상황인지 컴퓨팅을 위한 경량의 서비스 추론과 상황정보 관리를 위한 프레임워크의 설계

한지연, 이기혁, 한형진, 최원철, 한경훈, 김태현, 손기락
한국외국어대학교 컴퓨터공학과
e-mail : hanjiyeon@hufs.ac.kr

Design of Light-weight Service Reasoner & Context Information Base Framework for Context-aware Computing

Ji-Yeon Han, Ki-Hyuk Lee, Hyung-Jin Han, Won-Chul Choi, Kyung-Hoon Han, Tae-Hyun Kim, Kirack Sohn
Dept of Computer and Information Communication Engineering,
HanKuk University of Foreign Studies

요 약

상황인지(Context-awareness)란 사용자의 상황을 인지하여 이를 통해 사용자에게 적합한 최적의 정보를 제공하는 것으로 현재 많은 연구가 진행되고 있다. 이미 상황인지를 위한 프레임워크들이 존재하지만, 작은 메모리가 요구되는 제한적인 하드웨어 상황에서 적용하기에 적합하게 설계된 프레임워크는 흔하지 않다. 이에 대한 문제점을 인식하여 하드웨어 리소스가 부족한 상황에서 빠르게 동작할 수 있는 프레임워크를 구상하여, 기존과는 다른 제한적인 환경 안에서 상황인지를 통해 사용자에게 맞는 최적의 서비스를 판단, 제공하는 모바일 오브젝트를 설계하였다.

1. 서론

상황인지(Context-awareness)란 사용자의 상황을 인지하여 이를 바탕으로 사용자에게 적합한 최적의 정보를 제공하는 것이다. 이러한 상황인지 컴퓨팅 기술은 소형화, 개인화, 이동화가 요구되는 유비쿼터스 컴퓨팅 환경에 많이 접목되고 있다.

상황인지를 위한 기존의 많은 프레임워크들은 자원이 풍부한 클라이언트-서버 환경에 적합하게 설계되었기 때문에 작은 메모리가 요구되는 제한적인 하드웨어 환경에서 적용하기에는 무리가 있다. 예를 들어, 추론기능을 제공하는 JESS[1]나, 데이터 저장을 위해 많이 사용되는 JENA[2]와 그에 대한 질의언어인 SPARQL[3]을 사용하는 경우에 자원이 제한적인 환경에 적용하기는 힘들다. JENA 의 경우 Java 기반으로 운영되어 동작하는 데만 약 120MB 의 메모리를 소모하기 때문에 작은 메모리의 제한적인 하드웨어를 가진 디바이스에서 실행하기에는 무리가 있다.

본 논문에서는 이러한 문제점을 해결하기 위해 필요한 기능만 추출하여 단순화한 질의언어와 데이터 타입를 정의하였고, 추론을 위해서 Rete Matching 알고리즘[4]을 사용하였다. 이러한 방법으로 유비쿼터스 지능공간에서 자체 상황인지를 통해 사용자에게 맞는 서비스를 판단하여 제공해주는 모바일 오브젝트(U-MO: Ubiquitous Mobile Object) 를 설계하였다[5].

2. U-MO

U-MO 는 각종 리소스에 대한 정보를 저장하고, 그 정보를 관리하고 질의하는 역할을 수행한다. U-MO 디바이스는 시계 크기의 Watch 타입과 중간 크기의 Pocket 타입, 가장 큰 크기의 Hand 타입으로 나누어진다. 그 중에서도 본 논문의 타겟 디바이스인 Watch 타입은 사용할 수 있는 하드웨어 리소스가 적어서 작은 메모리에서도 빠르게 수행할 수 있도록 프레임워크가 설계되어야 한다.

(그림 1) UMO Watch 타입

2.1 U-MO Watch 타입의 하드웨어

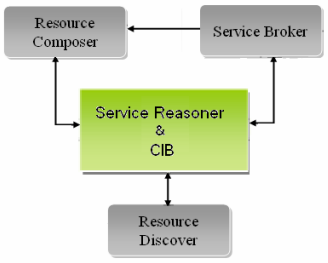
U-MO Watch 타입은 <표 1>과 같은 하드웨어 구성으로 이루어진다.

<표 1> U-MO Watch 타입의 하드웨어 구성

OS	Windows CE 6.0
프로세서	ARM 11
메인 메모리	64MB
저장장치	128MB Flash Memory
통신장치	ZigBee

U-MO Watch 타입 디바이스는 Windows CE 6.0 버전의 OS 에서 운영되며 ARM11 의 프로세서를 가진다. 메인 메모리는 64MB 로 제한적이며 저장장치 또한 128MB 의 Flash Memory 로 구성되어서 다른 종류의 U-MO 디바이스들에 비해 하드웨어 리소스가 적다.

