CNRP 서버/클라이언트 시스템
(A CNRP Server/Client System)

유 영 호 † 이 종 환 † 이 중 화 †† 김 경 석 †††
(Young-Ho Yu) (Jong-Hwan Lee) (Jung-Hwa Lee) (Kyong-Sok Kim)

요 약 IETF는 보통이름을 인터넷 리소스의 URI로 풀이하기 위한 서버와 클라이언트간의 객체 교환 프로토콜인 CNRP를 제안하였다. CNRP는 하나의 인터페이스를 통해 다양한 서비스들에 접근할 수 있고, 원하는 리소스를 쉽게 얻도록 해주는 것은 물론, 결과로 얻은 리소스를 단순한 데이타가 아닌 정보로서 재사용 할 수 있도록 한다. 이러한 장점으로 인해 CNRP는 인터넷 서비스의 통합이나, 보통이름 풀이가 필요한 응용에서 널리 사용될 것으로 본다. 하지만, CNRP 명세에서는 구체적인 구현 방법에 대해서는 언급하지 않고 있다. CNRP를 이용한 몇몇 연구에서 프로토타입 시스템을 개발하여 구현 방안을 제시하고 있지만, CNRP 객체를 모두 구현하지 않거나 하나의 서버를 가지는 형태로 개발되어 다양한 응용에서 이용하기엔 부족하다. 따라서, 본 논문에서는 CNRP 서버/클라이언트를 구현할 때 발생하는 문제점들을 분석하여 그에 대한 해결 방안을 제시함으로써 구체적인 구현 방안을 제시하고, CNRP 명세에서 정의한 모든 객체를 구현한다. 또한, 본 논문에서 제시하는 방안으로 구현한 CNRP 서버와 클라이언트를 사용하여 회사이름 풀이 서비스의 프로토타입 시스템을 개발함으로써, 구현 방안의 타당성을 검증한다. 본 논문에서 구현된 CNRP 서버/클라이언트는 다양한 CNRP 응용 시스템 개발에 활용될 수 있다.

키워드 : CNRP, 보통이름, 보통이름 풀이, XML

Abstract IETF has proposed CNRP that is a protocol exchanging CNRP objects between server and client for resolving a common name to URIs of the desired Internet resouces. CNRP enables users to access various services via the integrated interface, to easily get the desired resources, and to reuse the results not as the data but as the information. Whereby these advantages, CNRP will be generally used for the integration of the various Internet services or the applications where the resolution of common name is needed. But, CNRP specification doesn't describes the practical implementation method for server and client. Though a few prototype systems are developed in some researches using CNRP, they are not enough to be generally used for the various Internet applications because they doesn't include all objects specified in CNRP specification or they construct systems with one server. So, this paper proposes the practical implementation method for CNRP server/client through analyzing and solving the problems occurred when implementing them, and implements all objects specified in CNRP specification. This paper also verifies the feasibility of the proposed method by developing the prototype system of the company name resolution service using the CNRP server/client implemented in this study. The CNRP server/client implemented in this paper are used to develop various CNRP application systems.

Key words : CNRP, common name, common name resolution, XML

† 비 회 원 : 부산대학교 전자계산학과
yhyou@asadal.pnu.edu
jhwlee@asadal.pnu.edu
†† 정 회 원 : 동의대학교 컴퓨터-영상공학부 교수
yijh@dongeui.ac.kr
††† 종신회원 : 부산대학교 전자전기정보컴퓨터공학부 교수
gimgs@asadal.pnu.edu
논문접수 : 2001년 8월 20일
심사완료 : 2002년 6월 12일

1. 서 론

WWW(World Wide Web)의 등장으로 인터넷이 대중화되면서 인터넷은 다양한 정보들을 얻을 수 있는 원천으로 인식되고 있다. 인터넷 리소스(resource)는 URL (Uniform Resource Locator)을 사용하여 접근할 수 있으며, 단체나 개인은 사용자들이 기억하기 쉽고, 친숙한 도메인 이름(domain name)을 만들어 자신들의 리소스

를 잘 접근할 수 있도록 하기 위해 노력한다[1,2]. 그러나, 현재 도메인 이름 공간(domain name space)은 이미 그 한계를 드러내고 있으며, 또한 다국어 지원이 되지 않아 영어권이외의 국가에서는 이름 자체가 친숙하지 않은 경우가 일반적이다[2]. 따라서, 일반 인터넷 사용자들은 복잡한 URL을 기억해서 입력창에 직접 타이핑하거나, 검색 서비스를 이용해서 나온 결과를 통해 자신이 원하는 리소스에 접근한다[3,4]. 물론 포탈(portal) 서비스의 한 부분으로 검색 엔진이 사용자들에게 정보 검색 서비스의 역할을 해주고 있긴 하지만, 키워드에 의해 검색된 데이타는 양이 너무 많을 뿐만 아니라 사용자가 원하는 정보가 아닐 경우도 많다[4,5]. 또한 메타 검색 서비스가 있어서 여러 검색 엔진의 결과를 하나의 인터페이스로 볼 수 있지만[3], 하나의 질의에 대한 여러 검색 엔진의 결과를 하나의 결과 집합으로 통합하는 것은 상당히 어려운 작업일 뿐만 아니라, 검색결과는 사용자에게 보여주기 위한 단순한 검색결과일 뿐이기 때문에 검색된 데이타를 재사용하기가 어렵다[1]. 그러므로 하나의 인터페이스를 통해 다양한 서비스들에 접근할 수 있고, 손쉽고 정확하게 원하는 리소스를 얻도록 해주는 것은 물론, 결과로 얻은 리소스를 단순한 데이타가 아닌 정보로서 재사용 할 수 있는 서비스에 대한 필요성이 점차 커지고 있다[1,2,3].

이러한 필요에 따라 최근 IETF(Internet Engineering and Task Force)에서는 보통이름(commnon name)을 인터넷 리소스의 URI(Uniform Resource Identifier)로 풀이(resolution)하기 위한 서버와 클라이언트간의 객체 교환 프로토콜인 CNRP(Common Name Resolution Protocol)를 제안하였다[2]. CNRP는 사용자들로 하여금 회사 이름, 브랜드 이름, 제품 이름, 또는 책제목과 같은 보통이름을 사용해서 원하는 인터넷 리소스의 URI를 얻을 수 있도록 해준다[1,2,3].

하지만 CNRP 명세(Specification)에서는 서버와 클라이언트간의 객체 교환 방식만을 제공하고 있고, CNRP 시스템의 구현에 관한 구체적인 방안을 제시하지는 않고 있다[7,8]. 단지 클라이언트를 구현하기 위한 방법으로 클라이언트의 두 가지 형태만을 추상적으로 제시하고 있을 뿐이다. 따라서, 본 논문에서는 CNRP 명세에서 정의하는 모든 객체를 지원하도록 서버와 클라이언트를 설계하고, 구현할 때 발생할 수 있는 문제점들을 분석하고 그에 대한 해결 방안을 제시함으로써 구체적인 구현 방안을 제시한다. CNRP 서버/클라이언트 시스템은 다양한 응용 서비스에 이용될 수 있다. 본 논문에서는 회사이름의 풀이를 통해 회사의 URI를 반환하는 서비스 개발에 서버/클라이언트 시스템을 적용해 봄으로써 제시하는 구현 방안의 타당성을 검증한다.

