DOI: 10.3745/KIPSTA.2009.16-A.1.1

상황인식 기반 멀티 에이전트 시스템을 위한 계층적 P2P 네트워킹과 2단계 압축기법

추 정 훈[†] · 윤 희 용[††]

요 약

유비쿼터스 컴퓨팅 환경이 추구하는 바는 인텔리전트한 컴퓨팅 환경을 만드는데 있다. 사람이 하나하나 개입하지 않아도 스스로 판단하여 일을 처리할 수 있는 환경을 만드는 것이다. 에이전트 기술은 에이전트 플랫폼이 에이전트들 사이에서 효율성과 안정성을 제공하는 동안 유비쿼터스 시스템의 효과적인 구현을 허용한다. 본 논문에서는 메시지들이 에이전트 플랫폼에서 병합되고 2레벨로 압축되는 계층형P2P 네트워킹 방법을 제안한다. 제안된 방법은 상황인식 어플리케이션의 대표적인 메시지의 정적 특성을 이용한다. 실제의 멀티에이전트 시스템의 실험은 기존 방법과 비교하여 응답시간과 처리량의 상당한 향상을 나타낸다.

키워드 : 멀티에이전트, 퍼베이시브 컴퓨팅, 에이전트 플랫폼, 분산 시스템, 센서 네트워크

Hierarchical P2P Networking and Two-level Compression Scheme for Multi-agent System Supporting Context-aware Applications

Jeong Hun Chu[†] · Hee Yong Youn[††]

ABSTRACT

Ubiquitous computing requires an intelligent environment where the users do not need to be involved in the operation. The agent technology allows effective implementation of ubiquitous system, while agent platform provides efficient and stable interaction between the agents. In this paper we propose a hierarchical P2P networking approach where the messages are combined inside the agent platform and then compressed in two levels. The proposed approach capitalizes the static nature of the messages of typical context-aware applications. Experiment with an actual multi-agent system reveals that the response time and throughput are significantly improved compared to the existing scheme.

Keywords : Multi-Agent, Pervasive Computing, Agent Platform, Distributed System, Sensor Network

1. 서 론

유비쿼터스 컴퓨팅은 어디서나 컴퓨팅을 실현할 수 있어야 하며, 컴퓨팅 기능이 주위환경에 내재되어 이로부터 정보를 획득하여 활용하거나 사용자가 인식하지 못하는 상태에서도 컴퓨팅 기능을 수행할 수 있어야 한다.

에이전트 플랫폼은 에이전트와 에이전트들 사이의 통신을 위해 표준 언어인 ACL(Agent Communication Language)를 사용한다. 그리고 에이전트 운영과 관리를 수행하기 위한 화이트 페이지(White Page) 서비스를 제공하는 AMS(Agent Management Service) 모듈을 가진다. 또한, 옐로우 페이지(Yellow Page) 서비스와 같은 추가적인 에이전트 서비스를 제공하기 위한 모듈도 필요하다[1].

멀티 에이전트 시스템은 자율적인 행동(behavior)들을 수행하는 에이전트들이 상호 협력을 통해 목표를 수행한다. 이러한 시스템에서는 지능적인 에이전트들 사이에 효율적이고 안정적인 상호작용을 위해 플랫폼이 필요하다. 따라서 에이전트들 사이의 모든 통신은 플랫폼을 거쳐서 일어나기 때문에 플랫폼으로 집중되어 있다. 이런 네트워크 모델의 경우 에이전트 플랫폼으로 과도한 부하가 집중되어 전체적인 네트워크 속도를 급격히 저하시킬 수 있으며, 각각의 에이전트들도 급격하게 트래픽이 증가하여 QoS(Quality of

※ 이 논문은 2008년도 중소기업청의 산학 공동기술개발지원사업, ETRI 연구개발 사업의 일환으로 추진되고 있는 USN 미들웨어 플랫폼 기술개발 사업의 지원에 의하여 연구되었음.
† 준 회 원 : 성균관대학교 전기전자컴퓨터공학과 석사과정
†† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2008년 7월 28일
수 정 일 : 1차 2008년 11월 25일, 2차 2008년 12월 23일
심사완료 : 2008년 12월 23일

Service)가 떨어질 수 있다.

이러한 에이전트 플랫폼과 에이전트 시스템의 과부하로 인한 전체 시스템의 성능 저하를 막고 연속적인 서비스를 제공하기 위해 기존의 중앙집중 형 네트워킹 방식의 플랫폼 부하를 분산하고자 상황인식 기반 그룹 에이전트의 부하 분산을 통해 에이전트 플랫폼 전체의 성능을 향상시킨 네트워크 모델링 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 다루며, 3장에서는 제안된 플랫폼을 소개한다. 4장에서는 제안된 에이전트 플랫폼의 구현을 보여준다. 마지막으로 5장에서는 제안된 접근 방법에 대한 결론과 앞으로의 발전 방향에 대해 기술한다.

