XML 기술을 이용한 비 동기 RPC 자원 서비스 시스템

Asynchronous Remote Procedure Call Service System using the XML Technology

김 정 희* 곽 호 영**
Jeong-Hee Kim Ho-Young Kwak

요 약

본 논문에서는 XML 기술을 이용한 RPC 비 동기 자원 서비스 시스템을 설계하고 구현한다. 이를 위해 클라이언트의 요청(Request) 정보는 XML의 DOM에 기반하여 XML 문서로 포장된 후 분산 환경의 서버로 전송된다. 서버는 XML-DOM을 받아들일 수 있는 객체를 사용하여 클라이언트의 요청을 일반 응용프로그램과 XML-RPC 서비스로 구분하여 처리한다. 또한 비 동기성을 지원하기 위해 클라이언트의 요청 결과를 바로 전송하지 않고 XML-DOM 구조 내에 저장하며, XML-DOM 정보 내에 또 다른 요청을 해당 서버로 Redirect 되도록 한다. 시스템 구현 결과 일반적인 RPC 서비스와 XML-RPC 서비스가 통합 되었으며, 클라이언트의 요청이 서버들 사이로 Redirect 되는 비 동기성이 구현 되었고, 시스템 실행 환경은 전통적인 RPC 요청 보다 단순화되었다.

Abstract

The purpose of this paper is the design and implementation of asynchronous remote procedure call (RPC) resource service system using the XML technology. For this purpose, the request information of client is encoded into XML document based on XML-DOM, and transferred to server. Server classifies the client requests into general application program and XML-RPC service using the object which can deal with the XML-DOM. In addition, server saves the request result of client in XML-DOM structure not transmitting it immediately in order to support asynchronous service, and makes the client request redirected to another request server in XML-DOM information. As a result, general RPC and XML-RPC services were attained and client request was redirected tc servers, and the execution environment was simplified compared to common RPC.

Keyword : asynchronous remote procedure call, XML-RPC, RPC

1. 서 론

웹(WWW)은 현재 우리 생활에 가장 큰 영역을 자리 잡으면서 비즈니스 영역으로 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. 이러한 흐름 속에서 XML[1,2] 기술을 활용하여 응용 프로그램간 통신 수단으로 웹을 이용하도록 분산 환경을 어떻게 더욱 발전된 모습으로 지원해 나갈 수 있을 것인지를 두고 다양한 시도가 이루어지고 있다. 분산 환경에서 XML기술을 활용하여 가장 중심적인 역할을 수행할 수 있는 분야는 데이터를 표현하여 모든 곳에서 통합화/표준화된 채널을 통하여 모든 사용자들이 접근 가능하도록 하는 기반 구조이다[3].

하지만, 최근 널리 사용되고 있는 분산 기술인 Microsoft의 COM/DCOM, OMG의 CORBA[4], JAVA Bean 등은 그 기술들 간 서로 호환되지 않으며, 객

* 정 회 원 : 제주산업정보대학 컴퓨터정보계열 겸임교수
carina@cheju.ac.kr(제1저자)
** 정 회 원 : 제주대학교 통신컴퓨터공학부 교수
kwak@cheju.ac.kr(공동저자)

체기반 분산 컴퓨팅 환경 하에서의 클라이언트는 ORB(Object Request Broker) 소프트웨어에 의존적으로 서버와 데이터 교환을 하기 때문에 CORBA, DCOM, JAVA RMI 등의 ORB 등은 상호운영성의 제약과 방화벽에 의한 메시지 차단이 되며, 원격 프로시저를 호출하지 않고 웹과의 연동 시에 원격지 응용 프로그램과의 상호작용이 되지 않아 서로 통신을 할 수 있는 표준 프로토콜이 필요시 되고 있다[5,6,7]. 따라서, 현재 XML이라는 표준 데이터 포맷을 이용하여 분산 환경의 표준화된 접근 채널을 제공하기 위해 위에 기술된 분산 기술들보다 계산 능력이나, 통신 대역폭, 그리고 효율성 면에서 뛰어난 IIOP[7]나 RPC(Remote Procedure Call)를 사용하고 있는 추세이며, 또한 XML은 Publishing과 Data Exchange라는 두 가지 주요한 Application 영역과 프로토콜로 XML-RPC[7,9], SOAP[10,11], WDDX[3] 등이 제공되어 응용 프로그램 간 데이터 교환을 지원하면서 단순히 데이터 교환뿐 아니라 그에 적합한 처리를 할 수 있는 응용 프로그램까지 데이터를 전달할 수 있게 되었다[3]. 하지만 새로운 분산 환경의 패러다임을 위해서 XML 기술과 표준 프로코콜이 지원되고 있지만 분산 환경상의 서비스의 사용은 요청과 제공(Reply)이라는 1대 1의 종속적 관계를 여전히 유지하고 있다.

