모바일 환경에서 분산된 이동 에이전트에 동일한 수준의 QoS를 보장해주기 위한 구조

전규영, 윤희용
성균관대학교 전자전기컴퓨터공학과

An architecture which guarantees the same level QoS to each distributed nomadic agent in mobile environment

Kyu-Yeong Jeon, HeeYong, Youn
Sungkyunkwan University
E-mail : reo726@nate.com, youn@ece.skku.ac.kr

요 약

멀티 에이전트 시스템은 다수의 에이전트들이 서로 협력하여 특정 목적을 달성하는 것을 목표로 한다. 이러한 시스템에서는 지능적인 에이전트들 사이에 효율적이고 안정적인 상호작용을 위한 에이전트 플랫폼이 존재한다. 본 논문에서는 각 에이전트의 네트워크 상태를 실시간으로 감지하여 통신에 있어서 모든 에이전트에 동일한 QoS를 보장하는 방법을 제안한다. 제안된 QoS 보장 방법은 동적인 가중치 변환 방법을 통해 이동하는 에이전트에도 늘 동일한 QoS를 보장해 줄 수 있다.

1. 서론

오늘날 우리는 디지털 시대에 살고 있다. 누구나 쉽게 주변에서 “유비쿼터스” 라는 용어를 들을 수 있고, 사용한다. 유비쿼터스 환경에서 사용자는 언제 어디서나 컴퓨터를 활용한 지능적인 서비스를 제공받을 수 있으며, 그 서비스는 계속적이며 또한 인지하지 못하는 사이에도 계속 제공된다. 이러한 유비쿼터스 환경에서 멀티 에이전트 시스템은 서비스를 제공하는데 있어 꼭 필요한 플랫폼이 될 것이다.

멀티 에이전트 시스템은 자율적으로 동작하는 지능적인 에이전트와 에이전트들 사이에 효율적이고 안정적인 상호 작용을 위해 에이전트 컨테이너를 통해 효과적인 에이전트 관리 기능을 제공하는 에이전트 플랫폼으로 이루어진다. 멀티 에이전트 시스템에서는 에이전트 간의 상호 협력을 통해 목표를 수행한다[1].

하나의 목표를 수행하는 에이전트들 혹은 동일하거나 비슷한 목표를 수행하는 에이전트들을 관리상의 편의를 위해 그룹으로 묶어서 관리하는 에이전트 그룹핑 기술이 사용되는데, 같은 그룹에 속한 에이전트는 같은 수준의 통신 품질을 제공받아야 한다.

하지만 에이전트 간의 통신은 공공 통신망을 통해서 이루어지며, 공공 통신망의 기본적인 성격은 "best effort“ 에 해당한다. 이는 에이전트 간의 통신에 있어서 QoS를 보장받지 못한다는 것을 의미한다. 특히 시간에 있어서 각기 다른 지연 시간을 갖는 네트워크에 속한 두 에이전트는 동일한 수준

본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스컴퓨팅 및 네트워크원천기반기술개발사업의 B2B3-B1-10M 과제로 지원된 것임

의 QoS를 보장 받을 수 없다. 더욱이 멀티 에이전트 시스템은 모바일 환경에서 동작한다. 모바일 환경은 특성상 지연시간이 길고, 패킷 손실이 발생할 수 있으며, 특히 통신 위치에 따라 통신 품질의 편차가 크다.

에이전트 플랫폼은 이러한 문제점을 해결하기 위해 각 에이전트에 적절한 가중치를 부여함으로써 상대적인 메시지의 기아현상을 줄일 수 있다. 하지만 고정적으로 결정된 가중치는 에이전트가 다른 장소로 이동하게 되면 오히려 그 격차를 크게 만들 수 있다.

따라서 본 논문에서는 이동하는 에이전트의 위치에 따라 실시간으로 통신 품질을 감지하고 그에 따라 가중치를 변경시켜 동일 그룹의 에이전트에 가능한 한 유사한 통신 품질을 제공하는 방법에 대해 연구하고, 실제로 에이전트 플랫폼에 적용하여 그 유용성을 검증해 본다.

2. 관련 연구

에이전트 간 통신의 QoS를 보장하기 위한 연구로는 다음과 같은 연구들이 진행되고 있다.

