유비커터스 환경에서 이동 에이전트를 이용한 서비스 디스커버리 기법 설계

김경인, 선승상, 엄영익
성균관대학교 전기전자 및 컴퓨터공학과
e-mail: {kimki95, threes, yieom}@ece.skku.ac.kr

Service Discovery Scheme Design Using Mobile Agent in Ubiquitous Environment

Kyung-In Kim, Seung Sang Sun, Young ik Eom
Dept of Electrical and Computer Engineering,
Sungkyunkwan University

요 약

유비쿼터스 환경은 인간이 생활하는 각종 기기들에 컴퓨팅 능력과 네트워킹 능력을 가지고 상호간에 정보를 교환하며 서비스를 제공하는 환경을 말한다. 이러한 유비쿼터스 환경이 구축되기 위해서는 컴퓨팅 인프라 및 네트워킹 인프라를 기반으로 다양한 서비스 프레임워크가 지원되어야 한다. 여러 서비스 프레임워크에서 중요한 한가지로 서비스 및 자원에 대한 디스커버리 엔진을 들 수 있다.

본 논문에서는 유비쿼터스 환경에서 이동 에이전트를 이용하여 효율적인 디스커버리 기법을 제안한다. 본 기법을 이용하면 빠른 서비스의 검색뿐만 아니라 서비스 목록들의 일관성을 유지해 높은 신뢰성을 제공할 수 있다.

1. 서론

유비쿼터스는 라틴어 어원으로 '동시에 도처에 존재하는', '편재하는' 등의 뜻으로 쓰인다. 즉 모든 사물에 컴퓨터가 내장되어 있어 장소나 시간에 구애받지 않고 항상 컴퓨터를 사용할 수 있는 새로운 패러다임을 말한다. 다가오는 유비쿼터스 컴퓨팅 환경은 다양한 기기들이 현실 세계의 사물과 환경 속으로 스며들어 유·무선 네트워크를 통해 상호 연결되고 언제 어디서나 이용할 수 있는 최적의 컴퓨팅 환경을 보일 것이다[1,2].

본 논문에서는 이러한 유비쿼터스 환경에서 사용자가 원하는 다양한 서비스 및 자원을 좀 더 효율적으로 검색하기 위해 이동 에이전트를 이용하는 메커니즘을 제공한다. 본 기법을 이용하면 하나의 홈 도메인뿐만 아니라 다중 도메인에서도 사용자가 원하는 서비스를 제공 받을 수 있으며, 네임 캐시 기법을 이용하여 좀 더 빠른 서비스 검색 및 서비스 목록들의 일관성을 유지하여 높은 신뢰성을 제공한다.

2. 관련연구

2.1 이동 에이전트(Mobile Agent)

이동 에이전트는 이기종 분산 환경에서 자율적으로 이동이 가능한 객체로 네트워크의 트래픽 감소, 비동기적인 상호 작용, 서비스의 분산 및 병렬처리 등을 지원할 수 있다. 이동 에이전트는 각각의 독립된 역할뿐만 아니라 정보와 자원을 서로 공유할 수 있기 때문에, 여러 에이전트간의 협업에 의해 한 에이전트가 해결하기 힘든 복잡한 문제들을 해결할 수 있으며, 충분한 자원을 보유한 호스트로 이주하여 계속 작업을 수행할 수도 있다[3,4].

2.2 네이밍(Naming)

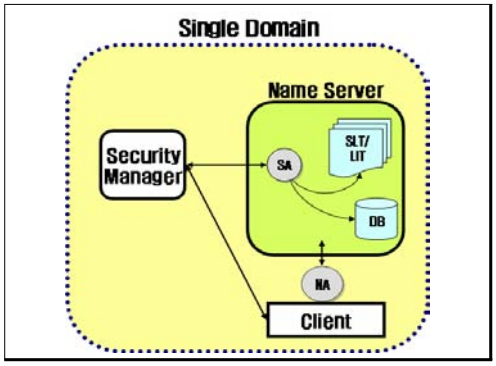
분산 환경에서 네이밍은 객체가 실제로 저장되어 있는 위치와 저장된 세부 내용을 투명하게 제공한다. 이는 사용자가 객체의 위치를 모르더라도 객체의 이름과 위치가 연관(mapping)되어 있기 때문에 원하는 서비스 이름으로 위치를 참조할 수 있다. 네이밍 서비스를 지원하기 위해서는 사용자가 먼저 이해하기

쉬운 이름으로 서비스를 등록하고, 이후 등록된 이름으로부터 객체의 위치를 참조할 수 있다[5].

3. 제안기법

3.1 시스템 구조

본 논문에서 제안하는 단일 도메인(Single Domain) 시스템 구조는 그림 1에서 보인다.

