생태계 모방 임베디드 미들웨어의 설계 및 구현

최현식*, 한성민*, 윤현준*, 박성용*
*서강대학교 컴퓨터공학과
e-mail:hs38317@gmail.com

Design and Implementation of Bio Inspired Embedded Middleware

Hyun-Sik Choi*, Sung-Min Han*, Hyun-Jun Yun*, Sung-Yong Park*
*Dept of Computer Science and Engineering, Sogang University

요 약

본 논문에서는 미래의 네트워크 모델에 적합하도록 설계한 생태계 모방 임베디드 미들웨어(BIEM : Bio Inspired Embedded Middleware)를 구현하고, 그의 구조와 특성을 기술하며, 성능을 평가하여 BIEM의 정확성을 검증한다. BIEM의 정확성을 검증하기 위해, BIEM 위에서 응용프로그램을 실행시켰을 때와 독립적으로 일반적인 해를 구했을 때의 결과를 비교한다. 또한 서비스 제공을 위해 요구되는 통신 오버헤드를 측정하여, 오버헤드가 네트워크 트래픽에 얼마나 영향을 미칠지에 대해서 알아보고, 마지막으로는 이주서비스에 소요되는 시간을 비교 및 분석한다. 이를 통해, BIEM이 미래 네트워크에 이용될 수 있는 하나의 시스템임을 결론지을 수 있다.

1. 서론

현존하는 네트워크 인프라와 응용 서비스들은 정보의 공유를 가능케 하여, 필요한 정보에 편하게 접근하여 그를 이용할 수 있도록 해준다. 하지만 현존 네트워크 서비스의 구조는 결함에 대처할 수 있는 가용성(Availability) 측면이나 다양한 환경 변화에 대한 적응성(Adaptability) 측면에서 문제점을 가지고 있을 뿐만 아니라, 유비쿼터스 컴퓨팅, P2P, 스트리밍 서비스 등에 의해 네트워크 호스트의 수와 트래픽이 증가할 때의 확장성(Scalability)에도 문제가 있을 수 있다. 따라서 미래 환경에서 효율적인 서비스를 제공하기 위해서는 단순하게 네트워크 인프라의 처리속도를 높이는 것 보다는 네트워크 소프트웨어의 구조를 발전시키는 방향으로의 접근이 이루어져야 한다[1]. 이에 대한 대안으로, 진화를 통해 변화하는 주위환경에 적응하고 생존하는 생태계를 모방한 생태계 모방 임베디드 미들웨어(BIEM: Bio Inspired Embedded Middleware)를 개발하도록 한다.

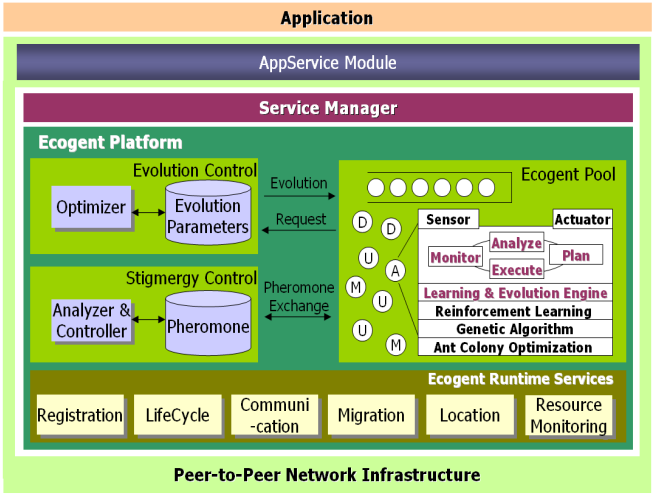
BIEM은 에이전트 기반의 시스템으로, 에이전트는 스스로의 판단에 의해 자율적 행동을 하는 특징을 가진다. 이러한 에이전트 개념에 진화, 적응, 생존 등의 생태계 현상을 모방한 바이오 알고리즘을 적용하여 다양한 서비스 및 응용 프로그램에 확장성, 적응성 등을 제공할 수 있다. 또한 성능 측정을 위해서 임베디드 보드에 BIEM을 올려서 테스트를 하도록 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 BIEM의 구조와 기능에 대해 알아본다. 3장에서는 플랫폼에서 응용프로그램이 정확히 수행되는지를 측정하고, BIEM의 기능 중에서 통신 및 이주서비스에서 발생하는 오버헤드를 측정하여 미들웨어의 적합성 여부를 평가하도록 하며, 4장에서는 결론과 향후 과제에 대해 살펴본다.