2. 관련연구

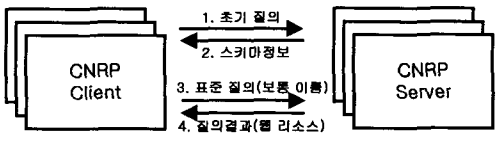
2.1 CNRP

CNRP는 2000년 IETF에서 제안된 것으로 보통이름의 풀이를 위한 서버와 클라이언트간의 객체 교환 프로토콜이다. CNRP 객체들은 XML DTD로 정의되어 있으며, 서버와 클라이언트는 데이타를 CNRP 명세에서 명시한 XML 태그로 캡슐화(encapsulation)한 질의(Query)와 결과(Results) 객체를 서로 주고받음으로서 통신하게 된다. 이러한 질의와 결과 객체 내에는 해당 리소스에 대한 힌트(hint)로서 프로퍼티를 포함할 수 있어 보통이름에 대한 리소스의 풀이를 더욱 정밀하게 할 수 있다[1,2].

2.1.1 CNRP 인터랙션(Interation) 모델

서버와 클라이언트는 간단한 질의/응답 방식을 사용하는데, 이때 클라이언트가 보내는 질의의 종류에는 초기 질의(initial query)와 표준 질의(standard query)가 있다[2,7,8]. 초기 질의는 서버가 제공하는 서비스에 대한 정보를 얻기 위한 질의이며, 그 질의에 대한 응답으로 받은 결과 객체는 서버가 제공하는 서비스에 대한 서비스 스키마 정보이다. 표준 질의는 서버로부터 받은 서비스 스키마 정보를 참조하여 풀이를 원하는 보통이름과 힌트로 사용되는 프로퍼티를 서버로 보내는 질의이다. 이러한 표준 질의에 대한 응답으로 온 결과 객체 내에는 리소스를 기술하는 리소스 디스크립터(resource descriptor) 객체, 리퍼럴(referral) 객체, 또는 상태 메시지(status message)를 나타내는 상태(status) 객체가 포함될 수 있다. 그림 1은 서버와 클라이언트의 인터랙션을 나타내고 있다.

CNRP의 모든 인터랙션은 XML 객체 교환을 통해 이루어지며, 전송 프로토콜에 대해 독립적이지만 클라이언트가 초기 질의를 서버에 보내기 위해서는 서버와 클



1. 초기 질의 : 해당 CNRP 서버에 대한 스키마 정보요구
2. 스키마 정보 : 서비스 가능한 질의 스키마 정보로 응답
3. 표준 질의 : 스키마 정보에 맞도록 질의
4. 질의결과 : 표준 질의에 대한 결과

그림 1 CNRP 인터랙션 모델

라이언트간에 표준화된 방식을 지원해야 하므로, 클라이언트는 HTTP의 POST 방식으로 기본 CNRP 포트인 1096 포트를 통해서 서버에 초기 질의를 보내야 한다[2,7,8].

2.1.2 CNRP 객체

CNRP에서 사용되고 있는 모든 객체는 CNRP 객체에 대한 XML DTD에 정의되어 있다. CNRP에서 최상위에 있는 기본적인 객체로 초기 질의 객체 <servicequery>, 표준 질의 객체 <query>, 그리고 이러한 질의들에 대한 응답 객체인 <results>가 있다. EMPTY 객체인 <serviceqeury>를 제외한 나머지 객체들은 자신을 기술하는 여러 내부 객체들을 포함하고 있다. 주요 CNRP 객체에 대한 XML DTD는 그림 2와 같다.

```
<! ELEMENT cnrp (servicequery | query | results)>
<! ELEMENT serviccquery EMPTY>
<! ELEMENT query (id | (commoname, property*)
<! ELEMENT results (status? |
 (service+,(status | resourcedescriptor | referral)*)*)>
```

그림 2 CNRP 객체의 DTD

CNRP에서는 프로퍼티의 리스트를 통해 각 객체를 기술하며, 이때 사용되는 프로퍼티의 종류에는 핵심(core) 프로퍼티, 추상(abstract) 프로퍼티, 맞춤(custom) 프로퍼티, 바탕(base) 프로퍼티가 있다[2,7,8]. 프로퍼티는 보통이름의 풀이에 사용되는 힌트로, 풀이의 정밀도를 높이기 위해 질의에 포함되는 파라미터로 볼 수 있다.

핵심 프로퍼티는 CNRP 서비스에서 제공해야할 최소의 프로퍼티로써, CNRP 서비스들간의 상호 호환성을 위해 반드시 지원되어야 한다. 따라서, CNRP 서비스를 제공하는 서버의 서비스 스키마는 핵심 프로퍼티를 모두 포함하여야 하며, 이를 핵심 스키마라고 한다[2,7,8]. 각 서버의 서비스 스키마는 핵심 스키마를 확장한 형태가 된다. 핵심 프로퍼티의 종류는 표 1과 같다.

추상 프로퍼티는 CNRP의 확장을 위해 정의된 중요한 개념의 프로퍼티로 핵심 스키마의 확장에 사용된다. 맞춤 프로퍼티는 확장을 위해 사용된 로컬의 추상 프로퍼티의 이름(name)과 타입(type)이 IANA(Internet Assigned Numbers Authority)에 등록되었을 때 맞춤 프로퍼티라고 한다.

표 1 핵심 프로퍼티

프로퍼티 이름	프로퍼티 설명
common name	리소스와 관련된 보통이름
ID	서비스로부터 온 결과에 대한 식별자
resourceURI	리소스의 absolute URI
Description	리소스에 대해 기술하고 있는 텍스트

바탕 프로퍼티는 구현상에 있어 의무적인 것은 아니지만, 대부분의 CNRP 서비스에서 지원하기를 권고하는 맞춤 프로퍼티의 부분집합으로 Language, Geography, Category, Range 등이 있다.

하나의 프로퍼티가 만들어지기 위해서는 반드시 이름과 하나 이상의 타입이 정의되어야만 한다. 타입은 영역(domain)이 제한되는 형태와 그렇지 않은 freeform 형태가 있다. 바탕 프로퍼티인 Geography의 타입을 예로 들면, freeform 타입과 영역이 제한되는 타입으로 ISO3166-1, ISO3166-2, lat-long 등을 사용할 수 있다[2,7,8].

2.2 CNRP 관련 연구

현재 CRNP는 IETF에서 표준화 작업 중인 I-D (Internet-Draft) 상태로 draft-ietf-cnrp-10까지 나와 있으며, 여러 기관 및 기업체에서 CNRP를 활용하기 위한 활발한 연구가 진행중에 있다[7,8].

Network Solution은 CNRP 명세에 따라서 두 가지의 프로토타입을 구현하였는데, 그 첫 번째가 NSI CNRP Pilot이라는 프로토타입 서비스로 회사 이름을 입력하면 해당 회사의 홈페이지 URL을 돌려준다. 회사 이름외에 산업분류, 지역, 언어를 추가 입력하면 결과를 제한할 수 있다. 두 번째로 CNRP 서비스 접근 방법 중 하나인 'go' URI를 지원하는 플러그인을 만들어 웹브라우저가 'go' URI를 지원하도록 함으로써 CNRP 서비스를 이용하도록 하였다[9]. Catalogix는 GIDS (Global Indexed Directory Service)라는 프로토타입을 구현하였는데, 이 서비스는 CNRP를 사람의 이름에 응용한 것으로 사람의 이름을 입력하면 그 사람과 관련된 정보가 있는 링크를 결과로 돌려준다[10]. [9]와 [10]에서는 더 많은 결과를 제공하는 다른 서버에 대한 리퍼럴 객체나 처리중의 오류 등을 위한 상태 객체 등을 제대로 구현하지 않았다.