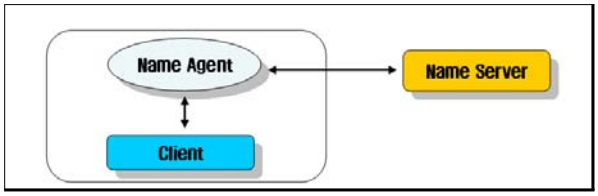


(그림 1) 단일 도메인 구조

단일 도메인은 세 가지 컴포넌트로 구성된다. 첫 번째 컴포넌트에는 사용자가 원하는 서비스를 요청하기 위해 네임 에이전트(Name Agent: NA)를 생성 시키는 클라이언트가 있으며, 두 번째 컴포넌트에는 실질적인 네이밍 서비스를 제공하는 네임 서버(Name Server: NS)가 있다. 네임 서버는 정적으로 상주하는 SA(Stationary Agent)를 통해 서비스를 요청 받고 해당 서비스를 제공하는 역할과 함께 서비스 목록이 저장되는 SLT (Service Location Table)와 LIT(Location Information Table)를 관리하는 기능도 담당한다. 마지막 컴포넌트로는 보안과 관련된 보안 관리자(Security Manager: SM)가 있다. 보안 관리자는 인증된 클라이언트만이 NS에 접속할 수 있게 하며, 인증된 클라이언트라도 접근 권한에 따라 서비스의 차등적 지원을 제공한다.

3.2 네이밍 모델(Naming Model)

본 논문에서 개발하고자 하는 네이밍 모델은 그림 2에서 보인다.



(그림 2) 네이밍 시스템

3.2.1 네임 서버(Name Server)

네임 서버는 사용자가 원하는 서비스 이름에 대한 서비스의 위치로 매핑 시켜주는 프로세스이다. 또한 네임 서버는 네임 스페이스(name space)들을 관리하고, 객체에 대한 속성을 관리하게 된다.

3.2.2 네임 에이전트(Name Agent)

네임 에이전트는 네임 서버의 위치와 네임 서비스의 분산을 사용자에게 투명성하게 제공한다. 따라서 네임 에이전트는 네임 서버와 클라이언트들 사이에서 동작하며, 존재하는 네임 서버들에 대한 정보를 유지하고, 사용자의 요청을 네임 서버에게 전송하는 기능을 수행한다.

3.2.3 네임 캐시(Name Cache)

본 논문에서는 네임 캐시를 이용하여 디스커버리를 위한 절차를 줄이고 서비스 목록들의 일관성을 유지하도록 한다.

■SLT(Service Location Table)

SLT는 홈 도메인에 존재하는 서비스 네임과 동일한 서비스에 대한 네임 참조를 가지는 테이블로서, 그 생성 과정을 보면 홈 도메인에 존재하는 서비스를 먼저 SLT에 등록시키고 동일한 서비스에 대한 네임 참조는 SA(original requestor)가 생성시킨 QM(Query Message)을 전송하며 그 결과에 따라 SLT에 기록하게 된다. SLT는 call-back 기법을 이용해 SLT간 일관성을 유지하도록 한다.

■LIT(Location Information Table)

LIT는 홈 도메인에 존재하지 않는 서비스에 대한 네임 참조를 가지는 테이블로서, 그 생성 과정을 보면 SA(original requestor)가 필요한 서비스를 찾기 위해 QM을 전송하며 그 결과에 따라 LIT가 업데이트 된다. LIT는 time-out 기법을 이용해 유효하지 않는 서비스 목록은 일정한 시간이 지나면 자동 삭제 되도록 한다.

3.3 자료구조

적응적 디스커버리 메커니즘을 지원하는 네이밍 기술 개발을 위한 자료구조는 그림 3에서 보인다.

Service table format

S_ID	S_Type	S_Attr	S_Loc	NS_List

SLT(Service Location Table)

S_ID	S_Type	S_Attr	NS_List

LIT(Location Information Table)

Message format

S_ID	S_Type	S_Attr	C_ID	OR_NS_ID	TTL	M_ID

QM(Query Message)

S_ID	S_Type	S_Attr	C_ID	OR_NS_ID	S_Time	NS_ID	M_ID

RM(Reply Message)

S_ID	S_Type	S_Attr	Dest_NS_List	M_ID

DM(Delete Message)

