ONC RPC 표준을 지원하는 Java RPC의 설계 및 구현

이국희°, 한욱신, 이민재, 황규영
한국과학기술원 전산학과/첨단정보기술연구센터

Design and Implementation of a Java RPC Supporting ONC RPC Standard*

Kook-Hee Lee, Wook-Shin Han, Min-Jae Lee and Kyu-Young Whang
Department of Computer Science
and
Advanced Information Technology Research Center
Korea Advanced Institute of Science and Technology

요 약

로컬 환경의 함수 호출 기능을 분산 환경으로 확장한 RPC는 분산 시스템 개발을 위한 필수 패키지이다. 본 논문에서는 ONC RPC 표준을 지원하는 RPC를 Java 프로그래밍 언어를 사용하여 설계하고 구현하였다. 본 논문의 Java RPC를 이용하면 웹 환경을 지원하는 분산 시스템을 쉽게 개발할 수 있다. 본 논문의 Java RPC는 외부 데이터표현의 표준 규약인 XDR 프로토콜을 처리하는 라이브러리 클래스와 TCP/IP 기반의 RPC 라이브러리 클래스를 제공한다. 또한 자동으로 RPC 응용 클래스를 생성해주는 RPCGEN 유틸리티를 제공하여, Java 프로그래밍 언어를 이용한 RPC 응용 프로그램을 손쉽게 작성할 수 있도록 한다.

1. 서 론

인터넷이 급속히 발전해 감에 따라 인터넷에 연결된 수많은 컴퓨터들을 보다 편리하게 사용할 수 있게 해주는 분산 시스템의 중요성이 높아지고 있다. 그러나 분산 환경에서의 응용 프로그램 작성은 로컬 환경에 비해 난이도가 높아 응용 프로그래머들에게 어려움을 주고 있다. 따라서 로컬 환경에서와 같이 쉽게 분산 응용프로그램을 작성할 수 있도록 해주는 RPC는 필수적이라 할 수 있다.

RPC는 로컬 환경에서의 컨트롤 및 데이터의 전달 방법인 함수 호출(Procedure Call) 기능을 분산 환경으로 확장하여, 다른 컴퓨터에 위치한 함수를 호출할 수 있도록 해서 네트워크를 통해 컨트롤과 데이터를 전달할 수 있도록 해주는 분산 패키지이다. RPC는 개념이 간결하여 이를 이용해 쉽게 분산 시스템을 개발할 수 있고, 구현방법이 복잡하지 않아서 네트워크를 통해 빠르게 정보를 주고 받을 수 있다는 장점을 갖는다[Bir84].

본 논문에서는 현재 분산 시스템 구축에 널리 쓰이고 있는 ONC RPC 표준을 지원하는 RPC 패키지를 Java 프로그래밍 언어를 사용하여 설계하고 구현하였다. Java를 개발언어로 선택한 이유는 Java 언어가 웹 환경을 잘 지원하므로 Java RPC가 제공되면 웹 환경을 지원하는 분산시스템을 쉽게 개발할 수 있다는 장점을 갖기 때문이다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구로서 ONC RPC와 XDR 프로토콜에 대해 설명한다. 제 3 장에서는 본 논문에서 설계하고 구현한 ONC RPC 지원 Java RPC에 대해 설명한다. 끝으로 제 4 장에서는 결론을 내리고 앞으로의 연구 방향을 제시한다.

2. 관련 연구

본 장에서는 본 논문에 관련된 연구들을 설명한다. 제 2.1 절에서는 본 논문에서 설계하고 구현한 Java RPC가 지원하는 ONC RPC 표준에 대해 설명한다. 그리고 제 2.2 절에서는 사용 컴퓨터 기종에 관계없이 데이터를 자유로이 주고 받을 수 있게 해주는 외부 데이터 표현규약(XDR Protocol: eXternal Data Representation Protocol)에 대해 설명한다.

