2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

이벤트 통지 구조를 기반으로 한 이동 에이전트 통신 언어

서효정, 방대욱
계명대학교 컴퓨터공학과
e-mail : hjseo@jinri.kmu.ac.kr, dubang@kmucc.keimyung.ac.kr

An Mobile Agent Communication Language based on Event Notification Architecture

Hyo-Jeong Seo, Dae-Uk Bang
Dept. of Computer Science, Kei-Myung University

요 약

최근 들어 에이전트의 관심이 높아지면서 에이전트 시스템의 연구도 많아졌다. 특히 에이전트가 가지는 특성에 의해 여러 분야에서 에이전트의 이용도 높아졌다. 에이전트들은 특성에 의해 고정 에이전트, 이동 에이전트등 여러 가지로 분류된다. 하지만 이들 모든 에이전트 시스템이 갖추어야 할 요소 중 에이전트 통신은 아주 중요한 부분을 차지한다. 에이전트 통신을 위해서는 에이전트간 통신언어는 없어서는 안될 부분이다. 이제까지 KQML, FIPA ACL과 같이 여러 에이전트 통신 언어가 연구되었지만 이동 에이전트를 위한 통신 언어는 없었다. 이들 언어를 이동 에이전트 통신 언어로 사용하기에는 여러 가지 문제점이 있다. 본 논문은 이를 위해서 이동 에이전트에 적합한 이벤트 통지 통신 구조를 기반으로 한 이동 에이전트 통신 언어인 MACL(Mobile Agent Communication Language)를 제시한다.

1. 서론

최근 전자 상거래 등 여러 인터넷 응용 사업의 발달과 함께 에이전트에 관한 관심도 높아졌다. 여러 에이전트의 기술 중에서 에이전트 통신에 관한 기술은 아주 중요한 부분을 차지한다. 통신 기술에서 에이전트 통신 언어는 없어서는 안될 항목이다. 이런 에이전트 통신 언어는 다중 에이전트나 지적 에이전트에서 사용하는 KQML[2,3], FIPA ACL등이 있다. 이들 언어를 이동 에이전트에 적용 시키기에는 이동 에이전트만이 가지는 특성 때문에 많은 문제점이 있다. 또한 고정 에이전트의 통신 메커니즘 역시 이동 에이전트를 위해서는 부적합하다. 본 논문에서는 이동 에이전트를 위한 이벤트 기반의 통신 메커니즘을 기반으로 한 새로운 이동 에이전트 통신 언어인 MACL를 제시한다. 이벤트 통지 기반의 메커니즘에 의해 에이전트들은 자신이 원하는 이벤트들만 주고 받을 수 있어 자료의 필터링이 가능하다. MACL은 이벤트 통지 구조를 기반으로 FIPA ACL의 확장된 형태로써 XML[7] 형태로 표현 된다.

본 논문은 이벤트 통지 기반의 통신 구조와 XML을 이용한 에이전트 통신 언어인 MACL의 표현 그리고 기존의 통신 언어와 비교 분석 등을 기술한다.

2 이동 에이전트를 위한 이벤트 통지 서비스

이벤트 서비스는 이벤트를 보내는 생산자와 이들 이벤트를 받는 소비자가 있다. 일반 이벤트 채널에서는 어떤 이벤트가 생성되었을 때 이 이벤트에 아무런 명시 없이 보내게 되면 모든 소비자들은 관계도 없는 이벤트들까지도 받게 된다. 이런 경우 네트워크 부하가 높아지게 된다. 이러한 문제점을 해결하기 위해 이벤트에 통지라는 개념을 두었다. 이벤트 통지 서비스[1]란 이벤트를 보내거나 받을 쪽에서 자신이 원하는 이벤트의 내용을 명시하는 것이다. 생산자와 소비자는 자신이 원하는 이벤트만을 받게 된다. 이런 이벤트 통

지 서비스는 고정된 위치의 에이전트가 통신하는 방법인데 이동 에이전트에 적용시키기에는 많은 문제점이 있다. 에이전트의 이동, 신뢰성, 이동 후 재연결 등을 고려하여야 한다.