2. 관련 연구

2.1 에이전트 컴퓨팅

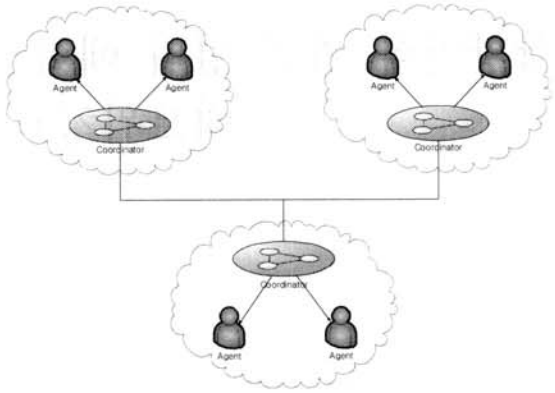
에이전트란 동적이고 복잡한 유비쿼터스 환경에서 공통된 목표를 달성하기 위해 사용자를 대신해서 작업을 수행하는 자율적인 프로세스이다[2].

이 에이전트는 다른 일반 소프트웨어와 구별되는 다음과 같은 가장 핵심적인 특성을 갖추고 있다. 사용자나 다른 에이전트의 직접적인 지시나 간섭 없이도 스스로 판단하여 행동하는 자율성(autonomy), 지식베이스와 추론능력을 바탕으로 사용자의 의도를 파악하여 계획을 세우고 학습을 통해 새로운 지식을 깨닫게 되는 지능(intelligence), 다른 에이전트와의 협업을 통해 공통된 목적을 달성하는 협동성(cooperation), 독립된 하나의 에이전트로는 처리하기 힘든 작업을 수행하기 위해 에이전트간의 통신으로 도움을 받는 사교성(social ability)이 있다. 그 외에 환경변화에 대해 반응할 수 있는 반응성(reactivity), 올바른 정보만을 주고받는 정직성(veracity), 그리고 반드시 목적을 달성하는 방향으로 작업을 수행한다는 이성적 행동(rationality), 학습할 수 있는 능력을 말하는 적응성(adaptation), 에이전트가 초기에 부여받은 임무를 수행하기 위해 자신이 생성된 호스트를 벗어나 네트워크 상의 다른 호스트에서 작업을 수행하거나, 임무 수행에 필요한 정보를 구하기 위해 네트워크 사이를 이동할 수 있는 이동성(Portability), 환경에 대하여 능동적으로 목표 지향적인 행동을 취하는 전향성(Pro-Activeness) 등을 만족해야 한다. 전향성 이란 에이전트가 자신의 환경 변화에 단순히 반응하는 것이 아니라, 스스로 판단하여 목표 지향적인 행동을 보여주는 것을 말한다[3]. 이러한 특성들을 완벽하게 갖추어져야만 에이전트가 되는 것은 아니다. 다만 이런 성질들을 많이 만족하면 할수록 더 지능 적인 에이전트에 가깝다 말한다.

2.2 Platform based Multi-Agent

멀티에이전트 시스템은 독립적인 에이전트로 해결하기 힘든 복잡한 문제를 에이전트간의 협업을 통해 해결한다.

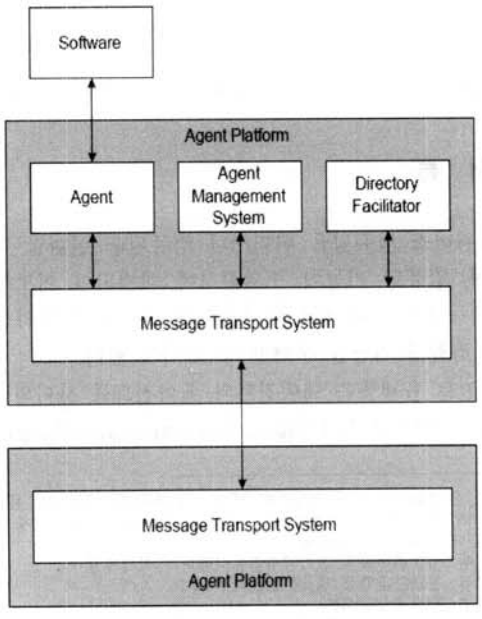
하나의 에이전트는 지식습득과 추론 등을 통해 문제를 해결할 수 있지만 개개의 에이전트들이 모든 기능을 다 갖출 수는 없으므로 다른 에이전트와 협업하여 문제를 해결해야 한다. 전체 시스템의 기능을 하나의 에이전트에 구성하는 것 보다는 각각의 공통된 기능들로 구성된 에이전트를 별도로 두어 서로 협업을 통해 문제를 해결한다면 시스템의 중복을 막을 수 있으며, 복잡한 문제를 해결 할 수 있다. 반면 기존 응용 프로그램의 특성에 따라 서로 다른 형태를 지니는 이형질성은 가장 중요한 고려사항이다[4]. 모든 에이전트를 처음부터 동일한 환경 하에서 새로이 개발하는 것은 자원의 낭비이고 시간도 많이 걸리므로 기존에 많은 노력을 들여 개발한 유용한 프로그램을 에이전트로 재사용하는 것은 매우 경제적이기 때문이다[5].



(그림 1) 멀티에이전트 구조

2.3 JADE Platform

JADE는 FIPA 표준을 따르는 멀티에이전트 구현을 위한 프레임워크이다. 자바언어를 사용하여 구현이 가능하며 에이전트를 관리하기 위해 컨테이너를 사용한다.