2.1 ONC RPC

RPC(Remote Procedure Call)는 기존의 함수 호출(Procedure Call) 기능을 확장하여 네트워크 상의 다른 컴퓨터에 위치한 함수를 호출할 수 있는 기능을 제공하는 패키지이다[Bir84]. 예를 들어 컴퓨터 A의 프로세스가 RPC를 통해 컴퓨터 B의 함수를 호출 한 경우, 컴퓨터 A의 프로세스는 기존의 함수 호출에서와 마찬가지로 함수 호출결과가 반납될 때까지 대기하게 된다. 그리고 함수 호출 시 입력한 인수와 함수 호출결과의 반납을 통해 컴퓨터 A와 컴퓨터 B는 메시지 패싱과 같은 복잡한 작업을 수행하지 않고도 정보를 서로 교환할 수 있다[Sun95].

이와 같이 RPC는 기존 함수 호출과 같은 간단한 형식을 갖으면서도 정보교환을 자유로이 수행할 수 있는 강력한 기능을 제공하므로, 많은 분산시스템에서 채택되어 사용되고 있다. 대표적인 예로 현재 각 분야에서 널리 쓰이고 있는 미국 SUN사의 SOLARIS 운영체제의 네트워크 파일 시스템 NFS(Network File System)가 RPC를 기반으로 하는 것을 들 수 있다.

RPC을 기반으로 하는 응용 프로그램의 구조는 그림 1과 같다[Bir84].

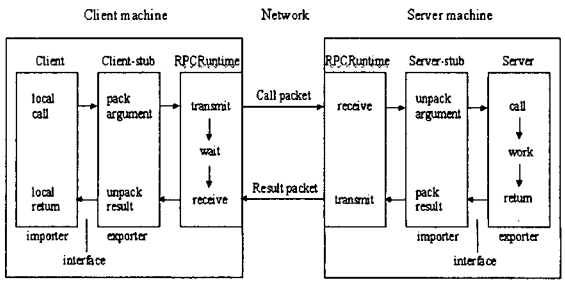

그림 1. RPC 기반 응용 프로그램의 구조

* 본 연구는 웹기반 정보검색 OODBMS를 위한 Web-DBMS 통합기술의 개발 과제를 통하여 과학기술부의 지원을 받았음

먼저 클라이언트를 살펴보면, 사용자에게 기존의 함수 호출과 같은 인터페이스를 제공하는 user-stub 과 네트워크를 통한 통신을 담당하는 RPC-Runtime 으로 구성된다. Client-stub 에서는 사용자 입력 인수의 표준데이터형식으로의 변환과 표준데이터형식으로 반납된 호출 결과를 클라이언트 표현 형식으로 변환하는 기능을 담당한다. RPC-runtime 에서는 표준데이터형식의 입력인수를 네트워크를 통해 서버에게 전달하고 호출결과를 네트워크를 통해 전달 받는다.

다음으로 서버를 살펴보면 네트워크를 통해 입력인수를 전달 받고 호출결과를 네트워크를 통해 전달하는 RPC-runtime 과 표준데이터형식의 입력 인수를 서버 표현 형식으로 변환하고 호출결과를 표준데이터형식으로 변환하는 server-stub 으로 구성된다.

이와 같은 구조를 갖는 RPC 응용 프로그램을 작성할 때 client-stub 과 server-stub 은 RPC 패키지에 포함되어 제공되는 RPCGEN 유틸리티를 통해 자동으로 생성되므로, 사용자는 기존 함수 호출과 같은 방식의 프로그램 만을 작성하면 된다. 따라서 RPC 패키지를 사용하면, 분산환경의 응용프로그램을 손쉽게 작성할 수 있고 로컬환경에서 작성된 응용프로그램을 분산환경으로 간단히 이식할 수 있다.

현재까지 제안되어 널리 사용되고 있는 RPC 표준으로는 미국 SUN 사에서 제안한 ONC RPC[Sun95][Sri95a], OSF 에서 제안한 DCE RPC, 미국 Xerox 에서 제안한 Courier RPC[Xer81] 등이 있다. 본 논문의 Java RPC 는 이러한 RPC 표준 중 가장 많이 사용되고 있는 ONC RPC 표준을 만족하도록 설계하고 구현하였다.