2.1 이벤트 통지 서비스의 사용자 인터페이스

이동 에이전트를 위한 이벤트 통지 서비스에서 에이전트 서버들의 위상을 acyclic peer-to-peer 형태로 가정한다.
이동 에이전트가 이벤트 통지 서비스를 하려면 [그림 2]와 같은 5가지 인터페이스 함수를 사용한다.

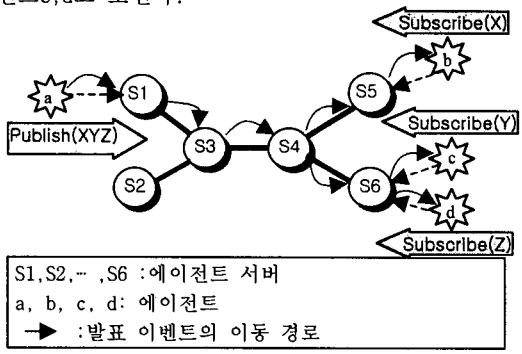
```
<인터페이스 함수>::= {<publish> | <subscribe> |
    <unsubscribe> | <advertise> |<unadvertised>}
<publish>::= publish (<통지>)
<subscribe>,<unsubscribe>
            ::=(un)*subscribe(<명령어>,<패턴>)
<advertise>,<unadvertise>
            ::=(un)*advertise(<명령어>,<필터>)
<명령어>::={create| move| arrive| join_group| meet}
<통지> ::= (속성, 값)
<필터> ::= (속성, 연산자, 값)
<패턴> ::= <필터> {조합 연산자 <필터>}*
```

[그림 2] 사용자 인터페이스

Publish 함수는 <통지>라는 인자만 가지고, 이를 제외한 모든 함수는 첫번째 인자로 <명령어>를 사용한다. Create명령어는 에이전트가 처음 예약이나 광고를 하여 경로를 설정 할 때 사용하고, 이동을 원할 때는 move명령어, 그룹 통신을 위하여 그룹에 소속할 때는 join_group명령어등을 사용한다. 함수의 두 번째 인자는 함수에 따라 달라진다. subscribe함수는 <패턴>이 advertise함수는 <필터>가 들어 간다. <통지>는 속성과 값의 쌍으로 이루어 진다. <필터>는 <통지>에 연산자가 들어가서 속성들의 범위를 표현하고, <패턴>은 <필터>들의 조합으로 이루어 진다

2.2 이벤트 라우팅

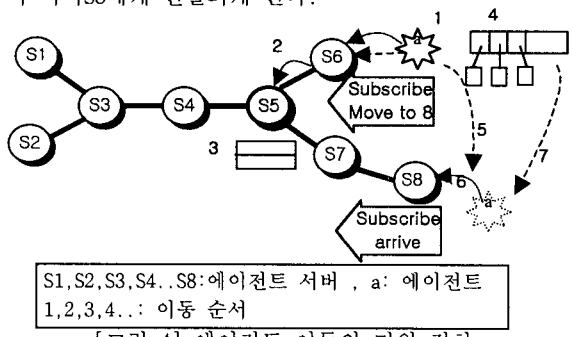
특정 이벤트 만을 받겠다는 예약(subscribe)과 특정 이벤트 만을 보낼 수 있다는 광고(advertise)에 의해 각 에이전트 서버에는 통지 라우팅 테이블이 생성된다. 이때 통지 라우팅 테이블은 예약과 광고한 내용을 기반으로 생성된다. 이벤트를 발표(publish)하면 이벤트는 각 에이전트 서버에 있는 이벤트 통지 테이블을 보고 경로를 설정한다.
예를 들어, [그림 3]은 에이전트 서버S1, S2, S3, S4, S5, S6가 acyclic peer-to-peer 형태로 구성되어 있고 여기에 에이전트a,b,c,d 4개가 에이전트 서버에 연결되어 있다. 에이전트b는' X' 을 에이전트c는 ' Y' 을 에이전트d는' Z' 을 예약하였다. 서버S1부터 서버S6에는 이벤트 통지 테이블이 생성되고 예약 정보가 저장된다. 에이전트a가 ' XYZ' 을 발표를 한다. 이 이벤트는 S1서버의 이벤트 통지 테이블을 정보를 보고 S3,S4로 전달된다. 서버S4의 테이블 정보를 보고 복사가 이루어져 각각은 서버S5,S6로 보낸다. 다시 서버S6의 테이블 정보를 보고 복사가 이루어져 에이전트c,d로 보낸다.