RealNames의 경우 인터넷 키워드란 이름으로 키워드를 해당 URL로 풀이하는 보통이름 풀이와 유사한 서비스를 제공하고 있는데, 검색창에 기업체의 브랜드명 또는 상품명 등을 입력하게 되면 관련 사이트로 바로 갈 수 있도록 하고 있다[11].

최근 발표된 IETF의 I-D(Internet-Draft)에서 제안된 SLS(Service Lookup System)에서도 CNRP를 이용하고 있다. SLS는 사람에게 친숙한 국제화된 (internationalized) 인터넷 식별자를 제공할 목적으로

만들어진 시스템이다. SLS는 클라이언트가 원하는 인터넷 리소스의 식별자를 얻을 수 있는 서비스의 3개의 계층(layer)을 정의하고 있다. 계층 1은 현재의 DNS를 그대로 사용하며, 계층 2는 한정된 식별자만으로 원하는 인터넷 리소스의 식별자를 얻을 수 있게 하는 서비스를 제공하며, 계층 3은 검색 엔진 형태의 서비스를 제공한다. SLS의 계층 2를 위해 CNRP를 사용하고 있다[12].

3. 서버/클라이언트 설계 및 구현

본 논문에서는 CNRP 명세에서 정의한 모든 객체들을 지원하고 서버와 클라이언트를 구성하는 각 모듈을 설계 및 구현하고, 각 모듈들의 인터랙션을 위해 필요한 객체를 CNRP와 일관성을 가지도록 XML을 사용하여 새로이 정의한다. 뿐만 아니라, CNRP 서버/클라이언트를 실제 구현시 발생할 수 있는 문제점들을 분석하고 이에 대한 처리 방법을 제시함으로써 구체적인 구현 방안을 제시한다. 본 논문에서 구현하는 CNRP 서버/클라이언트는 앞서 언급한 두 가지 프로토타입에서 사용된 CNRP 서버/클라이언트에서 제공하지 않는 객체를 포함하여 CNRP 명세에서 정의된 모든 객체들을 구현함으로써 다양한 응용에서 활용할 수 있도록 한다.

3.1 CNRP 시스템 구성

CNRP 명세에서 정의한 객체들과 그 객체들을 통한 인터랙션을 기반으로 보통이름 풀이 서비스 시스템을 구성하기 위해서는 클라이언트, 서버, 데이타베이스, 사용자 환경이 필요하다. 본 논문에서 구현하는 CNRP 시스템의 구성요소와 인터랙션은 그림 3과 같다. 그림에서 보는 바와 같이 서버와 클라이언트의 인터랙션은 CNRP의 정의를 그대로 따르고 있다.

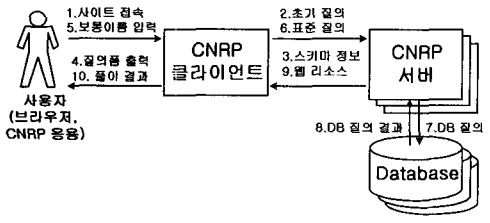


그림 3 CNRP 시스템 구성도

사용자가 브라우저를 이용해서 클라이언트에 접속하게 되면 클라이언트는 초기 질의인 <servicequery>객체를 HTTP의 POST 방식으로 서버들로 보낸다. 서버들은 자신이 제공하는 서비스의 스키마를 <results> 객체에 담아 클라이언트에게 응답한다. 클라이언트는 서버들로부터 받은 서비스 스키마를 통합하여 사용자 질의 인터페이스를 동적으로 생성한다. 이렇게 생성된 인터페이스에서 사용자가 입력한 보통이름과 보통이름 풀이에 힌트로 사용되는 프로퍼티들은 클라이언트로 전달된다. 사용자로부터 전송된 보통이름과 프로퍼티를 통해 클라이언트는 각 서버들에 맞게 질의를 생성해서 보낸 후, 각 서버들로부터 온 결과를 통합해서 사용자가 볼 수 있도록 브라우저로 전송한다.

3.2 서버 설계

3.2.1 서버의 기능

서버의 가장 중요한 역할은 사용자가 질의한 보통이름을 풀이하여 질의한 보통이름에 적합한 리소스를 반환하는 것이다. 이를 위해 서버는 다음과 같은 기능을 제공해야 한다.

▶ 클라이언트와의 통신: CNRP 명세에서는 클라이언트의 초기 질의는 항상 1096 포트를 통해 HTTP 방식으로 전송하도록 하고 있으며, 표준 질의는 초기 질의 결과에서 정의한 프로토콜에 따라 전송하도록 하고 있다. CNRP 명세에서 표준 질의 전송은 어떤 통신 프로토콜로도 가능하도록 규정하고 있지만 HTTP와 SMTP의 두 가지 프로토콜만을 소개하고 있다. 본 논문에서는 HTTP를 사용해서 표준질의를 전송하도록 설계한다.

▶ XML 문서의 처리: CNRP는 XML에 기반한 프로토콜로서 모든 인터랙션은 XML 문서를 교환하는 것으로 이루어진다. 따라서 서버는 클라이언트와의 인터랙션을 위해 XML 문서를 처리할 수 있어야 한다.

▶ 보통이름 풀이: 서버는 사용자가 입력한 보통이름과 힌트들을 이용하여 보통이름을 제공하는 서비스의 목적에 적합한 인터넷 리소스로 풀이할 수 있어야 한다.

3.2.2 서버 구성도

3.2.1에서 살펴 본 서버의 기능을 제대로 수행하기 위해서 필요한 모듈들을 독립적 설계 및 구현이 가능하도록 하고, CNRP 인터랙션과 일관성을 유지하기 위해서 각 모듈간의 통신은 XML 객체 교환 방식을 사용하도록 설계한다. 그림 4는 본 논문에서 설계한 서버의 구성도이다.

▶ 파서/결과처리기: 파서/결과처리기는 클라이언트가 직접 접속하는 모듈로 클라이언트로부터 전송된 XML 형태의 질의를 파싱하고 초기 질의이면 초기질의 처리기를, 표준 질의이면 표준질의 처리기를 호출하고, 초기질의 처리기 또는 표준 질의 처리기로부터 받은 결과를 클라이언트로 전송하는 기능을 한다.

▶ 초기질의 처리기: 초기질의 처리기는 파서/결과처리기로부터 호출되었을 때 CSSD(CNRP Service

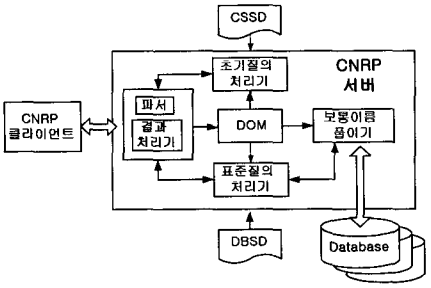


그림 4 서버 구성도

Schema Document)를 이용하여 초기질의 결과를 XML 문서로 만들어서 파서/결과처리기로 전송하는 기능을 한다. CSSD는 서비스 스키마 정보가 들어있는 XML 파일이다. CSSD를 유지함으로써 서비스 스키마의 변경이 있을 때, 서버 프로그램을 수정하지 않고 CSSD만 변경하여 처리할 수 있도록 한다.