2.1 QoS Negotiator

QoS 협상자는 에이전트 기술을 이용하여 통신망과 로컬 자원을 감시 및 예약, 선점하는 방식으로 QoS를 보장해준다[2]. QoS 협상자는 서로 상호통신하는 몇 개의 에이전트들로 구성된다. 구성되는 시스템/자원은 다음과 같다:

- 로컬 자원 (시피유와 메모리 등)
- 네트워크 자원 (로컬 통신 드라이버, 대역폭 중개자, 피어 제어/모니터링 에이전트 등)
- 로컬 및 원격 응용들

이런 상호작용들에 의해 응용에 적절한 네트워크에 전달되는 QoS 파라미터와 로컬 QoS 파라미터를 결정하는 것이 가능해 진다.

하지만 이러한 방법은 기본적으로 각각의 에이전트에 원하는 QoS를 보장해 주기 위한 방법이기 때문에 서로 다른 지연시간을 가진 네트워크에 있는 같은 그룹의 에이전트들에 동일한 수준의 에이전트를 보장하는 데에는 적절하지 않고, 또한 로컬과 네트워크 자원의 감시 및 예약 등의 모니터링 및 제어작업으로 인한 오버헤드가 크다는 단점이 있다.

2.2 FIPA Specifications for QoS Support

FIPA(Foundation for Intelligent Physical Agents) 이동 응용 지원 명세는 미들웨어가 응용들의 변경을 통해 이동 환경을 지원하는 것에 대해 정의한다[2][3]. 그리고 명세서는 MTP(Message Transport Protocols)와 그 아래의 MTC(Message Transport Connections)를 관리하고 제어하는 에이전트들을 정의한다. 또한 메시지 전송 서비스의 QoS를 표현하기 위한 온톨로지도 정의한다.

미들웨어 에이전트는 다음과 같이 정의된다.

- MTP와 MTC의 QoS에 대한 감시, 측정, 그리고 분석을 담당하는 모니터 에이전트(MA).
- MTC를 제어하고 MTP의 선택을 수행하는 컨트롤 에이전트(CA)

FIPA 장치 온톨로지 명세는 장치 프로필이 명시된 온톨로지에 의해 장치의 속성을 취급한다[4]. 예를 들어 D1, D2와 같이 두개의 장치가 있을 때, 직접적으로나 중개자를 통해서 장치 프로필을 서로 교환할 수 있고, 제공되는 서비스의 목록을 이웃 장치로부터 얻어올 수 있다. 이러한 목록은 마이크로폰이나 핸즈프리, GPS와 같은 하드웨어와 소프트웨어 서비스를 포함할 수 있다. 이러한 명세 내용들은 EU가 후원하는 프로젝트인 CRUMPET에서 사용자를 위한 개인화된 모바일 서비스를 위해 데이터 통신 프레임워크의 한 부분에서 사용된다[5].

3. QoS 보장을 위한 에이전트 통신 설계

본 장에서는 동일한 그룹에 소속된 에이전트들에 동일한 QoS를 보장해 주기 위한 과정을 설명하고 가상의 시나리오를 상정하여 실제 동작하는 프로세스를 통해 각 에이전트에 비슷한 수준의 QoS가 적용 되는지를 살펴본다.

3.1 기본 컨셉

제안된 방법에서 우리는 각 에이전트에 비슷한 수준의 QoS를 제공하기 위해 가중치 변경 방식을 사용하였다. 이는 여러 에이전트 가운데 상대적으로 통신 속도가 느린 에이전트에 높은 가중치를 부여해 주고, 통신 속도가 빠른 에이전트에는 낮은 가중치를 부여한다. 높은 가중치를 가진 에이전트는 통신하기 위한 좀 더 많은 기회를 가질 수 있게 된다. 또한 가중치는 유연하고 동적으로 변화함으로써 모바일 장치의 이동에 따른 네트워크 상황의 변화에도 적용될 수 있도록 하였다.

우리가 제안한 방법을 적용하기 위해 MTL(Message Transport Layer) 에 Circular queue를 사용하였고, 가중치를 구하기 위해 기준값을 사용한다. 원형큐는 최근 통신 딜레이 값을 저장하여 현재 소속된 네트워크의 통신 스피드를 감지하기 위해서 사용한다. 기준값이란 현재 통신 속도가 얼마나 빠른지 느린지의 정도를 구하기 위한 값으로 가중치를 구하는데 사용되며 에이전트 플랫폼이 각 에이전트에 알려준다. 가중치가 정해지면 각 에이전트는 가중치만큼의 통신 기회를 더 가지게 되고 그 방법에는 멀티 큐, 스레드 등 다양한 방법을 적용할 수 있다. 본 논문에서는 스레드를 이용하여 각 에이전트에 가중치를 적용하여 그 시뮬레이션 결과를 보인다.