[그림 3] 발표의 이동 경로

2.3 에이전트 이동의 지원

에이전트의 이동은 통지 라우팅 테이블의 정보에 의한 이벤트의 역추적을 이용한다. 이동을 하려는 에이전트가 move명령을 보내면 이동전 에이전트 서버에 임시 기억장소가 생기고 이동 후 arrive 명령어를 보낼 때까지는 이벤트 통지 테이블은 그대로 유지된다. Arrive명령이 도착되면 이벤트 통지 테이블의 에이전트 위치를 갱신 시키고 이동중에 이동하는 에이전트에게 보내어지는 이벤트를 저장해둔 이동전의 서버에서 이벤트를 새로운 서버로 가져온다.
예를 들어, [그림 4]에는 서버S6 에 있던 에이전트a가 서버S8 로 이동하려고 한다. 에이전트a가 move명령을 예약 시키면 각 서버들은 통지 라우팅 테이블에 예약 이벤트의 정보를 넣을 것이다. 그때 서버S6 에는 이동중에 있는 에이전트a에게 오는 메시지를 저장하기 위한 메모리를 할당한다.
에이전트가 서버S8로 이동한 후 arrive명령을 예약한다. 그러면 서버S5에 있는 통지 라우팅 테이블이 갱신 된다. 서버S6에 임시로 저장시켜 둔 이벤트 들은 서버S6,S5,S7의 통지 라우팅 테이블을 보고 역추적하여 서버S8에게 전달되게 된다.



[그림 4] 에이전트 이동의 지원 절차

3. 이동 에이전트 통신 언어(MACL)

3.1 MACL의 구문

이동 에이전트 통신 언어는 기존의 에이전트 통신 언어에 에이전트의 이동성과 이 이동성을 위해 제안한 이벤트 통지 서비스를 기반으로 확장된 형태를 이루고 있다. 이벤트 통지 기반의 이동 에이전트 통신 언어이기 때문에 이벤트 통지 서비스에 맞는 행위자가 있어야 한다.

행위자를 나누어 보면, 이벤트 통지 기반의 발표 즉 정보 전달에 필요한 행위자인 confirm, disconfirm, inform, inform-if, inform-ref가 있고 에이전트에게 직접 영향을 미치는 행위자인 create, move, arrive, join_group, meet가 있다 그리고 이벤트 통지 기반의 예약, 광고를 위한 subscribe, propose가 있다.

[그림 5]은 메시지의 구조를 XML로 나타내고 있다. 메시지 종류는 앞에서 언급한 것이고 인자들은 메시지의 종류에 따라 다르다.

```
<?xml version="1.0" encoding="UTF8"?>
<!DOCTYPE MACL SYSTEM "macl.dtd".>
<MACL >
<MESSTYPE>    </MESSTYPE>
<MESSPARM>
<SENDER>      </SENDER>
<SEN_SER>     </SEN_SER>
<RECEIVER>    </RECEIVER>
<REC_SER>     </REC_SER>
<LANGUAGE>    </LANGUAGE>
<ONTOLOGY>    </ ONTOLOGY>
<CONTENT>     </CONTENT>
<IN_REPLY_TO>  </IN_REPLY_TO>
< REPLY_ WITH >    </ REPLY _WITH >
</MESSPARM>
</MACL>
```

[그림 5] 메시지의 구조

3.2 MACL의 DTD

DTD(Document Type Definition)는 XML문서 작성시 사용자가 자신만의 마크업 요소를 정의하고 문서의 구조를 설계 해두는 것이다.