▶ 표준질의 처리기: 표준질의 처리기는 파서/결과처리기로부터 호출되었을 때 DBSD(DataBase Schema Document)를 이용하여 질의를 분할하고 순차적으로 보통이름 풀이기로 전달하여 처리한다. 또한, 부분 질의의 결과들을 하나의 결과로 통합하여 파서/결과처리기로 전송한다. DBSD는 프로퍼티와 데이타베이스의 애트리뷰트의 연관성을 기술해 놓은 XML 파일이다. DBSD를 유지함으로써 데이타베이스 스키마의 변경이 있을 때, 서버 프로그램을 수정하지 않고 DBSD만 변경하여 처리할 수 있도록 한다.

▶ DOM: 서버는 클라이언트의 질의를 처리하는 과정에서 많은 XML 문서를 다루게 된다. DOM 모듈은 클라이언트에서 받은 질의 등의 XML 문서 조작이 필요할 때마다 호출된다. 설계한 DOM 모듈은 XML 문서 파싱, 생성, 관리할 때 이용되며, 또한 조작중인 DOM을 XML 문서로 출력하는 기능을 가진다.

▶ 보통이름 풀이기: 보통이름 풀이기는 데이타베이스에 직접 접근하여 보통이름을 적합한 리소스로 풀이해주는 기능을 한다. 보통이름 풀이 방식이 바뀔 경우 풀이 논리만 바꾸어 수정하면 되도록 독립된 모듈로 설계한다.

3.3 서버 구현시 문제점 분석 및 해결 방안

본 절에서는 CNRP 서버의 기능을 구현할 때 발생할 수 있는 문제점들을 알아보고, 이를 해결하기 위해 본 논문에서 제안하는 처리방법을 살펴본다.

3.3.1 표준질의에 포함된 같은 프로퍼티 처리

CNRP 명세에서는 표준 질의는 보통이름과 함께 여러 프로퍼티들을 포함할 수 있고, 보통이름을 풀이할 때 표준질의에 포함된 서로 다른 프로퍼티를 모두 만족하는 리소스로 풀이를 하도록 정의하고 있다. 즉, 서로 다른 프로퍼티들 간에는 AND 논리로 처리한다는 의미이다. 또한 표준 질의에는 같은 이름을 가진 프로퍼티를 포함할 수 있고, 같은 이름을 가진 프로퍼티들은 OR 논리의 의미로 처리하도록 정의하고 있다. 같은 이름의 프로퍼티들에 대해 먼저 기술된 프로퍼티를 만족하는 것이 앞쪽에 오도록 우선순위를 두고 있다.

표준 질의를 한번의 풀이로 처리할 경우 이러한 의미적인 연산과 우선순위를 지키기는 거의 불가능하므로, 이러한 내부연산과 우선순위를 보장하기 위해 본 논문에서는 표준 질의를 서로 다른 프로퍼티로만 구성되는 여러 개의 부분 질의로 재구성하여 순차적으로 풀이한 후 생성된 결과 XML 문서들을 단순한 병합(merge)을 이용해서 부분 질의 결과를 통합하는 방법을 사용한다.

그림 5의 경우 보통이름이 삼성인 리소스들 중에서 "city" 프로퍼티와 "province" 프로퍼티의 값이 각각 "수원"과 "경기도"를 만족하는 리소스가 질의 결과에서 가장 먼저 나타나야 한다. 두 번째는 "수원", "경상남도", 세 번째는 "마산", "경기도", 마지막으로 "마산", "경상남도"를 만족하는 리소스의 순서로 결과가 구성되

```
<query>
<common name>삼성</common name>
<property name="city" type="freefrom"> 수원 </property>
<property name="city" type="freefrom"> 마산 </property>
<property name="province" type="freeform"> 경기도 </property>
<property name="province" type="freeform"> 경상남도 </property>
</query>
```

그림 5 같은 이름의 property가 포함된 표준질의의 보기

```
재구성된 질의 1
<query>
<common name>삼성</common name>
<property name="city" type="freefrom"> 수원 </property>
<property name="province" type="freeform"> 경기도 </property>
</query>

재구성된 질의 2
<query>
<common name>삼성</common name>
<property name="city" type="freefrom"> 수원 </property>
<property name="province" type="freeform"> 경상남도 </property>
</query>

재구성된 질의 3
<query>
<common name>삼성</common name>
<property name="city" type="freefrom"> 마산 </property>
<property name="province" type="freeform"> 경기도 </property>
</query>

재구성된 질의 4
<query>
<common name>삼성</common name>
<property name="city" type="freefrom"> 마산 </property>
<property name="province" type="freeform"> 경상남도 </property>
</query>
```

그림 6 재구성된 표준질의의 보기

어야 한다. 이를 위해 그림 5의 표준 질의를 본 논문에서 사용한 방법으로 재구성한 네 개의 부분 질의는 그림 6과 같다.

3.3.2 상태 메시지 처리

CNRP 명세에서는 풀이 과정에서 오류가 발생하거나, 클라이언트에게 추가적인 정보를 전달하고자 할 때 상태 메시지를 이용하도록 한다. 이를 위해 CNRP에서는 상태 메시지를 4가지 레벨로 나누어 정의하고 있다. 하지만 구체적으로 상태 메시지에 대한 구현방안을 언급하고 있지 않기 때문에 본 논문에서는 표 2와 같이 표준절의 처리기에서 상태메시지를 발생시키는 경우를 정의하고 구현한다.

표 2 주요 상태 메시지

코드 레벨	상태 코드	상태 코드 설명
Informative	1.0.0	서버가 보통이름 풀이에 관계없이 클라이언트로 보낼 정보가 있음을 알림
Success	2.1.0	질의에 대한 풀이는 성공했지만 풀이 결과가 하나도 없음
Partial Success	3.1.1	질의에 서비스에서 제공하지 않는 프로퍼티가 포함되어 있음
	3.1.3	서버가 데이타셋을 지원하지 않음
Transient Failure	4.1.0	질의 파싱중에 에러가 발생
	4.1.3	서버의 설정 파일 파싱중에 에러가 발생

3.3.3 보통이름 풀이기

보통이름 풀이기는 제공하는 서비스에 따라 다양한 형태로 구현될 수 있다. 본 논문에서는 보통이름을 풀이하기 위해 단순 문자열 일치(string match)를 사용하며, 풀이에 필요한 데이타가 데이타베이스에 저장되어 있는 것을 가정하고 있다. 보통이름 풀이기는 풀이에 필요한 서비스 스키마 정보, 데이타베이스 스키마 정보들을 XML 문서 형태로 넘겨받아 XML 문서를 조작하여 데이타베이스에 대한 질의를 구성하도록 한다.

