Experiment TCP
TCP Socket

Introduction

The Transmission Control Protocol (TCP) provides reliable, two-way transmission of data. In a TCP communication, two sockets are *connected*, allowing a reliable byte-stream to flow between them in either direction. TCP is referred to as a *virtual circuit* protocol, because it behaves as though a circuit is created between the two sockets.

A good analogy for TCP communications is a telephone system. Connecting two sockets is similar to calling from one phone to another. After the connection is established, you can write and read data (talk and listen).

Table shows the steps in establishing socket communications with TCP, and the analogy of each step with telephone communications.

Table : **TCP Analogy to Telephone Communication**

Task 1 Waits	Task 2 Calls	Function	Analogy
socket( )	***socket( )***	Create sockets.	Hook up telephones.
bind( )		Assign address to socket.	Assign phone numbers.
listen( )		Allow others to connect to socket.	Allow others to call.
	connect( )	Request connection to another socket.	Dial another phone's number.
accept( )		Complete connection between	Answer phone and establish

		sockets.	connection.
write( )	***write( )***	Send data to other socket.	Talk.
read( )	***read( )***	Receive data from other socket.	Listen.
close( )	***close( )***	Close sockets.	Hang up.

Example : Stream Sockets (TCP)

The code example (tcpClient.c, tcpServer.c)uses a client-server communication model. The server communicates with clients using stream-oriented (TCP) sockets. The main server loop, in *tcpServerWorkTask( )*, reads requests, prints the client's message to the console, and, if requested, sends a reply back to the client. The client builds the request by prompting for input. It sends a message to the server and, optionally, waits for a reply to be sent back. To simplify the example, we assume that the code is executed on machines that have the same data sizes and alignment.

Experiment

1. tcpClient.c 와 tcpServer.c 를 컴파일 하여 실행해 보시오. (실행 방법은 소스 코드 내에 있음.) 이때, tcpClient 에서 호출할 remote system 은 자기 자신이므로

-> tcpClient "166.104.225.251"

이라고 실행 한다.

2. Windows 2000 host 에서 socket.exe 프로그램을 실행 시키고 메뉴에서 Answer 를 선택한다. Port number 는 5001 로 설정한다. 다음, target 에서 tcpClient program 을 실행하여 target 에서 전송한 메시지가 Windows 2000 host 에 제대로 전달 되는지 확인하시오. 이때, tcpClient 에서 호출할 remote system 은 Windows 2000 host 이므로

-> tcpClient "166.104.225.252"

이라고 실행 한다.

3. Windows 2000 host 에서 GraphServer.exe 를 실행 시키고, 메뉴에서 Answer 를 선택한다. 그리고, target 에서 cosine.c program 을 실행하면 cosine graph 가 실시간으로 그려지는 것을 볼 수 있다. 프로그램 cosine.c 는 cosine graph 를 그리기 위한 x-y data 를 발생 시켜서 socket message 로 전송한다. 이때 전송하는 메시지는 x 축 데이터와 y 축 데이터에 대해서 각각 2 바이트씩 총 4 바이트의 메시지를 한번에 전송한다. 프로그램 cosine.c 를 수정하여, cosine 데이터를 발생 시키는 task 와 데이터를 받아서 Windows 2000 host 로 전송하는 task 로 나누어서 동일한 결과가 나오는지 확인 하시오. 이때, 두 task 사이의 데이터 전송은 message queue 를 이용하시오.