2.2 U-MO 프레임워크의 구성



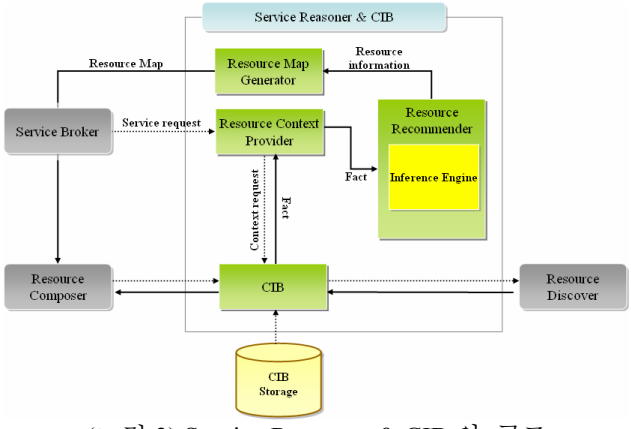
(그림 2) U-MO 프레임워크

U-MO 디바이스 안에 구축된 U-MO 프레임워크는 (그림 2)로 간략하게 나타낼 수 있다. U-MO 프레임워크는 사용자에게 필요한 서비스에 대한 요청을 받아서 Service Reasoner & CIB 에 전달된다. Service Reasoner & CIB 는 Resource Discover 를 통해 다른 U-MO 로부터 전달 받은 정보를 저장 및 관리하고 이 정보를 바탕으로 서비스를 추론한다. (Service Reasoner & CIB 의 CIB 는 정보의 저장, 수정을 담당하고, 데이터에 대한 추론은 Service Reasoner 에서 담당한다.) Service Reasoner & CIB 를 통해서 최적의 서비스 추론이 완료되면 Resource Composer 에게 알린다.

3. Service Reasoner & CIB 프레임워크

3 절에서는 U-MO 프레임워크에서 정보의 저장, 추론, 관리의 핵심적 역할을 수행하는 Service Reasoner & CIB 에 대해 설명한다.

3.1 Service Reasoner & CIB Overview



(그림 3) Service Reasoner & CIB 의 구조

Service Reasoner & CIB 는 데이터 처리와 추론 면에서 중추적인 역할을 수행한다. Service Reasoner & CIB 는 유비쿼터스 지능공간 안의 U-MO 에 대한 정보의 저장기능, 저장된 정보에 대해 관리 및 질의 기능과 저장된 정보를 바탕으로 최적의 서비스를 추천해주는 기능을 한다. 저장되는 정보는 Local U-MO 디바이스의 정보 및 주변 U-MO 디바이스에 대한 상세정보, 각 U-MO 디바이스들이 제공하는 공유 리소스들에 대한 속성 및 부가 기능에 대한 상세 정보이다.

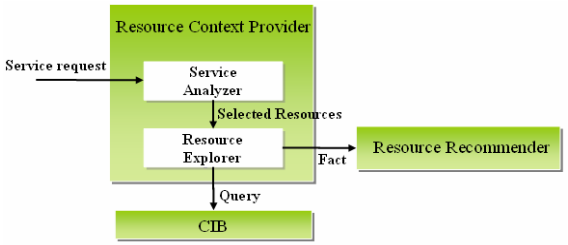
3.2 Service Reasoner & CIB

Service Reasoner & CIB 에는 Resource Map Generator, Resource Context Provider, Resource Recommender, CIB 로 이루어져 있는데, 각 모듈의 역할은 <표 2>와 같다.

<표 2> 각 모듈 별 역할

모듈	역할
Resource Context Provider	• 요청된 서비스 분석 • Context 검색을 위한 질의언어 생성 • Context 를 추론에 필요한 Fact 로 변환
Resource Recommender	• 서비스 추론 • 추론엔진에 Context 정보 추가 • 추론엔진에 Context 정보 삭제 • rule 의 추가
CIB	• Context 추가, 삭제 • Context 질의 처리
Resource Map Generator	• 결정된 Context 에 대한 서비스 구성 요청

◆ Resource Context Provider



(그림 4) Resource Context Provider 의 구조

Resource Context Provider 는 (그림 4)와 같이 CIB 에게 리소스에 대한 정보 조회를 요청하는 부분으로 필요한 상황정보를 판단하는 Service Analyzer 와 해당하는 상황정보를 찾기 위해 질의를 생성하는 Resource Explorer 로 이루어져 있다. Resource Context Provider 에서는 제한적인 하드웨어 환경에 적합하도록 정의된 질의언어를 사용한다. 이에 대해서는 4 절에서 설명한다.

