Design Document

CprE 450/550 Project 1

Ray Kinsella

## Problem Statement:

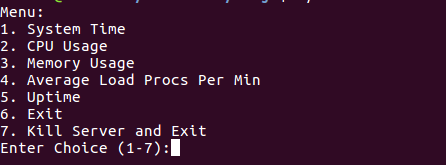
The objective of the machine problem is to write a network management application that tracks CPU usage and other statistics on a host and allows querying by a RPC-based network management system. Its output can be used to feed into an analysis component for deciding on corrective actions in self-managing distributed systems. The system will have multiple clients and single/multiple servers. Clients can send request to a server running at a different machine to get the current system statistics of the server machine. Track Current system time (can be in different formats such as date, time, or a combination of both.), CPU usage, Memory usage, Load procs per min, and uptime of the server.

## Approach:

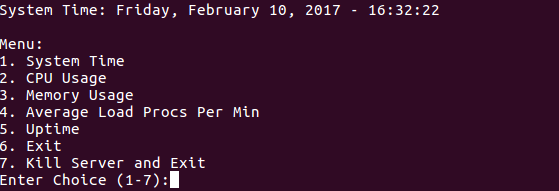
I began working on this project my practicing with the RPC sample code provided to us, and working from there. The sample code provided a structure, where the client could choose a request from a menu based on the number associated with that request. I chose to expand this system to include several new statistics, and add a debugging option in the form of an exit and kill sever option. I knew that I wanted to retain the System Time functionality seen in the sample code, so I first worked on consolidating that request into a single option in the client’s menu. From there I worked down the recommended statistics list, implementing one at a time and adding them to the client menu. The second statistic I added was CPU Usage, I found some code on github (Cited in code) which opened the machines proc/stat file and parsed the information on current CPU Usage. The next three statistics I implemented all use the sysinfo library’s sysinfo struct and function. The function returns information about the system found in the proc/stat and proc/load files respectively. This allows for me to easily access numerous system statistics including Memory Usage, Load Procs per time, and system uptime. By maintaining the sample code’s structure, the program is very modular and can easily be expanded on in the future by adding more menu options in client, and adding the information gathering corresponding with that request in the server code.

## Usage:

1. Generate RPC code from date.x file
   1. **Command:** rpcgen -k date.x
2. Compile client and server code with corresponding RPC date code.
   1. **Command:** gcc client.c date\_clnt.c -o client
   2. **Command:** gcc server.c date\_svc.c -o server
3. Run Server in background
   1. **Command:** ./server &
4. Run Client with localhost argument.
   1. **Command:** ./client localhost
   2. **Client Menu:**



* 1. Select any option. Example (1)



* 1. Select any option. Example (2)