3.3.4 데이타셋(dataset) 객체

서버는 하나의 서비스에 대한 여러 개의 데이타셋을 가질 수 있다. 데이타셋은 다른 서버 데이타의 미러링(mirroring)일 수 있으며, CNRP 서비스를 제공하는 다른 서버의 데이타를 직접 나타낼 수도 있다. 데이타셋 개념은 풀이와 풀이에 사용될 데이타를 분리해서 고려할 수 있도록 한다. 즉, 다른 서버에 있는 데이타를 이용해서 풀이를 수행할 수 있는 메카니즘을 제공함으로써 서버들의 데이타 공유를 가능하게 한다. 이를 위해 데이타셋은 유일(globally unique)하게 명시할 수 있어야 한다. 현재 CNRP 명세에서는 URN을 사용하여 데이타셋을 유일하게 명시하는 예를 보여주고 있다. URN은 인터넷 리소스를 불변의 의미를 가지는 이름으로 접근 할 수 있게 해주는 방식이다. URN으로는 직접 해당 리소스를 접근할 수 없으므로, URN을 URL로 풀이하는 과정이 필요하다. 그러나 현재 이용 가능한 URN 풀이 시스템이 없기 때문에 본 논문에서는 URN 풀이를 시뮬레이션하는 모듈을 구현하여 이를 대체한다.

3.3.5 CSSD와 DBSD

서버에서 제공하는 서비스 스키마는 데이타베이스에 저장된 스키마와 다를 수 있고, 접속하는 클라이언트에 따라 제공하는 서비스의 제한이 있을 수 있다. 또한, 하나의 서버에서 여러 가지 서비스를 제공할 수 있으며, 서비스 스키마의 구조와 데이타베이스의 구조가 다를 수 있다. 따라서, 서버는 클라이언트에 제공될 서비스 스키마 정보를 표현하고 서비스 스키마와 데이타베이스 스키마의 관계를 표현할 수 있는 방법이 필요하다. 본 논문에서는 두 개의 XML DTD, 즉, 서버의 서비스 스키마 정보를 표현하는 객체를 포함하는 CSSD DTD와 데이타베이스 스키마 정보를 표현하는 객체를 포함하는 DBSD DTD를 정의하여 사용한다. CSSD DTD는 CNRP DTD에 포함된 <service> 객체 부분을 그대로 사용하여 정의하고, DBSD는 본 논문에서 새로이 정의한다. 각각의 DTD에 따라 표현된 XML 문서가 CSSD와 DBSD이다. 초기 질의 처리기는 CSSD에 표현된 정보를 이용하여 클라이언트에 맞는 서비스 스키마를 결과로 제공한다. 보통이름 풀이기에서는 CSSD와 DBSD를 이용하여 자동으로 표준질의를 SQL 질의로 변환하도록 한다. 이렇게 서비스 스키마와 데이타베이스 스키마를 표현하는 XML DTD를 정의해서 사용함으로써 다른 서비스에 쉽게 적용할 수 있고, 서비스 스키마나 데이타베이스 스키마가 바뀌더라도 CSSD 또는 DBSD의 내용만을 바꾸면 된다.

▶ CSSD

서버가 제공하는 서비스 정보들을 저장하고 있는 XML 문서이다. 서버에서는 이 문서를 통해서 필요한 서비스 정보들을 유지한다. CSSD에 저장되는 정보들은 다음과 같고, CSSD DTD는 그림 7과 같다.

- 서비스 URI, 데이타셋, 서버 정보
- 프로퍼티 정보(이름, 스키마 등)

그림 8은 그림 7의 DTD를 바탕으로 구성된 CSSD의 보기로서 부분만 표시되어 있다.

▶ DBSD

DBSD에는 서버에서 유지하는 데이타베이스의 스키마 정보가 저장된다. DBSD에 저장되는 데이타베이스 정보들은 다음과 같고, DBSD DTD는 그림 9와 같다.

```
<!ELEMENT service (serviceuri,dataset*,servers?, description?,
property*,propertyschema?,queryschema?,resourcedescriptionsc
hema?,
serviceschema?)>
<!ATTLIST service ttl CDATA '0'>
<!ELEMENT dataset (property*)>
<!ATTLIST dataset id ID #IMPLIED>
<!ELEMENT serviceuri (#PCDATA)>
<!ATTLIST serviceuri id ID #IMPLIED>
<!ELEMENT servers (server+)>
<!ELEMENT server (serviceuri,property*)>
<!ELEMENT property (#PCDATA)>
<!ATTLIST property name CDATA #REQUIRED type CDATA
'freeform'>
<!ELEMENT description (#PCDATA)>
<!ELEMENT propertyschema (propertydeclaration*)>
<!ELEMENT propertydeclaration (propertyname,propertytype*)>
<!ATTLIST propertydeclaration id ID #IMPLIED>
<!ELEMENT propertyname EMPTY>
<!ELEMENT propertytype EMPTY>
<!ATTLIST propertytype default (no|yes) 'no'>
<!ELEMENT queryschema (propertyrefrence*)>
<!ELEMENT resourcedescriptionschema (propertyrefrence*)>
<!ELEMENT serviceschema (propertyrefrence*)>
<!ELEMENT propertyrefrence EMPTY>
<!ATTLIST propertyrefrence ref IDREF #REQUIRED required
(no|yes) 'no'>
```

그림 7 CSSD의 DTD

```
<service>
<serviceuri>http://asadal.cs.pusan.ac.kr/cnrp/CnrpServ</serviceuri
>
<dataset id="d1">
 <property name="datasetуri">
urn:oid:jdbc:msql://164.125.164.185:1114/netpia</property>
 <property name="language">English</property>
</dataset>
<servers>
<server>
<serveruri>http://asadal.cs.pusan.ac.kr/cnrp/CnrpServ</serveruri>
</server>
</servers>
<description> This is the Korean Company Information Cnrp
Server </description>
<property name="category" type="freeform">company</property>
<propertyschema>
 <propertydeclaration id="i1" > <!-- city -->
 <propertyname>city</propertyname>
 <propertytype default="yes">freeform</propertytype>
 </propertydeclaration>
</propertyschema>
<queryschema>
 <propertyreference ref="i1" required="yes" />
</queryschema>
<resourcedescriptionschema>
 <propertyreference ref="i1" required="yes" />
</resourcedescriptionschema>
<serviceschema>
 <propertyreference ref="i1" required="yes" />
</serviceschema>
</service>
```

그림 8 CSSD의 보기

```
<!ELEMENT database (databaseschema+)>
<!ATTLIST database id ID #IMPLIED>
<!ELEMENT databaseschema (databaseuri,databasename,table)>
<!ATTLIST databaseschema id ID #IMPLIED>
<!ELEMENT databaseuri (#PCDATA)>
<!ELEMENT databasename (#PCDATA)>
<!ELEMENT table (tablename,attribute-id,attribute-uri,attribute-
desc,tableproperties?)>
<!ELEMENT tablename (#PCDATA)>
<!ELEMENT attribute-id (#PCDATA)>
<!ELEMENT attribute-uri (#PCDATA)>
<!ELEMENT attribute-desc (#PCDATA)>
<!ELEMENT tableproperties (tableproperty*)>
<!ELEMENT tableproperty (propertyreference,propertyname)>
<!ELEMENT propertyreference EMPTY>
<!ATTLIST propertyreference id ID #IMPLIED>
<!ELEMENT propertyname EMPTY>
```

그림 9 DBSD의 DTD

```
<database>
 <databaseschema id="d1">

<databaseuri>jdbc:msql://164.125.164.185:1114/netpia2</databas
euri>
 <databasename>netpia2</databasename>
 <table>
  <tablename> company </tablename>
  <attribute-id> id </attribute-id>
  <attribute-uri> uri </attribute-uri>
  <attribute-desc> description </attribute-desc>
  <tableproperties>
  <tableproperty>
  <propertyreference id="i1" />
  <propertyname> location_city </propertyname>
  </tableproperty>
  </tableproperties>
 </table>
 </databaseschema>
</database>
```

그림 10 DBSD의 보기

- 데이타베이스 이름 및 URI
- 테이블 및 애트리뷰트 정보
- 서비스 스키마와의 맵핑 정보

그림 10은 그림 9의 DTD를 바탕으로 구성된 DBSD의 보기로서 부분만 표시되어 있다.