(그림 2) FIPA의 에이전트 관리 참조 모델

FIPA의 에이전트 관리 참조 모델의 각 항목들은 논리적인 서비스 집합들이며 물리적인 구성은 포함하지 않는다. 즉, 에이전트 관리 참조 모델은 (그림 2)에서와 같은 논리적인 주요 컴포넌트들로 구성된다. 에이전트는 애플리케이션의 기능들과 커뮤니케이션을 하며, 자율적인 특성을 지니는 프로세스이다. 에이전트들 사이는 ACL (Agent Communication Language)을 사용하여 통신한다.

DF (Directory Facilitator) : 에이전트 플랫폼의 부가적으로 선택할 수 있는 컴포넌트로 다른 에이전트에게 엘로우 페이지 서비스를 제공한다.

AMS (Agent Management System) : 에이전트 플랫폼에 꼭 필요한 필수 컴포넌트이다. AMS는 에이전트 플랫폼에 대한 접근과 사용에 대한 전반적인 제어를 한다. 오직 하나의 AMS가 하나의 에이전트 플랫폼에 존재한다.

MTS (Message Transport Service) : 서로 다른 에이전트 플랫폼 상에 에이전트들 사이에 기본적인 통신 방법을 제공한다[6].

2.4 P2P Service

P2P는 크게 순수(pure) P2P와 혼합형 (hybrid) P2P의 2가지 방식이 존재하고 있다. 순수 P2P 방식은 최초의 상대자 정보를 가지는 피어들이 상호간에 IP주소 리스트 등 개인 정보를 공유하여 서버 없이 직접 연결하는 방식이다. 혼합형(hybrid)P2P 방식은 상대자의 접속 상태 및 요청 자료의 검색을 중앙서버를 이용하여 제공해주는 방식이다 [7-9].

2.5 압축 알고리즘

압축 알고리즘은 그 중요성으로 인해 오랫동안 연구되어 왔고, 많은 알고리즘이 존재한다. 가장 대표적인 알고리즘은 Run-Length 압축 알고리즘과 Lempel-Ziv 압축 알고리즘이다.

Run-Length 알고리즘은 동일한 문자가 이어서 반복되는 경우 그것을 문자와 개수의 쌍으로 치환하는 방법이다.

0 부터 255 까지의 모든 문자가 사용된 파일을 압축한다면 일반적인 방법으로는 압축이 불가능하다. 그래서 탈출 문자(Escape Code)라는 것을 사용하여 문자가 반복되는 모양을 압축할 때 <탈출 문자, 반복 문자, 개수>와 같이 표현한다.

Source : ACDDDDBCBAAAAAACDB
Compression : AC*D4BCB*A6CDB

(그림 3) Run-Length 알고리즘

Source : ACDDDBCBCDB
Compression : ACD <1, 9> CDB

(그림 4) Lempel-Ziv 알고리즘

Lempel-Ziv 알고리즘은 현재의 패턴이 가까운 거리에 존재한다면 그것에 대한 상대적 위치와 그 패턴의 길이를 구해서 <탈출 문자, 상대 위치, 길이>로 패턴을 대치하는 방법이다.

Lempel-Ziv 알고리즘의 가장 대표적인 방법은 정적 사전(Static Dictionary)법과 동적 사전(Dynamic Dictionary)법이다. 동적 사전법은 파일을 읽어 들이면서 사전을 구성해야 하는 부담이 생기기 때문에 속도가 느리다는 단점이 있으나, 임의의 파일에 대해 압축률이 좋은 경우가 많다.

에이전트가 빈번하며, 반복적인 정보를 주고받는 점을 감안한다면 좋은 효과를 가져다 줄 것으로 확신한다[10-14].

3. 제안하는 모델

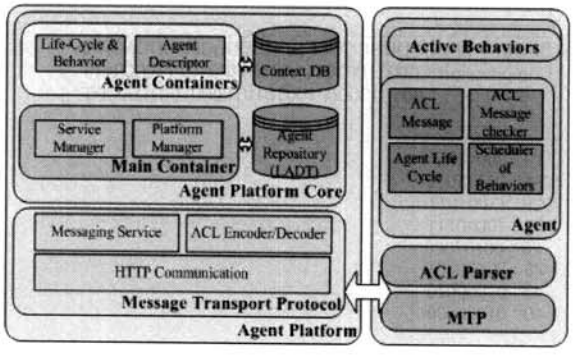
본 장에서는 AP(Agent Platform) 아키텍처에 대한 소개를 한 다음, 에이전트 사이에서의 정보를 Huffman Tree 와 Lempel-Ziv 알고리즘을 이용한 압축기법을 통해 효과적이며 신뢰적인 전송을 위한 혼합형 P2P를 적용한 플랫폼 모델에 대해 기술한다.

3.1 Agent Platform Architecture

에이전트 플랫폼은 내부적으로 에이전트 관리를 위한 주요한 정보와 기능 및 상황인식 정보들을 포함한다. AMS와 DF는 각각 에이전트로서 플랫폼에 등록 된다. (그림 5)에서 에이전트 플랫폼은 HTTP를 이용하여 신뢰성 있게 FIPA의 ACL 메시지를 처리하는 모듈인 MTP를 포함한다. 또한, 에이전트 플랫폼과 에이전트들을 관리하는 코어 모듈을 포함한다.