2. BIEM 아키텍쳐

본 장에서는 서론에서 제기한 문제를 해결하기 위해 구현한 BIEM의 아키텍쳐와 그 기능에 대해서 알아보도록 한다. 구현된 BIEM은 바이오 에이전트와 멀티 에이전트 기반의 바이오 플랫폼으로 구성된다. 편의상, 바이오 에이전트는 에코전트(Ecology와 Agent의 합성어)로, 바이오 플랫폼은 플랫폼으로 칭하기로 한다.

BIEM의 구조는 두 가지 방법으로 분석될 수 있다. 첫 번째 방법은 사용자에게 응용서비스를 제공하는 메커니즘을 분석하는 것으로, 플랫폼과 에코전트를 구분하여 그들 사이의 관계를 분석하는 것이다. 두 번째 방법은 플랫폼과 에코전트를 하나의 객체로 보고 그 기능을 분석하여, BIEM이 가용성, 적응성, 확장성 등을 제공할 수 있는지를 분석하는 것이다. 본 장에서는 BIEM에 대한 이해를 위해 두 가지 방법 모두를 사용하여 BIEM의 구조를 분석하도록 한다. (그림 1)은 BIEM의 구조를 도식화 한 것으로, 위의 두 가지 분석 방법을 적절히 보여준다.



(그림 1) BIEM 아키텍처

우선, BIEM 내부의 플랫폼과 에코전트의 관계를 분석하도록 한다. 에코전트는 생태계 개체들의 특성을 갖는 모바일 에이전트로 사용자에게 서비스를 제공하는 객체이고, 플랫폼은 에코전트의 운영환경에 해당하는 소프트웨어이다. 즉, 에코전트는 플랫폼 위에서 실행되며, 플랫폼이 제공하는 서비스를 이용하여, 서로 통신, 이주 및 협력하여 사용자가 원하는 서비스를 제공한다.

에코전트는 생성된 후, 모니터, 분석, 설계, 실행의 과정을 순환적으로 수행한다. 모니터 과정을 통해 에코전트에 필요한 정보들을 수집하고, 분석 과정에서 수집한 정보를 분석하며, 설계 과정에서는 이전에 분석한 정보에 의거하여 수행할 작업을 결정하고, 결정된 명령들을 명령 큐에 입력시킨다. 이 후 그를 수행하여 일련의 의미 있는 작업을 진행한다. 명령 큐에서 명령을 하나씩 가져와 순차적으로 수행하는 것을 에코전트의 실행이라고 정의한다.

플랫폼은 에코전트를 생성, 소멸하고 각각의 목적에 맞게 에코전트의 상태를 변화시키는 에코전트 Lifecycle 서비스를 기본적으로 제공하며, 에코전트 간의 통신(Communication) 서비스 및 에코전트 이주(Migration) 서비스를 제공하도록 한다. 에코전트 간의 통신 및 에코전트 이주 서비스를 위해서는 다른 플랫폼 및 내부의 에코전트를 찾는 기능(Location)도 있어야 한다. 추가적으로 플랫폼 자신의 메모리 상태와 CPU 사용률 등을 모니터하는 기능(Resource monitoring)을 갖도록 한다. 에코전트 간의 통신서비스는 FIPA의 규약인 ACL Message[2] 형식을 따르며, 비동기적으로 통신하여 효율을 높이도록 한다[3]. 에코전트 이주서비스는 송신 플랫폼에서 에코전트의 코드와 데이터를 직렬화하여 전송하고, 수신 플랫폼에서는 이를 역직렬화하여 수신한 코드와 데이터를 수행가능한 상태로 변환시키는 작업을 포함한다.