따라서, 본 논문에서는 이러한 종속적인 분산 환경의 서비스 요청을 해결하기 위하여, 비 종속성을 지원하는 분산 환경 자원 서비스 시스템을 설계하고 구현한다. 구현된 시스템이 사용하는 프로토콜은 자원 서비스 요청에 따라 TCP/IP 기반과 HTTP 기반을 모두 가능하도록 하였다. 특히 전통적인 클라이언트/서버(Request-Reply) 구조를 탈피하여, 요청 받은 서버의 서비스가 끝나면 또 다른 서버로 클라이언트의 요청을 전달할 수 있는 기능을 갖도록 하였다. 또한 요청한 자원 서비스에 대한 결과는 XML-DOM 객체 구조로 되어 있고, 해당 서버 간 Redirect 시 함께 전달되면서 정보를 구축하게 된다.

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

2. 관련연구

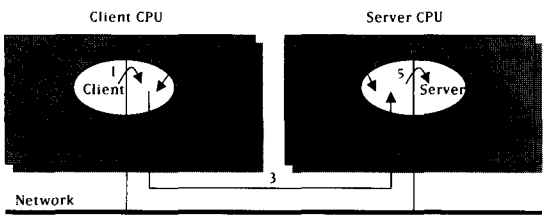
분산 기술은 웹과의 호환성을 직접 가지지 못하므로 웹에서 데이터 교환에 사용하려는 어플리케이션들은 별도의 프로토콜을 지원해야 한다. 이러한 컴포넌트 기술과 웹과의 호환성 문제를 해결하기 위해서 XML을 이용하고자 하는 많은 노력이 이루어져 왔으며, XML은 시스템간의 상호 운영성(Interoperability) 측면에서 볼 때 기존의 복잡했던 문제를 상당 부분 단순화시키는데 큰 역할을 수행하고 있다

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, 원격진료, 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[12].

2.2 RPC(Remote Procedure Call)

Birrell과 Nelson에 의해 제안된 RPC[13]는 클라이언트와 서버간 디스패치(Dispatch) 방식으로 운영되는 서비스이다. 즉, 다른 CPU에 위치한 프로시저를 호출하는 프로그램을 허용하는 것이다. 머신



(그림 1) 원격 프로시저 호출 단계

(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 필요 되어진다[14].

반면 통합 서비스를 위한 RPC는 데이터 전송 과정에서 데이터의 구조를 표현하는 인코딩에 문제가 있다. 그림 1은 일반적인 RPC 호출 단계이며, 요청과 응답의 관계는 1대 1이다.