Agent Platform Core: 에이전트 플랫폼에 대한 전반적인 관리 및 에이전트 정보관리 및 등록, 삭제, 메시지 전달을 수행하며 에이전트 상태를 모니터링 한다. 에이전트 사이의 협업에 관한 정보 관리 기술 및 안정성을 지원한다.

Message Transport Protocol: Agent와 원격 AMS, DF 사이의 커뮤니케이션을 지원하고, FIPA 스펙에 따라 HTTP를 사용하여 Agent 사이에서 ACL 메시지를 사용하여 커뮤니케이션을 지원한다.



(그림 5) 에이전트 플랫폼의 구조

3.2 압축 알고리즘

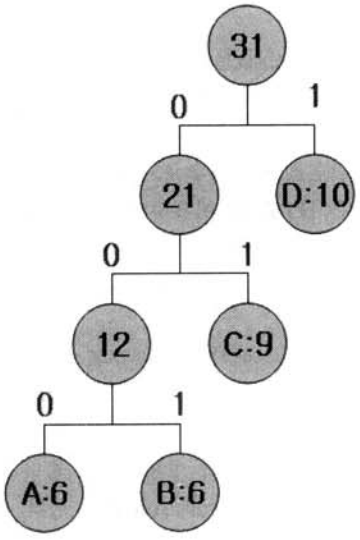
에이전트의 특성상 빈번하며 반복적인 정보들을 주고 받게 된다. 이러한 정보들을 Huffman Tree 알고리즘으로 압축을 한 후 8Byte씩의 일정한 사전을 작성한 후 그 사전을 근거로 Lempel-Ziv 알고리즘으로 다시 압축하여 정보를 구성한다.

위 문자열은 총 글자수 31개, 총31Byte이다. 각 문자당 중복되는 빈도수를 나타낸다.

(그림 7)에서와 같이 가장 작은 값을 기준으로 트리를 구성한다. 각 트리의 왼쪽 가지에는 0, 오른쪽 가지에는 1을 부여한다. 상위 노드부터 순차적으로 표현하여 문자열을 압축한다.

Source : ADDBCDDDCCDBCDDACDDBCABBCCACBAA				
Character :	A	B	C	D
Frequency :	6	6	9	10

(그림 6) Huffman 알고리즘



(그림 7) Huffman Tree 압축

A = 000
B = 001
C = 01
D = 1

Source : ADDBCDDDCCDBCDDACDDBCABBCCACBAA
Compression : 000110010111101011001011100001110010100000100101010000010010000000

19 : 00011001
7A : 01111010
CB : 11001011
87 : 10000111
28 : 00101000
25 : 00100101
42 : 01000010
40 : 01000000

(그림 8) 메시지 압축

00 : 19 7A CB 87 28 25 42 40
01 : 03 85 AB 85 55 34 65 FF
...

(그림 9) Data Dictionary

표현된 문자열을 이용하여 압축을 진행하여 새로운 1Byte 문자로 재구성하면 8Byte로 압축된다. 31Byte였던 문자열이 약 1/4로 줄어든 것을 (그림 9)에서 볼 수 있다.

새로이 압축된 데이터를 전송하게 되면, 학습을 통해 (그림 9)와 같은 사전을 구성한다. 이러한 사전을 구성하기까지는 시간이 필요하다. 반복적인 메시지패턴의 학습이 필요하다. 학습된 데이터는 사전에 등록되고, 만약 동일한 패턴의 메시지를 다시 보내게 된다면 00의 인덱스만 전송하면 되므로 결과적으로 8Byte에서 1Byte로 약 1/8을 더 압축하는 효과가 발생한다. 위의 경우 만약 같은 패턴을 만나서 다시 똑같은 메시지가 발생 했을 경우 1/31 의 압축률을 보여준다.

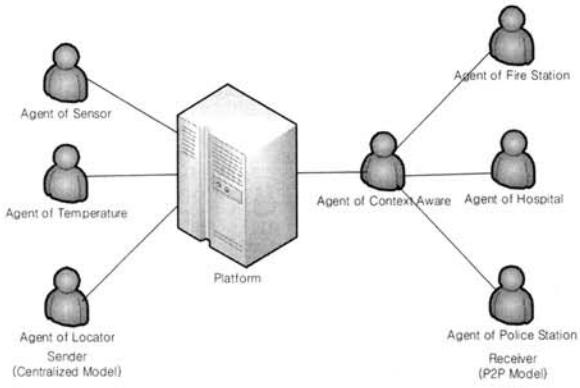
메시지가 처음 발생할 경우에는 기본적인 압축만을 통해 메시지를 구성하기 때문에 압축기법을 적용하기 전보다 오히려 시간이 더 소요 될 가능성이 있으며, 메시지의 변화가 빈번하여 학습량이 늘어나게 된다면 그에 따른 오버헤드가 발생 할 수도 있다. 하지만 시간이 지나면서 같은 패턴의 메시지가 동일하게 발생할수록 압축률은 높아져 플랫폼으로의 네트워크부하는 현저하게 떨어진다.