위에서는 플랫폼과 에코전트의 협업을 통해서 BIEM이 사용자가 요청하는 의미 있는 서비스를 제공하는 구조에 대해 분석을 했다. 두 번째 분석 방법으로 플랫폼과 에코전트를 하나의 객체로 보고, 네트워크 구조 문제를 해결하는 플랫폼의 핵심적인 기능을 기술하도록 하겠다.

BIEM의 설계 목표는 가용성, 적응성, 확장성이 기존 네트워크보다 우월한 미들웨어의 설계이다. 이 세 가지 기준들 중에서 가용성, 적응성과 확장성은 큰 관련이 없다. 따라서 이에 대한 접근으로는 Divide-and-Conquer 방법이 유용하다. 먼저 확장성을 위해서 기존의 지능형 멀티에이전트 시스템[4]을 미래의 대형화된 시스템에 보다 적합하도록 P2P 네트워크 기반으로 구성한다. (그림 1)에서도 볼 수 있듯이 플랫폼은 P2P 네트워크 인프라 위에 존재하며, 따라서 플랫폼 간의 모든 메시지 전달 등의 통신은 P2P 방식으로 이루어진다. 이를 통해 클라이언트/서버 방식보다 우월한 확장성을 갖게 된다[5]. 가용성과 적응성을 위한 적응, 진화 기능을 추가하는 부분을 살펴보도록 한다. 앞에서도 설명했듯이, BIEM은 일반적인 서비스에 추가적으로 생태계의 협동·경쟁, 진화·적응 모델을 적용하여, 생존성과 적응성, 확장성을 높인 분산 이동 에이전트 시스템이다. 이를 가능하도록 하기 위해서 플랫폼 서비스 내부에 Evolution Control과 Stigmergy Control을 추가했다.

Evolution Control 은 에코전트를 네트워크 환경전체에 적합하도록 진화시켜, 갑작스러운 네트워크 환경변화에도 에코전트가 빨리 적응할 수 있도록 한다. 또한 작업을 더욱 효율적으로 수행할 수 있도록 진화시킨다. 즉, 에코전트 진화와 적응에 대한 제반의 역할을 담당한다. 이를 위해 Genetic Algorithm(GA)[6]에 기초하여, 분산 환경에서 에코전트의 특성이나 수행방식을 결정짓는 파라미터 값들에 대한 진화와 적응을 관리하도록 한다. 하지만, GA는 문제에 의존적이어서 서비스에 따라 각기 다른 일을 하고, 서로 전혀 다른 파라미터 항목들을 가지고 있다. 따라서 서로 다른 특성을 지닌 에코전트를 모두 진화 및 적응시킬 수 있도록 범용적인 Genetic Algorithm Framework를 구현했다. Stigmergy Control은 특정 응용프로그램의 문제해결을 위한 방법을 제공하거나 에코전트의 진화를 담당한다. 이를 위해 생태계를 개미군집으로 모방하는 Ant Colony Optimization Algorithm(ACO)[7]을 사용하며 페로몬을 이용한 정보전달과 같은 간접교신을 통해 문제를 해결하도록 한다. 에코전트는 ACO에서처럼 플랫폼을 돌아다니며 페로몬을 남기는 방법으로 다른 에코전트와 정보를 공유한다. 이를 통해 얻은 정보를 분석 및 추출함으로써 사용자에게 최적의 해를 제공할 수 있다.

3. 성능 평가

본 장에서는 BIEM을 임베디드 보드에 포팅해서 측정한 결과를 분석한다. 이는 BIEM이 사용되게 된다면, 네트워크에 참여하는 어떤 노드에서도 동작할 수 있어야 하므로, 일반 PC에서 뿐만 아니라, 임베디드 보드에서도 정확히 수행되는지를 확인할 필요가 있기 때문이다.

