Teague Hall (Group 28)
hall1945@umn.edu
CSCI5105 - Introduction to Distributed Systems
Project #1 - Design Document

Project 1 will be implemented in C and will leverage the Linux RPC system. The project consists of a client, server, and shared module for decoding article strings. These are all detailed below.

Client (client code is contained within client.c with calls to article.c and the auto generated RPC code)

The client application accepts up to three input arguments:

1. Group Server (**required**) - Name or IP address of the RPC group server.
2. Test Article (**optional**) - Article for either publishing or subscribing (for automated testing)
3. Test Unsubscribe Duration (**optional**) - Duration after which client will unsubscribe (for automated testing)

The optional arguments assist with automated testing. Specifically, the inputted test article will automatically be sent on start-up (corresponding to either a subscribe request or publish request) while the test unsubscribe duration indicates a time duration after which an unsubscribe message will be sent. Thus testing can be performed automatically using a BASH script rather than having to manually type in test cases.

Once started, the client will perform initialization tasks and create three threads: **1)** RPC send thread **2)** message receive thread **3)** ping thread. The function and operation of each are described below.

1. The RPC send thread waits on user inputs, validates those inputs, and then communicates the specific RPC request (e.g. subscribe, publish) to the server. This thread will last indefinitely until a user enters the “leave” command, after which the client program will terminate. If an invalid input is submitted by the client, the client program will communicate an error and wait for a new user input. This thread is also responsible for handling the automated test inputs previously described. If test messages are provided to the client, those will be sent and then the client will automatically terminate (thus not waiting for further user inputs).
2. The message receive thread is responsible for monitoring incoming messages from the server in which a client has subscribed. This is implemented by listening on a previously created UDP socket that will be read indefinitely until the client terminates.
3. The ping thread sends RPC “ping messages” once a second to monitor the server status. If a ping message does not receive a response, this indicates the server is unavailable and thus terminates the client application.

The client features a single mutex lock to control access to the RPC object between threads. This ensures that a user submitted message will not disrupt the ping thread (and vice versa).

Server (server application code is contained with server_routines.c with calls to article.c and the auto generated RPC)

The server manages an active client list implemented as a statically sized array (MAXCLIENTS #define). On server start-up, this list is empty and will grow in size per subsequent “RPC join” messages. The list will decrease in size when “RPC leave” messages are received or if a client times out (more on that later). Access to this list requires a mutex lock such that join and leave messages do not collide.

The server also manages a subscriber list with is implemented as a statically sized array (MAXSUBSCRIPTIONS #define). This list starts empty on server start-up, and grows in size as “RPC subscribe” messages are received. This list will decrease in size when “RPC unsubscribe” messages are received, or if a client leaves the server. If a client leaves the server, all related subscriptions are removed. This list is iterated upon when published messages are received to identify any matches. If a match is found, the server will forward the received published article to the subscribing client via UDP. The server does not restrict duplicate subscriptions for simplicity purposes, thus a client could accidentally submit the same subscription request and thus receive duplicate messages. Lastly, access to this list is restricted by use of a mutex lock to avoid race conditions when messages are received simultaneously.

There also exists a timeout monitor thread that is responsible for removing clients that have not gracefully left the server. Under normal conditions, a client will submit a leave request thus triggering clean up on the server. However, if a client

were to crash, the server would be left in a state waiting on a non-existent client. This problem is remedied by use of a timeout thread which monitors received ping messages from a client. If a client does not send a ping message at least once per five seconds, the server will remove the client from the active list (and all related subscriptions).

Article.c / Article.h (shared module that is referenced by both client and server)

The code within this module provides a single helper function “articleDecode” responsible for decoding an article string into an article object that can then easily be checked for article type, originator, org, and contents. This function is used by the client to validate correctly formatted user input messages, and then used by the server for message validation, and subsequent publish and subscribe requests.