2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

윈속을 이용한 윈도우간 메시지교환시스템의 구현

김희태*, 오재철**
*순천대학교 컴퓨터과학과
**순천대학교 컴퓨터과학과
e-mail:ds4ajn@hanmail.net

A Implementation of Communication System from Windows to Windows

Hui-Tae Kim*, Jae-Chul Oh**
*Dept of Computer Science, Sunchon University
**Dept of Computer Science, Sunchon University

요약

컴퓨터가 보편화됨으로써 여러 가지 방법의 메시지교환을 위한 환경이 제공되고 있다. 일반적으로 웹 서버를 통한 메시지교환이 이루어지고 있으며, 이러한 방법은 보안에 취약한 형태를 지니고 있음이 분명하다. 경우에 따라서는 보안을 염두에 두고 통신을 할 필요가 있으므로 본 논문에서는 통신하는 두 컴퓨터이외의 다른 연결서버를 이용하지 않고, 메시지를 교환하고자 하는 컴퓨터 상호간에 어느 한 대는 서버, 다른 한 대는 클라이언트의 역할을 맡아 직접 메시지를 교환하는 시스템을 설계 및 구현하고자 한다.

1. 서론

인터넷의 확장으로 인해 여러 방향에 있어서 컴퓨터의 사용 빈도가 급속히 증가하고 있다. 가정이나 사무실을 막론하고 컴퓨터는 일상 생활의 일부분이 되어가고 있으며 이로 인해 업무처리는 물론이고 각종 연락사항도 컴퓨터를 이용하여 이루어지고 있다. Windows운영체제하에서 컴퓨터를 이용한 의사 전달의 경우 일반적으로 웹 상에서의 메시지 교환이 가장 적합한 통신형태라고 본다. 그러나 보안의 필요성이 강조되는 메시지의 송수신에는 적합하다고 볼 수가 없다. 이러한 문제를 해결하는 방법으로는 중간에 연결서버를 통하지 않고 두 대의 컴퓨터가 직접 통신을 할 필요가 있으며 이때 필요한 것이 윈속이다. 본 논문은 이러한 윈속을 이용하여 메시지 교환 Program을 설계 및 구현하여 인터넷의 웹을 통한 메시지 교환보다 통신의 보안을 한 단계 강화하였다.

2. 관련연구

1) Client/Server구조

클라이언트/서버 구조는 시스템의 자원에 접근할 때 두 종류의 구별되는 개체가 관여한다는 사실을 반영한다. 첫번째 개체는 클라이언트로 시스템의 자원에 접근하고자 요청을 보내는 쪽이다. 두번째 개체는 서버로 클라이언트의 요청이 들어올 때만 작동한다. 서버는 자원들을 직접 다룰 수 있으며 그 결과를 클라이언트에게 보낸다. 서버는 전적으로 클라이언트의 요청을 기다리는 컴퓨터이고 요청이 들어오면 즉시 반응한다. 데이터 서버의 경우 클라이언트쪽 컴퓨터의 응용프로그램에서 SQL질의가 담긴 요청을 보내면 데이터서버는 그 질의를 받은 후 자신의 데이터베이스를 가동해서 질의에 해당하는 정보들을 모아 그 내용을 네트웍을 통해 원래의 요청자에게 보낸다. 클라이언트/서버 구조에서 가장 일반적인 것이 데이터베이스 서버이긴 하나 요청과 응답 구조를 가지는 모든 네트웍이 클라이언트/서버의 구조이다.

2) 네트웍의 계층구조

네트웍은 구조가 복잡하여 계층구조를 사용한다. 자신보다 아래쪽 계층이 제공하는 서비스를 이용해서 자신의 위쪽 계층에 서비스를 사용하는 것이 계층 구조이다. 계층구조에서 한 컴퓨터의 한 계층은 다른 컴퓨터의 동급계층(peer layer)과 직접 통신하는 것처럼 보이지만 사실은 한 계층씩 위아래로 오르내리는 과정이 존재한다. 그리고 네트웍 서비스를 제공하는 계층의 기능은 통신하고자하는 상대편 컴퓨터에도 동급의 서비스 제공자가 있어야 가능하다. 상대편 서비스 제공자는 자신과 같은 수준의 계층이 보낸 명령과 응답을 이해할 수 있어야 한다.