2.3 SOAP

분산 컴포넌트 기술인 OMG(Object Management Group)의 CORBA(Common Object Request Broker Architecture)와 웹과의 연동을 위해 SOAP(Simple Object Access Protocol)을 이용한 SCOAP(Simple Corba Object Access Protocol)[15]이 제안되어 있는 상태이다. SOAP이라는 프로토콜은 Microsoft사에 의해 제안되었으며, 플랫폼에 독립적으로 인터넷에 분산된 서비스, 오브젝트, 또는 서버에 접근할 수 있도록 HTTP 프로토콜 위에서 구현된 XML 프로토콜이다. SOAP에서는 HTTP 혹은 SMTP 상에서 단순히 서비스 요청/응답에 대하여 표준 XML 문서 구조를 정의하여 서비스 제공자와 사용자간에 상호 교환하도록 구성되어 있다. 서비스를 요청하는 문서에는 사용할 서비스의 정보와 입력값을 채워서 보내주면 서비스 제공자는 이를 분석하여 해당 서비스를 수행한 후 그 결과 값을 응답 문서에 넣어서 돌려주는 방식이다. 현재 SOAP에서 제공되는 프로토콜은 그 효율성에 기대가 크지만 표준화가 계속 진행 중인 상태이기 때문에 구현 적응성에 있어서 전체 효율성이 낮다[16].

2.4 XML-RPC

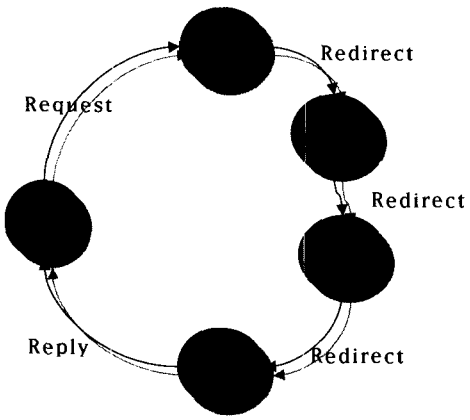
XML-RPC(XML-Remote Procedure Call)[18,19,20]는 XML 프로토콜의 한 종류로 XML을 이용하여 원격지 머신(Machine)에 대한 함수 호출을 캡슐화 한 후 원격지 머신에 XML 문서를 전달하고 그 결과 값으로 XML 문서를 돌려 받는다. 그림 2는 XML-RPC에서 Request 정보의 인코딩 예를 보여준다.

XML의 출현으로 전통적으로 RPC(원격 프로시저 호출, Remote Procedure Call)를 사용하는데 있어서 가장 큰 방해물 이였던 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[21].