(그림 3) 자료구조

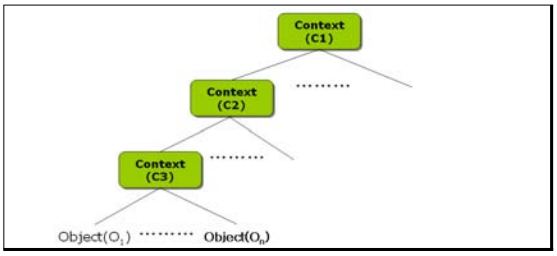
S_ID 필드는 등록된 서비스를 구분하며, S_Type 필드는 해당 서비스의 종류, S_Attr 필드는 서비스의 속성을 나타낸다. SLT의 S_Loc 필드는 서비스의 위치정보를 나타내며, NS_List는 동일한 서비스에 대한 네임 서버들의 위치 목록들을 저장하는 필드이다. 반면에 LIT의 NS_List는 홈 도메인에 존재하지 않는 서비스들에 대한 네임 서버들의 목록들을 참조하는 필드로 사용된다. 메시지 형식은 중복을 방지하기 위해 M_ID 필드를 가지며, C_ID와 OR_NS_ID 필드는 서비스를 요청한 클라이언트의 ID와 네임 서버의 ID를 나타낸다. TTL필드는 메시지의 무한루프를 방지하기 위해 최대 홉 수를 제한하고 있다. S_Time 필드는 유효하지 않는 서비스 목록 유지를 방지하기 위해 사용되며, NS_ID 필드는 해당 네임 서버의 ID를 저장한다. 마지막으로 Dest_NS_List 필드는 삭제할 네임 서버들의 목록을 나타낸다.

3.4 디스커버리 기법

본 논문에서 제안하는 디스커버리 메커니즘은 홈 내에서 서비스를 찾을 때와 홈 외부에서 서비스를 찾을 때로 나누어서 설명한다.

3.4.1 홈 내부에서 디스커버리 메커니즘

홈 내에서 사용자가 원하는 서비스를 SA가 검색하기 위해 context qualified name 구조에 따르며 이는 그림 4에서 보인다.



(그림 4) 홈 내에서 디스커버리

context는 네임 공간을 지리적, 구조적, 기능적 의미를 갖는 하나의 컴포넌트로 정의하며, context /name 쌍으로 형성되는 이름으로, 각 객체를 유일하게 식별하기 위해 사용된다. 유비쿼터스 환경에서, 산재되어 있는 객체들의 네임 관리를 위해 네임 정보들을 다중 네임 서버로 분산시킴으로써 효율적인 네임 관리를 지원하는 context 기반 qualified name 구조를 제안한다. 알고리즘 1에서는 홈 내에서 SA의 동작과정을 보인다.

```
when a SA receives a request from client
 find an entry E in SLT;
 if(there exists E){
   send E to the client;
   send QM to each NS in NS_List;
 }
 else{
  find an entry E' in LIT;
  if(there exist E'){
    send QM to each NS in NS_List
  }
  else
    send QM to the neighbor NSs;
 }
when the SA receive RM from other SAs
 update the NS_List information in RLT;
```

(알고리즘1) 홈 내에서 SA의 서비스 디스커버리 과정

클라이언트로부터 서비스 요청을 받은 SA는 먼저 홈 내에 등록된 서비스인지 확인하기 위해 SLT를 검색한다. SLT에 원하는 서비스가 없으면 LIT를 검색해 다른 NS에 있는 서비스임을 인식하고 인접한 NS에게 QM을 전송한다.

3.4.2 홈 외부에서 디스커버리 메커니즘

홈 외부에서 서비스를 검색할 때에는 그림 5와 같이 3가지 절차를 통해 찾는다.

(그림 5)홈 외부에서 디스커버리

NS farm은 여러 NS가 모인 하나의 그룹을 뜻한다. NS는 자신이 속한 NS farm에 등록된 다른 NS들에게 원하는 서비스를 질의 할 수 있으며, 인증된 NS farm간의 서비스 검색도 가능케 한다. 알고리즘 2에서는 홈 외부에서 SA의 동작과정을 보인다.

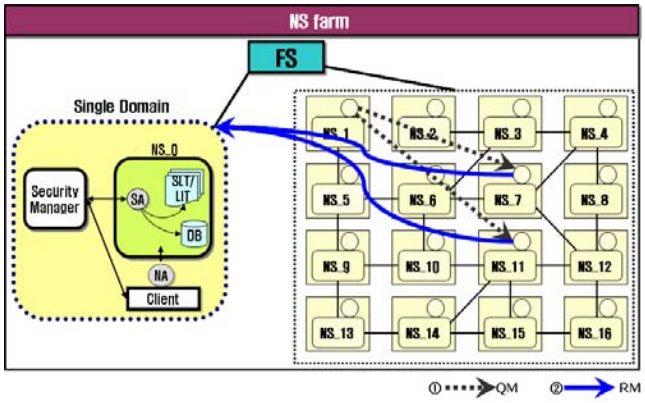
```
when SA receives a QM from another SA
 if(M_ID of the QM has already been registered in current NS)
   discard the QM;
 else{
   find an entry E in SLT;
   if(there exist E){
     send RM to the original requester;
     send QM to each NS in NS_List;
   else{
     find an Entry E in LIT;
     if(there exist E){
      send QM to each NS in NS_List;
     }
     else{
      send QM to the connected NS except the NS that sent QM;
     }
   }
 }
```