3.3 The Hierarchical P2P Networking model

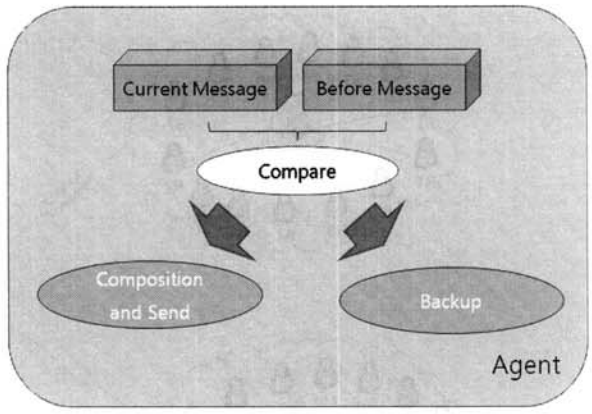
Super peer P2P 모델은 서버를 거치지 않고 사용자간의 파일을 교환할 수 있는 방식이다. 반면에 제안된 기법에서는 사용자간의 디스커버리를 제공하지 않는다. 이 점은 유비쿼터스 환경에서 사용자의 위치나 그룹의 변화가 빈번하기 때문에 모든 전송 위치는 AP가 주관하며, 슈퍼 에이전트는 AP 로부터 받은 데이터를 그룹 에이전트들에게 브로드캐스팅 한다.

기존의 중앙집중 형 네트워크 방식의 AP는 모든 통신이 AP를 통해서 이루어지기 때문에 유비쿼터스 환경에 유연하게 동작할 수 있는 반면, AP로 엄청나게 많은 부하가 몰리는 것을 피할 수 없다. 때문에 때로는 메시지의 손실(Message Loss)이 생기기도 하여 신뢰적인(Reliability) 전송을 기대할 수 없다. 그리하여 그러한 트래픽이 AP에 집중되는 것을 분산 시키고자 부분적인 P2P통신을 제안한다.

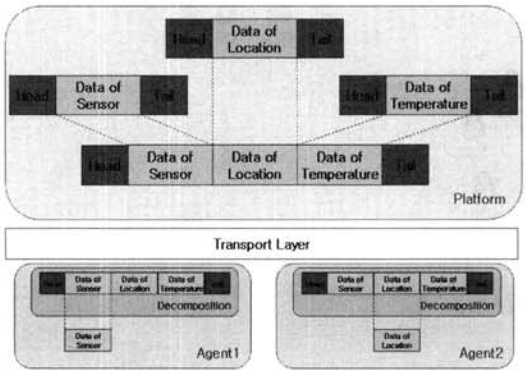
그룹별로 전송 받을 데이터들을 AP가 새로이 구성하여 임의의 슈퍼 에이전트인 한 에이전트에게만 메시지를 전달하면 이 에이전트는 자신의 그룹에 속해있는 에이전트들에게 데이터를 전송한다. 기존의 AP가 각각의 에이전트들에게 각각의 메시지들을 전달하는 것에 비해 전송량을 대폭 감소시킨다.



(그림 10) 혼합형 P2P 모델



(그림 12) 메시지 구조



(그림 11) 메시지 구조

기존의 중앙집중 형 네트워크 모델의 플랫폼은 모든 에이전트의 메시지들을 플랫폼을 통해서 전송하였다. 하지만 센서 데이터와 같은 빈번한 전송을 요구하는 에이전트가 많아질수록 플랫폼에 걸리는 부하가 기하급수적으로 늘어나기 때문에 때로는 정작 중요한 데이터를 전송하지 못하는 경우가 발생하거나 서비스를 받는 시간이 급격하게 늘어 날수가 있다. 이러한 상황으로 말미암아 정작 중요한 데이터를 서비스 받지 못하는 에이전트가 발생하여 그로 인해 전체 시스템에 영향을 미치게 된다. 예를 들어 어떤 지역의 온도가 상승하면서 사람들의 위치변화가 급격하게 변할 시에 그러한 상황인식을 통해 위급 상황을 발생시켜 소방서에 위급 상황을 알리고 환자의 수용이 가능한 가장 가까운 병원의 위치를 사용자에게 알려주는 시스템이 있다고 가정하자. 온도의 변화는 감지 되나 사람들의 위치변화가 어떤 다른 서비스로 인해 감지 되지 못한다면 원래의 에이전트 시스템의 목표를 이룰 수도 없을 뿐 아니라 전체 시스템에 치명적인 오류를 야기시킨다. 더 나아가서 소중한 사람의 생명을 앗아 갈 수 있는 심각한 상황이 발생하게 된다.

이러한 문제점을 해결하고자 혼합형 P2P 네트워크 모델(그림 10)을 제안한다. 데이터를 전송하는 에이전트는 기존의 방식을 그대로 따르고, 플랫폼은 그 메시지들을 (그림 11)과 같이 동일한 메시지의 셋으로 데이터를 구성한다. 그리고 데이터를 전송 받을 에이전트 그룹의 임의의 슈퍼 에이전트를 정해 그 에이전트로 메시지를 전송한다. 슈퍼 에이전트로 정해진 에이전트는 자신의 그룹에 속해있는 에이전트들에게 메시지를 브로드캐스팅 한다.

