서버 간 요청 위임을 고려한 XML 기반 RPC 자원 서비스 시스템 설계 및 구현

김정희[†] · 곽호영[††]

요 약

본 논문에서는 분산 환경의 자원 활용성과 처리의 효율성을 위하여 클라이언트의 요청(Request)이 서버 간 위임되는 XML 기반 RPC 자원 서비스 시스템을 설계하고 구현한다. 이를 위해 클라이언트가 이용할 복수개의 서버들에 대한 요청 정보들은 XML의 DOM에 근거하여 XML 문서로 포장된 후 분산 환경의 서버로 전송되고, 서버는 XML-DOM을 받아들일 수 있는 객체를 사용하여 클라이언트의 요청을 처리하도록 한다. 또한 서버 간 요청을 위임하기 위해 처리 결과를 바로 클라이언트에게 응답하지 않고 XML-DOM 구조 내에 저장하고, XML-DOM 정보 내의 또 다른 요청 정보에 따라 해당 서버로 클라이언트의 요청이 위임되도록 한다. 또한 분산 환경의 자원 투명성을 위해 자원 저장소(Resource Repository)를 운영토록 한다. 시스템 구현 결과 RPC와 XML-RPC 서비스의 통합 및 서버 간 요청 위임으로 인한 응답 시간(Response Time)의 단축으로 자원 활용성과 처리의 효율성을 향상시켰으며, 부가적으로 클라이언트의 비동기성의 확장과 평균 블록화를 줄일 수 있었다. 또한 클라이언트가 이용할 수 있는 자원에 대한 정보 제공이 가능해졌다.

Design and Implementation of Remote Procedure Call Resource Service System based on the XML Considering the Request Delegation between the Servers

Jeong-Hee Kim[†] and Ho-young Kwak[††]

ABSTRACT

Design and implementation of remote procedure call (RPC) resource service system based on the XML considering the request delegation between the servers are described for the resource application and efficient processing of distributed environment. For this purpose, the request information about several server, which would be used by client, is encoded into XML document based on XML-DOM and transferred to server. Server classifies the client requests using objects which can deal with the XML-DOM. For the request delegation between the servers, server saves the request result of client in XML-DOM structure without replying it immediately, and makes the client request delegated to another server in XML-DOM's request information. In addition, the resource repository are managed for resource transparency of distributed environment. As a result, resource application and processing efficiency are improved by reduction of response time caused by integration of RPC and XML-RPC services and request delegation between the servers. Extension of asynchronous and average blocking time of client were reduced and information of resource which can be used by client was offered.

Key words: RPC, XML-RPC, XML, XML-DOM, Asynchronous, Delegation

---

접수일 : 2003년 2월 25일, 완료일 : 2003년 4월 9일
[†] 정회원, 제주산업정보대학 컴퓨터정보계열 겸임교수
[††] 정회원, 제주대학교 통신컴퓨터공학부 교수

1. 서 론

전자상거래, 인터넷 정보 검색과 같은 분산 응용 프로그램들이 인터넷 사용의 증가와 분산 컴퓨팅 기술의 발전에 따라 점차 증가하고 있으며 이러한 분산 응용 프로그램에서 프로세스 사이의 상호협력을 위한 통신(Inter Process Communication)에는 전통적으로 원격 프로시저 호출(Remote Procedure Call)에 기반 한 클라이언트-서버 모델이 사용되어 오고 있다.[1-3]. 하지만, RPC는 개념적으로 단순하고 구현이 간단하기는 하지만 낮은 대역폭과 통신량이 많은 네트워크 환경에는 적합하지 않은데, 이는 분산 환경과 응용 프로그램이 갈수록 복잡해지고, 사용자들은 더 많은 편리함을 추구하면서 새로운 형태의 상호협력 메커니즘을 필요로 하고 있는 실정이며[4], 이러한 요구로 현재 이동 에이전트(Mobile Agent)가 등장하고 있다.[5,6]. 이동 에이전트는 이동성과 자율성이라는 특성을 가지고 호스트간의 통신량을 줄임으로써 네트워크 부하를 줄일 수 있고, 클라이언트와 서버간의 비동기성을 증가시켜 클라이언트와 서버 사이의 지속적인 연결을 요구하지 않으며, 사용자 중심적인 기능을 수행하기에 용이한 특징을 갖는다. 최근 연구에 따르면 이동 에이전트가 전통적인 클라이언트-서버 패러다임에 비해 정보검색, 네트워크 관리, 이동 컴퓨팅과 같은 응용 분야에서 유용함을 보이고 있으며[6-8], 또한 네트워크 부하와 실행시간에 영향을 주는 파라미터를 사용하여 프로세스 간 통신(IPC) 패러다임의 성능을 비교, 분석하는 연구가 진행 중이다[9-12].

웹(WWW)은 현재 우리 생활에 큰 영역을 차지하면서 비즈니스 영역으로 그 활용 분야를 넓혀가고 있다. 현재 많은 프로그램들은 HTTP를 기반 전송 규약으로 사용하는 웹 응용 프로그램으로 구축되고 있으며, 웹 브라우저를 통해서 임의의 플랫폼에서도 응용 프로그램에 접근하여 정보 공유와 경제 활동을 하고 있다[13]. 이러한 흐름 속에서 표준 데이터 교환 형식은 XML[14,15] 기술을 활용하면서, 응용 프로그램 간 통신 수단으로는 웹을 이용하도록 하는 분산 환경을 어떻게 더욱 발전된 모습으로 지원해 나갈 수 있을 것인지를 두고 다양한 시도가 이루어지고 있다[16], 분산 환경에서 XML 기술을 활용하여 가장 중심적인 역할을 수행할 수 있는 분야는 데이터 표현의 통합화 및 표준화된 채널을 제공하는 부분과, 또한 이를 통하여 모든 사용자들이 접근 가능하도록 하는 기반 구조들에 있다[17,18]. 또한 XML 기술은 현재에도 사용되고 있는 분산 객체 기술인 Microsoft의 COM/DCOM, OMG의 CORBA[19], JAVA Bean 등의 호환성 문제 및 객체기반 분산 컴퓨팅 환경 하에서의 클라이언트 ORB(Object Request Broker) 소프트웨어 의존성(클라이언트는 서버와 데이터 교환을 하기 때문에 CORBA, DCOM, JAVA RMI 등의 ORB 등은 상호운영성의 제약과 방화벽에 의한 메시지 차단, 원격 프로시저를 호출하지 않고 웹과의 연동 시에 원격지 응용 프로그램과의 상호작용이 되지 않아 서로 통신을 할 수 있는 표준 프로토콜 필요한 사항[20-22])등의 문제점을 극복하기 위해 IIOP[22]나 RPC를 사용하고 있는 추세이며, Publishing과 Data Exchange라는 두 가지 주요한 Application 영역과 XML-RPC[17,22,23], SOAP[24,25], WDDX[18] 등이 프로토콜이 제공되어 응용 프로그램 간 데이터 교환을 지원하면서 단순히 데이터 교환뿐 아니라 그에 적합한 처리를 할 수 있는 응용 프로그램까지 데이터를 전달할 수 있도록 그 활용 범위를 넓혀가고 있다[18].

