COMP 512 Project 1 Report

Introduction

This description uses the attached class diagram with application boundaries as a reference.

Design Description

Communication

Communication between client, middleware, and RM servers is achieved using descriptor objects passed via ObjectOutputStream and ObjectInputStream over TCP. The descriptor objects allows us to encode strongly typed data in a meaningful way. We have one descriptor for requests (RequestDescriptor) and one descriptor for responses (ResponseDescriptor).

The RM servers and middleware uses a WelcomeManager object which listens for new client connections on a ServerSocket and creates a Socket wrapped in a ClientConnectionThread when a new client is accepted. ClientConnectionThread forwards requests it receives to the appropriate request handler. For the RM server it is RMRequestHandler and for the middleware it is MiddlewareRequestHandler.

The client and middleware creates and opens a Socket to the server based on command line input. The hostname and port of the server is stored in a ServerConnection object which contains an operation for sending out a request and returning the response. This operation also creates a new socket for each request which creates a ClientConnectionThread on the server-side. Since the middleware must connect to multiple RM servers, it keeps track of its server connections using ConnectionManager.

Concurrency

We achieve concurrency by creating a new thread (ClientConnectionThread) for each new request to the middleware and RM servers via the WelcomeManager. Thus one client can make a request while another client is still waiting on a response. While there is overhead in creating a new thread and socket each time, we expect to lower this overhead in future deliverables with thread and socket pooling.

We know the underlying data structures are not damaged by concurrent requests because the basic operations on RMItem in ResourceManagerImpl (readData, writeData, and removeData) are synchronized. That is to say, only one thread can be inside any of these operations at a time. Therefore, after re-implementing web-service-like behaviour over TCP, there should not be any difference in concurrency safety

Handling Requests

The RequestHandler unpacks the RequestDescriptor at the middleware to determine which RM server the request should be sent to. Or if it is a customer related operation, it sends out the request to each RM server and executes the request on the middleware's RM for customers.

Similarly, the RMRequestHandler unpacks the RequestDescriptor at the RM server to execute the operation on the RM server. RMRequestHandler is essentially a wrapper for ResourceManagerImpl that converts a RequestDescriptor to the appropriate operation on ResourceManagerImpl.

Testing Suite

We initially tested our application simply by starting it up and making a few commands and checking on the server application that the output was indeed what was accepted.

We automated this process by changing the ant build script to start up the RM servers, middleware, and a test client which sends a series of requests and compares the responses to what would be expected from that request. For example, adding a new flight with correct parameter should return true. Attempting to book that flight before it is created should return false.

We additionally manually tested and passed the following cases:

- Make a request while a RM server is down.
- Make a request while the middleware server is down.
- Modifying the RM server code so that a certain operation takes blocks for 20 seconds. Then running two clients and making the blocking request from the first client followed by a typical request from the second client. (simulates concurrent requests to middleware)
- Making the same change as the above case but the second client makes a request that ends at the same RM server as the first client. (simulates concurrent requests to RM server)
- Running two clients simultaneously and repeatedly make requests from both clients. (simulates concurrent data access)

Application Use Notes

Since the middleware and webserver (client) can take connection information via the command line, any of the three applications can be started in any order.

Resource Manager Servers

1. Start this application via java. The main method is located in “server.RMServer.main()”.
2. Enter the port to listen on via command line, you are prompted for this information.
3. Press enter to quit the application.

Middleware Server

1. Start this application via java. The main method is located in “middleware.MiddlewareServer.main()”.
2. Enter the port to listen on via command line, you are prompted for this information.
3. You are then given a command-line menu with which you can add, remove, or list resource manager servers. You will be prompted for the type of server to add and the connection information for that server.

WebServer (Client)

1. Start this application via java. The main method is located in “webserver.WebServer.main()”.
2. If you did not provide a port as an argument, you can enter it via the command line if prompted.
3. Open your browser and navigate to http://localhost:8000/ where 8000 should be replaced by the port number you used in step 2.
4. Click on the drop down menu to select the operation you want to run.