◆ Resource Recommender

Resource Recommender 는 서비스에 필요한 최적의 리소스를 추론하여 Resource Map Generator 에게 알려주는 역할을 수행한다. 내부에는 Rete Matching 알고리즘을 사용하는 추론엔진(Inference Engine)이 내재되어 있어 빠른 추론을 가능하게 한다. Resource Recommender 는 내부 추론엔진에 사용할 데이터를 전달받아 전달 받은 데이터를 Fact 로서 추론엔진에 입력하고, 저장되어 있는 Rule 을 기반으로 최적의 추

론을 수행한다.

◆ CIB

(그림 5) CIB 의 구조

CIB 는 수집된 정보로부터 리소스 및 상황정보를 저장하고 이를 관리 하는 역할을 수행하는데 (그림 5)와 같이 Manipulation Processor, Query Processor 그리고 CIB Storage 로 구성된다. CIB 의 정보를 실제로 물리적으로 저장하는 역할을 수행하는 모듈을 CIB Storage 라고 한다. Manipulation Processor 는 CIB 의 정보에 대한 추가, 삭제의 요청을 관리하는 역할을 한다. 그리고 Query Processor 는 원하는 상황정보를 찾기 위해 CIB Storage 로 질의를 전달 후 결과를 반환하는 역할을 수행한다. CIB 에 저장되는 정보는, 자체적으로 데이터 타입을 정의하여 사용하였다. 이에 대한 설명은 4 절에서 설명하였다.

◆ Resource Map Generator

어떤 디바이스를 사용할 것인지 결정된 것을 Resource Composer 에게 전달해주며 해당 드라이버를 설치하게 한다.

4. 구현

4.1 상황 정보베이스(CIB)

아래 데이터 테이블은 자체적으로 지정한 데이터 타입의 형식과 그 예시를 나타낸다.

<표 3> 상황 정보 베이스의 예

Context Type	Context Value	Confidence	Entity Id	Source
output. display .monitor. brightness	900	1	Monitor_01	UMO1. LocalRes ource
…	…	…	…	…

하나의 Row 는 Local UMO 디바이스 정보 및 Local Resource 에 대한 정보를 나타낸다.

◆ Context Type

상황정보에 대한 온톨로지 형태의 타입을 나타낸다. <표 3>의 예에는 output 디바이스의 하위 카테고리 중에 monitor 의 brightness 속성에 대한 IS-A 관계를 나타내는 정보이다.

◆ Context Value

해당 속성의 값을 나타낸다. <표 3>의 예시에는 해당 monitor 의 brightness 가 900 이라는 것을 의미한다.

◆ Confidence

Confidence 는 Context Value 의 신뢰성의 정도를 나타낸다. 신뢰도 100% 일 때 Confidence 는 1 이다.

◆ Entity Id

디바이스의 ID 를 나타내는 것으로 각 디바이스의 ID 는 중복 없이 고유의 값을 가진다.

◆ Source

해당 디바이스의 출처를 나타낸다. U-MO 에 소속된 디바이스가 자기 U-MO 가 원래 가진 디바이스인지, 외부 U-MO 의 디바이스인지를 나타낸다.

즉, <표 3> 의 예로 입력된 정보는 UMO1 의 LocalResource 인 Monitor_01 의 밝기는 900 이고 이 값은 100% 정확하다.' 라는 의미가 된다.

4.2 Entity 데이터 타입

Entity 에 대한 정보를 나타내는 데이터 테이블이 존재하는데 그 형식은 아래와 같다.

<표 4> Entity Data 타입 의 예

Entity Id	In Use	Time Stamp	Entity Info
Monitor_04	Y	1	Local Resource

◆ Entity Id

각 디바이스가 가진 고유한 Id 를 의미한다.

◆ In Use

해당 Entity 를 사용하고 있는가를 의미하는 것으로, Y 이면 사용 중, N 이면 비사용 중을 나타낸다.

◆ Time Stamp

정보의 기록 시간을 의미한다.

◆ Entity Info

Entity 의 정보를 나타내는 것으로 LocalResource, VirtualResource, 또는 DiscoverdResource 로 나타날 수 있다.