따라서, 본 논문에서는 전통적인 RPC와 XML에서 지원되는 분산 환경 접근 기술(XML-RPC)을 통합하고, 분산 환경상의 서비스 요청과 제공(Request-Reply)의 종속적 관계(일대 일)로 인한 클라이언트의 평균 블록화(Blocking) 시간의 단축과 비동기성의 확장, 그리고 클라이언트의 요청을 위임하는 RPC 자원 서비스 시스템을 제안하고 구현한다. 시스템은 서비스 요청에 따라 TCP/IP 및 HTTP 기반을 제공하여 접근 기술을 통합하며, 위임 기능을 위한 메시지 구조체는 XML-DOM을 사용하도록 하며, 그럼으로써 자원의 활용성과 처리의 효율성을 향상시키도록 한다.

본 논문의 2장은 분산 환경과 XML과의 관련기술에 대하여 살펴보고, 3장에서는 본 논문에서 제안하는 요청 위임 RPC 자원 서비스 시스템 모델에 대하여 기술하며, 4장에서는 시스템 구현, 5장은 실험 결과, 그리고 6장에서는 결론 및 향후 연구에 대해 기술한다.

2. 관련연구

분산 기술은 웹과의 호환성을 직접 가지지 못하므

로 웹에서 데이터 교환에 사용하려는 애플리케이션들은 별도의 프로토콜을 지원해야 한다. 이러한 컴포넌트 기술과 웹과의 호환성 문제를 해결하기 위해서 XML을 이용하고자 하는 많은 노력이 이루어져 왔으며, XML은 시스템간의 상호 운영성(Interoperability) 측면에서 볼 때 기존의 복잡했던 문제를 상당부분 단순화시키는데 큰 역할을 수행하고 있다.

2.1 분산 시스템(Distributed System)

분산 시스템의 특징은 각 노드들이 서로 다른 운영체제와 자신의 파일 시스템 및 다른 관리 하에 놓이며, 인터넷으로 서로 약하게 연결되는 수많은 시스템으로 구성된다. 전형적인 인터넷 응용은 원격 컴퓨터에 대한 접근(telnet, rlogin), 원격 정보에 대한 접근(웹, FTP), 개인 간 통신(전자우편, 채팅), 그리고 방대한 응용(전자상거래, 원격 진료, 원격 강의)등을 포함한다. 또 다른 특징은 전체 시스템을 같은 방법으로 바라볼 수 있게 하는 공통 패러다임이다. 즉, 시스템을 단일화하는 패러다임으로, 운영체제 위에 미들웨어(middleware) 소프트웨어 층을 둔다[26].

2.2 RPC(Remote Procedure Call)

Birrell과 Nelson에 의해 제안된 RPC[27]는 클라이언트와 서버간 디스패치(Dispatch) 방식으로 운영되는 서비스이다. 즉, 다른 CPU에 위치한 프로시저를 호출하는 프로그램을 허용하는 것이다. 머신(Machine) 1에 있는 프로세스가 머신 2에 있는 프로시저를 호출할 때 머신 1에 있는 프로세스는 중단되고, 머신 2에 있는 호출된 프로시저가 수행된다. 호출자와 피호출자 간에 결과를 포함한 정보는 파라미터로 전달된다. 그리고 원격 프로시저 호출을 지역 호출과 같이 보이게 하기 위해 각각 클라이언트 스터브(Client Stub)와 서버 스터브(Server Stub)가 필요되어진다[28].

반면 통합 서비스를 위한 RPC는 데이터 전송 과정에서 데이터의 구조를 표현하는 인코딩에 문제가 있다[29]. 그림 1은 일반적인 RPC 호출 단계와 내부 인터페이스를 보여주며, 요청과 응답의 관계는 일대 일이다.

2.3 XML-RPC

XML-RPC(XML-Remote Procedure Call)[17,22,23,30,31]는 XML 프로토콜의 한 종류이다. 이는 XML을 이용하여 원격지 머신에 대한 함수 호출을 캡슐화한 후 XML 문서를 전달하고 그 결과값으로 XML 문서를 돌려 받는다. 그럼으로써, 전통적으로 RPC를 사용하는데 있어서 가장 큰 방해물이였던 인코딩 문제를 해결하게 되었으며 또한 데이터 전송은 HTTP 방식을 사용하면서 매우 간단한 데이터를 텍스트로 표현할 수 있도록 할 뿐만 아니라 데이터 구조를 나타내는 표준을 제공하게 되었다[13].

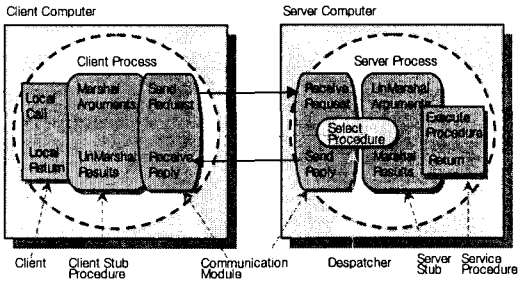


그림 1. Stub Procedure 원격 프로시저 호출 단계

또한 DOM[32]과 SAX[33]의 XML 접근 API들로 인해 XML-RPC 라이브러리를 쉽게 구현할 수 있게 되었으며, 따라서 XML을 인코딩으로서 사용할 경우 빠르고 경량이 되며, 애플리케이션 관련 부분 중에서 서로 다른 환경을 연결한다든지, 응답 시간이 절대적으로 중요하지 않는 프로젝트에서 매우 효과적이고 좋은 성능을 제공한다는 것이 증명되었다[13]. 그림 2는 XML-RPC에서 요청에 대한 정보의 인코딩 예를 보여준다.