2.2 XDR 프로토콜

본 절에서는 공간 질의를 효율적으로 지원하는 공간 액세스 방법의 한 종류인 MLGF 의 구조와 특징에 대해 자세히 설명한다.

본 절에서는 외부데이터표현 규약의 한 종류인 XDR 프로토콜의 특징과 데이터 표현 방법에 대해 설명한다. XDR 프로토콜은 미국 SUN 사에서 제안한 이기종 컴퓨터간 데이터 교환 시 사용되는 데이터 부호화(encoding) 방식의 표준이다[Sri95b].

XDR 프로토콜은 프로그래밍 언어, 운영체제, 컴퓨터의 종류에 관계없이 데이터를 표준형식으로 표현할 수 있게 함으로써 이식성이 뛰어난 프로그램을 작성할 수 있도록 한다.

XDR 이 지원하는 타입의 종류에는 signed integer, unsigned integer, enumeration, boolean, signed hyper integer, unsigned hyper integer, floating point, quadruple-precision floating point, fixed-length opaque data, variable-length opaque data, counted byte strings, fixed-length array, variable-length array, structure, discriminated union, void, constant, typedef, optional-data 가 있다.

XDR 프로토콜은 이러한 지원 타입 각각에 대해 표준 부호화(encoding) 방식을 정의하고 있다. 이와 같이 다양한 타입 지원을 통해 사용자가 요구하는 대부분의 데이터를 저장할 수 있다. 또한 XDR 프로토콜은 C 언어와 유사한 데이터표현언어를 사용하여 복잡한 구조를 갖는 데이터 형식을 간결하게 표현할 수 있도록 한다.

본 논문에서 지원하는 ONC RPC 표준에서도 외부데이터표현 형식으로 XDR 프로토콜을 채택하여, 이기종 컴퓨터들로 구성되는 분산환경에서도 문제없이 사용 가능하도록 하고 있다. ONC RPC 의 클라이언트에서 다른 컴퓨터에 위치한 서버의 함수를 호출할 때, 입력인수를 XDR 형식으로 변환하여 보냄으로써 이기종 컴퓨터에 위치한 함수도 호출 가능하게 된다. 특히 RPC 메시지에 포함된 헤더 정보 역시 XDR 형식을 사용하여 이식에 문제가 없도록 하고 있다.

3. Java RPC 의 설계 및 구현

본 장에서는 본 논문에서 설계하고 구현한 ONC RPC 지원 Java RPC 에 대해서 자세히 설명한다.

본 논문에서는 구현 시 사용한 프로그래밍 언어인 Java 의 객체지향 속성을 이용해 RPC 기능들을 모듈화 하였다. 먼저 기존의 RPC 패키지를 분석하여 기능 별로 XDR 프로토콜 처리 모듈, 통신처리 모듈, RPCGEN 유틸리티 모듈로 구분한 후, 각 기능 별로 기본 클래스를 정의하여 세부 기능을 갖는 클래스들은 기본 클래스를 상속 받아 구현되도록 하였다. 이를 통해 간단하면서도 확장이 용이하도록 RPC 패키지를 설계하고 구현하였다.

본 장의 구성은 다음과 같다. 먼저 제 3.1 절에서는 ONC RPC 표준에서 채택하여 사용하는 XDR 프로토콜을 처리하는 클래스의 설계와 구현에 대해 설명한다. 제 3.2 절에서는 RPC 패키지의 통신을 담당하는 클래스의 설계와 구현에 대해 설명한다. 마지막으로 제 3.3 절에서는 RPC 응용프로그램 작성을 도와주는 RPCGEN 유틸리티 구현을 위한 클래스들의 설계와 구현에 대해 살펴본다.

3.1 XDR 처리 클래스

본 절에서는 XDR 처리 클래스의 설계와 구현에 대해 자세히 설명한다.

본 논문에서 XDR 처리를 위해 구현한 클래스에는 XDR 형식으로 변환된 데이터의 입출력을 관리하는 XDRStream, XDRInputStream, XDROutputStream 과 각 타입별 XDR 변환을 담당하는 XDRType 과 XDRint, XDRshort, XDRfloag, XDRstring, 등이 있다.