3.2 QoS를 지원하는 에이전트 통신을 위한 인터페이스

각 에이전트에 동급의 QoS를 보장해 주기 위한 통신 라이브러리의 주요 인터페이스는 다음과 같다.

- getStatusOfAgent(gid, id)

에이전트의 현재 통신 환경에 대한 정보를 가져오는 함수로써 인자값으로 에이전트의 그룹 ID와 에이전트의 ID를 가진다. 리턴값으로는 쿼리를 보낼 때의 시간을 기록하고 응답이 오는 시간을 측정하여 메시지를 주고 받는데 걸리는 전체의 시간을 ms 단위로 측정하여 반환한다. 이 결과값을 바탕으로 각 에이전트에 부여되는 가중치를 변경한다. 반환되는 타임은 최근 주고 받은 메시지의 평균값을 반환하는데 정책에 따라 최근 몇 번 이내의 값의 평균을 구하거나 최근 일정 시간동안 주고 받은 메시지 지연시간의 평균을 구하는데, 시간이나 횟수가 짧을수록 에이전트의 통신 환경 변화에 민감하게 반응할 수 있는 반면 시스템에 부하가 커지고 작은 노이즈와 같은 통신 환경 이외의 요소에 영향을 받을 가능성이 커진다. 반면 평균을 구하는 시간이나 횟수를 증가시키면 잦은 통신 환경 변화에 민감하게 반응하기 힘들지만 지터와 같은 외부 요소에 영향을 적게 받게 된다.

- setWeight(gid, id)

각 에이전트에 실질적으로 가중치를 설정해 주는 부분으로써 그룹ID와 에이전트 ID를 인자값으로 받는다. 각 에이전트는 가중치를 변수로 갖는데, 이 함수는 가중치를 변경시킬 때 사용하게 되며, 그 가중치는 실제 통신을 하게 될 때 참고하여 통신 환경이 좋지 않은 에이전트에 좀 더 많은 통신 기회를 준다. 가중치는 setBaseValueOfDelay 함수에서 주어지는 값을 기준으로 산정된다.

- setRoundTripTime(gid, id)

setRountTripTime 함수는 다른 함수와 마찬가지로 에이전트 그룹 ID와 에이전트 ID를 인자값으로 가진다. 이 함수는 플랫폼이 에이전트와 통신하기 위해 메시지를 전송하는 함수 내부에서 호출되는 함수로, 메시지를 전송하고 다시 응답을 받았을 때 까지 걸린 시간을 계산하여 원형 큐나 벡터와 같은 각 에이전트의 RTT값의 내역을 가지는 자료구조에 저장된다. 주로 getStatusOfAgent()의 정책에 따라 최근 데이터의 횟수만큼을 크기로 가지는 원형 큐를 사용한다.

- setBasisValueOfDelay(int)

이 함수는 통신 속도가 느린 에이전트에 추가적인 가중치를 줄 것인지를 결정하는 기준이 되는 응답시간을 설정해 둔다. 이 값을 기준으로 추가적인 가중치의 여부와 그 정도를 결정한다.

3.3 QoS 보장을 위한 에이전트 통신 구조

QoS 보장을 위한 에이전트 통신 구조는 다음 그림 1과 같다.