우리의 에이전트 플랫폼은 유비쿼터스 환경에서의 이동성을 고려하여 위치의 빈번한 이동으로 말미암아 발생하는 잦은 IP의 변화에 능동적으로 대처하고자 모든 도메인 및 에이전트들의 IP 의 관리를 플랫폼에서 하고 있다. 유비쿼터스 환경의 이동성에 유연하게 고려되었으나 그로 인해 모든 메시지가 에이전트로 집중되는 것을 피할 수가 없다.

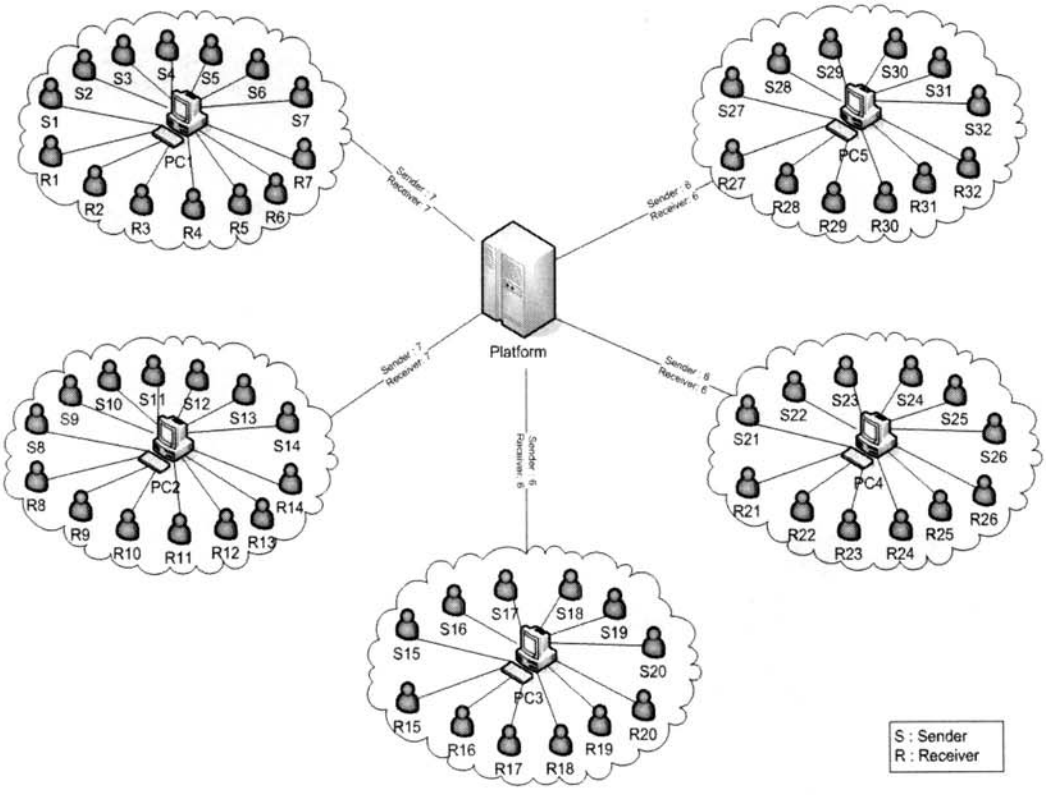
위와 같은 방법은 전송 받는 에이전트 쪽의 부하는 감소시켜주나, 데이터를 전송하는 에이전트 쪽의 부하는 감소시킬 수 없다. 이러한 점을 해결하고자 메시지를 전송하는 에이전트의 메시지를 (그림 12)과 같이 구성한다. 보내는 메시지를 구성하는 방법은 동일하나 보내기 전에 이전에 데이터와의 값을 비교하여 값이 일치 하다면 전송하지 않는다. 값이 일치 하지 않는다면 메시지를 저장하고, 전송하게 된다. 저장된 메시지는 다시 다음 전송 시 비교된다. 온도나 위치와 같은 데이터들의 값이 변하지 않을 경우에는 대부분의 전송 부하를 감소 시킨다.

이런 네트워크 모델을 적용함으로 인해서 기존의 하나하나의 메시지들을 전송하기 위해서 네트워크의 연결을 빈번하게 맺고 끊음으로써 네트워크의 부하를 야기 시켰던 부분을 대부분 감소 시킬 수 있어, 더 많은 에이전트들의 서비스를 보장 할 수 있다.

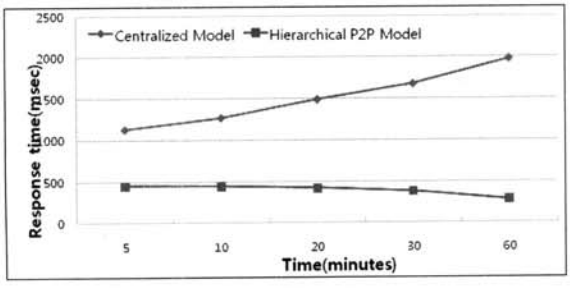
4. 실 험

본 논문에서는 제안 기법의 평가를 위한 실험 환경에 대해 설명하고, 클라이언트 수, 동일 시간에 얼마나 많은 데이터를 처리하는지를 기존기법과의 성능 비교를 수행한다.