(알고리즘2) 홈 외부에서 SA의 서비스 디스커버리 과정

홈 외부에 있는 SA가 QM 메시지를 수신하면 먼저 메시지 중복 체크를 한다. SLT에 해당 서비스가 있으면 최초 서비스를 요청한 SA에게 RM을 보내고 해당 서비스가 없으면 LIT를 검색한다. LIT 검색 결과 존재하는 서비스면 해당 NS에게 QM을 전송하고 LIT에 존재하지 않는 서비스면 인접 노드에 QM을 전송한다.

3.6 시나리오

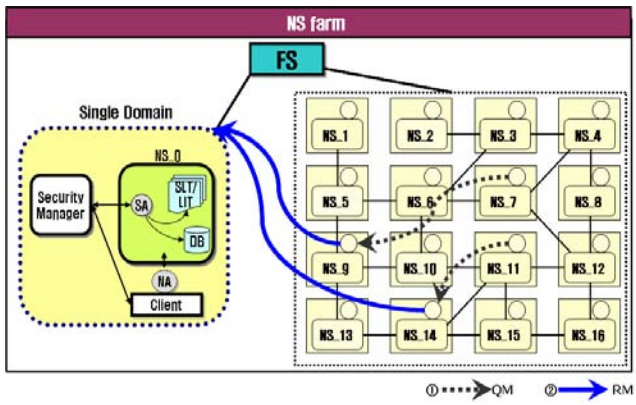
3.6.1 LIT를 이용한 Discovery 시나리오



(그림 6) LIT에 의한 discovery

그림 6에서 클라이언트는 원하는 서비스를 NS_0에게 요청했지만 NS_0의 SLT, LIT의 서비스 목록에 해당 서비스가 없을 때, SA(original requestor)는 NS farm에 등록 되어 있는 모든 NS를 관리하는 FS (Farm Server)를 통해 인접한 NS_1에게 QM을 전송한다. QM을 수신한 NS_1의 SA는 SLT에는 서비스가 등록 되어 있지 않지만 LIT에는 NS_7과 NS_11이 해당 서비스로 등록되어 있어서 QM을 전송한다. QM을 수신한 NS_7과 NS_11의 SA는 RM을 SA(original requestor)에게 전송하여 서비스를 등록 시킨다.

3.6.2 SLT를 이용한 Discovery



(그림 7) SLT에 의한 discovery

그림 7에서 NS_7과 NS_11는 SLT에 해당 서비스와 동일한 서비스가 NS_9와 NS_14에도 등록되어 있어서 QM 을 해당 네임서버로 전송한다. QM을 수신한 NS_9와 NS_14는 NS_0에게 RM을 전송하며, RM을 수신한 SA(original requestor)는 서비스를 등록하게 된다.

4. 결론 및 향후 연구계획

본 논문은 유비쿼터스 환경에서 사용자가 원하는 다양한 서비스 및 자원을 효율적으로 검색하기 위해 이동 에이전트를 이용한 메커니즘으로 하나의 홈 도메인뿐만 아니라 다중 도메인에서도 사용자가 원하는 서비스를 제공 받을 수 있게 한다. 또한, 네임 캐시 기법을 이용하여 좀 더 빠른 서비스 검색과 서비스 목록들의 일관성을 유지하며 신뢰성 있는 서비스를 제공한다.

향후 강건한 신장 트리(Robust Spanning Tree)에 대한 연구와 함께 다양한 환경에서 성능 평가하겠으며, 인증 및 위임 기술과 접목하여 에이전트 보안에 관련된 취약 부분을 개선해 나가도록 하겠다.

참고문헌

[1] M. Weiser, "The Computer for the Twenty-First Century," Scientific American, pp. 94-10, September 1991.

[2] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," Communications of the ACM, Vol. 36, No. 7, July 1993.

[3] J. Baumann, et. al., "Communication Concepts for Mobile Agent Systems," Lecture Notes in Computer

[4] J. Gomoluch and M. Schroeder. "Information Agents on the Move: A Survey on Load-Balancing with Mobile Agents," Software Focus, Vol. 2, No. 2, Wiley, 2001.

[5] P. K. Sinha, "Distributed Operating Systems concepts and Design," IEEE PRESS, USA.