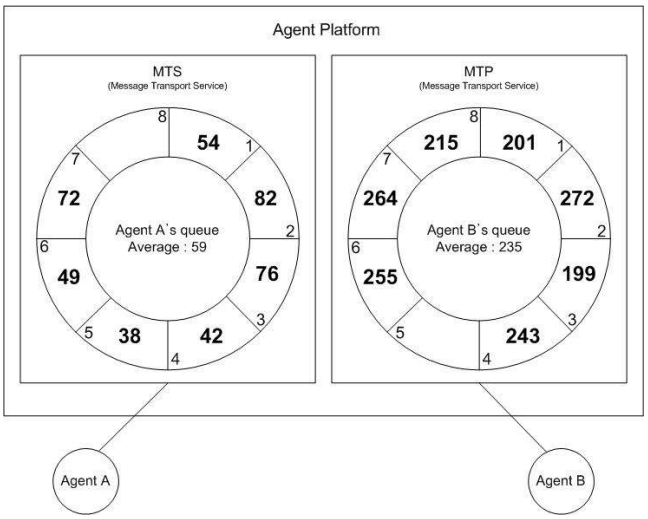


그림 1. QoS 보장을 위한 에이전트 통신 구조

위 그림 1에서 두 에이전트 A와 B는 tail을 나타내기 위한 한 칸을 제외하고 7칸을 가진 원형 큐를 가진다. 즉 이 에이전트 플랫폼은 최근에 주고 받은 7개의 메시지의 RTT의 평균값으로 가중치를 계산한다. 3.2절에서 설명한 인터페이스에서 setRound TripTime() 함수는 원형큐의 가장 오래된 값을 빼고 가장 최근에 측정된 값을 저장한다. getStatus OfAgent() 함수는 원형큐 내부 값의 평균값을 리턴해 준다. 그리고 적절하게 수립된 각각의 환경에 맞는 전략 또는 정책을 수립하여 동일 그룹 내의 에이전트의 평균값들을 연산하여 각 에이전트에 부여될 가중치를 계산하고 setWeight() 함수로 각각의 에이전트에 가중치를 준다. 그리고 에이전트와 통신할 때 가중치에 따라 가중치가 높은 에이전트에 더 많은 메시지를 전송할 기회를 준다.

3.4 QoS 보장을 위한 에이전트 통신 시나리오

같은 그룹 내의 한 에이전트가 이동중일 때에도 비슷한 QoS를 보장해 주기 위한 시나리오를 살펴본다.