먼저 XDR 형식 데이터의 입출력을 관리하는 클래스들에 대해 살펴본다. XDR 형식 데이터의 입출력 관리 클래스의 기본 클래스로 XDRStream 을 정의하였다. XDRStream 클래스에는 XDR 형식 변환시 입출력 전담 메쏘드로 사용하는 get_byte(), put_byte()와 이들을 이용한 기본 타입에 대한 XDR 변환 메쏘드인 xdr_encode_xxx(), xdr_encode_xxx()이 포함된다. 기본 클래스 XDRStream 에 대해서는 메모리상에서 입출력을 수행하도록 get_byte()와 put_byte() 메쏘드를 구현하였다.

기본 클래스 XDRStream 을 상속 받아 구현된 XDRInputStream 과 XDROutputStream 은 각각 지정된 대상으로부터 입력과 출력을 수행할 수 있도록 한다. XDRInputStream 은 상위 클래스인 XDRStream 의 get_byte() 메쏘드를 오버라이드하여 지정된 대상으로부터 입력을 수행하도록 하고, XDRInputStream 은 상위 클래스인 XDRStream 의 put_byte() 메쏘드를 오버라이드하여 지정된 대상으로 출력을 수행하도록 한다. 입출력을 수행하는 대상은 각 클래스의 생성자의 인수로 지정할 수 있다. 예를 들어 TCP/IP 에 대한 출력 스트림을 인수로 사용하여 XDROutputStream 타입의 객체를 생성하면, 이 객체를 통한 XDR 변환 결과는 TCP/IP 상의 출력 스트림에 출력되게 된다.

다음으로 각 타입별 XDR 변환을 담당하는 클래스들에 대해 설명한다.각 타입별 XDR 변환 클래스의 기본 클래스로 XDRType 을 정의하였다. XDRType 은 실제 코드는 포함하지 않고 XDR 변환을 위한 메쏘드인 xdr_encode()와 xdr_decode()에 대한 인터페이스 만을 정의한다. XDRType 에 속한 xdr_encode(), xdr_decode() 메쏘드에 대한 실제 구현은 XDRType 을 상속받은 각 타입별 XDR 변환 클래스들에서 각각 수행하였다. xdr_encode()과 xdr_decode() 메쏘드는 입출력 대상을 지정하는 XDRInputStream 혹은 XDROutputStream 타입의 객체를 인수로 갖는다. 따라서 인수로 입력하는 객체의 종류만 변경하면 메모리, 네트워크 등 어떤 대상에 대해서도 XDR 변환을 수행할 수 있다. 그리고 RPCGEN 유틸리티가 자동으로 생성해주는 client-stub, server-stub 에서 함수 인수와 함수 수행 결과의 XDR 형식으로 변환할 때에도 XDRType 의 하위 클래스들을 사용한다.

3.2 통신 처리 클래스

본 절에서는 본 논문에서 설계하고 구현한 Java RPC 패키지의 통신을 담당하는 클래스에 대해 자세히 설명한다.

본 논문에서 통신 처리를 위해 설계하고 구현한 클래스는 크게 클라이언트 부분의 통신을 담당하는 클래스인 RPCClient, ClientGeneric, ClientTCP, RPCCallMsg, XDRRPCCallMsg 와 서버 부분의 통신을 담당하는 클래스인 RPCServer, RPCReplyMsg, XDRRPCReplyMsg 로 구분된다.

클라이언트 부분에서 수행하는 작업은 다음과 같은 순서로 진행된다. 먼저 사용자가 stub 함수 호출을 통해 전달한 입력인수를 XDR 형식으로 변환하고, 변환된 입력인수를 포함하는 호출 메시지를 구성한다. 다음으로 구성된 호출 메시지를 서버로 전달하고 서버로부터 함수 호출 결과를 전달받는다. 마지막으로 서버로부터 전달받은 호출 결과를 사용자가 사용하는 형식으로 다시 변환하여 사용자에게 반환한다.