3.1. 알고리즘의 정확성

Knapsack 문제는 조합최적화 문제이다. 배낭에 담을 수 있는 무게의 최대값이 정해져 있고, 일정 가치와 무게가 있는 짐들을 배낭에 넣을 때, 가치의 합이 최대가 되도록 짐을 고르는 방법을 찾는 문제이다. Knapsack 문제는 짐을 쪼갤 수 있는 경우와 짐을 쪼갤 수 없는 경우로 나누어지는데 짐을 쪼갤 수 없는 경우 0-1 Knapsack 문제라고 하며, 이는 대표적인 NP-Complete 문제이다.

짐의 무게와 가치의 데이터 쌍 (w_j, p_j)가 n개 주어졌을 때 Knapsack 문제를 수식으로 나타내면 아래와 같다.

$$\sum_{j=1}^{n} w_j x_j \le c,\ x_j = 0 \text{ or } 1,\quad j = 1, \ldots, n \quad \text{<수식 1>}$$

그러므로 Knapsack 문제의 결과는 <수식 1>을 만족하는 데이터 중에 <수식 2>가 최대가 될 때의 값이다.

$$\sum_{j=1}^{n} p_j x_j \quad \text{<수식 2>}$$

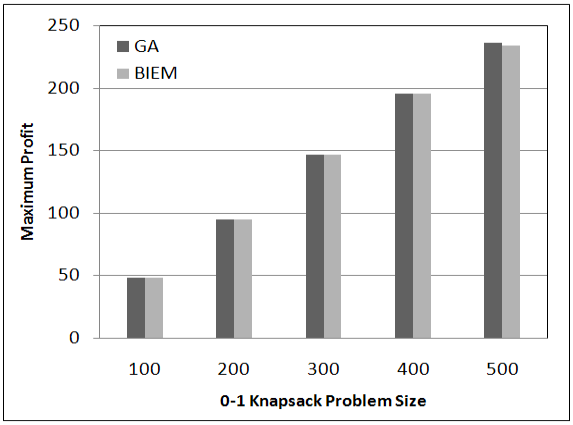
앞서 제시한 BIEM의 정확성을 측정하기 위하여 BIEM이 제공하는 GA를 사용하여 0-1 Knapsack 문제의 최적해를 구하는 프로그램을 실행하였다.

<표 1> 테스트 케이스

데이터 개수	c (무게의 최대값)
100	12.5
200	25
300	37.5
400	50
500	62.5

Knapsack 문제의 무게와 가치인 w와 p는 1보다 작은 실수를 만족하는 쌍으로 설정하였다. w와 p의 데이터 개수와 그에 따른 배낭에 담을 수 있는 무게의 최대값은 <표 1>과 같이 5개의 테스트 케이스로 실험하였다.

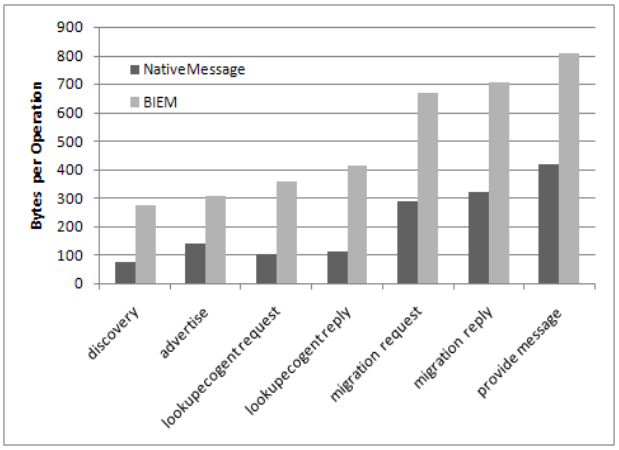
(그림 2)는 BIEM이 제공하는 GA를 사용하여 Knapsack 문제를 풀 때와 GA를 단독으로 사용하여 수행하였을 때의 결과이다. 두 경우 GA의 환경 변수를 동일하게 설정하여 실험하였다. BIEM에서 수행한 결과가 GA를 단독으로 수행한 결과와 거의 일치하는 것을 알 수 있으며, 이는 BIEM이 정확히 동작함을 의미한다.



