 COM과 OLS로 이루어지는 온라인 서버의 기본이 되는 HoServer 기능의 서버 시스템 컴포넌트이며, SS(2)는 에이전트 생성 및 통신을 위해 7개의 매크로 컴포넌트들로 구성되는 컴포넌트로 HumanPlace 기능을 수행할 수 있는 서버 시스템 컴포넌트이다. SS(3)는 3개의 매크로 컴포넌트로 구성되는 컴포넌트로 HumanServer 구축을 위해 기본이 되는 서버 시스템 컴포넌트로 BasicServer라고 한다. SS(4)는 에이전트의 기능을 구축할 수 있는 컴포넌트로 BasicHuman이라 하고, SS(5)는 Place들의 수행 환경이며 이들을 관리하는 Metaplace를 구축할 수 있는 컴포넌트로 MetaPlace라고 한다. 서버 시스템 컴포넌트 SS(k)는 자체로써 수행이 가능하다. 이는 각 컴포넌트가 모바일 에이전트 시스템에서 능동적 실행 부분을 담당하고 있다는 것을 의미하고 있다. 즉, 서버, Place (혹은 Metaplace) 그리고 모바일 에이전트들의 기본 동작 기능 등이 모두 시스템 컴포넌트인 것이다. 그러한 측면에서, EmHUMAN은 시스템 컴포넌트들의 집합이며, 그들을 서로 서로 적절하게 조합함으로써 다양한 수준의 모바일

에이전트 시스템들이 구성될 수 있는 것이다. 즉, EmHuman은 고정된 시스템이 아니라 기존의 프리미티브 컴포넌트들과 매크로 컴포넌트들을 변경하거나, 새로운 컴포넌트를 추가 혹은 제거함으로써 더 다양한 모바일 에이전트 시스템들을 구성할 수 있는 것이다. 그러므로 일단 사용할 시스템 컴포넌트들이 정해지면 그들로 구성되는 모바일 에이전트 시스템의 기능과 동작이 정해진다고 할 수 있다.

표 1로부터 유틸리티 시스템 컴포넌트들을 구성하는데 사용되는 매크로 컴포넌트는 현재 UTL을 포함하여 7개가 구현되어 있다. 이 7개의 컴포넌트들을 기반으로, 구성할 수 있는 유틸리티 시스템 컴포넌트들은 매우 다양하게 존재할 수 있는데, 그 중 US(1) 부터 US(9)까지 9개의 유틸리티 시스템 컴포넌트 예를 보여주고 있는 것이 표 3이다.

표 3. 유틸리티 시스템 구성 테이블
Table 3. Utility System Configuration Table (USCT)

	PM	유틸리티 시스템 컴포넌트 구성 테이블: US(k)								
		1	2	3	4	5	6	7	8	9
1	UTL		●	●	●	●	●	●	●	●
2	IPF		●	●			●		●	●
3	RPF				●	●		●	●	●
4	FPF						●	●		●
5	ROM		●	●			●		●	●
6	RPM				●	●		●	●	●
7	RCI	●		●		●				●

2. 구성적 임베딩

예를 들면, US(1)은 하나의 매크로 컴포넌트 RCI로 구성되어 있지만, 이를 가짐으로써 분산 응용의 주요 기능인 원격 클래스 구동 기능을 가지는 것이다. 그리고 US(2)는 에이전트가 거쳐야할 노드들의 리스트인 여정 경로와 응답 모드를 설정하는 인터페이스를 제공하여 자율적으로 에이전트들이 네트워크상의 노드들을 이동할 수 있도록 하는 기능을 지원하게 된다. US(9)은 EmHUMAN이 제공할 수 있는 모든 유틸리티 기능을 지원하는 구성이라고 할 수 있다. 이러한 방식으로 매우 다양한 유틸리티 시스템 컴포넌트들을 구성할 수 있는 것이다.