3.4 클라이언트 설계

3.4.1 클라이언트의 기능

본 논문에서 설계하는 클라이언트는 CNRP 명세에서 제안하는 두 가지 방법 중 웹을 통해 접근 가능한 프론트엔드(front-end) 형태로 여러 서버들에 동시에 질의할 수 있도록 설계한다. 따라서, HTTP를 이해하고 사용자의 질의를 분배하고 여러 서버들의 결과를 통합하는 기능을 가져야 한다. 이를 위해 클라이언트는 다음 기능을 지원하여야 한다.

▶ XML/HTML 변환: 본 논문에서는 웹에서 접근 가능한 형태의 클라이언트를 설계하고 구현하므로, 브라우저와 클라이언트간의 통신을 위해 XML/HTML 변환 기능이 있어야 한다.

▶ 서버와 통신: 초기질의를 보낼 서버들의 URI 정보를 유지하고, 그 정보를 가지고 서버들로 초기질의를 보낼 수 있어야 한다. 또한 초기질의의 결과에서 서버와의 표준질의 통신방식을 분석하고 그 방식에 따라 서버와 통신할 수 있어야 한다.

▶ XML 문서의 처리: 클라이언트는 서버와의 인터랙션을 위해 XML 문서를 처리할 수 있어야 한다.

▶ 질의 분배 기능: 사용자가 원하는 정보를 얻을 수 있는 서버들을 선택하고 각 서버들로 초기질의를 분배할 수 있어야 하며, 초기질의 결과로 받은 서버들의 서비스 스키마 정보를 바탕으로 각 서버들의 서비스 스키마에 적합한 표준질의를 생성하고 분배할 수 있어야 한다.

▶ 동적 질의 인터페이스 생성: 초기질의의 결과로 받은 서버들의 스키마 정보를 통합하여 동적으로 사용자

질의 인터페이스를 생성할 수 있어야 한다.

▶ 질의 결과 통합 기능: 각 서버에서 받은 표준질의 결과를 통합해서 사용자에게 제공할 수 있어야 한다.

3.4.2 클라이언트 설계

서버들의 서비스 스키마 정보를 이용해서 각 서버들에 적합하게 질의를 분배하고 그 서버들로부터의 결과를 통합해서 사용자에게 제공할 수 있도록 하기 위해서 클라이언트에 필요한 모듈을 설계하고 각 모듈간 인터랙션을 정의한다. 서버에서 처럼 모든 인터랙션은 XML을 사용하도록 설계한다. 본 논문에서 설계한 클라이언트의 모듈간 인터랙션과 구성은 그림 11과 같다.

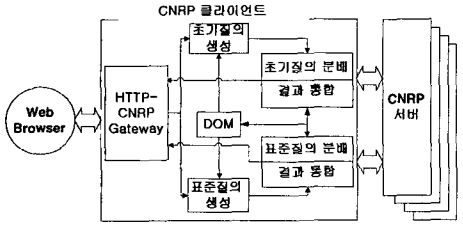


그림 11 CNRP 클라언트 구성도

▶ HTTP-CNRP 게이트웨이: 브라우저와 직접 통신하는 모듈로, 브라우저에 의해 호출되었을 때 추가 결과를 위한 연결이 아니라면 초기 질의 생성 모듈을 호출한다. 또한 초기질의 분배/결과 통합 모듈에 의해서 통합된 CNRP 서비스 스키마 정보를 바탕으로 사용자가 표준질의를 할 수 있도록 동적질의 인터페이스를 생성하고, 표준질의 분배/결과 통합 모듈에 의해서 통합된 XML 형태의 풀이 결과를 HTML로 변환하여 브라우저에 전달한다.

▶ 초기질의 생성 모듈: HTTP-CNRP 게이트웨이의 호출에 의해, 초기질의를 생성하여 초기 질의 분배 모듈로 보낸다.

▶ 초기질의 분배/결과 통합 모듈: 초기질의 생성 모듈로부터 받은 초기질의를 각 서버들로 분배하고, 각 서버에서 온 분배된 질의의 결과를 통합하여 HTTP-CNRP 게이트웨이 모듈로 전송한다.

▶ 표준질의 생성 모듈: 표준질의 생성 모듈은 사용자가 동적 질의 인터페이스에 입력한 보통이름과 프로퍼티들, 그리고 DOM 모듈에 저장된 각 서버들의 스키마 정보를 바탕으로 서버들에 맞게 표준질의를 생성한다.

▶ 표준질의 분배/결과 통합 모듈: 표준질의 분배/결과 통합 모듈은 표준 질의 생성 모듈에서 각 서버들에 맞게 생성된 질의들을 실제 각 서버들로 전송하고, 서버들로부터 받은 풀이 결과를 통합해서 HTTP-CNRP 게이트웨이 모듈로 전송한다.

3.5 클라이언트 구현시 문제점 분석 및 해결 방안

본 절에서는 CNRP 클라이언트의 기능을 구현할 때 발생할 수 있는 문제점들을 알아보고, 이를 해결하기 위해 본 논문에서 제안하는 처리방법을 살펴본다.

3.5.1 초기질의를 보낼 서버 설정

클라이언트는 초기질의를 보낼 서비스의 URI를 어떤 방식으로든 알고 있어야 한다. 초기질의를 보낼 서비스의 URI 정보를 가져올 수 있는 방법은 여러 가지가 있을 수 있겠지만 CNRP 명세에서는 이에 대한 정의가 없기 때문에 본 논문에서는 다양한 서비스로의 확장성을 제공하기 위해서 초기질의를 보낼 서비스의 URI를 XML 문서 형태로 저장해서 구현한다. 이것은 서비스를 제공하는 서버의 URI가 추가될 때마다 프로그램을 수정할 필요가 없이 단순히 초기질의를 보낼 서비스의 URI를 문서에 추가하면 되기 때문이다. 그림 12는 클라이언트가 유지하고 있는 초기질의를 보낼 서버의 리스트로, 본 논문에서 정의한 DTD에 따라 작성한 XML 문서로 저장되어 있는 보기이다.

```
<?xml version="1.0"?>
<!DOCTYPE cnrp SYSTEM "CNRP_client.dtd" >
<frontcfg>
 <servernum>2</servernum>
 <server>http://asadal.cs.pusan.ac.kr/cnrp/CnrpServ</server>
 <server>http://asadal.cs.pusan.ac.kr/cslee/CnrpServ</server>
</frontcfg>
```

그림 12 초기질의를 보낼 서비스 리스트

3.5.2 프로퍼티 타입에 대한 처리

사용자가 브라우저로 클라이언트에 접속했을 때, 클라이언트는 서버에서 제공된 초기 질의의 결과를 기반으로 질의 인터페이스를 생성하여야 한다. 이 때, 프로퍼티의 타입이 Freeform일 경우는 입력폼만을 생성하고, Freeform이 아닌 영역이 정해져 있는 타입일 경우에는 사용자가 입력할 프로퍼티의 도메인을 정확히 이해하지 못할 수 있으므로 가능하면 사용자가 프로퍼티의 값을 선택할 수 있도록 선택 리스트를 만드는 것이 좋다. 이런 경우 선택 리스트를 만들려면 사용될 프로퍼티 타입에 대한 정보를 미리 알고 그 타입에 대한 목록이 준비되어 있어야 한다.