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181
<?xml version="1.0"?>
<methodCall>
 <methodName>examples.getStateName</methodName>
 <params>
   <value><i4>41</i4></value>
 </params>
</methodCall>
```

그림 2. XML-RPC 인코딩

3. 요청 위임을 고려한 RPC 자원 서비스

3.1 제안 시스템 개요

본 논문에서 제안하는 시스템의 목적은 클라이언

트의 요청이 서버 간 위임되도록 하는데 있으며, 그럼으로써, 클라이언트의 블록 상태의 단축 및 비동기성을 확장 지원하는데 있다. 그림 3의 빗금 친 부분에서 클라이언트가 사용할 자원의 위치는 특정 서버 하나에 한하고 있는 전형적인 RPC 환경을 나타내고 있으며, 이러한 환경에서 비동기는 하나의 서버에 대해 다중의 자원을 적절한 시간 차이를 두고(그림 4, Asynchronous RPC) 요청할 때 지원되는 상황에 있다. 하지만 본 논문에서는 특정한 서버의 자원에서 여러 곳의 서버 자원을 사용할 수 있도록 요청을 서버 간 위임하며, 최초 요청 후 또 다른 요청을 적절한 시간 차이를 두지 않고 최초 요청 시 한꺼번에 요청하는 메커니즘을 고려한다. 그림 3은 제안한 시스템의 동작 개요이며, 그림 4의 빗금친 부분은 제안하는 시스템의 비동기 메커니즘이다.

그림 3과 같은 동작이 이루어지기 위해서 시스템은 각 서버가 처리한 결과를 저장하고, 또 다른 서버로 클라이언트의 요청을 위임, 또는 클라이언트로 모든 처리 결과를 응답 여부를 판단할 때 본 논문에서는 XML-DOM 구조를 사용하는 Gathering System

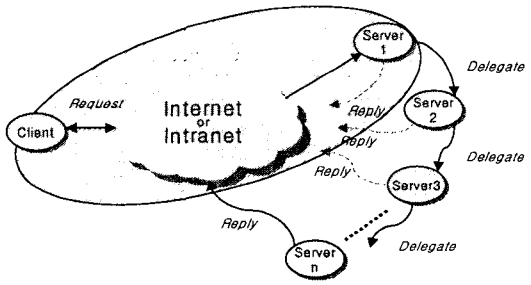


그림 3. 제안한 시스템의 동작 개요

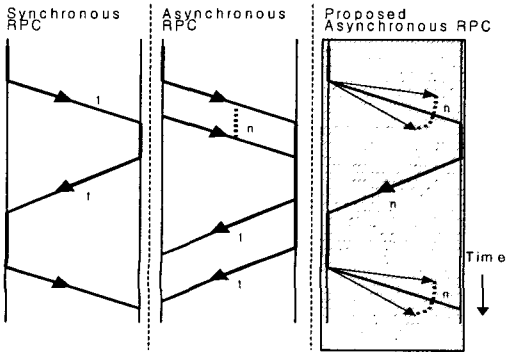


그림 4. 제안한 시스템의 비동기 개요

모듈을 구현함으로써 해결한다. 즉, 최초 요청 시 XML-DOM 객체 클래스를 생성하고, XML-DOM 내용에는 서버 URL과 일반 응용프로그램 또는 XML-RPC 서비스 등의 서버 자원 목록을 지정하고, 요청할 메소드와 데이터가 포함되며, 추가적으로 해당 서버가 처리한 결과가 XML-DOM 내에 저장되어 전달되도록 한다. 그리고 각각의 서버들은 XML-DOM을 파싱하고 클라이언트의 요청 정보를 처리하게 된다. 따라서 클라이언트는 전형적인 요청-응답의 종속 관계(일대 일)를 벗어 날 수 있는 블록 상태 단축 및 비동기를 확장 구현하게 되고, 그럼으로써, 동시에 이용할 수 있는 자원은 분산 환경 상에 존재하는 여러 개의 서버들로 확장되게 된다.

제안하는 시스템 전체 동작 구조는 그림 5와 같고, 서버 동작 환경은 Gathering System을 유기적으로 사용하며, 서버 자체는 일반 응용프로그램과 XML-RPC를 처리하고 또 다른 서버로 클라이언트의 요청을 위임하는 구조가 된다.

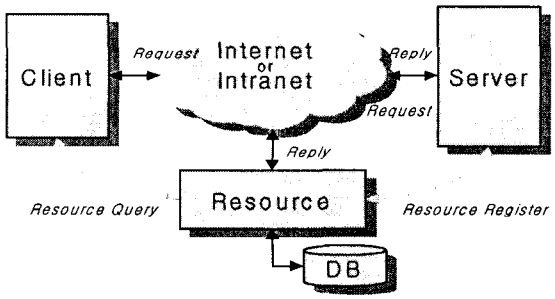


그림 5. 시스템 동작 구조

3.2 클라이언트 시스템(Client System)

클라이언트는 서비스를 요청할 서버의 URL 주소(targetURL), 일반 응용프로그램 서비스 요청인지 XML-RPC 서비스 요청인지를 구분하는 Service, 수행될 서버의 Function과, Function이 사용할 데이터(Data)를 인자로 갖는 함수(그림 6)를 이용하여 XML 요청 문서를 생성한다. 그 결과 그림 7과 같이 XML 문서로 포장되어 분산

```
createXML(targetURL, Service, Function, Data)
```

그림 6. Request XML 문서 생성 함수