시스템 컴포넌트들을 기반으로 시스템 구성 테이블을 사용하면, 다양한 기능과 적합한 자원 규모의 모바일 에이전트 시스템이 구성적으로 임베딩 될 수 있는데, 그러한 구성적 임베딩 개념이 그림 4에서 보여주고 있다. 그림 4의 HUMAN#1부터 HUMAN#k 까지 생성되는 시스템은 모두 모바일 에이전트 시스템이며 제각기 그 기능과 자원 사용 규모가 다르다. EmHUMAN 자체도 모든 시스템 컴포넌트들로 구성되는 모바일 에이전트 시스템이며, 임베딩 시키고자 하는 목표 장치의 자원 사양에 맞춰, 서버 시스템 컴포넌트와 유틸리티 시스템 컴포넌트들을 선택하여 구성할 수 있는 것이다.

표 2와 표 3에서 보여준 SSCT와 USCT를 이용하면 단순 기능으로부터 복합적 기능 까지를 가질 수 있는 모바일 에이전트 시스템을 구성할 수 있다. 시스템 컴포넌트는 모바일 에이전트 시스템을 구성하는 중요한 능동 컴포넌트이다. 그것들은 대부분의 모바일 에이전트 시스템들이 가져야만 하는 최소한의 기능으로부터 상당한 고급의 기능까지를 내포하고 있다.

표 2의 SSCT로부터 얻어져서 현재 EmHUMAN이 가지고 있는 5개의 서버 시스템 컴포넌트인 SS(1) 부터 SS(5) 까지를 사용하여 여러 수준의 에이전트 서버 시스템을 구축할 수 있는데, 그러한 예를 보여주고 있는 것이 표 4이다. 표 4에는 4가지 형태의 에이전트 서버, S1부터 S4를 구축하는 경우를 보여주고 있다. 첫 번째 에이전트 서버, 즉 S1의 경우가 모바일 에이전트의 정상적 동작을 위한 최소 수준이라고 할 수 있다. 이렇게 구축할 경우, MetaPlace를 통하는 보호 모드 기능이라든지 HoServer 기능을 제공하지 않는다. S2의 경우에는, MetaPlace에 의한 보호 모드 기능이 가능하나, HoServer의 기능이 지원되지 않는다. 마찬가지로 세 번째인 S3의 경우는 HoServer 기능은 제공하나 MetalPlace 기능이 없으므로 2단계 보호 모드를 지원할 수 없게 된다. 그리고 네 번째의 경우 S4는 EmHUMAN이 제공할 수 있는 최고 사양의 에이전트 서버라 할 수 있는데, 왜냐하면 모든 서버 기능을 제공하고 있기 때문이다. 표 3의 USCT에서 처럼 유틸리티 시스템 컴포넌트들을 조합하여 모바일 에이전트 시스템이 가지는 기능 부분을 필요에 맞춰 구성할 수 있다. 표 3에서는 9개의 유틸리티 시스템 컴포넌트만을 보여주고 있지만 7개의 매크로 컴포넌트들을 사용하면 이론적으로는 그 보다 훨씬 많은 유틸리티 시스템 컴포넌트들을 구성할 수 있다. 서버 시스템 컴포넌트로 구성할 수 있는 에이전트 서버 시스템의 수를 K라 하고 매크

로 컴포넌트를 가지고 구성할 수 있는 유틸리티 시스템 컴포넌트의 수를 N이라고 하면, 총 K x N 개의 서로 다른 기능과 자원 규모가 다른 모바일 에이전트 시스템을 구성할 수 있는 것이다.

표 4. 모바일 에이전트 시스템의 구성 예
Table 4. Feasible server configurations

	HoServer	Meta Place	Place	Basic Server	Basic Human	Sizs
S1			●	●	●	70 K
S2		●	●	●	●	105 K
S3	●		●	●	●	83.4 K
S4	●	●	●	●	●	113.4 K

현재 EmHUMAN은 서버와 유틸리티 기능을 18개 매크로 컴포넌트로 분류하고 있고, 이들로부터 5개의 서버 시스템 컴포넌트를 지원하고 있지만, 필요시 새로운 매크로 컴포넌트들과 시스템 컴포넌트들로 변경, 확장 될 수 있으며, 이에 따라 기능이 변경되거나 확장된 서버 시스템 컴포넌트들이 생성될 수 있다. 동일한 개념으로 유틸리티 시스템 컴포넌트들도 생성될 수 있다. 이는 서비스 규모와 범위 등에 따라 서버를 분리 운영하느냐 아니면 통합하여 운영하느냐에 따른 정책적 결정 혹은 지원 가능한 하드웨어 플랫폼의 규모에 의해 이루어질 수도 있는 것이다.