CNRP 명세에서는 프로퍼티의 타입에 특별한 제한을 두지 않고 있다. IANA에 등록되지 않은 타입에 대해서는 처리하지 않아도 된다고 정의하고 있지만, 보다 나은 서비스를 위해서는 다양한 타입에 대한 처리가 필수이다. 본 논문에서는 CNRP DTD를 확장하여 타입에 대

한 처리를 표준화된 방식으로 할 수 있도록 할 것을 제안한다. CNRP DTD의 각 서비스에서 제공하는 프로퍼티를 정의하는 <propertyschema> 객체에 포함된 <propertytype> 객체에 그 타입에 대한 목록이 있는 URI를 제공하는 속성을 포함하도록 확장하고, 타입의 목록을 정의하는 XML DTD를 정의하여 그 DTD에 따라 해당 URI에서 목록을 제공한다면 타입에 대한 표준화된 처리를 할 수 있다.

본 논문에서는 현재의 CNRP 명세에 충실하게 서버와 클라이언트를 구현하므로, CNRP DTD를 확장하여 사용하진 않는다. 대신 사용자에게 보다 나은 질의 인터페이스를 제공하기 위해, 특정 도메인의 값의 목록을 얻을 수 있는 URI를 제공하는 XML 문서를 작성하여 사용한다. 타입의 목록에 대한 URI를 제공하는 XML 문서는 그림 13과 같은 형태를 가진다.

```
<typedef>
 <domain>
  <name>iso1066</name>
  <uri>http://asadal.cs.pusan.ac.kr/~cnrpcli/domain/iso1066.xml</uri>
 </domain>
 <domain>
  <name>rfc1766</name>
  <uri>http://asadal.cs.pusan.ac.kr/~cnrpcli/domain/rfc1766.xml</uri>
 </domain>
</typedef>
```

그림 13 프로퍼티 타입 처리를 위한 XML 문서 보기

3.6 서버/클라이언트 시스템 구현 환경

서버와 클라이언트는 UNIX 기반의 Solaris 운영체제에서 JAVA 언어를 사용하여 구현하였다. 구체적인 구현 환경은 표 3과 같다.

표 3 구현 환경

	서버/클라이언트
하드웨어	Sun UltraSparc 2
운영체제	Solaris 2.5.1
컴파일러	JDK 1.2 , JSDK
XML 파서	Xerces JAVA XML Parser

4. 회사이름 풀이 시스템 구현

3장에서 구현한 CNRP 서버/클라이언트를 사용하여 회사이름 풀이 서비스를 직접 개발함으로써 본 논문에서 제안하는 구현 방안의 타당성과 다양한 CNRP 응용에 사용할 수 있는 가능성을 검증한다.

4.1 회사이름 풀이 시스템 구성도

본 논문에서 구현한 CNRP 서버/클라이언트를 이용하여 사용자가 질의한 회사이름을 풀이하여 회사의 URI를 반환해 주는 서비스의 프로토타입 시스템을 그림 14와 같이 구현한다. 본 논문에서 구현한 CNRP 서버에서 보통이름 풀이기를 회사이름 풀이기로 수정하고 데이타베이스를 회사이름에 관련된 리소스로 구축하였으며, 서비스 스키마와 데이타베이스 스키마를 회사이름 풀이에 적합하게 설계해서 구현하였다. CNRP 인터랙션이 모두 이용될 수 있도록 회사이름 풀이를 위해 사용되는 서버는 2개를 가지도록 한다.

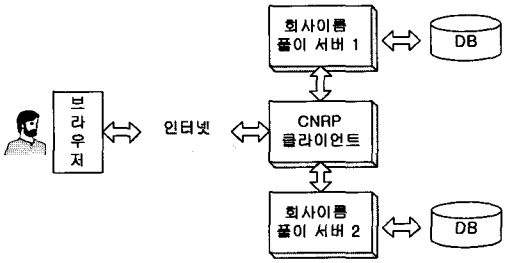


그림 14 회사이름 풀이 시스템 구성도

4.2 구현환경

회사이름 풀이 시스템의 구현 환경은 CNRP 서버/클라이언트에서 보통이름 풀이기를 회사이름 풀이기로 바꾸고, 사용자가 브라우저를 통해 접속하는 것으로 구성하였으므로 CNRP 서버/클라이언트 시스템 구현 환경과 같다. 단, 데이타베이스는 mSQL을 사용해서 구축하고, 데이타베이스 연결을 위해서 JDBC 드라이버를 사용하였다.

4.3 동작 예

그림 15는 회사이름 풀이 시스템에서 초기질의를 바탕으로 동적으로 생성된 질의 인터페이스이로, 두 개의 회사이름 풀이 서버의 서비스 스키마를 통합해서 만들어진 것이다. 그림 15에 나타난 것처럼 회사이름 풀이 서버들이 제공하는 프로퍼티는 geography, language, category이며, 각 프로퍼티의 값을 3개까지 입력하는 것으로 제한하였다. 서버에서 제공한 서비스 스키마에서 geography와 category는 freeform 타입이므로 단순히 입력폼으로 만들어지고, language는 RFC 3122로 표현되는 영역을 제한하는 타입이므로 그림 15와 같이 선택리스트로 만들어진다.

그림 16은 그림 15의 질의 인터페이스에서 보통이름을 "삼성"으로, geography는 "서울"과 "부산"으로 입력하고, language를 "kr"로 선택했을 때, 회사이름 풀이 서버로부터 받은 풀이 결과를 나타내고 있다. 결과로 나온 URI가 4개이므로 결과목록을 보여주지만, 결과 URI