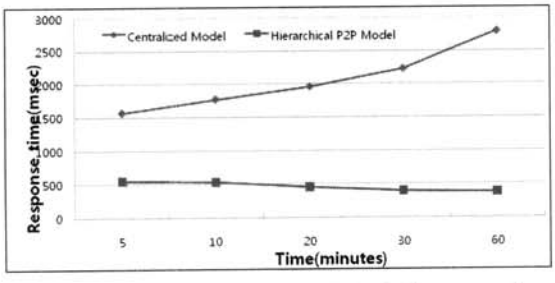
Intel Xeon Server로 구성된 에이전트 플랫폼과 Pentium-4 프로세서 및 1GB 메모리로 구성된 PC 10대로 에이전트들을 구성하여 실험을 수행했다.



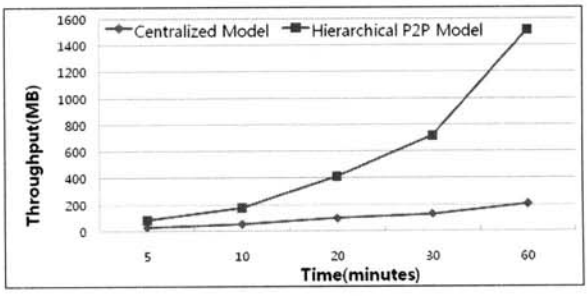
(그림 13) 테스트 환경



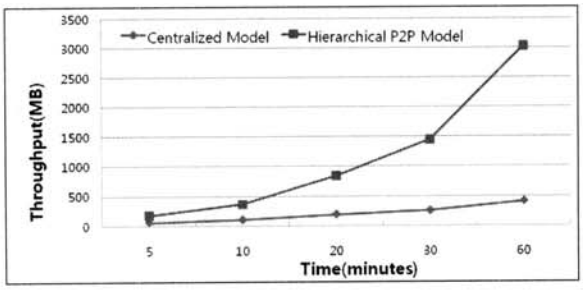
(그림 14) 시간에 따른 응답시간의 변화(16coupled)



(그림 15) 시간에 따른 응답시간의 변화(32coupled)



(그림 16) 시간에 따른 처리량의 변화(16coupled)



(그림 17) 시간에 따른 처리량의 변화(32coupled)

10KB의 데이터를 1000ms의 간격으로 보낼 때, 시간이 지남에 따라 부하로 인한 네트워크 성능을 측정하였다. (그림 14-15)에서 보는 바와 같이 시간이 지남에 따라 중앙집중 형 네트워크 모델의 경우 트래픽이 증가하여 응답시간이 크게 늘어나는 것을 볼 수 있다. 반면에 제시된 혼합형 P2P 모델에서는 시간이 지날수록 데이터의 학습률이 높아진다. 데이터 사전에 존재 확률이 높아짐으로써 데이터의 양이 줄어들기 때문에 그만큼의 부담이 줄고 그에 따라 응답시간

또한 줄어 드는 것을 볼 수 있다. 기존 시스템에 비해 플랫폼에 주었던 부하를 각각의 에이전트들로 분산을 시켰다. 유비쿼터스 환경에서의 이동성을 고려해 모든 통신이 플랫폼을 통해서 이루어지기 때문에 플랫폼으로 부하가 집중되고, 그로 인해 응답 시간이 늘어나면 서비스를 요청했던 에이전트는 그만큼의 시간을 더 기다려야 한다. 에이전트는 작업을 하는 시간보다 하지 않는 시간이 더 늘어나게 된다. 그런 유휴한 컴퓨팅 파워를 사용함으로써 플랫폼으로의 집중되는 부하를 감소 시켜 주는 것을 보여 준다.

(그림 16-17)의 차트는 단위 시간 동안 처리되는 데이터의 양을 보여준다. 기존의 시스템에서 작업시간이 늘어감에 따라 작업량 또한 늘어난다. 하지만 시간이 지날수록 연속적인 서비스 요청으로 인해 네트워크의 부하도 늘어나기 때문에 눈에 띄게 처리량이 늘어나지는 못한다. 반면에 제안된 기법을 적용한 시스템에서는 시간이 지나면서 반복되는 데이터가 늘어감에 따라 전송 데이터의 학습률이 증가하여 압축률 또한 증가하기 때문에 실제 전송되는 데이터의 크기는 점차 줄어들게 된다. 그림에서 보는 것처럼 16coupled 에서 32 coupled 로 에이전트 수의 변화로 인한 부하의 증가에도 제안된 방법에서는 학습을 통해 기존의 방식보다 개선된 응답시간의 감소와 처리량의 증가를 볼 수 있다. 처리량이 증가되었다는 것은 그만큼 부하가 분산되어 데이터의 처리량뿐만 아니라 한 플랫폼당 가용한 에이전트의 수가 늘어난 것이다. 본 논문에서는 가용한 에이전트의 수에 관한 성능평가는 하지 않았다. 앞으로 진행될 동적 재구성에 관한 연구 진행 시 평가될 것이다.