V. EmHUMAN의 사용

EmHUMAN은 모바일 에이전트 시스템의 개발을 위한 프레임워크로 모바일 에이전트 응용 개발을 위한 도구로서 사용될 수 있으며, 동시에 그 자체로서 모바일 에이전트들의 동작을 위한 서버로서도 사용될 수 있다. Java 기반의 모바일 에이전트 시스템은 메쉬(mesh), Ad-hoc 혹은 Peer-to-Peer 등과 같은 유무선 네트워크 상에서의 안드로이드 기반의 분산 응용을 위한 프레임워크가 될 수 있다.

개발 환경으로서의 EmHUMAN을 사용하는 경우, 구현되는 응용 에이전트는 모든 프리미티브, 매크로 그리고 시스템 컴포넌트들을 사용할 수 있기 때문에, 응용 에이전트가 수행되는 장치 상에 전체 컴포넌트가 설치되어 있어야 한다. 이는 목적하는 장치에 개발 환경인 EmHUMAN이 필요로 하는 충분한 자원을 보유하고 있어야 한다는 것을 의미하고 있다. 그 대신 응용 에이전트의 크기를 최소화 할 수 있어, 이동을 위해 필요로 하는 대역폭을 절약할 수 있다는 장점을 가진다. 그런데 목적하는 장치에 필요로 하는 자원이 충분하지 않아, 일부분만 탑재된다면, 유틸리티 등을 이용할 수가 없으므로 그 상에서 수행을 하는 응용 에이전트 자체가 그 기능을 가져야하므로 그 크기가 증가하게 되며, 그로 인해 이동에 필요한 대역폭 점유가 늘어나는 단점을 가지게 되는 것이다. 그러므로 개발 및 수행 환경으로 사용하는 경우 이러한 상호 장단점을 적절하게 고려하여야 할 것이다. EmHUMAN을 에이전트 동작을 위한 서버 시스템으로만 사용할 경우, 표 4처럼 최소 기능 서버 시스템인 S1부터 최대 기능 서버 시스템인 S4까지 4 가지의 모바일 에이전트 서버 시스템을 구성할 수 있다. 이론적으로는 5개의 서버 시스템 컴포넌트를 사용하면, 총 31가지 서버 시스템 구성이 이루어질 수 있으나, 에이전트의 정상적인 동작을 위해서는 최소한 S1에서 보여준 3가지 서버 시스템 컴포넌트가 존재하여야 하기 때문에 여기서는 3개의 컴포넌트를 포함하는 4가지 경우의 모바일 에이전트 서버 시스템만을 보여주고 있다. 무선 메쉬 네트워크, 혹은 Ad-hoc 네트워크 등에서 에이전트 서버로 동작할 장치는, 최소한 4가지 경우 중 하나를 탑재할 수 있어야 한다. 그리고 서버로만 작동을 하기 때문에 유틸리티를 지원하지 못하므로 서버 장치에서는 에이전트의 작업 수행이 이루어지기 어렵다. 그것은 응용 에이전트 자체가 유틸리티 기능을 포함하고 있어야 하므로 에이전트의 크기가 증가하고 그로인해 에이전트 이동을 위한 대역폭 사용의 증가를 초래할 수 있다.

