Herman Chin
CMPE 150L
Winter 2012
Section: Tues 4PM

eVote Project – Socket Programming

Quick Start:

This is a program demonstrating a voting system between a server and its clients. It is written in C and takes in the nominee's name and voter's user ID as input then displays the current results of the votes to the client with a debug screen shown on the server side. The program uses the fork method to allow multiple connections at once and stores data using txt documents instead of integrating them into arrays in the code.

Design:

Design considerations:

The considerations taken into this design is the use of a txt document to store the results of the votes and nominees instead of placing them inside of arrays because of the use of the fork method. The fork method sets a child process and data that is changed in the child process does not get saved in the main program because the data cannot be shared between the two processes. So, a solution that was implemented is the use of a txt document to store the values.

System architecture:

The architecture for this program is the same as shown in the diagram in the lab. The server creates a socket, binds it to a port, listens for incoming connections, and sends/receives. The client creates a socket, connects, and sends/receives.

User Interface:

The user interface for this program is pretty basic. The server can start up the program by simple running the program. The client can send in a vote by using the input format: client Nominee hostIP UserID

Server Side:

This file creates a socket and binds it to port 1337. It then takes in the ID and Nominee from the client and adds it to nominee.txt file. It does this to store the values so that the program can access the data outside of the fork child process loop.

The program opens up the votelist.txt and appends it to an array. The program then compares the nominee name to the names from nomlist.txt, if it exists then it will increment the votes associated with the name, if it does not exist then it will add it to the list and increment its votes.

The program then exits its fork loop and draws data from nomlist.txt and votelist.txt to send to the client. The program sends an initial packet to the client to notify it of how many recv to prepare for, then, delivers the packets.

Client Side:

The client side of the project, this program creates a socket and binds it to a port then sends the packets of the ID and Nominee taken from the command line in the form: client Nominee HostIP UserID.

Then the client will listen on port 1337 for the preparation packet from the server to prepare for a certain amount of recv. The client then recv and prints the results to the terminal.