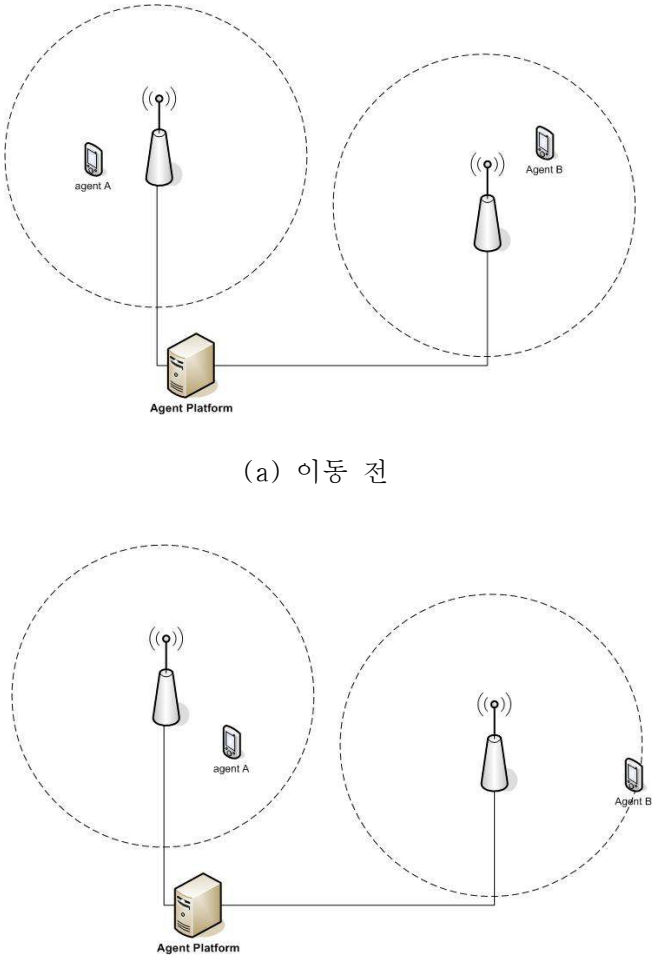


그림 2. 이동하는 에이전트의 QoS 보장

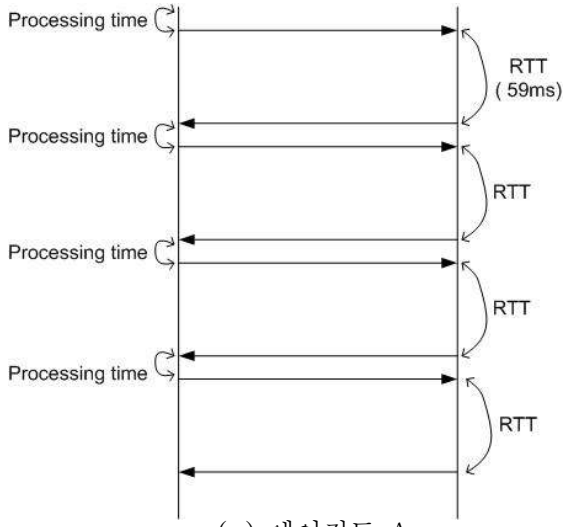
그림 2는 유선 통신망에 연결된 Agent Platform Server와 Access Point(AP), 그리고 무선으로 AP를 거쳐 Agent Platform과 통신하는 두 에이전트 A와 B로 이루어 져 있다. (a)에서는 두 에이전트 A와 B가 모두 AP와 가까운 거리에 있어서 원활한 통신이 가능하며 각각 비슷한 가중치를 가진다. 반면 에이전트 A와 B가 각각 (b)의 위치로 이동하였을 때, 에이전트 A는 여전히 통신 지연 시간이 짧지만, 에이전트 B는 AP의 범위의 가장 바깥쪽으로 이동하여 통신 지연시간이 길어지게 된다. 7번의 메시지를 주고받은 후, 에이전트 B의 원형 큐의 값이 위 그림 1에서의 에이전트 B의 그것과 같아 졌다고 가정해 본다.

에이전트 A는 보다 통신 환경이 좋은 영역에 속해 있고 평균 RTT는 59ms이다. 에이전트 B는 통

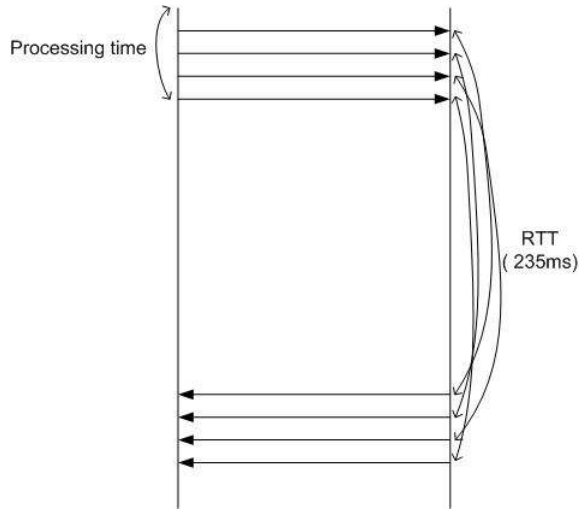
신 지연시간이 큰 네트워크에 속해 있고 평균 RTT는 235ms이다.

각 에이전트에 가중치를 계산하기 위한 방법은 각각의 환경에 따라 다양하게 달라질 수 있겠지만, 가장 단순한 방법으로 에이전트 A와 B의 메시지 전송 비율을 활용하는 방법이 있다. 즉 A:B = 17:4라는 비율을 이용하여 21/17: 21/4의 가중치를 적용하는 방법이다. 즉 대략 1:4의 가중치를 갖게 되며 이는 에이전트 A로 하나의 메시지를 전송할 때 에이전트 B에는 4개에 상당하는 메시지를 전송하는 것을 의미한다.

그림으로 표현하면 다음과 같다.