```
<Service_list>
  <list>
      <url>carina.cheju.ac.kr</url>
      <service>app</service>
      <function>reverse</function>
      <data>carina</data>
  </list>
  <list> ····· </list>
</Service_list>
```

그림 7. 생성된 Request XML 문서의 예

환경의 서버로 전달된다. 그림 8은 클라이언트 시스템의 메커니즘이다.

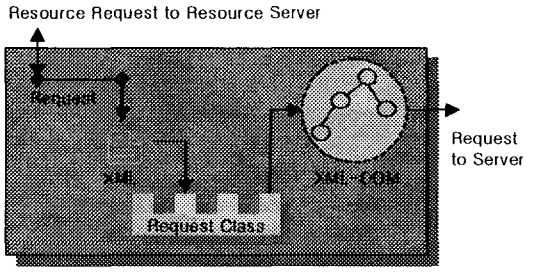


그림 8. 클라이언트 시스템

3.3 서버 시스템(Server Systems)

서버 시스템은 클라이언트의 요청이 일반 응용프로그램인지, 혹은 XML-RPC 인지를 XML-DOM 정보에서 파악하여 처리하며, 처리된 결과를 클라이언트로 전송할 것인지, 아니면 또 다른 서버로 클라이언트의 요청을 전달할 것인지를 XML-DOM 정보에 따라 판단한다. 일반 응용프로그램의 처리는 전형적인 RPC 처리를 말하며, XML-RPC 처리는 HTTP 기반의 메시지 기반 처리를 말한다. 본 논문은 통합을 위해 일반적인 RPC 처리에 추가적으로 XML

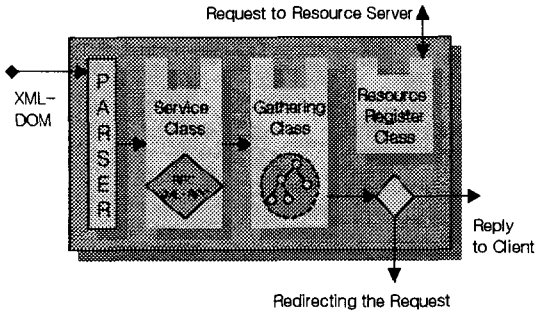


그림 9. 서버 시스템

기술을 이용한 XML-RPC 서비스도 지원하도록 하며, 이렇게 함으로써 분산 환경의 분산 객체를 사용하기 위한 실행환경(Execution Environment)에 보다 자유로우면서, ASP, PHP, JSP 등 일반 웹에서 제공되는 서비스까지 이용할 수 있는 장점을 지니게 한다. 즉, 서버 시스템은 요청에 따라 서비스를 구별하여 처리하고 그 결과를 Gathering System에 의해 XML-DOM에 추가로 저장하고 또 다른 서버로 클라이언트의 요청을 위임하거나 최초 요청 클라이언트로 DOM 정보를 응답하도록 한다.

또한 서버 시스템들은 Resource Register를 이용하여 제공할 수 있는 자원 정보들에 대해 자원 데이터베이스에 등록을 요청할 수 있고, 그럼으로써, 분산 환경의 자원 관리가 자동화 되도록 한다. 그림 9는 서버 시스템이 클라이언트의 요청을 처리하는 절차이며, 그림 10은 Gathering System의 흐름도, 그림 11은 Resource Register의 함수 구조이다.

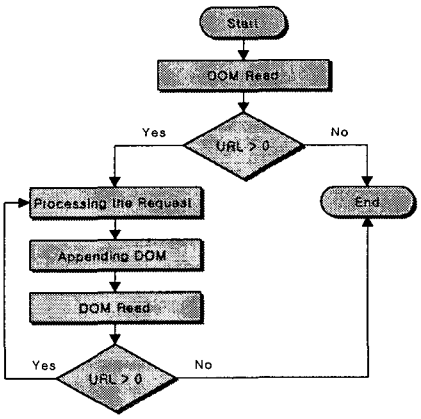


그림 10. Gathering System 흐름도

3.4 자원 저장소(Resource Repository)

분산 환경 상에서 자원의 위치를 제공하기 위한 저장소이다. 소규모 환경 상에서는 대부분 자원의 위치를 클라이언트가 묵시적으로 알고 서비스를 사용하게 되는데 범위를 확장했을 때는 이런 가정이 불가능하게 된다. 따라서 분산 시스템의 자원 투명성을 지원하기 위한 개념으로 클라이언트가 이용할 수 있는 자원들의 위치를 제공하기 위한 목적으로 자원 저장소를 운영한다. 서버 URL, 제공하는 Method, 서비스 유형, 파라미터 유무 등에 대한 정보를 관리하고 클라이언트의 Query에 대해 적절하게 응답하도

록 한다. 스키마 구조는 그림 11의 파라미터 형식을 갖는다.

```
RR(serverURL, Service, Function, Data_format)
```

그림 11. Resource Register 함수 구조

4. 시스템 구현

시스템의 구현 환경은 다음과 같고, 표 1은 클라이언트와 서버의 배치 상황이며 reverse, echo, sum은 서버에서 제공하는 메소드 명이다.

- ▶ IBM PC Pentium (III)
  - · Windows XP
  - · Windows 2000 Server
  - · Linux(redhat) 7.3
  - · J2SDK 1.4.0.02
  - · Xerces Parser for XML
- ▶ helma.xmlrpc : XML-RPC Library
- ▶ Testing Program
  - · ASP, PHP : XML-RPC Program
  - · JAVA : Application Program
- ▶ Oracle 8i : Resource Database

표 1. 클라이언트와 서버 시스템

	OS	IP/DNS/Method
Client	2000 XP	203.253.213.169
Server_1	2000 Server	carina.cheju.ac.kr/echo
Server_2	Linux	mincando.cheju.ac.kr/sum
Server_3	Linux	sslab.cheju.ac.kr/reverse

4.1 클라이언트 시스템(Client System)

클라이언트 시스템은 분산 환경에 존재하는 서버들의 자원들을 이용하기 위해 3.2절에서 설명한 내용을 기반으로 요청 정보를 XML-DOM 구조를 사용하여 XML 문서로 포장한다. 클라이언트의 요청은 운영체제 쉘(Shell) 상에서 직접 인자를 넘겨주는 방식으로도 가능하지만, 미리 프로그램 상에 코딩된 형식으로도 처리될 수 있도록 하였다. 요청 정보를 XML 문서로 포장하는 방식은 그림 12와 같이 처리하였다.

