Project 1: Client-Server Socket Programming

Department of Computer Science
University of Stellenbosch
7600 Stellenbosch
South Africa

1 Introduction

This project serves as an introduction to client-server programming. You will be expected to implement an abstract protocol called Simple Registration Protocol using simple TCP socket programming. To illustrate the use of the protocol, you will also implement a concrete application: a simple appointment management system. This application will introduce you to the use of LAMP (Linux, Apache, MySQL and PHP), a free, powerful and commonly-used combination for Web-based applications.

2 The Simple Registration Protocol

2.1 General Description

The SRP uses a client-server model where multiple client processes request services from a server using request messages. These requests are answered using reply messages. The SRP includes only one mandatory request message (called REQ) and one mandatory reply message (REP). REQ is sent by a client to the server to indicate a specific application-defined service (e.g. reservation, confirmation, information request). The server responds with a REP message, the contents of which is specific to each type of request. The server should check the validity of each request and *should not fail if an invalid request is received.*

The client follows these three steps to make a request:

1. Establishes a TCP connection with the server,
2. Sends the request to the server, and
3. Closes the connection.

If the request is valid, the server

1. Opens a TCP connection to the client,
2. Sends the reply, and
3. Closes the connection.

Note that the REQ and REP messages are sent in two separate TCP sessions. The client will close the existing REQ TCP session and wait for the server to open a new TCP session for the REP.

2.2 Specification of Messages

The exact format of the REQ and REP messages is left for you to decide,[1] but each of the messages should comply with the specifications below.

2.2.1 The REQ Message

The REQ message must contain the following fields:

TYPE: type of the request

REQID: unique identifier for this request message

CONTENT: content of the request

PASSWD: password authentication in case the request requires this

RESR: resource requested or to be reserved

DATEr: Date on which the request was sent

TIMEr: Time at which the request was sent

Observe that

- RESR is an abstract resource such as a conference to attend, a classroom to be reserved, a book to be ordered, a tub of ice-cream, etc.
- PASSWD is an optional parameter which is displayed by the application program only when user authentication is required such as in the use of database updates.

2.2.2 The REP Message

The REP message must contain the following fields:

REPID: unique identifier for this reply message

RESP: response to the request (e.g. a Y for ’yes’ or N for ’no’, or even a text message or picture)

DATEa: date on which the request was accepted or denied

TIMEa: time at which the request was accepted or denied

2.2.3 NB!!!

This is an abstract protocol. Do not define fixed meanings for the content of the fields in the messages, or even make assumptions about their length! The content of the messages is completely independent of the underlying protocol in the same way that TCP does not know whether it is transporting a web page

[1]Note that in the project on security and authentication, we will be making SRP more secure, so make sure your message format is flexible (does not rely on fixed-length messages, etc.) if you want to save some work later on.

or video or file contents. You must define some message format that is capable of transporting unknown application-specific information. You might consider using a binary message header containing the byte offsets of the various fields in the message, or a text-based human-readable messages like those used in HTTP or SMTP, or just send serialized objects.

Your implementation of the protocol must give applications a clean interface that provides the ability to create new messages, pack contents into the message fields and extract the contents from received messages. For example,

```
...
REQMessage req = new REQMessage();
REPMessage rep = new REPMessage();
req.setType(new String("A message for booking holidays"));
rep.setResp(new Image("../Denied.gif"));
String s = (String) req.getType();
...
```

3 Concrete application: Appointment Management System

Since the above protocol is pointless without some example application, also implement the following to show that your protocol implementation works. Do not take too much effort with realism or eye candy – the application should merely show the correctness of your protocol implementation, your understanding of basic client-server socket programming and that you have played with LAMP. [2]

Having completed the Databases course, it is assumed you have enough SQL knowledge to get on your feet. A good MySQL/PHP Tutorial is available at THE URL OF THE TUTORIAL HERE. To save you time, a sample database creation script, a Java access layer class and a PHP web page using the sample database is available for download at THE URL OF THE SAMPLES HERE. *You will not be penalized for using these (even without any modification) in your assignment, since this project is about socket programming, not Web development.*

Here are the details of the application you must develop:

[2]Note that all the LAMP software is installed in Narga and the Linux Lab for your use. Should you wish to install the software on your own computer, details are given below.

Linux needs no introduction and is available form `http://sulug.sun.ac.za`

Apache is a free open-source web server available from `http://www.apache.org`, but should be available with your Linux distribution. For details on setting up your web page, see the FAQ on `http://bach.sun.ac.za`

PHP is a language used for client-side Web scripting and modules should come with your Apache or Linux system. For more details see `http://www.php.net`

MySQL is a light-weight, free open-source relational database, which you can download from `http://www.mysql.org` but should also come with your Linux distribution. PHP has excellent, easy-to-use interfaces for accessing MySQL databases from web pages. A MySQL account has been created for you on `bach.sun.ac.za` with your student number as user-name and password. The only database you have the right to access is the one already created for you; it has your student number as its name. Note that for security reasons, you do not even have the right to create and drop this database. The MySQL JDBC Driver is available from INSERT URL HERE.

Of course, Apache, PHP and MySQL are available for Microsoft Windows, but nobody wants to run something called WAMP.

Overwhelmed by requests and invitations from an ever-expanding social circle, and running into financial and time-management problems, you decide to implement an automatic appointment management system. Friends run your client application, which they use to lodge pleas for your company, stating the date, time, duration and nature of the social activity, and how much the outing is likely to cost you. The clients send these requests to a server which consults your database and checks whether you are already busy, whether the social activity is in your list of current favorites and if the event is within your current financial capability. The server responds to the request with appropriate affirmative or apologetic responses. If an appointment is accepted by the server, your balance in the database should decrease by the appropriate amount. You can view your database from anywhere using your PHP-based web interface and check your social obligations and financial plight.

Implement both a sequential server (one request at a time) and a concurrent server (will require multi-threading, see a Java resource like the ones in the References).

4 Implementation Details

Although many other technologies are suitable for this exercise (e.g. C or C++, asp, jsp, JavaScript, Servlets, PostgreSQL etc.), you *must* stick to these implementation requirements, please!

- Your protocol implementation, as well as the client and server programs, should be coded in Java, and work in the Linux Lab. See a good Java book or `http://java.sun.com` for details on using the network package.
- The web page should be in HTML or XHTML and use PHP [3] to access the database. Ensure that your web page works in Mozilla or Konqueror.
- Use MySQL for the database.
- Your code should be neat and well-documented.
- Your report should document what you have done: the format of your SRP messages, the fields in your database, how your application uses the SRP fields, any assumptions you might make – anything non-obvious should be clearly explained. The sources for *all* your work, including HTML, PHP and SQL scripts, Java classes, database creation scripts and test cases must be included in your electronic submission.

References

[1] PHP Manual, `www.php.net`

[2] MySQL Manual, `www.mysql.org`

[3] Linux in a Nutshell, O'Reilly

[4] Dave Raggett's Introduction to HTML, `http://www.w3c.org/MarkUp/Guide`

[5] Deitel, H. and Deitel, P., Java How To Program, Fifth Edition, Prentice Hall, 2003.

[6] Java Network Programming Tutorial, `java.sun.com`

[3] use the phpinfo() function to ensure that the PHP functions you are using are compatible with the version of PHP installed in the lab