(그림 2) 알고리즘의 정확성

3.2. 통신 오버헤드

본 실험에서는 BIEM 간의 통신을 위한 위치, 통신, 이주 서비스의 Native Message와 그에 따른 오버헤드를 비교한다. BIEM은 P2P로 연결된 여러 플랫폼이 통신을 위해 어플리케이션 수준의 통신 프로토콜을 정의하여 사용한다. 통신 프로토콜은 핵심 내용인 Native Message의 처리를 용이하게 하고 확장성(Extensibility)을 높이는 내용을 부가한다. 특히 통신서비스에서 사용하는 FIPA ACL Message[2] 역시 내부의 통신 프로토콜을 통해 직렬화, 역직렬화되며 플랫폼간의 통신에 사용된다. (그림 3)은 Native Message와 실제 BIEM의 통신 프로토콜에서 사용하는 Message의 바이트 수를 측정하여 비교한 것이다.



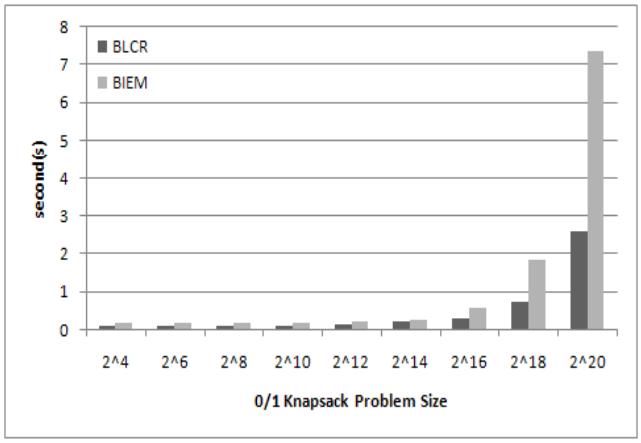
(그림 3) 통신 프로토콜에 따른 오버헤드

Discovery, Advertise, Lookup Ecogent Request / Reply는 위치서비스, Migration Request / Reply는 이주서비스, 그리고 Provide Message는 통신서비스에서 필요로 하는 연산이다. 각 연산에서 발생하는 오버헤드는 Native Message에 비해서 상대적으로 높다. 하지만 수백 킬로바이트 이상을 필요로 하는 이주 데이터에 비해 매우 작은 수치이며, 1 킬로바이트 이하의 작은 메시지이기 때

문에 통신 프로토콜에 의한 오버헤드는 성능에 큰 영향을 주지 않는다. 또한 메시지의 구분을 위한 태그 방식의 문자열이 오버헤드의 대부분을 차지하므로, 쉽게 Native Message의 크기에 근접하도록 최적화 할 수 있다.

3.3 이주 오버헤드

본 실험에서는 Knapsack 문제를 풀 때, BIEM이 제공하는 이주서비스를 통해 Evolution Ecogent를 이주하는 동안 발생하는 오버헤드를 측정한다. 문제를 푸는 과정에서 이주서비스를 이용하여 다른 플랫폼으로 이동하는데 걸린 시간과 실행중인 프로세스를 다른 PC로 이동하고 실행을 재개하는 BLCR(Berkeley Lab Checkpoint / Restart)[8]의 수행시간을 비교한다. Knapsack 문제에서 입력 데이터의 크기는 2^4 에서 2^{20} 까지 증가한다.