```
public static void createXML()
{
  xmlDOM a=new xmlDOM(); /* XML-DOM class */
  a.create_root();       /* 루트 엘리먼트 생성 */
  /* DOM 정보  targetURL + Service + Function + data */
  /* 클라이언트 주소(최종 목적지) */
  a.appendFinal("203.253.213.169");
  /* XML-RPC */
  a.appendAgent("carina.cheju.ac.kr","rpc","echo",-
                "Hello Mr. Jwa");
  a.appendAgent("mincando.cheju.ac.kr","rpc","sum",-
                "20,40");
  /* Application */
  a.appendAgent("sslab.cheju.ac.kr","app","reverse",-
                "carina");
  PrintDocument.Print(a.theDocument.-
                      getDocumentElement());
  SerialClient.sendToHost(a,a.getFirstURL());
}
```

그림 12. XML-DOM 내의 Request 정보

4.2 서버 시스템(Server Systems)

본 논문에서는 아파치에서 진행하는 Java 기반의 Xerces Parser[34]를 사용하여 구현하였으며, 서버 시스템은 이를 기반으로 XML-DOM 구조의 내용을 파싱하여 클라이언트의 요청을 하나씩 처리하도록 하였다. 그 과정은 일반 응용 프로그램 서비스인지 XML-RPC 서비스인지 판단하여 처리하는 것과 서비스 결과를 XML-DOM에 저장하고 클라이언트의 요청을 또 다른 서버로의 위임 여부를 판단하는 것이다. 그림 13에 필요한 메소드를 정리하였다.

```
private String processFunction(String, String)
{ RPC 서비스와 Application 서비스 호출 }
private void processService()
{
  /* 처리 결과 값 result를 XML-DOM에 추가 */
  result = processFunction(myAgent.getFirstMethod()
  appendResult();
  if(클라이언트의 요청 서버 주소 수 > 0)
  { 또 다른 서버의 주소로 클라이언트 request 위임 }
  else
  { 최종 목적지 주소로 XML-DOM 정보 전달(응답) }
}
```

그림 13. 서버 시스템의 처리 루틴

4.2.1 Gathering System

서버들이 해당 요청을 처리하고 그 결과를 저장할 때 유기적으로 사용되는 시스템이다. 서버의 처리 결과는 3.3절에서 설명한 바와 같이 XML-DOM 구조 내에 저장 된다. 따라서 이를 위한 XML-DOM 객체와 루트 엘리먼트 생성, 자식 엘리먼트 추가, 결과를 저장하기 위한 결과 저장, 최종 목적지 정보, 전체 결과 리스트 등에 관련된 메소드들이 필요하며, 그림 14에 이들 메소드들을 정리하였다.

```
public xmlDOM() {   /* 문서 타입 정의 */ }
public void create_root() { /* request_list 루트 엘리먼트
                              생성하고 추가 */ }
public void appendxmlDOM
{  /* DOM에 다음 구조에 해당되는 자식 엘리먼트를 생성하고
      그 값을 Text Node로 삽입 */
   <list>
     <url>Value </url>  <targetURL>Value </targetURL>
     <Service>Value </Service>
     <Function>Value </Function> <Data>Value </Data>
   </list>
   <list>  .....  </list>
}
public void appendResult(String data)
{ /* <result> 엘리먼트를 생성 및 결과 값을 DOM에 추가 */ }
public void appendFinal(String data)
{/*<final> 엘리먼트 생성, 최종 목적지 주소 값을 DOM에 추가*/}
public void reportResult()
{ /* <result> 엘리먼트의 값을 출력(전체 request 결과) */ }
public String getFirstURL()
{ /* 또 다른 서버로 클라이언트의 request를 전달할 값 */ }
public String getFinalURL()
{ /* 요청한 클라이언트 주소 파악 */. }
```

그림 14. Gathering System의 메소드

4.2.2 Resource Register

분산 환경 상에 존재하는 각각의 서버들은 클라이언트가 이용할 수 있는 자원에 대한 정보를 Resource Register를 이용하여 Resource Database에 등록할 수 있다. ***RR("carina.cheju.ac.kr","RPC","echo","string_data")***을 호출할 경우 그림 15의 함수가 실행된다.

```
public static void RR(s_URL,s_Service,s_Function,s_Para)
{
  query q = new dbQuery(); /* DB Query Class */
  q.insert(s_URL,s_Service,s_Function,s_Para);
}
```

그림 15. Resource Register의 메소드

4.3 자원 저장소(Resource Repository)

서버들이 제공할 수 있는 자원을 관리한다. Resource Register의 요청에 의해 해당 서버의 자원들이 등록되며, 클라이언트의 서버 자원 질의에 대해 응답하도록 되어있다. 그림 16은 오라클 데이터베이스에 등록된 서버 정보를 웹페이지로 출력한 것이다.

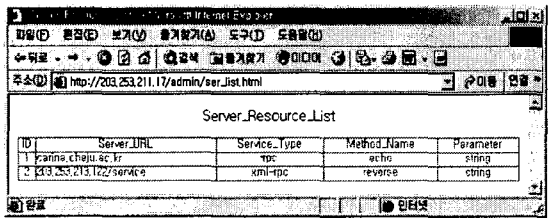


그림 16. Resource Repository 등록 현황

5. 실행 결과

실행 환경은 표 1과 같이 배치하였으며, 검증을 위해 클라이언트의 요청을 4가지로 분류하여 테스트하였다.

5.1 RPC 요청 결과(reverse)

지정한 서버의 일반 응용 프로그램(JAVA Program)서비스를 요청한 결과이다. 그림 17의 결과화면을 살펴보면 <Service_list> 엘리먼트 안에 <final> 엘리먼트는 최종 목적지 URL이 되며, <url> 엘리먼트는 서버 URL이며,.<method> 엘리먼트는 "app"이므로 일반 응용프로그램을 요청한 것이며, <function>은 "reverse" 함수를 지정한 것이며, <data>는 넘겨주는 인자 값으로 "How are you?"인 것을 볼 수 있다. 그리고 그 밑으로 서버에서의 응답이 올 때까지 Waiting이 되었다가 마지막 라인에 그 결과가 얻어졌다. 이 결과 화면은 클라이언트의 요청을 한 개의 서버에게만 요청한 결과이다.