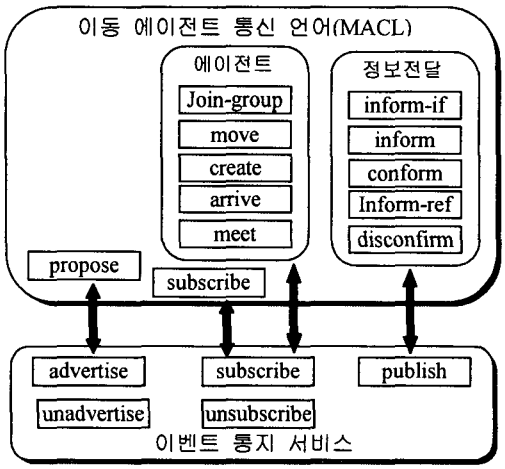
```
<!ELEMENT  XACL  (MESSTYPE, MESSPARM+) >

<!ELEMENT  MESSTYPE (subscribe | propose | confirm |
disconfirm |  inform | inform-if | inform-ref | create | move
| arrive | join_group | meet ) >

<!ELEMENT   MESSPARM  ( SENDER* | SEN_SER* |
RECEIVER* | REC_SER* | LANGUAGE* | ONTOLOGY* |
CONTENT* | IN_REPLY_TO* | WITH_REPLY*)  >

<!ELEMENT  SENDER  (#PCDATA, %sender;) >
<!ELEMENT  SEN_SER  (#PCDATA, %sen_ser;) >
<!ELEMENT  RECEIVER  (#PCDATA, %receiver;) >
<!ELEMENT  REC_SER  (#PCDATA, %rec_ser;) >
<!ELEMENT  LANGUAGE  (#PCDATA, %language;) >
<!ELEMENT  ONTOLOGY  (#PCDATA, %ontology;) >
<!ELEMENT  CONTENT  ((#PCDATA, #PCDATA)+) >
<!ELEMENT  IN_REPLY_TO  (#PCDATA, %in_reply_to;) >
<!ELEMENT  WITH_REPLY  (#PCDATA, %with_reply;) >
```

[그림 6] MACL의 DTD

XML문서 중 well-formed 파일인 경우 DTD가 필요 없다. 하지만 MACL의 구조는 변하는 것이 아니라 항상 고정적이어서 MACL의 DTD를 정의하면 MACL의 파싱이 용이하다. [그림 6]은 MACL의 DTD이다.

3.3. MACL과 이벤트 통지 서비스의 매핑

이동 에이전트 통신을 위해 하위에서는 이벤트 통지 서비스 구조와 상위에서는 이벤트 통지 언어인 MACL를 사용한다. 그러나 이 두쪽이 사용하는 메시지 타입이 정확이 일치하지는 않는다. 언어 레벨에서 propose은 통지 서비스의 advertise와 매핑이 되고, subscribe은 통지 서비스의 subscribe와 매핑이 된다. 정보를 전달하는 많 은 언어는 하위의 publish에 대응된다. 상위 레벨에서의 에이전트를 조작하기 위한 언어는 하위 레벨에서 subscribe내에 명령어로 들어 간다.



[그림 7] MACL과 이벤트 통지 서비스의 매핑

```
<?xml version="1.0" encoding="UTF8"?>
<!DOCTYPE XACL SYSTEM "xacl.dtd".>
<XACL >
<MESSTYPE>  propose  </MESSTYPE>
<MESSPARM>
<SENDER> :sender   b </SENDER>
<SEN_SER> :sen_ser   S_5 </SEN_SER>
<ONTOLOGY> :ontology   motor </ONTOLOGY>
< CONTENT> :content ((종류, 승용차),
                     (cc>2500 ),
                      (회사,'삼성','현대'))
                       </CONTENT>
</MESSPARM>
</XACL>
```

[그림 8] 광고 행위자

```
{ Advertise(-,P),
    (Sender,=,b),
    (sen_server,=, S_5),
    (ontology,=,motor),
    (종류,=,승용차),
    (cc, >,2500),
    (회사,=, '삼성','현대') }
```

[그림 9] 광고 이벤트 통지

[그림 7]은 이벤트 통지 서비스와 MACL의 행위자의 매핑을 나타낸다.

[그림 8]의 광고 행위자인 MACL은 [그림 9]의 광고

이벤트 통지로 매핑 된다.