VI. 파일 탐색 에이전트의 구현

IBM에서 개발된 모바일 에이전트 시스템인 AGLET[10]을 기반으로 네트워크상의 목적하는 파일의 탐색을 위한 간단한 파일 탐색 응용이 설계 구현되었다[14]. 본 장에서는 그를 EmHUMAN을 기반으로 하는 파일 탐색 응용으로 변환하여 구현하였으며, 아래에 보여준 바와 같이 그 변환 결과는 AGLET을 기반으로 하는 경우에 비해 적은 노력으로 이루어질 수 있었다. 여기서는 프로세스의 핵심 부분인, 관심 있는 노드들로부터 원하는 파일들에 관

한 정보를 얻는 부분만을 보여준다. 첫 번째 부분 DesiredFileSearch는 원하는 파일의 탐색을 수행하는 에이전트인 FileSearchAgent를 생성하는 부모 에이전트로 Place에 포함되어 있으며, 여정모드 혹은 응답 모드 등을 생성하는 에이전트에게 지정하는 사용자 인터페이스 기능도 수행한다. 두 번째 부분은 부모 에이전트에 의해 생성된 모바일 에이전트로 Place로부터 지정된 노드로 이동하여 자기가 맡은 작업, 즉 사용자가 원하는 파일을 탐색하는 작업을 수행하고 그 결과물인 파일 리스트를 응답해 준다. 이때 이동 및 응답은 사용자가 지정한 여정모드를 따라 이동하며 지정해준 응답 방식에 따라 결과물을 응답한다. 이러한 일련의 과정이 AGLET에 비해 매우 간단하게 이루어질 수 있는데, 이는 EmHUMAN에서 내장 API로 제공해주는 여정 모드 및 응답 모드 그리고 파일 비교와 같은 유틸리티를 그대로 사용할 수 있기 때문이다. 만약에 그러한 유틸리티들을 제공해주지 않는다면 그 모든 과정을 사용자가 직접 프로그래밍 하여야 하며, 그로 인해 에이전트 코드의 크기가 상당히 증가하게 되며 이동 시에도 대역폭을 많이 점유하는 문제를 나타낼 수 있다.

DesiredFileSearch Agent(Parent agent)

```
//Child Agent creation : FileSearchAgent(FSA)
public void createFSA()
  FileSearchAgent agent =
   new FileSearchAgent(parentPlace.generateName());
  parentPlace.addHumanOnCreation(agent, null);

//message handling (SEL, RESULT kind)
public void handleMessage(HumanMessage msg)
  if (msg.kind.equals("SEL"))
          sendAck(msg.recipient, msg.sender);
  else if (msg.kind.equals("RESULT"))

    //other  message handling
```

FileSearchAgent(Child agent with mobility)

```
public void run()
          //Agent set up
          setAgent(this, getName());
          //moving agent according to routing mode
          doItinerary();
public void onArrival()
          ..............
          //perform the agent task
          doTask();
public void doTask()
          ..............
          //File compare using HumanFile
          HumanFile hf = new HumanFile();
          hf.setDir(dirname);
          hf.setFiles(fileList);
          hf.compareFile();
          //the result form File Compare
          result = hf.getDesiredFileList();

          //send the result according to reply mode
          setReplyResult(false, result);
          doReply();
```

VII. 결 론

Java 기반의 모바일 에이전트 시스템은 분산 응용을 위한 좋은 프레임워크가 될 수 있다. 본 논문에서 다양한 유형의 자원 제약 특성을 가지는 Java 기반의 장치로의 구성적 임베딩을 위한 컴포넌트 기반의 모바일 에이전트 시스템 프레임워크가 설계, 구현 되었다. 또한 에이전트 프로그래밍을 수월하게 하게 위해 다양한 유틸리티를 API로 내장 지원했다. 모바일 에이전트 시스템의 구축은 임베딩될 장치의 저장장치 규모, 컴퓨팅 능력, 대역폭 등의 자원 제약 특성을 고려하여 이루어진다. 또한 에이전트와 호스트의 보호를 위해 Place와 더나아가 MetaPlace를 사용하는 2단계 보호 방법이 제안되어 더욱 안전한 에이전트 활동을 가능하게 하였다.