본 논문에서는 이와 같은 작업을 수행하는 메쏘드들의 인터페이스

를 포함하는 ClientGeneric 클래스를 정의한 후, 실제 구현은 이를 상속받은 ClientTCP 에서 수행하였다. 이와 같은 구조를 선택한 이유는 현재 본 논문에서 사용하고 있는 TCP/IP 프로토콜 이외의 네트워크 프로토콜을 사용한 클라이언트를 쉽게 추가할 수 있도록 하기 위해서 이다.

RPCClient 클래스는 네트워크 프로토콜에 종속적인 ClientGeneric 클래스의 하위 클래스들에 대한 wrapper 역할을 한다. 따라서 사용자는 RPCClient 클래스를 사용하여 네트워크 프로토콜에 관계없이 RPC 응용프로그램을 작성할 수 있다.

RPCCallMsg 클래스는 ONC RPC 에서 정의한 호출 메시지 내용을 저장하기 위한 클래스이다. 제 2.2 절에서 설명한 바와 같이 RPCCallMsg 클래스에 포함된 호출 메시지 내용을 서버에 전달하는 경우에도 XDR 형식을 사용한다. 따라서 이를 처리하는 클래스로 XDRRPCCallMsg 를 별도로 정의하여 편리하게 호출메시지를 서버에 전달할 수 있도록 하였다.

지금까지 클라이언트 부분의 통신 관련 클래스들을 살펴보았다. 다음으로 서버부분의 통신 관련 클래스들에 대해 살펴본다.

서버 부분에서 수행하는 작업은 다음과 같은 순서로 진행된다. RPC 서버는 항상 무한 루프를 돌면서 클라이언트로부터의 호출 메시지를 기다린다. 그리고 클라이언트로부터 호출 메시지가 도착하면 이로부터 함수 인수를 추출한 후, 서버에서 사용하는 형식으로 변환하여 대응되는 함수에 전달한다. 함수 수행이 끝나면 수행 결과를 XDR 형식으로 변환한 후, 반납메시지에 포함시켜 클라이언트에게 다시 전달한다. 본 논문에서는 이와 같은 작업을 수행하는 메쏘드를 포함하는 클래스인 RPCServer 를 정의하였다. RPCServer 클래스는 JDK 의 쓰레드 클래스를 상속받아 구현되었다. RPCServert 클래스의 메쏘드 중 상속받은 run()에서 무한루프를 돌면서 클라이언트로부터의 호출 메세지를 기다리게 된다. 그리고 전달받은 호출 메시지에 대응되는 함수를 호출하는 dispatch 메쏘드인 DoCall()는 응용프로그램에 따라 dispatch 내역이 달라지므로 인터페이스 만을 정의하고, 실제 구현은 사용자가 RPCServer 클래스를 상속 받아 정의하는 새로운 클래스 내에서 수행된다. 사용자는 이와 같이 RPCServer 를 상속 받아 자신의 응용프로그램에 맞는 새로운 클래스를 정의하여 사용한다. 그런데 RPCServer 를 상속 받는 클래스는 RPCGEN 유틸리티를 통해 자동 생성되므로 사용자는 어려움 없이 JavaRPC 를 사용하여 응용프로그램을 작성할 수 있다.

RPCReplyMsg 클래스는 ONC RPC 에서 정의한 반납메시지 내용을 저장하기 위한 클래스이다. 호출메시지와 마찬가지로, RPCReplyMsg 클래스에 포함된 반납 메시지 내용 역시 XDR 형식으로 클라이언트에 전달된다. 따라서 이를 처리하는 클래스로 XDRRPCReplyMsg 를 별도로 정의하여 편리하게 반납메시지를 클라이언트에 전달할 수 있도록 하였다.

3.3 RPCGEN 유틸리티를 위한 클래스

본 절에서는 RPC 응용프로그램 작성을 도와주는 RPCGEN 유틸리티 구현을 위한 클래스들의 설계와 구현에 대해 살펴본다.