4. FIPA ACL , KQML와의 비교 분석

많은 에이전트 통신 언어 중에서 KQML[4]과 ACL을 비교 대상으로 삼은 이유는 다음과 같다. 많은 에이전트 시스템들이 KQML과 FIPA의 ACL을 사용한다. 우선 FIPA ACL와 KQML은 기본적인 개념이나 구문상으로 많은 유사성을 가지고 있다[5]. 이것은 FIPA ACL의 중요한 특징이다. KQML구문에서 FIPA ACL의 구문으로 변화를 용이하기 위해 이런 특징을 가진다. MACL은 FIPA ACL을 이동 에이전트에 적용시키기 위하여 확장하였다. 그래서 FIPA ACL과 비슷한 구분을 가지지만 XML형태로 인코딩 되어 있는 특징이 있다. 이런 XML형태는 에이전트 개발에서 큰 오버헤드로 작용하던 ACL메시지 파싱을 용이하게 한다. 왜냐하면 DTD를 가지고 있어 파서를 개발하기 더욱 쉽기 때문이다.

KQML과 FIPA ACL은 문법과 온토로지에 독립적으로 정보 교환을 하는 프로토콜이다[5]. 즉 TCP/IP, SMTP, IIOP와 같은 전송 메커니즘에 독립적이고, KIF, SQL, Prolog와 같은 내용 언어에 독립적이고, 내용에 의해 추측되어지는 온토로지에 독립적이다. 하지만 이런 언어들은 에이전트의 이동성을 전혀 고려하지 않았다. MACL은 에이전트의 이동성을 고려하여 설계한 이벤트 통지 서비스를 기반으로 하여 설계하였다. 또한 MACL과 이벤트 통지 서비스의 매핑을 고려하므로 하였다.

KQML과 FIPA ACL의 구조에서는 중재자가 있어 등록과 중재자의 역할이 필수적이다. 이동 에이전트에서는 이런 점이 큰 제약이 된다[6]. MACL의 경우 통신에 있어 중재자가 필요 없는 이벤트 통지 구조를 설계하므로 통신 메커니즘 구조가 다르다.

일반적으로 에이전트가 앞으로는 WWW의 내용을 다룰 것이고 에이전트의 활동 영역으로 인터넷이 주류를 이룰 것이다. 이런 점에서 MACL은 XML로 기술되어 있어 더욱 WWW에 가깝다. 에이전트의 능력, 에이전트의 정보, 프로토콜등을 나타내는 부분은 XML을 사용하므로 더욱 설명하기 편하다.

5. 결론

기존의 KQML, FIPA ACL은 이동 에이전트에서 사용하기에는 문제가 있다. 본 논문은 이동 에이전트를 위한 새로운 통신 메커니즘인 이벤트 통지 서비스를 제시하고 그 위에서 사용될 수 있는 ACL의 확장된 언어인 MACL를 제시하였다. MACL은 XML로 표현함으로써 이동 에이전트 사이에 주고 받는 언어가 된다. 무엇보다 XML을 사용함으로써 에이전트 통신언어의 파싱이 쉬워졌고 통신언어 자체가 계층적 구조를 따르므로 XML로 표현하기 좋다. 또한 언어에 독립적으로 적용될 수 있고 에이전트 시스템이 에이전트의 능력, 에이전트 정보, 프로토콜등을 표현하기 좋다.

참고문헌

[1]A. Carzaniga, " Architectures for an Event Notification Service Scalable to Wide-area Networks" , PhD Thesis, Milano, 1998

[2] ARPA Knowledge Sharing Initiative. Specification of the KQML agent-communication language. ARPA Knowledge Sharing Initiative, External Interfaces Working Group, July 1993.

[3] Yannis Labrou. Semantics for an Agent Communication Language. PhD thesus, University of Maryland, Baltimore County, August 1996

[4] Yannis Labrou and Tim Finin. A proposal for a new kqml specification. Technical Report TR-CS-97-03, University of Maryland, Baltimore County, August 1997

[5] Yannis Labrou, Tim Finin, and Yun Peng. Agent Communication Language : the current landscape. IEEE Intelligent Systems, May 1999.

[6] Yannis Labrou, Tim Finin, and Yun Peng. The Interoperability Problem : Bringing together Mobile Agents and Agent Communication Languages. IEEE Intelligent Systems, May 1999.

[7] W3C. Extensible Markup Language(XML)1.0