(그림 4) 이주서비스와 BLCR의 수행시간 비교

이주시간은 에코전트를 정지하고 관련 데이터를 모두 정리하여 목적지 플랫폼에 전송한 후, 에코전트가 다시 실행되기까지 걸리는 시간으로 정의할 수 있다. BLCR은 checkpointing을 통해 프로세스의 모든 데이터를 파일로 저장하고, 대상 PC에 전송한 후 프로세스의 재실행 시작 순간까지의 시간을 측정하였다. (그림 4)처럼 입력 크기 2^{14} 까지는 이주서비스와 BLCR의 수행시간 차이가 거의 없지만, 수십 메가바이트 단위로 이주를 위한 전송량이 증가하면 BLCR이 더 좋은 성능을 보여주었다. 이것은 BIEM의 이주서비스는 유저영역에서 구현되어 있는 반면에, BLCR은 커널의 도움을 받는 디바이스 드라이버 영역에서 효율적인 방법으로 동작하기 때문이다. 또한 BIEM 환경은 크기가 큰 에코전트가 적은 수로 동작하기 보다는 크기가 작은 에코전트가 많이 동작하며 서로 상호작용한다. 따라서 (그림4)에서 입력 데이터의 크기가 2^{16} 일 때부터 급격하게 증가하는 오버헤드는 실제 동작 환경에서 매우 드물게 발생할 것이며, 따라서 BIEM의 이주서비스가 충분한 성능을 낸다고 할 수 있다.

4. 결론 및 향후 과제

본 논문에서는 현존 네트워크 서비스의 구조적인 한계를 극복하고 더불어 확장성이 있는 시스템을 구축하기 위해, 생태계의 진화, 적응 등을 모방한 BIEM을 설계하고 구현했다. 개발한 미들웨어는 확장성을 잃지 않기 위해 P2P 네트워크를 기반으로 하여 설계되었으며, 플랫폼 내부에 Evolution Control, Stigmergy Control을 구현하여 적응과 진화 기능을 추가했으며, 이를 통해 적응성과 가용성을 갖는다. 성능 평가에서는 Knapsack 문제를 BIEM이 정확히 풀어낸다는 것을 검증했으며, 플랫폼간의 통신에서 메시지 오버헤드가 존재하지만 이것이 확장성에 영향을 미칠 정도로 크지 않음을 확인했다. 이주서비스도 충분한 성능을 낼 수 있을 것임을 밝혔다. 이를 통해, BIEM이 미래 네트워크에 이용될 수 있는 하나의 시스템임을 결론지을 수 있다. 그렇지만 구현한 미들웨어를 테스트하기 위해 임베디드 보드에 올리고 Knapsack 문제해결 프로그램을 수행시켰을 때, 그 수행속도가 PC에서보다 현저히 떨어지는 문제가 발생했다. 이는 근본적으로 PC와 임베디드 보드의 성능차이에 기인하지만, BIEM이 임베디드 환경에서도 이용될 수 있도록 내부적인 코드의 최적화가 이루어져야 할 것이다.

참고문헌

[1] Scott Shenker, *"Fundamental design issues for the future Internet"*, IEEE Journal on Selected Areas in Communications (JSAC), Vol. 13, No. 7, September, 1995, pp. 1176--1188.

[2] http://fipa.org/specs/fipa00061/

[3] Matsuda, Motohiko et al, *"The Design and Implementation of an Asynchronous Communication Mechanism for the MPI Communication Model"*, 2004, Proceedings of the IEE,E International Conference on Cluster Computing (Cluster 2004).

[4] Nicholas Jennings, Katia Sycara and Michael Wooldridge. "*A Roadmap of Agent Research and Development*", Autonomous Agents and Multi-Agent Systems, vol. 1, pp. 7-38, 1998.

[5] Dinesh C. Verma, "*Legitimate applications of Peer- to-Peer networks*", Wiley-Interscience, 2004

[6] M. Mitchell, "*An Introduction to Genetic Algorithms*", MIT Press, Cambridge, Mass. 1996.

[7] M. Dorigo and T. Stützle, "*Ant Colony Optimization*", MIT Press, Cambridge, 2004.

[8] Paul H. Hargrove and Jason C. Duell "*Berkeley Lab Checkpoint/Restart (BLCR) for Linux Clusters*" In Proceedings of SciDAC 2006: June 2006. (publication LBNL-60520)