또한 DOM[22]과 SAX[23]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 프로

```
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



(그림 3) 제안한 시스템의 동작 개요

젝트에서 매우 효과적이고 좋은 성능을 제공한다는 것이 증명되었다[21].

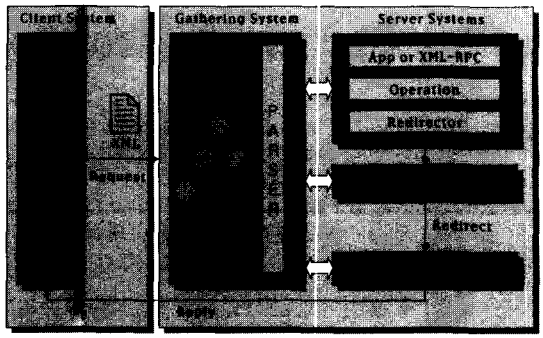
3. 비 동기 RPC 자원 서비스 시스템

3.1 제안 시스템 개요

본 논문에서 제안하는 시스템은 분산 환경의 RPC 서비스와 XML-RPC 서비스를 처리함에 있어서, 전형적인 요청-응답의 종속 관계(1대 1)를 XML 기술을 이용하여 클라이언트의 요청이 Redirect 되도록 하는데 있다. 그럼으로써 비 동기성을 지원하도록 한다.

전형적으로 RPC는 클라이언트와 서버 시스템 사이에서 요청-응답을 기반으로 운영된다. 즉, 클라이언트는 요청 메시지를 보내고 서버는 그에 해당하는 연산을 수행하고 응답 메시지를 클라이언트로 되돌리는 구조라는 특성으로 클라이언트는 계속해서 자신의 실행을 수행하기 전에 응답 메시지를 기다려야 하는(Wait) 부담이 존재하게 되지만, 제안하는 시스템에서는 클라이언트는 요청 후에 서버의 응답을 기다려야 하는 상황에서 자유롭게 된다. 그림 3은 제안한 시스템의 동작 개요이다.

그림 3과 같은 동작이 이루어지기 위해서 시스템은 각 서버가 처리한 결과를 저장하고, 또 다른



(그림 4) 시스템 구조

서버로 클라이언트의 요청을 전달해야 하는지, 아니면 클라이언트로 모든 처리 결과를 응답해야 하는데, 본 논문에서는 이를 위해 XML-DOM 구조를 사용하는 Gathering System 모듈을 구현함으로써 해결한다. 즉, 최초 요청 시 XML-DOM 객체 클래스를 생성하고, XML-DOM 내용에는 서버 URL과 일반 응용프로그램 또는 XML-RPC 서비스를 지정하고, 요청할 메소드와 데이터가 포함되며, 추가적으로 해당 서버가 처리한 결과가 XML-DOM 내에 저장되어 전달되도록 한다.

따라서 각각의 서버들은 XML-DOM을 Parsing 하고 클라이언트의 요청 정보를 처리하게 된다. 제안하는 시스템 구조는 그림 4와 같고, 서버 동작 환경은 Gathering System을 유기적으로 사용하며, 서버 자체는 일반 응용프로그램과 XML-RPC를 처리하고 Redirector에 의해 또 다른 서버로 클라이언트의 요청을 전달하는 구조가 된다.

3.2 클라이언트 시스템(Client System)

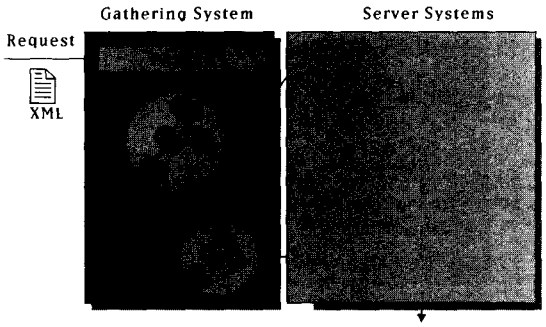
클라이언트는 서비스를 요청할 서버의 URL 주소(targetURL), 일반 응용프로그램 서비스 요청인지 XML-RPC 서비스 요청인지를 구분하는 Service, 수행될 서버의 Function과, Function이 사용할 데이터(Data)를 인자로 갖는 함수(그림 5)를 이용하여 XML 요청 문서를 생성한다. 그 결과 그림 6과 같이 XML 문서로 포장되어 분산 환경의 Server로 전달된다.

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

(그림 5) 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 6) 생성된 Request XML 문서의 예



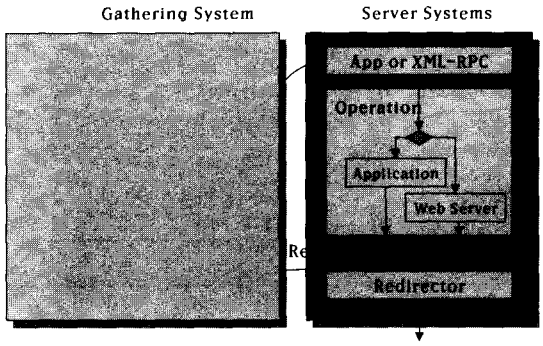
(그림 7) Gathering System

3.3 Gathering System(GS)

Gathering System은 클라이언트의 요청에 대한 서버의 처리 결과를 저장한다. GS는 각각의 서버 환경에서 동작되며, XML Parser와 XML-DOM을 사용한다. 분산 환경의 최종 서버가 가지는 DOM 정보는 클라이언트의 전체 요청 처리 결과를 인코딩 한다. GS의 구조는 그림 7과 같다.

3.4 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 Request를 전달할 것인지를 XML-DOM 정보에 따라 판단한다. 일반 응용프로그램의 처리는 전형적인



(그림 8) Server Systems

RPC 처리를 말하며, XML-RPC 처리는 HTTP 기반의 메시지 기반 처리를 말한다. 본 논문은 통합을 위해 일반적인 RPC 처리에 추가적으로 XML 기술을 이용한 XML-RPC 서비스도 지원하도록 하며, 이는 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 XML-DOM에 추가로 저장하고 Redirector에 의해 또 다른 서버로 클라이언트의 요청을 전달하거나 최초 요청 클라이언트로 DOM 정보를 전송하게 된다. 그림 8에서 서버 시스템의 내부 구조를 보인다.

4. 시스템 구현

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

- ▶ IBM PC Pentium (III)
  - · Windows 2000 Server
  - · J2SDK 1.4.0.02
  - · Xerces Parser for XML
- ▶ helma.xmlrpc : XML-RPC Library
- ▶ Testing Program
  - · ASP, PHP : XML-RPC Program
  - · JAVA : Application Program

(표 1) 클라이언트와 서버 시스템

	Client	Server
OS	2000 Server	2000 Server
DNS		carina.cheju.ac.kr
Method		reverse(Java) : AP echo(PHP) : XML-RPC SUM(ASP) : XML-RPC