```
Shell>java serialClientMain
Requesting RPC and APP Service via Agent.....
<Service_list>
 <final> 203.253.213.169 </final>
 <list>
   <url> sslab.cheju.ac.kr </url>
   <method> app </method>
   <function> reverse </function>
   <data> How are you? </data>
 </list>
</Service_list>
Waiting for Response of Agent on port 5005
sslab.cheju.ac.kr's Result : ?uoy era woH
Shell>
```

그림 17. RPC 요청 결과(Reverse)

그림 18은 서버 쪽에서 클라이언트의 요청을 분석한 것을 보여준다. "Next Host name"은 처리 후 이동할 URL이며, "Total Host Count"는 클라이언트가 서비스를 지정한 서버 수를 말하며, "Current Host name"은 서버를 말하며, "Current Method"는 요청한 RPC를, "Current Function"은 서버가 실행할 함

```
Next Host name : 203.253.213.169
Total Host Count : 1
Current Host name : sslab.cheju.ac.kr
Current Method : app
Current Function : reverse
Current Data : How are you?
```

그림 18. 클라이언트 요청에 대한 모니터링 결과

수명이며, "Current Data"는 넘어오는 인자값을 나타낸다.

5.2 XML-RPC 요청 결과(sum)

그림 19는 지정한 서버의 웹 서버에게 XML-RPC를 이용한 클라이언트의 요청 처리 결과이다. 이는 분산 환경의 서버에게, 특히 웹 서버에게 XML-RPC를 요청한 것이며, 또한 웹 서버에서 실행될 함수는 ASP 함수이며 클라이언트가 넘겨주는 두개의 인자(20,40)을 받아서 덧셈 연산 후 넘겨주었다. 결과 화면의 정보는 5.1절을 참고하기 바란다. 그림 20는 서버에서 모니터링 한 화면이다.

```
Shell>java serialClientMain
Requesting RPC and APP Service .....
<Service_list>
 <final> 203.253.213.169 </final>
 <list>
   <url> mincando.cheju.ac.kr </url>
   <method> rpc </method>
   <function> sum </function>
   <data> 20,40 </data>
 </list>
</Service_list>
Waiting for Response of Agent on port 5005
mincando.cheju.ac.kr's Result : 60
Shell>
```

그림 19. XML-RPC 요청 결과 : ASP

```
Next Host name : 203.253.213.169
Total Host Count : 1
Current Host name : mincando.cheju.ac.kr
Current Method : rpc
Current Function : sum
Current Data : 20,40
```

그림 20. 클라이언트 요청에 대한 모니터링 결과

5.3 XML-RPC 요청 결과(echo)

서버의 웹 서버에게 XML-RPC를 이용한 클라이언트의 요청 결과이다. 결과 화면은 그림 21, 모니터링 화면은 그림 22이며, 상황은 5.2와 동일하다. 단, 요청한 서비스만 PHP 프로그램일 뿐이다.

```
Shell>java serialClientMain
Requesting RPC and APP Service .....
<Service_list>
 <final> 203.253.213.169 </final>
 <list>
   <url> carina.cheju.ac.kr </url>
   <method> rpc </method>
   <function> echo </function>
   <data> Hello Mr. Jwa </data>
 </list>
</Service_list>
Waiting for Response of Agent on port 5005
carina.cheju.ac.kr's Result : Hello Mr. Jwa
Shell>
```

그림 21. XML-RPC 요청 결과 : PHP

```
Next Host name : 203.253.213.169
Total Host Count : 1
Current Host name : carina.cheju.ac.kr
Current Method : rpc
Current Function : echo
Current Data : Hello Mr. Jwa
```

그림 22. 클라이언트 요청에 대한 모니터링 결과

지금까지의 5.1-3의 테스트 결과는 전형적인 RPC와 XML-RPC 서비스 결과들을 검증하기 위해 한 대의 서버를 지정하고 테스트 하였다.

5.4 요청 위임 RPC 자원 서비스 요청 결과

5.1-3절의 서비스 3가지를 혼합한 클라이언트의 요청에 대한 결과이며, RPC의 전형적인 서비스(요청-응답 : 일대 일)에서, 클라이언트는 서비스를 요청 한 후 해당 서버의 응답을 받기 위해 대기하지 않아도 되는 상황을 보여주면서, 클라이언트의 요청 정보 중에 "Total Host Count" 값에 따라 분산 환경의 또 다른 서버로 클라이언트의 요청 정보가 위임됨을 보여주고 있다.

그림 23은 클라이언트의 요청, 그림 24는 모니터링 결과 화면이다. 그리고 그림 24의 "Total Host Count" 값이 3은 서비스를 요청 받은 서버 수를 의미한다. 따라서, 클라이언트는 서비스를 이용할 서버 목록과 메소드(자원)만 명시함으로써, 서버에 종속적이지 않으면서 분산 환경의 서비스를 제공받을 수 있음을 알 수 있게 되었다.

```
Shell>java serialClientMain
Requesting RPC and APP Service .....
<Service_list>
 <final> 203.253.213.169 </final>
 <list> . 이하 생략 . </list>
 <list>  . 이하 생략 .
        <function> sum </function>
   <data> 20,40 </data>
 </list>
 <list>
   <url> sslab.cheju.ac.kr </url>
   <method> app </method>
   <function> reverse </function>
   <data> How are you? </data>
 </list>
</Service_list>
Waiting for Response of Agent on port 5005
sslab.cheju.ac.kr's Result : Hello Mr. Jwa
mincando.cheju.ac.kr's Result : 60
carina.cheju.ac.kr's Result : ?uoy era woH
Shell>
```

그림 23. 요청 위임 RPC 자원 서비스 요청 결과

```
Next Host name : mincando.cheju.ac.kr
Total Host Count : 3
Current Host name : sslab.cheju.ac.kr
Current Method : rpc
Current Function : echo
Current Data : Hello Mr. Jwa