본 논문에서 설계하고 구현한 Java RPC 패키지를 사용하여 응용프로그램을 작성하기 위해서는 패키지에서 제공하는 RPCServer 클래스를 상속 받아 응용에 맞는 새로운 RPC 서버 클래스를 정의해야 하고, RPC 에 사용되는 함수의 입력 인수와 수행 결과에 대한 XDR 변환 클래스를 작성해야 한다. 이러한 불편을 개선하기 위해 본 논문에서는 필요한 클래스를 자동으로 생성해주는 RPCGEN 유틸리티를 제공한다. 사용자가 XDR 프로토콜의 데이터표현언어의 일종인 RPC 언어를 사용하여 구현하고자 하는 응용 프로그램에 대해 기술하면, RPCGEN 유틸리티는 이를 입력으로 받아 응용프로그램 작성에 필요한 RPC 서버 클래스와 XDR 변환 클래스를 자동으로 생성해 준다.

이러한 RPCGEN 유틸리티를 구현하기 위해서는 유틸리티의 입력 형식인 RPC 언어를 파싱(parsing)하여 사용자가 요구하는 응용프로그램 내용에 대해 분석한 후, 이에 대한 클래스를 자동으로 생성해야 한다.

본 논문에서 설계하고 구현한 RPCGEN 유틸리티를 위한 클래스는 토큰 단위 분석을 위한 클래스인 JAVArpcgen_Token, JAVArpcgen_TokenManager 와 파싱을 위한 클래스인 JAVArpcgen_Declaration, JAVArpcgen_Definition, JAVArpcgen_ProgramDef, 등이 있다. 그리고 메인 메쏘드를 포함하는 JAVArpcgen 클래스가 있다.

JAVArpcgen_Token 클래스는 RPCGEN 유틸리티에서 처리하는 토큰의 종류에 대한 정보를 포함하고, JAVArpcgen_TokenManager 클래스는 입력 화일을 분석하여 토큰 단위로 나눠주고 Preprocessor, Comment 등도 함께 처리한다.

JAVArpcgen_Declaration, JAVArpcgen_Definition, JAVArpcgen_ProgramDef 등의 클래스는 토큰 단위 입력을 받아들여 입력 화일의 의미를 분석하는 파싱을 수행한다. 파싱된 결과는 enum-definition, const-definition, typedef-definition, struct-definition, union-definition, program-definition 과 같은 6 가지 종류의 정의문들로 분석된다. 이러한 분석 결과를 바탕으로 JAVArpcgen 클래스의 메인 메쏘드에서 응용 프로그램 작성을 위해 필요한 서버 클래스와 인수에 대한 XDR 변환 클래스를 생성한다.

4. 결 론

본 논문에서는 ONC RPC 표준을 지원하는 Java RPC 를 설계하고 구현하였다.

본 논문에서는 Java 프로그래밍 언어의 객체 지향 특성인 클래스간 상속을 이용하여 간단하면서도 확장이 용이하도록 RPC 패키지를 설계하고 구현하였다. 따라서 추후 확장이 필요한 경우 추가 기능을 쉽게 더할 수 있다.

본 논문에서 설계하고 개발한 ONC 표준을 지원하는 Java RPC 를 이용하면 웹환경을 지원하는 분산 시스템을 손쉽게 개발할 수 있다. 또한 기존의 ONC RPC 기반의 분산 시스템을 간단히 웹 환경을 지원하도록 이식할 수 있다.

참고 문헌

[Bir84] Birrell, A. D. and Nelson, B. J., "Implementing Remote Procedure Calls," *ACM Trans. On Computer Systems*, Vol. 2, No. 1, Feb 1984, pp.39-59.

[Sun95] Sun Microsystems, "ONC+ Developer's Guide", Sun Microsystems, 1995.

[Sri95a] Srinivasan, R., "XDR: External Data Representation Standard," *RFC1832*, Sun Microsystems, Inc., Aug. 1995.

[Sri95b] Srinivasan, R., "RPC: Remote Procedure Call Protocol Specification Version 2," *RFC1831*, Sun Microsystems, Inc., Aug. 1995.

[Xer81] Xerox Corporation, "Courier: The Remote Procedure Call Protocol," *Xerox System Intergration Standard 038113*, Xerox OPD, Dec 1981.