```
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("carina.cheju.ac.kr");
  /* XML-RPC */
  a.appendAgent("carina.cheju.ac.kr","rpc","echo",-
                "Hello Mr. Jwa");
  a.appendAgent("carina.cheju.ac.kr","rpc", -
                                "sum","20,40");
  /* Application */
  a.appendAgent("carina.cheju.ac.kr","app","reverse",-
                "carina");
  PrintDocument.Print(a.theDocument.-
                            getDocumentElement());
  SerialClient.sendToHost(a,a.getFirstURL());
}
```

(그림 9) XML-DOM 내의 Request 정보

4.1 클라이언트 시스템(Client System)

클라이언트 시스템은 분산 환경에 존재하는 서버들의 자원들을 이용하기 위해 3.2 절에서 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 9와 같이 처리하였다.

4.2 Gathering System

서버들이 해당 요청을 처리하고 그 결과를 저장할 때 유기적으로 사용되는 시스템이다. 서버의 처리 결과는 3.3절에서 설명한 바와 같이 XML-DOM 구조내에 저장 되도록 하였다. 따라서 이를 위한 XML-DOM 객체와 여러 가지 메소드들이 필요한데, 이들은 루트 엘리먼트 생성, 자식 엘리먼트 추가, 결과를 저장하기 위한 결과 저장, 최종 목적지 정보, 전체 결과 리스트 등에 관련된 메소드들이다. 그림 10에 이들 메소드들을 정리하였다.

4.3 서버 시스템(Server Systems)

본 논문에서는 아파치에서 진행하는 Java 기반의 Xerces Parser[24]를 사용하여 구현하였으며, 서버 시스템은 이를 기반으로 XML-DOM 구조의 내용을 Parsing하여 클라이언트의 요청을 하나씩 처리하도록

```
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 결
       과) */ }
public String getFirstURL()
  { /* 또 다른 서버로 클라이언트의 request를 전달할
       값 */ }
public String getFinalURL()
  { /* 요청한 클라이언트 주소 파악 */ }
```

(그림 10) Gathering System의 메소드

```
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 전달 }
  else
  { 최종 목적지 주소로 XML-DOM 정보 전달 }
 }
```

(그림 11) Server System의 처리

하였다. 그 과정은 일반 응용 프로그램(Application) 서비스인지 XML-RPC 서비스인지 판단하여 제공하며, 또한 서비스 결과를 XML-DOM에 저장하고 클라이언트의 요청을 또 다른 서버로의 전달(Redirect) 여부를 판단한다. 그림 11에 필요한 메소드를 정리하였다.

5. 실행 결과

클라이언트 시스템(Client System)과 서버 시스템(Server System) 모두 동일 시스템을 사용했으며, 검증을 위해 클라이언트의 Request를 4가지로 분류하여 테스트하였다.

5.1 RPC 요청 결과(reverse)

지정한 서버의 일반 응용 프로그램(JAVA Program) 서비스를 요청한 결과이다. 그림 12의 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