위의 성능 평가에 따르면 중앙집중 형 네트워크 모델의 경우 시간이 지남에 따라 트래픽이 증가하며 부하가 크게 늘어나는 것을 볼 수 있다. 반면에 제시된 혼합형 P2P모델에서는 에이전트의 수가 늘어나도 그 부하를 플랫폼에 집중되는 것이 아니라 각각의 에이전트로 분산되기 때문에 전송능력에는 크게 영향을 주지 못하는 것을 보여준다. 또한 시간이 지나면서 반복되는 데이터가 늘어감에 따라 전송 데이터의 학습률이 증가하여 압축률 또한 증가하기 때문에 실제 전송되는 데이터의 크기는 점차 줄어들게 된다.

위의 평가처럼 제시된 모델링 기법은 보다 많은 에이전트의 전송을 가능하게 하며, 플랫폼의 부하를 감소 시켜줌으로써 보다 안정적인 모델로 평가 받을 수 있다.

5. 결 론

본 논문은 지능적인 에이전트들 사이에 상호 커뮤니티 환경을 제공하는 에이전트 플랫폼의 새로운 네트워크 모델을 제시한다. 그리고 큐를 이용한 메시지의 흐름 제어 및 빈번한 메시지의 전송의 횟수를 감소 시키는 메시지의 구조화를 보여준다. 이로 인해 보다 많은 에이전트들 사이에서의 안정적인 커뮤니케이션을 보장한다. 제안한 모델은 커뮤니티 동적 재구성에 관한 정책에 관한 부분이 적용되지 않아 커뮤니티의 재구성 시 슈퍼 에이전트로부터 불필요한 데이터를 전송 받을 수도 있다. 앞으로 재구성에 따른 정책적인 부분의 연구가 필요할 것이며, 부하를 분산하는 과정에서 발생할 수 있는 고장에 관한 감내할 수 있는 연구가 진행될 것이다.

참 고 문 헌

[1] A.H. Park, S.H. Park, and H.Y. Youn, "A Flexible and Scalable Agent Platform for Multi-Agent Systems", Proceedings of WASET, Bangkok, pp.1-6, Jan., 2007.

[2] S.W. Han and H.Y. Youn, "A Middleware Architecture for Community Computing with Intelligent Agents", Ubiquitous Computing & Networking Systems, pp. 79-84, June, 2005.

[3] M. Wooldrige and N.R. Jennings, "Intelligent Agents: Theory and practice", The Knowledge Engineering Review, 10(2), pp.115-152, 1995.

[4] JADE, Java Agent Development framework, http://jade.org

[5] http://www.multiagent.com/

[6] JADE, Java Agent Development framework, http://jade.org

[7] A.T. Stephanos, "A survey of peer-to-peer content distribution technologies", Proceedings of ACM Computing Surveys 2004, pp.335-371, December, 2004.

[8] Bo Leuf, "Peer-to-Peer: Collaboration and Sharing over Internet", Addison-Wesley, June, 2002.

[9] L. Ciminiera, A. Sanna, and C. Zunino-Polito, "Survey on grid and peer-to-peer network technologies"Technical report, Proceedings of EGSO 2002, Oct., 2002.

[10] A. Gersho, "Advances in speech and audio compression", proceedings of the IEEE, pp.901-918 Vol.82, 1994.

[11] N.B. Body and D.G. Bailey, "Efficient representation and decoding of static Huffman code tables in a very low bit rate environment", Proceedings of ICIP 1998, pp.90-94 Vol.3, Oct., 1998.

[12] M. Bytrom and S. Kaiser, "Soft decoding of variable-length codes", Proceedings of ICC 2000, pp.1203-1207 Vol.3, June, 2000.

[13] J. Ziv and A. Lempel. "A Universal Algorithm for Sequential Data Compression", IEEE Trans. Inform. Theory, pp.337-343 Vol.IT-23 No.3, 1977.

[14] A. Anastasiadi, S. Kapidakis, C. Nikolaou and J. Sairamesh, "A computational economy for dynamic load balancing and data replication", Proceeds of 1st International Conference on Information and Computation Economies, Charleston SC, pp.166-180, 1998.

윤 희 용

e-mail : youn@ece.skku.ac.kr

1977년 서울대학교 전기공학과(학사)

1979년 서울대학교 전기공학과(석사)

1988년 Univ. of Massachusetts at Amherst 컴퓨터공학과(박사)

1988년~1991년 Univ. of North Texas. 조교수

1991년~1999년 Univ. of Texas at Arlington 부교수

1999년~2000년 ICU 교수

2000년~현 재 성균관대학교 정보통신공학부 교수 및 유비쿼터스 컴퓨팅기술연구소 소장

관심분야 : 모바일 컴퓨팅, 분산처리, 유비쿼터스컴퓨팅

추 정 훈

e-mail : occupier@skku.edu

2006년 한국산업기술대학교 컴퓨터공학과 (학사)

2007년~현 재 성균관대학교 전기전자 컴퓨터공학과 석사과정

관심분야 : 분산 처리, 센서 네트워크, 유비쿼터스 컴퓨팅