한 계층과 그에 해당하는 동급 계층은 서로 이해하고 있는 언어를 통해 요청과 응답을 주고 받는데, 네트웍 용어로 이러한 언어를 '프로토콜(protocol)'이라 한다. 실제로 두 동급 서비스가 물리적으로 연결되어 있는 것이 아니고 단지 자신보다 아래 계층이 제공하는 서비스를 통해서만 연결된다. 이렇게 내려가면 결국 가장 아래 계층을 만나는데, 비로소 동축케이블이나 전화선 같은 물리적 매체로 서로 연결되는 것이다. 한 계층이 자신의 아래 계층이 제공하는 서비스를 이용하기 위해 서비스를 전송하기 위한 메카니즘이 필요하다. 하나의 요청은 상대편 컴퓨터의 동급 계층에 직접 전해지는 것은 아니다. 대신 그 요청은 같은 컴퓨터의 바로 아래 계층에게 미리 정해진 방식으로 전해진다. 요청이 점차 아래로 전달될 때마다 각 서비스는 전달받은 요청에다 필요한 정보를 더해서 잘 꾸린 다음 자기 아래의 계층에 보내며, 맨 아래 계층에 도달하면 요청을 상대편 컴퓨터에 전달한다. 상대편 컴퓨터에서는 우선 맨 아래 계층이 요청을 받아들이고 자기 위의 계층으로 올려 보낸다. 올라가는 과정은 원래 요청이 시작된 계층과 동급인 계층에 도달할 때까지 계속된다. 그 동급 계층은 받은 요청에 따라 응답을 작성해서 다시 자기 아래로 보내고, 점차 내려가 원래 컴퓨터로 전송되고 원래의 계층으로 올라오게 된다. 한 계층에서 한 인터페이스로 전해지는 모든 요청은 두 부분, 즉 상대편 컴퓨터로 보내는 데이터와 그 데이터를 어떻게 처리할 것인지에 대해 다음 계층에게 보내는 제어정보로 이루어진다. 즉 동급 서비스로 보내는 데이터는 원래의 데이터 외에도 여러 가지 정보가 추가된다. 컴퓨터 네트웍 계층의 경우 요청이란 데이터의 버퍼들로 구성된다. 이 버퍼들은 제어 정보와 함께 인터페이스를 통해 다음 계층으로 전달되고 그 계층은 제어 정보를 이용해 자신과 동급인 계층이 무엇을 어떻게 처리해야 하는지 알 수 있는 제어정보를 추가한다. 결과적으로 원래의 데이터에는 추가적인 header가 추가되고, 종종 tailer가 붙기도 한다. 이렇게 꾸려진 데이터는 다음 계층으로 넘겨지며 또 추가적인 제어 정보가 붙는다. 이렇게 추가된 정보들은 상대편 컴퓨터의 최하위 계층으로부터 위로 올라갈 때마다 차츰 떨어져 나간다. 또 데이터를 받은 아래쪽 계층이 데이터가 너무 길어 전송하기 힘들다고 판단되면 작은 조각들로 데이터를 나누는 경우도 생긴다. 이러한 경우 이 계층은 각 조각마다 자신만의 header와 tailer를 추가해서 나중에 위쪽 계층으로 올라갈 때 적절히 조각들을 맞춰서 원래대로 복원할 수 있도록 해야한다. 그런데 서로 다른 계층에서 붙인 header와 tailer가 자신의 위 계층에서 받은 제어정보와 밀접하게 연관되어 있긴 하지만 그 제어 정보 자체는 네트웍을 통해 전송되는 실제 데이터에 포함되는 것이 아니다. 제어정보를 해석하고 그것을 이용해서 자신의 header와 tailer를 만드는 것은 그 계층의 책임이다.

3)Winsock에 대하여

소켓은 Unix 환경에서 만들어 졌고, Unix 환경과 Windows 환경은 다르다. Unix Programing 환경이 절차적이라면 Windows 환경은 Event에 좌우되는 환경이다. 이러한 차이를 극복하기 위해 Unix에서 사용되는 소켓 API를 Windows 환경에서 사용할 수 있도록 옮겨 놓은 것이 윈도소켓 즉 윈속(winsock)이다. 윈속의 TCP/IP에서 위치는 Application Layer와 Transport Layer사이에 있다.

윈속은 Windows에서 TCP/IP응용 Program을 만들 때 사용하는 API의 집합으로 많은 사람들이 사용할 수 있도록 정리하고 표준화하고, 일반에 공개한 일종의 표준화 문서라고 할 수 있다. 윈속이 처음 개발된 의도는 마이크로소프트에서 나온 각종 운영체제에 대하여 일관성 있는 네트웍 프로그래밍 모델을 제시하고 지원하기 위해서이다. 즉 윈속을 이용하여 TCP/IP Protocol 상에서 일관성 있게 Program을 작성할 수 있는 수단을 마련하기 위한 것이다.

MFC는 윈속 지원을 위해 CAsyncSocket, CSocket 등 몇 개의 클래스를 제공한다. 이 중 CAsyncSocket클래스가 가장 기본적인 클래스인데 이 클래스는 Windows 소켓API를 객체지향적으로

캡슐화하고 있다. CSocket클래스는 CAsyncSocket 클래스에서 파생된 클래스이며 CAsyncSocket에 비해 좀더 높은 수준의 추상화를 제공한다.