```
Shell>java serialClientMain
Requesting RPC and APP Service via Agent.....
<Service_list>
 <final> carina.cheju.ac.kr </final>
 <list>
   <url> carina.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
carina.cheju.ac.kr's Result : ?uoy era woH
Shell>
```

(그림 12) RPC 요청 결과(Reverse)

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

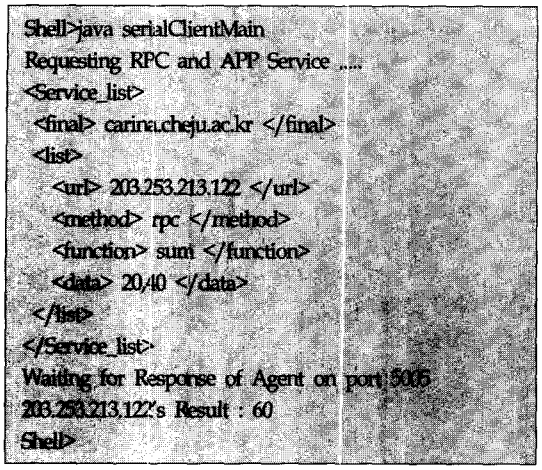
(그림 13) 클라이언트 요청에 대한 모니터링 결과

서버에게만 요청한 결과이다.

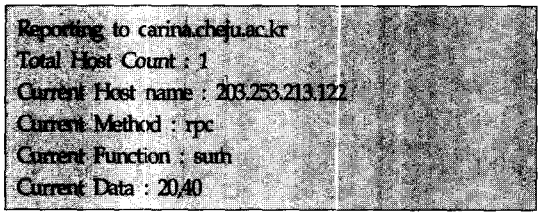
그림 13은 서버 쪽에서 클라이언트의 요청을 분석한 것을 보여준다. "Next Host name"은 처리 후 이동할 URL이며, "Total Host Count"는 클라이언트가 서비스를 지정한 서버 수를 말하며, "Current Host name"은 서버를 말하며, "Current Method"는 요청한 RPC를, "Current Function"은 서버가 실행할 함수명이며, "Current Data"는 넘어오는 인자값을 나타낸다.

5.2 XML-RPC 요청 결과(sum)

그림 14는 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



(그림 14) XML-RPC 요청 결과 : ASP



(그림 15) 클라이언트 요청에 대한 모니터링 결과

산 후 넘겨주었다. 결과 화면의 정보는 5.1절을 참고하기 바란다. 그림 15는 서버에서 모니터링 한 화면이다.

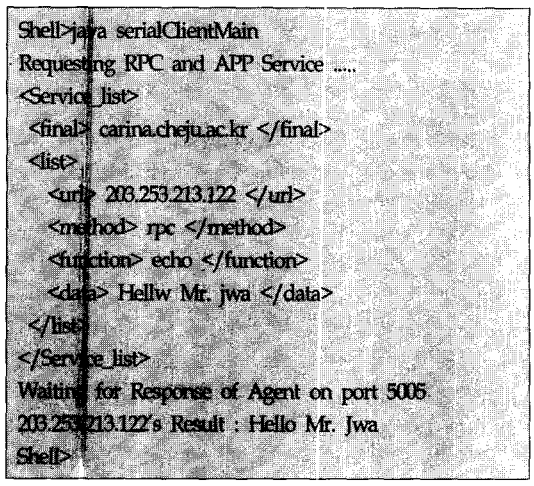
5.3 XML-RPC 요청 결과(echo)

지정한 서버의 웹 서버에게 XML-RPC를 이용한 클라이언트의 요청 결과이다. 결과 화면은 그림 16, 모니터링 화면은 그림 17이며, 상황은 5.2와 동일하다. 단, 요청한 서비스만 PHP 프로그램일 뿐이다.

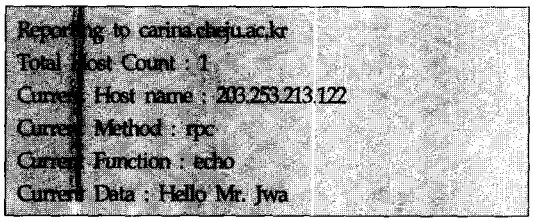
지금까지의 5.1, 5.2, 5.3의 테스트 결과는 전형적인 RPC와 XML-RPC 서비스 결과들을 검증하기 위해 "Total Host Count"를 하나만 지정하고 테스트하였다.

5.4 비 동기 RPC 자원 서비스 요청 결과

일반 응용프로그램(RPC), XML-RPC의 ASP와 PHP



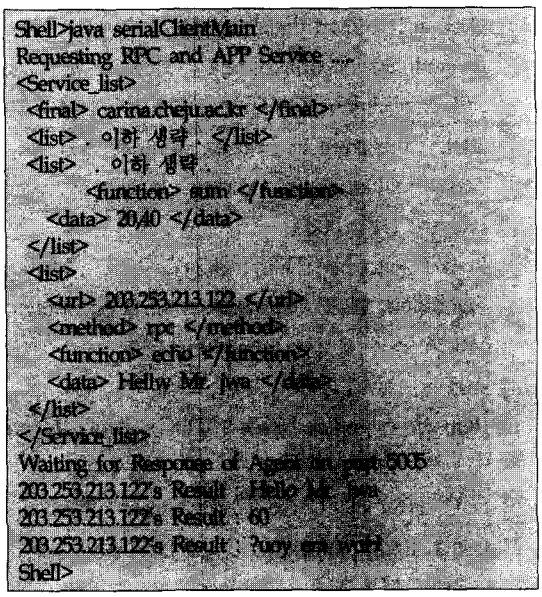
(그림 16) XML-RPC 요청 결과 : PHP



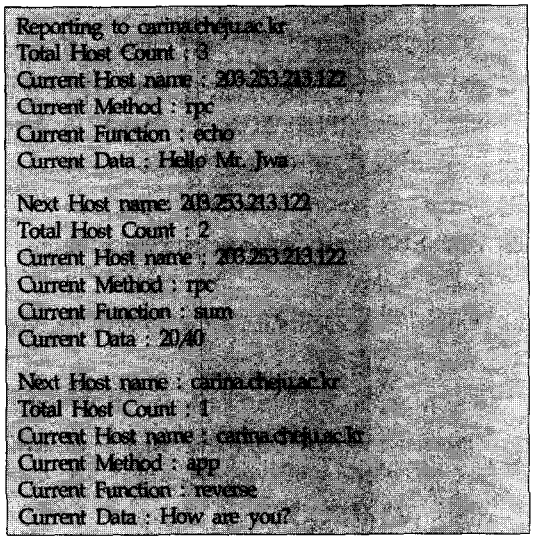
(그림 17) 클라이언트 요청에 대한 모니터링 결과

서비스 3가지를 혼합한 클라이언트의 요청에 대한 결과이며, RPC의 전형적인 서비스(요청-응답 : 1대 1)에서, 클라이언트는 서비스를 요청 한 후 해당 서버의 응답을 받기 위해 대기하지 않아도 되는 상황을 보여주고 있다. 이는 클라이언트의 요청 정보 중에 "Total Host Count" 값에 따라 분산 환경의 또 다른 서버로 클라이언트의 요청 정보가 Redirect 됨을 뜻한다.