참 고 문 헌

[1] B. Day, “Developing Wireless Applications using the Java2 Platform, Micro Edition,” a white paper, http://java.sun.com/j2me

[2] S. Ortiz Jr., "Embedded OSs Gain the Inside Track," IEEE Computer, Vol. 34, No. 11, Nov. 2001

[3] D. B. Lange & M. Oshima, "Programming and Deploying Java Mobile Agents with Aglets," Addison-Wesley, 1998

[4] S. Ilarri, R. Trillo & E. Mena, "SPRINGS: A Scalable Platform for Highly Mobile Agents in Distributed Computing Environments," Proc. of 2006 Int'l Symp. on a World of Wireless, Moble and Multimedia Networks (WoWMoM'06), 2006

[5] H. Qi, Y. XU & X. Wang, "Mobile-agent-based collaborative signal and information processing in sensor networks," Proceeding of IEEE, Vol. 91, No. 8 pp.1172-1183, Aug. 2003,

[6] M Chen, T. Kwon & Y. Choi, "Data Dissemination based on Mobile Agent in Wireless Sensor Networks," Proc. of IEEE Conf. on Local Computer Networks, 2005 (LNCS'05)

[7] M. Chen, S. Gonzalez & V.C.M. Leung, "Applications and Design Issues for Mobile Agents in Wireless Sensor Networks," IEEE Wireless Communications, Vol. 14, No. 6, pp. 20-26, 2007

[8] C.-L. Wu, C.-F. Liao & L.-C. Fu, "Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology," IEEE Trans. on Sys., Man, and Cybernetics, Part C: Applications and Reviews, Vol. 37, No. 2, pp.193-205, 2007

[9] S. Ilarri et al, "Using cooperative mobile agents to monitor distributed and dynamic environments," Information Sciences, Vol. 178, No. 9, pp.2105-2127, May 2008

[10] B. Chen & W. Liu, "Mobile Agent Computing Paradigms for Building a Flexible Structural Health Monitoring Sensor Network," Computer-Aided Civil and Infrastructure Engineering, Vol. 25, No. 7, pp.504-516, 2010,

[11] P. Bellavista et al, "A mobile computing middleware for location- and context-aware internet data services," ACM Trans. on Internet Technology (TOIT), Vol. 6, No. 4, pp.356-380, 2006

[12] C.-L. Fok et al, "Agilla: A Mobile Agent Middleware for Self-Adaptive Wireless Sensor Networks," ACM Trans. on Autonomous and Adaptive Systems, Vol. 4 No. 3, July 2009

[13] N. Migas et al, "Mobile Agents for Routing, Topology Discovery, and Automatic Network Reconfiguration in Ad-Hoc Metworks," Proc. of 10th Int'l Workshop on Engineering of Computer-Based Systems, 2003

[14] C. Spyrou et al, "Mobile Agents for Wireless Computing: The Convergence of Wireless Computational Models with Mobile-Agent Technologies," Mobile Networks and Applications, Vol. 9, No. 5, pp.517-528, 2004,

※ 본 연구는 덕성여자대학교 2011년도 교내연구비 지원에 의해 수행되었음

저자 소개

정 원 호(정회원)

- 1979년 2월 : 서울대학교 전자공학과 학사
- 1981년 2월 : 한국과학기술원 전기및 전자공학과 석사
- 1989년 2월 : 한국과학기술원 전기및 전자공학과 박사
- 1989년 3월 ~ 2010년 2월 : 덕성여자대학교 컴퓨터공학부 교수
- 1995년 8월 : IBM 왓슨 연구소 방문연구원
- 2010년 3월 ~ 현재 : 덕성여자 대학교 디지털미디어학과 교수

<주관심분야: P2P, 모바일플랫폼>

강 남 희(정회원)

- 1999년 2월 : 숭실대학교 공학사
- 2001년 2월 : 숭실대학교 공학석사
- 2004년 12월 : University of Siegen, 공학박사
- 2009년 3월 ~ 현재 : 덕성여자대학교 디지털미디어학과 조교수

<주관심분야 : 인터넷통신, 통신보안>