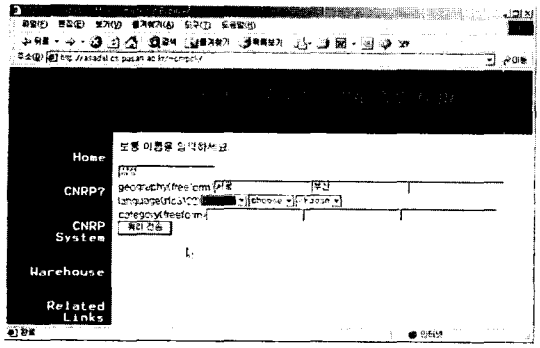


그림 15 회사이름 풀이 시스템의 동적 질의 인터페이스 보기

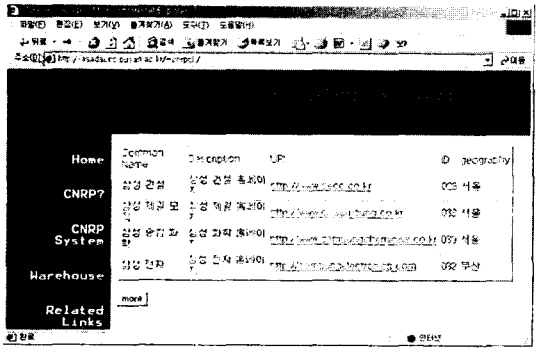


그림 16 회사이름 풀이 시스템의 풀이 결과

가 하나일 경우는 바로 해당 URI로 접속하게 된다. 그림 16은 CNRP 명세에서 정의하고 있는 표준질의의 함축적 논리 연산에 따라서 "서울" AND "kr", "부산" AND "kr" 순서로 결과를 나타내고 있다. 마지막으로 그림 16의 more 버튼은 리퍼럴 객체의 구현을 나타낸 것으로 또 다른 회사이름 풀이 서비스에서 다른 결과를 얻을 수 있음을 나타내고 있다.

5. 결 론

CNRP는 일상 생활을 통하여 흔히 접할 수 있는 보통이름의 풀이를 통하여 원하는 인터넷 리소스를 얻기 위해 제안된 프로토콜이다. 그러나 CNRP 명세에서는 서버와 클라이언트간의 객체 교환 방식만을 제공하고 있을 뿐, CNRP 시스템의 서버 및 클라이언트의 구체적인 구현방안을 제시하지는 않는다. 따라서, 본 논문에서는 보통이름 풀이를 위한 서버/클라이언트 시스템을 구현할 때 발생할 수 있는 문제점들을 분석하고, 그에 대한 해결책을 제시함으로써 구체적인 방안을 제시했다. 또한 제시한 방안에 따라 CNRP 서버/클라이언트 시스템을 설계 및 구현하였다. 또한, 본 논문에서 구현한 CNRP 서버/클라이언트를 이용하여 회사이름을 풀이하여 회사의 URI를 제공하는 서비스의 프로토타입 시스템을 개발함으로써 구현 방안의 타당성과 다양한 응용서비스에 적용할 수 있는 가능성을 검증해 보았다.

본 논문에서 구현한 서버/클라이언트 시스템은 풀이 결과가 단순한 검색 결과가 아닌 XML 형태의 데이타이므로 다양한 형태로 다양한 응용에서 이용될 수 있으며, 또한 확장성을 고려하여 설계 및 구현하였기 때문에 다른 서비스를 제공하는 시스템, 보통이름에 대한 풀이 방식이 다른 시스템, 서로 다른 데이타베이스나 데이타셋을 사용하는 시스템에도 쉽게 적용할 수 있다. 따라서, 본 논문에서 구현한 CNRP 서버/클라이언트 시스템은 다양한 CNRP 응용 서비스 개발 및 CNRP 연구를 위한 도구로서 활용될 수 있다.

CNRP는 아직 I-D 상태에서 활발히 연구되고 있지만, 보통이름 공간에서 알 수 있듯이 앞으로 다양한 응용분야에 적용되어 구현될 것으로 본다. 그러나, 실제 구현된 시스템들이 상호 동작하기 위해서는 프로퍼티에 대한 여러 문제점들의 해결방안이 필요하다. 이를 위해 프로퍼티 타입의 등록 기관인 IANA에서 프로퍼티 운영에 대한 구체적인 방안을 제시할 수 있어야 할 것이다.

참 고 문 헌

[1] Nicolas Popp, Michael Mealling, Larry Masinter, Karen Sollins "Context and Goals for common name Resolution," RFC 2792 October 2000.

[2] Moseley, M., Mealling, M. and N. Popp, "CNRP PROTOCOL SPECIFICATION," Internet-Draft draft-ietf-cnrp-10, January 23, 2001.

[3] 신봉기, 김영환, "인터넷 정보검색 동향", 정보과학회지 제 16권 제 8호 pp. 16-20, 1998년 8월.

[4] 권혜진 외, "국내 웹 정보 검색 기술 동향", 정보과학회지 제 15권 제 10호 pp. 16-23, 1997년 10월.

[5] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, BCP 14, March 1997.

[6] M. Mealling, "The 'go' URI Scheme for the common name Resolution Protocol," draft-ietf-cnrp-uri-06, January 23, 2001.

[7] 홍승우, 유영호, 김형수, 김경석, "XML을 이용한 CNRP기반의 통합 검색 시스템 설계", 2000 가을 학술 발표 논문집(I), 제 27권 2호.

[8] 김형수, 유영호, 이철숙, 정일동, 김경석, "CNRP 서버/클라이언트 시스템 설계", 2001 봄 학술 발표 논문집 (B), 제 28권 1호.

[9] Network Solution Inc. "NSI CNRP Pilot," http://www.cnrp.research.netsol.com May 30, 2001.
[10] Catalogix Corporation "GIDS(Global Indexed Directory Service)," http://gids.catalogix.se/gids.html
[11] RealNames Corporation "RealNames Internet Keyword interface," http://www.internetkeywords.org, May 2000.
[12] M. Mealling, L. Daigle, VeriSign, Inc., "Service Lookup System(SLS)," Internet-Draft draft-mealling-sls-01, November 21, 2001.
[13] W3C XML Specification, http://www.w3.org/TR/2000/REC-xml-20001006
[14] DOM (Document Object Model) Specification, http://www.w3.org/DOM/
[15] 맹성현, 주종철, "문서 구조화와 정보 검색", 정보과학회지 제 16권 제 8호, pp. 6-15, 1998년 8월.
[16] Steve Lawrence, C. Lee Giles, "Inquirus, the NECI Meta Search Engine," WWW7/Computer Networks 30(1-7), 1998, pp.95-105.

유 영 호
1994년 부산대학교 전자계산학과(이학사). 1997년 부산대학교 전자계산학과(이학석사). 1997년 ~ 현재 부산대학교 전자계산학과 박사과정. 관심분야는 데이타베이스, XML, 이동 통신

이 종 환
1996년 부산대학교 전자계산학과(이학사). 1998년 부산대학교 전자계산학과(이학석사). 1998년 ~ 현재 부산대학교 전자계산학과 박사과정. 관심분야는 데이타베이스, 정보가전, XML

이 중 화
1994년 부산대학교 전자계산학과(이학사). 1997년 부산대학교 전자계산학과(이학석사). 2001년 부산대학교 전자계산학 박사. 2002년 ~ 현재 동의대학교 컴퓨터-영상공학부 전임강사. 관심분야는 데이타베이스, 멀티미디어, XML, 한글/한말 정보처리

김 경 석
1977년 서울대학교 무역학과(경제학사). 1979년 서울대학교 전자계산학과(이학석사). 1988년 일리노이 주립대(어바나-샴페인) 전자계산학 박사. 1988년 ~ 1992년 미국 노스다코타 주립대학교 전자계산학과 조교수. 1992년 ~ 현재 부산대학교 전자전기정보컴퓨터공학부 교수. 관심분야는 데이타베이스, 멀티미디어, 한글/한말 정보처리