Software Requirements Specification
(SRS Document)

on

"Echo Server Using Socket Programming"

Title of Assessment:

Implement a multithreading application for echo server using socket programming in JAVA.

Problem Definition:

Java implements socket programming using "java.net.Socket" and "java.net.ServerSocket" classes. We must develop a program which will help us to understand the following concepts-

- Socket programming in Java
- Multithreading in Java

Scope:

An application implementing client-server architecture is the best way to understand socket programming. An echo-server is an example of an application using client-server architecture. Concepts of multithreading can be introduced by setting up multiple threads to handle different clients. It should be implemented for-

- Single system as a Client and Server
- Single system as a Server and Multiple Client
- Different System as a Client and Single Server

Potential applications of an echo server are-

- **Connectivity Testing:** This "echo server" can be set up to listen on any desired (tcp) port to simulate whatever application you want to run (eg web server = port 80, Microsoft SQL Server = port 1433, etc). From the client machine, you can then telnet to this port. When a telnet connection has been established, everything

you type will be echoed back to your screen, indicating that the telnet client and the echo server can talk to each other : you've established connectivity at the application level.

- **Troubleshooting:** You can use this echo server to troubleshoot networks, test a firewall (eg "if I have a server listening on port 123, will my firewall allow connections to it ?) and so on.

Other applications which use concept of multithreading are-

- **Image processing** can often be done in parallel (e.g. split the image into 4 and do the work in 1/4 of the time) depending on the algorithm being run.
- **Rendering of animation** is massively parallel as each frame can be rendered independently to others
- **GUI programming** often helps to have at least two threads when doing something slow, e.g. processing large number of files - this allows the interface to remain responsive whilst the worker does the hard work

Additional applications of socket programming include:

- Online banking service
- Email
- Network Printing
- Online examination system

Functional Requirements:

The system should-

- utilise client-server architecture through socket programming.
- multithreading should be implemented in order to allow multiple clients to connect to server on same system on different threads
- once client-server connection is established, communication can take place between them. A server will respond when client 'pings' it through 'echo' command.

Non-Functional Requirements:

- **Efficiency:** Any client which pings server should immediately be responded to by the server.
- **Maintainability:** A client should not lose connection to server once it connects to it.
- **Scalability:** Quality of service should remain the same whether number of clients is 1 or the maximum number.
- **Supportability:** There should not be any degradation in performance when it is implemented over one device as server and multiple other devices as clients, i.e. even if the different devices use different operating systems, there should not be any problem.
- **Fault Tolerance:** System should not fail if a single connection fails.
- **Reliability:** System should be reliable, i.e. we can expect only the correct response when a client pings the server. Response messages shouldn't be mixed up.

Constraints:

- Only a finite number of clients can connect with a server at any given time.
- Too many requests from the clients may lead to congestion.
- If server fails, then the whole network will collapse.

Conclusion:

An echo server program has been implemented utilising socket programming and multithreading on Java. In this application, after a client connects to a server, it can ping the server for response. This function can be used to check parameters like connection status, response time of server etc. The application has been developed to operate in 3 ways - (i) single device as single server and single client; (ii) single device as single server and multiple clients; (iii) multiple devices as single server and multiple clients.