Next Host name: carina.cheju.ac.kr
Total Host Count : 2
Current Host name : mincando.cheju.ac.kr
Current Method : rpc
Current Function : sum
Current Data : 20,40

Next Host name : 203.253.213.169
Total Host Count : 1
Current Host name : carina.cheju.ac.kr
Current Method : app
Current Function : reverse
Current Data : How are you?
```

그림 24. 클라이언트 요청에 대한 모니터링 결과

5.5 결과 분석

구현 결과를 분석하면 표 2와 그림 25, 26과 같다. 구분 항목 중에서 A에서 C까지는 전형적인 RPC나 XML-RPC 서비스와 크게 차이가 없다. 클라이언트의 요청 수가 한 개이기 때문에 해당되는 서비스가 처리되면 종료된다. 하지만, 구분 항목 D인 경우는 클라이언트의 요청 정보 중에 "Total Host Count"의 수만큼 클라이언트의 요청이 서버 간 위임되며, 처리 결과는 해당 서버들의 서비스 처리 후 최종적으로 응답됨을 알 수 있다. 또한 RPC와 XML-RPC 서비스가 통합됨을 보여줬다.

그리고 비동기는 자원 간 서비스에서 서버 간으로, 그리로 동시에 지정할 수 있는 서버 수의 제한도 확장되었다. 응답 시간(Response Time) 측면에서는 개개의 단일 요청 보다는 복수 요청이 클라이언트의 블록 상태를 단축시키는 결과를 보였으며, 부가적으로 서버들은 응답 시간에 구애 받지 않으면서 해당 서비스를 제공할 수 있게 되었다. 그림 25, 26은 연속적인 서비스 요청에 대한 응답 시간 결과이다. 그림 25는 로컬 시스템에서의 서비스 요청에 대한 결과이며, 그림 26은 클라이언트와 서버 간의 서비스 요청 결과이다. 특히 그림 25의 "e"와 그림 26의 "e"는 단일 서버내의 요청 위임 결과이며, 그림 26의 "f"는 다중 서버 간 요청 위임의 응답 시간을 나타내고 있다. 결과적으로 각각의 서비스에 대한 평균치보다 빠르게 처리되고 있음을 알 수 있다.

표 2. 구현결과 분석

구 분	A	B	C	D
RPC	○	○	○	○
XML-RPC	○	○	○	
위 임	이루어지지 않음			○
블록화	존 재			단 축
비동기	자원 간			서버 간
요청 서버 수	1 개			제한없음
서비스 제공	Individual			Integrate

A : RPC 요청 결과(reverse),
B : XML-RPC 요청 결과(sum)
C : XML-RPC 요청 결과(echo)
D : 요청 위임 RPC 자원 서비스 요청 결과

a - echo, b - sum, c - reverse
d - a,b,c 평균, e - 요청 위임
단위 - millisecond, 요청 수 - 20회

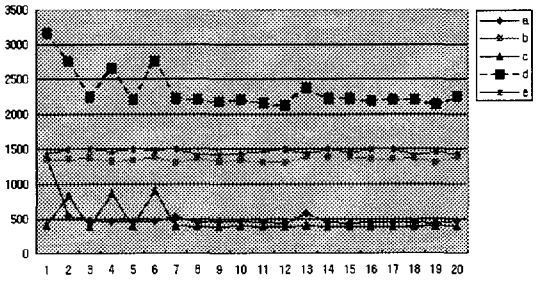


그림 25. 로컬 시스템에서의 응답 시간

a - echo, b - sum, c - reverse
d - a,b,c 평균, e - 요청 위임(단일 서버 내)
f - 요청 위임(다중 서버 간)
단위 - millisecond, 요청 수 - 20회

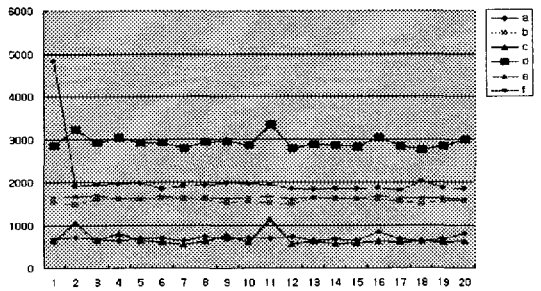


그림 26. C/S 및 다중 서버 간 요청 위임 응답 시간

6. 결 론

본 논문에서는 분산 환경의 자원을 이용하기 위해 전형적인 RPC 구조에서, 블록 상태를 단축시키면서 비동기성을 확장 지원하고, 일반 응용프로그램과

XML-RPC를 통합 제공하며, 클라이언트의 요청을 서버 간 위임하는 RPC 자원 서비스 시스템을 설계하고 구현하였다. 이를 위해 클라이언트의 요청은 XML 문서로 포장되어서 서버로 전송이 되고 서버는 XML-DOM 구조를 분석하여 클라이언트의 요청을 처리하고 또 다른 서버로 요청이 위임하도록 하였다.

본 논문의 구현으로 RPC 서비스와 웹 서비스와의 통합 연동이 가능할 것이며, 특히 서버 환경 또는 시스템 관리자에게는 자체 컴퓨팅 환경의 시스템 체크나 접속자 Log, 응답 시간이 절대적으로 중요하지 않은 프로젝트 등의 연성 실시간 응용, 또는 지속적인 통계 정보를 모니터링 하는 분산 환경 구축을 상당 부분 단순화 시키는데 큰 역할을 수행할 것이라 기대된다.

참 고 문 헌

[1] A.D. Birrell and B.J. Nelson, "Implementing Remote Procedure Calls", ACM Transactions on Computer Systems 2(1), pp.35-39 February 1984.

[2] R. Shrinivasan, "RPC:Remote Procedure Call Protocol Specification Version 2", RFC-1831, August 1995.

[3] J.W. Stamos and D.K. Gifford, "Remote Evaluation", ACM Transactions on Programming Languages and Systems, Vol.12, No.4, pp.537-565, October 1990.

[4] 한승완, 정기문, 박승배, 임형석, "보안 서비스를 고려한 이동 에이전트 모델과 클라이언트-서버 모델의 성능 비교", 정보과학회논문지, Vol.29, No.3, pp.286-298, 2002. 6

[5] J. Baumann, F. Hohl, N. Radouniklis, K. Rothermel, and M. Strasser, "Communication Concepts for Mobile Agents", Proceedings of the 1st International Workshop on Mobile Agents, Berlin(D), Lecture Notes in Computer Science, No. 1219, Springer-Verlag(D), pp. 123-135, April 1997.

[6] R. Gray, D. Kotz, S. Nog, D. Rus, and G. Cybenko, "Mobile agents for mobile computing", Technical Report PCS-TR96-285, Department of computer Science, Dartmouth College, Hanover, 1996.

[7] C.G. Harrison, D.M. Chess, and A. Kershenbaum, "Mobile Agents:Are they a good idea?", Research Report, IBM Research Division T.J. Watson Research Center, March 1995.

[8] J. White, "Mobile Agents White Paper, "General Magic, Sunnyvale, CA, USA, 1998, URL http://www.generalmagic.com/t-echnology/techwhitepaper.html