그림 18은 클라이언트의 요청 결과 화면이며, 그림 19는 모니터링 결과이다. 그리고 그림 18의 "Total Host Count" 값이 3으로 서비스를 요청한 server 수가 3이라는 것을 의미한다. 따라서, 클라이언트는 서비스를 이용할 서버 목록과 메소드(자원)만 명시함으로써, 서버에 종속적이지 않고 분산 환경의 서비스를 제공받을 수 있음을 알 수 있게 되었다.



(그림 18) 비 동기 RPC 자원 서비스 요청 결과



(그림 19) 클라이언트 요청에 대한 모니터링 결과

(표 2) 구현결과 분석

Test	RPC	XML-RPC	Redirect	서비스 유형
5.1	◉	◉	none	독립적
5.2	◉	◉		
5.3	◉	◉		
5.4	▣	▣	제한 없음	통합

(기존과 동일 : ◉, 제안 시스템 : ▣)

5.5 결과 분석

구현 결과를 분석하면 표 2와 같다. Test 항목 중에서 5.1에서 5.3까지는 전형적인 RPC나 XML-RPC 서비스와 크게 차이가 없다. 클라이언트의 요청 수가 1 개이기 때문에 해당되는 서비스가 처리되면 종료된다. 하지만, Test 항목 5.4인 경우는 클라이언트의 요청 정보 중에 "Total Host Count"의 수만큼 클라이언트의 요청이 서버 간 Redirect 되며 클라이언트는 최초 요청 후 대기에서 자유롭게 되었고(Asynchronous), 처리 결과는 해당 서버들의 서비스 처리 후 최종적으로 응답됨을 알 수 있다. 또한 RPC와 XML-RPC 서비스가 통합됨을 보여줬으며, 또한 부가적으로 서버들은 응답 시간(Response Time)에 구애받지 않고 해당 서비스를 제공할 수 있게 되었다.

6. 결 론

본 논문에서는 분산 환경의 자원을 이용하기 위해 전형적인 RPC 구조를 XML 기술을 활용하여 클라이언트와 서버의 요청-응답 관계가 비 동기성을 지원하면서, 일반 응용프로그램과 서버의 웹 서비스를 XML-RPC를 이용하여 접근할 수 있는 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 전달(Redirect)될 수 있도록 하였다. 또한 서비스 유형의 통합 및 XML-RPC 서비스로 인해 분산 환경의 분산 객체 이용에 따른 실행환경에 자유롭도록 할 수 있음을 보였다.

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, 또는 지속적인 통계 정보를 모니터링 하는 분산 환경 구축을 상당

부분 단순화시키는데 큰 역할을 수행할 것이라 기대된다.

향후 연구는 분산 환경의 Naming Service, 그리고 자원에 대한 위치 투명성(Location Transparency), Fault tolerance등을 구현한 시스템에 추가하는데 있다.

참 고 문 헌

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

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

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

[4] CORBA Specification, Ver 2.4.2, OMG. "http://www.omg.org".

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

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