4.3 질의언어 형식

지정된 데이터 타입으로 데이터가 메모리에 저장 되었을 때, 원하는 데이터를 얻어 내기 위해서는 이 데이터에 대한 질의언어가 필요하다. 질의언어의 형식은 아래와 같은 형식으로 작성된다.

```
subquery ::= RETURN CONTEXT
             FROM context_type
             WHERE search_condition
search_condition ::= property operator value

subquery ::= subquery UNION subquery
search_condition ::= search_condition {AND|OR} search_condition
property oprator value ::= > | < | <> | >= | <=
```

(그림 6) 질의언어 형식

질의언어는 RETURN CONTEXT, FROM, WHERE 로 이루어 지는데, RETURN CONTEXT 는 상황정보를 반환하라는 명령을 명시하는 예약어이다. FROM 절에는 찾고자 하는 상황정보의 타입을 입력한다. WHERE 절에서는 “작다”, “크다”, “다르다”, “작거나 같다”, “크거나 같다”의 5 가지로 구분 가능한 조건문을 통해서 조건을 입력하여 원하는 정보만 찾아 오는 것이 가능하다. 질의언어는 UNION 연산으로 여러 질의언어가 결합될 수 있고, WHERE 절의 조건문은 AND 나 OR 연산으로 여러 조건의 추가가 가능하다.

4.4 추론엔진

추론엔진에서는 Fact 와 Rule 을 이용하여 Rete matching 알고리즘을 기반으로 최적의 서비스를 추론한다. 추론엔진에 Fact 를 입력하면 저장된 Rule 을 바탕으로 추론이 이루어진다. Fact 는 “서술어 – 주어 – 목적어”의 Triple 형태를 사용한다.

<표 5> 추론엔진의 기능

기능	설명
define fact	Fact 의 추가
assert fact	실행 중 Fact 의 추가
retract fact	Fact 의 삭제
call	함수 호출
print	출력
halt	중지
reset	Fact 의 삭제
add production	Rule 추가

해당하는 Rule 을 만족 시에 지정된 기능을 수행하는데, 추론엔진은 이해 대해 define fact, assert fact, retract fact, call, print, halt, reset, add production 의 기능을 제공한다. define fact 는 새로운 Fact 를 생성하고, assert fact 는 새로운 Fact 를 추가하는 작업을 수행하지만 define fact 로 추가된 Fact 와는 다르게 reset 시에 소멸이 된다. retract fact 나 reset 으로 Fact 의 삭제가 가능하고, call 을 통해 호스트 언어의 함수를 호출할 수 있다. print 는 출력작업을 수행하고, halt 는 중지 작업을 수행한다. 그리고 add production 은 새로운 Rule 의 추가를 수행한다.

5. 결론

기존의 상황인지를 위한 프레임워크는 리소스가 풍부하고 클라이언트-서버 환경에 적합하게 설계되었기 때문에 작은 메모리의 제한적인 환경에 적용하기는 무리가 있었다.

본 논문에서는 이러한 문제점을 인지하고, 해결하기 위한 방법을 제시하였고 유비쿼터스 지능공간에서 자체적인 상황인지를 통해 사용자에게 맞는 서비스를 판단하여 제공해주는 U-MO 프레임워크에 대해 소개하였다. 그리고 U-MO 프레임워크 안에서 데이터의 저장, 삭제 및 추론의 역할을 수행하는 Service Resource & CIB 를 중점적으로 다루었다. U-MO 프레임워크는 사용자에게 받은 요청을 Service Reasoner & CIB 에게 전달하여 원하는 요청에 제공할 수 있는 최적의 서비스를 추론하여 구성하게 된다.

위에서 언급한 작은 메모리의 제한적인 하드웨어 환경에서 사용하기 위해 자체적으로 새로운 질의언어와 데이터 타입을 정의하여 사용한다. Service Reasoner & CIB 에서는 CIB 의 정보를 Triple 기반의 Fact 형태로 추론엔진에 입력하여 저장된 Rule 을 기반으로 사용자에게 추천할 서비스를 찾아내게 된다. 추론엔진에서는 Rete Matching 알고리즘을 사용하여 빠른 추론을 가능하게 한다.

본 논문에서 설명한 설계 방식을 이용한 프레임워크를 적용하여 개발이 이어진다면 작은 메모리의 제한적인 환경을 요구하는 유비쿼터스 컴퓨팅 환경에서의 상황인지가 가능할 것이다.

참고문헌

[1] http://www.jessrules.com/jess/index.shtml

[2] Dave Reynolds. Java 2 Inference support. Version 1.35, 2007/03/23, see Web page. http://jena.sourceforge.net/inference/.

[3] SPARQL Protocol for RDF, K. Clark, Editor, W3C Recommendation, 15 January 2008, http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/. Latest version available at http://www.w3.org/TR/rdf-sparql-protocol/ .

[4] Robert B. Doorenbos. Production Matching for Large Learning Systems. Technical Report CMU-CS-95-113. School of Computer Science, Carnegie Mellon University, 1995.

[5] 이기혁 외 3 인. 유비쿼터스 지능공간에서 상황인지 기능 지원을 위한 RIB 프레임워크. IT 서비스학회 춘계학술대회논문집. 2007 년 5 월.