(a) 에이전트 A



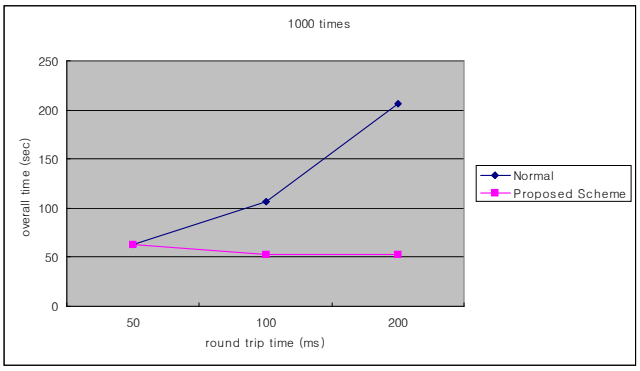
(b) 에이전트 B

그림 3. 시간에 따른 에이전트와 에이전트 플랫폼 간의 통신 순서

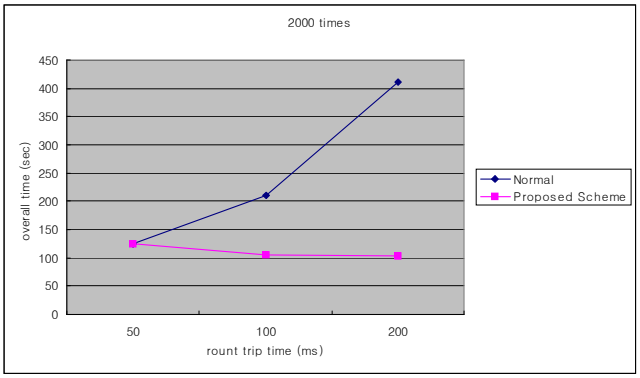
3.5 제안된 방법의 시뮬레이션 결과

본 논문에서 제안한 메커니즘을 실제 구현했을 때의 결과를 알아보기 위해 다음과 같이 임의로 환경을 구축하여 테스트 하였다. 임의로 설정한 환경은 다음과 같다;

- 메시지를 보내고 다시 응답을 받기까지 소요되는 시간을 대략 50, 100, 200에 가까운 값을 가지도록 설정하였다.
- 명확한 결과를 알아보기 위해 기반이 되는 디폴트 값을 50으로 설정하였다.
- 제안된 메커니즘을 적용한 방식과 적용하지 않은 방식으로 각각 1000번과 2000번씩 테스트 하였다.
- 세 번째 테스트에서는 네트워크의 상태가 변화하는 상황에서 동적으로 동일한 QoS를 보장해 줄 수 있는지를 테스트하기 위해 응답시간을 변경해 가면서 테스트 하였다.



(a) 1000 번 전송 시 전송 시간 비교



(b) 2000 번 전송 시 전송 시간 비교

그림 4. 제안된 방법을 적용한 경우의 성능 비교

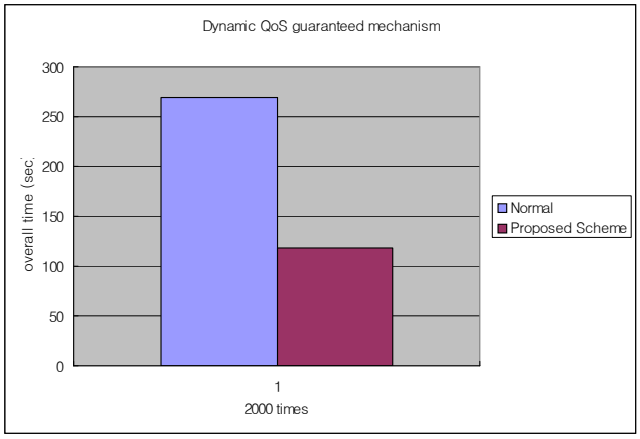


그림 5. 동적 가중치 변경 기법을 적용한 경우

그림 5에서의 결과를 보면 순차적으로 150ms, 50ms, 200ms, 100ms 순으로 응답 시간이 바뀌도록 했음에도 불구하고 위에서 일반적으로 전송했을 때와 동일한 결과를 보여주는 것을 알 수 있다.

4. 결론

본 논문에서는 멀티 에이전트 시스템에서 동일한 그룹 내의 에이전트가 서로 다른 지연시간을 가지는 네트워크에 포함될 때, 간단한 연산을 통해 전체의 에이전트 시스템의 성능에 큰 영향을 주지 않으면서 동일한 수준의 QoS를 보장 받는 방법에 대하여 연구한 결과를 정리하였다.

향후 실제 서로 다른 지연 시간을 가지는 유무선 네트워크 환경에서 많은 테스트를 통해 다양한 정책을 세우고 그에 맞는 가중치 산출 수식을 고안해 볼 것이다.

또한 여러 그룹이 존재할 때, 같은 그룹 내의 에이전트에는 동일한 수준의 QoS를 보장하면서 그룹의 중요도에 따라 서로 다른 QoS를 제공하는 방법으로 연구를 확장시켜 나갈 것이다.

[참고문헌]

[1] Ae Hee Park, So Hyun Park, Hee Yong Youn, "A Flexible and Scalable Agent Platform for Multi-Agent Systems", 2007 IEEE International Conference, 2007

[2] Nahrstedt K. and Smith J: The QoS Broker. IEEE Multimedia Magazine, Sprg, 1995

[3] XC00014D, FIPA Nomadic Application Support Specification, 2001

[4] PC00091B, FIPA Device Ontology Specification, 2001

[5] Poslad S, et.al: CRUMPET: Creation of User-Friendly Mobile Services Personalised for Tourism, Proceedings of 3G, 2001