[7] 구태완, 정연진, 엄상용, 이광모, "RMI-IIOP를 이용한 CORBA 환경에서 의 X-ML 객체 모델링", 정보과학회 학술대회, Vol.28, No.2, pp. 529~531, 2001. 10.

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

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

[10] W3C, SOAP version 1.2 Working Draft, http://www.w3.org/TR/2001/WD-soap/12-20010709/.

[11] 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.

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

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

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

[15] SCOAP(Simple Corba Object Access Protocol), OMG, ftp://ftp.mog.org/pub/orbos/00-09-03.pdf"

[16] George M. Doss, Wordware Publishing, Inc. "CORBA Developer's Guide with XML".

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

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

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

[20] XML-RPC Specification, http://www.xmlrpc.com/spec.

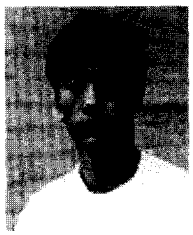
[21] Simon St. Laurent, Joe Johnston, Edd Dumbill, "Programming Web Services with XML-RPC, O'reilly, 2001. 6.

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

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

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

◐ 저 자 소 개 ◑



김 정 희

1994년 제주대학교 정보공학과(학사)
1997년 제주대학교 대학원 정보공학과(석사)
2002년 제주대학교 대학원 정보공학과 박사과정
1998년~현재 : 제주산업정보대학 컴퓨터정보계열 겸임교수
관심분야 : XML, 데이터베이스, Internet Application, 프로그래밍 언어론, GIS
E-mail : carina@cheju.ac.kr

곽 호 영

1983년 홍익대학교 전자계산학과(학사)
1985년 홍익대학교 대학원 전자계산학과(석사)
1991년 홍익대학교 대학원 전자계산학과(박사)
1990년~현재 : 제주대학교 통신컴퓨터공학부 교수
관심분야 : 객체지향 프로그래밍, 프로그래밍 언어론, GIS
E-mail : kwak@cheju.ac.kr