4) Winsock의 구성

윈속은 함수 Prototype을 가지고 있는 Header File이라 할 수 있다. 이 Header File에는 윈속 사양에서 정의되어 있는 모든 함수들이 선언되어 있다. 윈속 사양에서 언급하고 있는 내용은 각종 함수의 정의와 각 함수를 사용하는 방법에 대하여 언급하고 있다. 윈속 사양에서 정의되어 있는 함수의 종류는 보통 세 가지로 나누며 다음과 같다.

· 소켓을 이용해서 상대편과 연결하고, 각종 통신 기능을 수행하는 함수들의 집합. 소켓을 생성한 후 연결하고 없애는 함수, 소켓에 속성을 주는 함수, 소켓 정보를 얻어내는 함수, 데이터 변환 함수, 주소변환 함수, 데이터를 주고받는 함수 등

· 네트웍 상에서 각종 정보를 얻을 때 사용하는 데이터베이스 관련 함수들의 집합. 호스트 주소나 이름을 주고 호스트 정보를 얻어내는 함수, Protocol 이름이나 번호를 주고 Protocol 정보를 얻어내는 함수, 서비스 이름이나 Port를 주고 서비스에 관한 정보를 얻을 수 있는 함수 등

· Windows환경에서 새로이 정의되거나 기능이 확장된 윈속 확장 함수들의 집합. 소켓함수나 데이터베이스 관련 함수 중에서 비동기 적으로 액세스할 수 있도록 기능을 확장한 함수

5) Winsock의 이용

마이크로소프트가 윈속의 사양을 Windows에서 TCP/IP를 지원하기 위한 표준안으로 공개함으로서, TCP/IP스택을 구성하는 개발사들은 윈속 사양에서 정해진 인터페이스에 맞도록 윈속 동적 라이브러리를 제공하게 되었다. 따라서 응용Program 개발자들은 윈속 사양에 정의되어 있는 함수들을 사용하여 어느 시스템에서나 사용 가능한 Program을 만들 수 있다. 이런 식으로 시스템이 구성되기 위해서는 각 개발사들이 윈속 사양에 맞는 윈속 동적 라이브러리 파일을 제공하여야 한다. 다시 말하면 각 개발사들은 윈속 동적 라이브러리인winsock.dll을 제공해야 하는 것이다. 그러면 응용Program 개발자들은 정의되어 있는 윈속Header File을 사용하여 Program을 작성하고 실행시 동적으로 윈속라이브러리를 연결하여 윈도를 사용하는 시스템이면 어디서든 돌아가는 응용Program을 만들 수 있는 것이다.

3. 시스템 구현

윈속을 이용한 메시지교환Program은 서로 다른 두 Windows에서 대화를 하기 위한 통신시스템으로서 서버가 필요하지 않다는 장점을 가지고 있으며 두 시스템만의 통신이기 때문에 보안에 상당히 강하다는 의미를 포함하고 있다.

1) 시스템 분석

① 서버쪽 : 서버는 두 개의 소켓이 필요하다. 하나는 연결을 대기하는 소켓이고 나머지 하나는 연결이 들어왔을 때 실제로 통신을 하는 소켓이다. 일단 서버는 클라이언트의 연결을 받아들이기 위해 대기소켓을 먼저 만들고 이 소켓의 Listen 함수를 호출하여 연결을 대기하도록 한다. 이 상태에서 클라이언트로부터 연결요청이 들어오면 Onaccept 함수가 호출된다. 즉 Onaccept함수는 연결요청에 대한 이벤트 핸들러이다. 이 함수에서 통신을 위한 소켓을 생성하면 클라이언트와 연결이 된 것이다.

② 클라이언트쪽 : 서버가 두 개의 소켓이 필요한데 비해 클라이언트는 대기를 할 필요가 없으므로 통신을 위한 소켓 하나만 만들면 된다. 통신 소켓을 만든 후 이 소켓의 Connect 함수를 호출하여 서버와 연결한다. 이 때 연결하고자 하는 서버의 IP address 와 Port번호를 인수로 지정해 주어야 한다. 지정한 서버와 연결이 이루어지면 Connect 함수는 TRUE를 리턴한다.

2) 시스템 설계

객체지향시스템을 개발하기 위한 계획을 수립하는데 있어서 필수적으로 알아야 할 사항은 기능, 운영절차, 하드웨어 시스템 등의 운영 환경 시스템 체계를 인식하는 것이다.

메시지교환시스템의 흐름 구조는 다음과 같다.

