CS:425 Mini Project 1

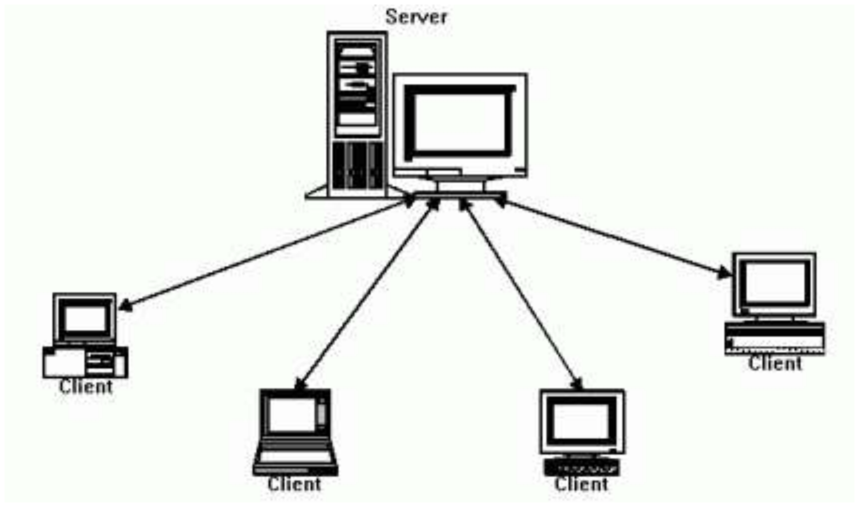
CHATBOT

Introduction

- Develop comprehensive client-server messaging application
- Application has basic security features of a chat room
- Application has support for broadcast and point to point messages
- Includes concepts of asynchronous messaging

Setup

- Client server based chat application



- Clients communicate with each other through "commands" sent to centralized server

Features - Authentication

- Sever maintains text file with username password
- Clients authenticate themselves and log into the system
- After 3 consecutive unsuccessful login attempts, the servers blocks the IP for that user for 60 seconds.
- Prohibit simultaneous duplicate logins. Eg While Columbia is logged in, restrict Columbia from other terminals

Features – whoelse, wholasthr

- Useful feature to list the other logged in users
- Implemented by sending commands
  - whoelse : Displays name of other connected users
  - wholasthr: Displays name of users connected within last hour

Messaging Features

- Broadcast
  - broadcast <message>
    - The message is immediately displayed on the logged in user terminals
- Private message
  - message <user> <message>
    - This message is sent only to a specific user
- Block and Unblock
  - The users can block and unblock other user(s) Based on this the private messages will be delivered
  - block <user>
  - unblock <user>

Asynchronous messaging

- If a user is not available, such offline message are displayed when the user logs in next time
- Only applicable for private messages

Miscellaneous features

- logout
  - To enable logging off functionality.

Environment

- Use of C/C++, Java, Python is permitted
- Environment:
  - Java 1.6, gcc version 4.6.3, python 2.7.3

Java Implement Hint

- 1. Class Socket

Socket (InetAddress address, int port)

Creates a stream socket and connects it to the specified port number at the specified IP address.

close()

Closes this socket.

Java Implement Hint

- **getInetAddress**()

  Returns the address to which the socket is connected.

- **getInputStream**()
- Returns an input stream for this socket.

- **getOutputStream**()
- Returns an output stream for this socket.

Java Implement Hint

- 2.**Class BufferedReader**
- Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

- **3. Class InputStreamReader**
- An InputStreamReader is a bridge from byte streams to character streams

Java Implement Hint

- Reading from a socket:

```
Socket Client;
BufferedReader buff;

buff =new BufferedReader(new InputStreamReader
      (client.getInputStream()));
```

C Implement Hint

- C is the **hardcore** choice.

- In Java:

```
ServerSocket s = new ServerSocket(port);
Socket in=s.accept();
```

- Equivalently, in C (omitting error handling)

```
int listenfd,connfd;
struct sockaddr_in servaddr, cliaddr;
listenfd = socket (AF_INET, SOCK_STREAM, 0)
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(SERV_PORT);
bind (listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
listen (listenfd, LISTENQ);
connfd = accept(listenfd, (struct sockaddr *) &cliaddr, &clilen);
```

C Implement Hint

- Data structure
- sockaddr_in, file descriptor (int)
- pid_t

- System calls:
- socket(), bind(), listen(), connect(), send(), recv(), inet_ntoa(), close()
- fork(),waitpid()

C Implement Hint

- Header files to include:
- sys/socket.h
- arpa/inet.h
- netinet/in.h
- unistd.h
- sys/wait.h

C Implement Hint

- Server:
- socket->bind->listen->accept->select/fork/pthread(!)->recv/send->close

- Client:
- socket->connect->recv/send->close

C Implement Hint

- Be careful.
- Buffers – Overflow Attacks
- Memory Allocation – Memory Leaks

- Read man pages!
- E.g., man socket, man send, man bind, …

- Start from examples
- TCP Echo server/client
- http://www.cs.dartmouth.edu/~campbell/cs50/conEchoServer.c
- http://www.cs.dartmouth.edu/~campbell/cs50/echoClient.c