[9] T-H Chia and S. Kannapan, "Strategically Mobile Agents," Proceedings of First International Workshop on Mobile Agents97, pp.149-161, April 1997.

[10] L. Ismail and D. Hagimont, "A Performance Evaluation of the Mobile Agent Paradigm," Proceedings of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 306-313, November 1999.

[11] A. Puliafto, S. Riccobene, and M. Scarpa, "An analytical comparison of the client-server, remote evaluation and mobile agents paradigms," Proceedings of First International Symposium on Agents Systems and Applications, pp.248-292, October 1999.

[12] M. Straber and M. Schwehm, "A Performance Model for Mobile Agent Systems," Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA97), Vol.II, CSREA, pp.1132-1140, 1997.

[13] Simon St. Laurent, Joe Johnston, Edd Dumbill Forword, "Programming Web Services with XML-RPC", O'reilly, 2001. 6.

[14] XML(eXtensible Markup Language) 1.0, W3C, http://www.w3.org/TR/REC-XML, 1998.

[15] Pokorny, j. "XML functionally", Database Engineering and Application Symposium, 2000 International, 2000, pp. 266-274.

[16] 김명희, 주수종, "분산 실시간 서비스를 위한

CORBA 객체 그룹 플랫폼의 구축", 정보과학회 논문지, Vol. 7, No. 6, 2001. 12.

[17] Dave Winer, UserLand Software Inc, "XML-RPC Specification", "http://www.xml-rpc.com/spec"

[18] 임정은, 윤용익, "XML-RPC를 이용한 문서교환 및 제어용 미들웨어 구조 연구", 정보과학회 학술대회, Vol.28, No.2, pp.514-516, 2001. 10.

[19] Mishra. S, Nija Shi, "Improving the performance of distributed CORBA applications", Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, pp. 36-41.

[20] 윤권섭, 이호섭, 홍충선, "분산 환경에서의 XML 기반의 서비스 관리 구조", 정보과학회 학술대회, Vol.29, No.1, pp. 361-363, 2002. 4.

[21] 이호섭, 홍충선, "분산환경에서 CORBA와 XML의 연동 구조", 정보과학회 학술대회, Vol.28, No.1, pp.424-426, 2001. 4.

[22] 구태완, 정연진, 엄상용, 이광모, "RMI-IIOP를 이용한 CORBA 환경에서의 XML 객체 모델링", 정보과학회학술대회, Vol.28, No.2, pp.529-531, 2001. 10.

[23] W3C, "XML Protocol Working Group Chapter", http://www.w3.org/2000/09/XML-Protocol-Chapter/

[24] W3C, SOAP version 1.2 Working Draft, http://www.w3.org/TR/2001/W-Dsoap/12-20010709/

[25] O'Connell. P, McCrindle. R, "Using SOAP to clean up configuration management", Computer Software and Applications Conference, 2001, COMPSAC 2001, 25th Annual International, 2001, pp. 555-560.

[26] Andrew S.Tanenbaum, "Modern Operating System", Prentice Hall, 2002.

[27] Remote Procedure Calls and Java Remote Method Invocation, IEEE Transactions on Concurrency, Vol. 6 Issue 3, pp. 5-7, 1998.

[28] Andrew Birrell, Bruce Jay Nelson, "Implementing Remote Procedure Calls", TOCS, Vol. 2, pp. 35-59, 1984.

[29] Brett Mclaughlin, "Java & XML, 2nd Edition", O'Reilly, 2001.

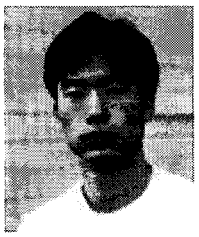
[30] XML-RPC HOWTO, http://xmlrpc-c.sourceforge.net/xml-rpc-howto/xmlrpc-howto.html

[31] XML-RPC Library for Java, http://classic.helma.at/hannes/xmlrpc

[32] W3C, DOM(Document Object Model), http://www.w3c.org/DOM, DOM Level 3 Core specification, 2002.

[33] SAX(Simple API for XML) 1.0 Overview, http://www.saxproject.org/

[34] XML-RPC(Apache XML-RPC), http://xml.-apache.org/xmlrpc



김 정 희

1994년 제주대학교 정보공학과 졸업(학사)
1997년 제주대학교 대학원 정보공학과 졸업(석사)
2002년 제주대학교 대학원 정보공학과 박사과정 수료
1998~현재 제주산업정보대학 컴퓨터정보계열 겸임교수
관심분야 : XML, 데이터베이스, Internet Application. 프로그래밍 언어론, GIS 등

곽 호 영

1983년 홍익대학교 전자계산학과 졸업(학사)
1985년 홍익대학교 대학원 전자계산학과 졸업(석사)
1991년 홍익대학교 대학원 전자계산학과 졸업(박사)
1990~현재 제주대학교 통신컴퓨터공학부 교수
관심분야 : 객체지향 프로그래밍, 프로그래밍 언어론, GIS 등

교 신 저 자

김 정 희 690-756 제주도 제주시 아라1동 산1번지 제주대학교 통신컴퓨터공학부