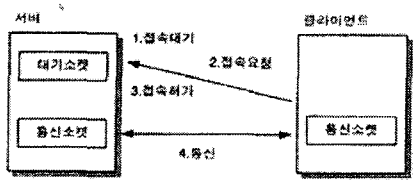

그림 1 소켓을 이용한 접속과정

3) 시스템 구현

멤버함수	기능
Creat()	소켓을 생성한다.
Bind()	소켓에 로컬주소(IP 주소)와 포트(Port)를 결합.
Listen()	클라이언트의 연결요청을 대기
Accept()	클라이언트와의 연결 허용.
Connect()	서버소켓에 연결 시도.
Send()/SendTo()	데이터 송신
Receive()/ReceiveFrom()	데이터 수신
GetSockName()	소켓에 대한 주소를 얻는다.
GetPeerName()	소켓에 연결된 상대방의 주소를 얻는다.
Close()	소켓을 닫는다.

표 2 CAsyncSocket 클래스의 주요함수

서버(Server)의 경우	클라이언트(Client)의 경우
· Creat()를 호출하여 소켓(SOCKET)생성	· Creat()를 호출하여 소켓(SOCKET)생성
· Listen()을 호출하여 클라이언트의 요청을 기다린다.	· Connect()을 호출하여 서버에 접속
· Accept()을 호출하여 연결 허용	
· 데이터 송수신	· 데이터 송수신

표 3 CAsyncSocket 클래스 통신

① 서버/클라이언트의 작업을 한 개의 프로그램 내에서 동시에 처리할 수 있도록 하였다.

② 프로그램이 두 번 이상 실행되지 않도록 하였으며 프로그램 실행과 동시에 자신의 IP를 보여주도록 하였다.

③ 대기종료, 접속종료가 가능하도록 한다. 이때 한쪽편이 종료되면 상대편에게도 신호를 보내 같이 대기상태로 만들도록 하였다.

④ 대화상태를 보여주는 리스트박스를 자동으로 스크롤시켜 항상 최근 대화한 내용을 보여주도록 하였다.

⑤ 접속할 IP를 레지스트리에 기억시켜서 프로그램을 실행할 때마다 IP을 입력하지 않아도 되도록 하였다. 또한 프로그램의 위치도 레지스트리에 저장하도록 하였다.

⑥ 프로그램이 실행되는 즉시 대기상태로 시작하도록 옵션을 만들어서 이 프로그램을 시작프로그램에 등록해 두면 상대편이 언제든지 접속하여 메시지를 보낼 수 있도록 한다. 또한 프로그램이 최소화 되어 있을 때에도 상대편이 접속하여 메시지를 보내면 프로그램이 활성화되어 메시지를 받을 수 있도록 하였다.

⑦ 메시지를 보낸 후 다음 메시지를 보낼 때마다 에디트박스에 입력된 이전 메시지를 지워야 하는 불편함을 없애기 위해 문자열을 선택상태로 만들어 곧바로 다음 메시지를 보낼 수 있도록 하였다.

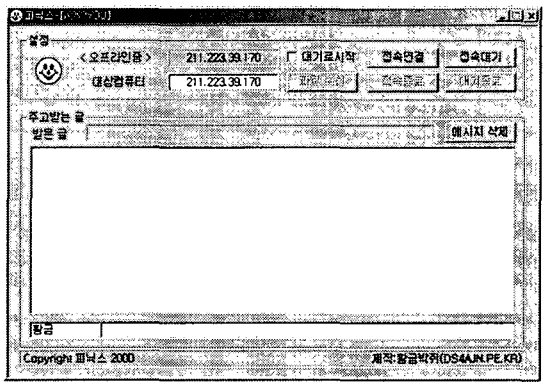

그림 2 메시지교환시스템 메인화면

4. 결론 및 향후 계획

본 논문은 현재 일반적으로 사용중인 웹서버를 이용한 메시지 교환시스템의 보안상 문제점을 해결하기 위하여 윈속을 이용한 연결서버가 필요 없는 메시지교환시스템을 설계 및 구현함으로써 보안상의 문제점을 해결하였다. 개발된 시스템은 객체지향 시스템을 채택하여 윈도우간 통신을 GUI환경으로 설계 및 구현하여 일반사용자들이 특별한 지식 없이 쉽게 사용 가능하도록 하였다. 현재는 텍스트를 기반으로 한 문자 메시지를 송수신하는데 그치고 있으나 향후 음성을 이용한 통신을 가능하게 하여 보다 효율적이고 편리한 프로그램으로 개선해 갈 예정이다.

[1] 류광, "Leonid Braginski and Matthew Powell"

[2] 김한규, 박동선, 이재광, "데이터 통신과 네트워킹"

[3] 류형규, 이순천, 류시원, 신성호, "UML 기반 객체지향 클라이언트/서버 구축", 2000

[4] 김상하, "통합객체지향법론 모델링 및 설계구축실무", 1999