Universidad Simón Bolívar.

Departamento de Computación y Tecnología de la Información.

Ingeniería de Software I (CI-3715).

Curso de Redes I CI-4815.

Profesor: Ricardo González.

**Proyecto 1**

Integrantes:

Gabriel Freites 08-10383

Vanessa Balleste 08-10091

Sartenejas, 10 de Mayo de 2013.

**Informe**

**Diagrama de Secuencia**

Un diagrama de secuencia con el intercambio de mensajes que debe ocurrir entre un schat y otros dos cchat, los dos suscritos a una misma sala de nombre actual, uno de los cuales ejecuta los siguientes comandos:

men Hola

usu

sal

cre nueva

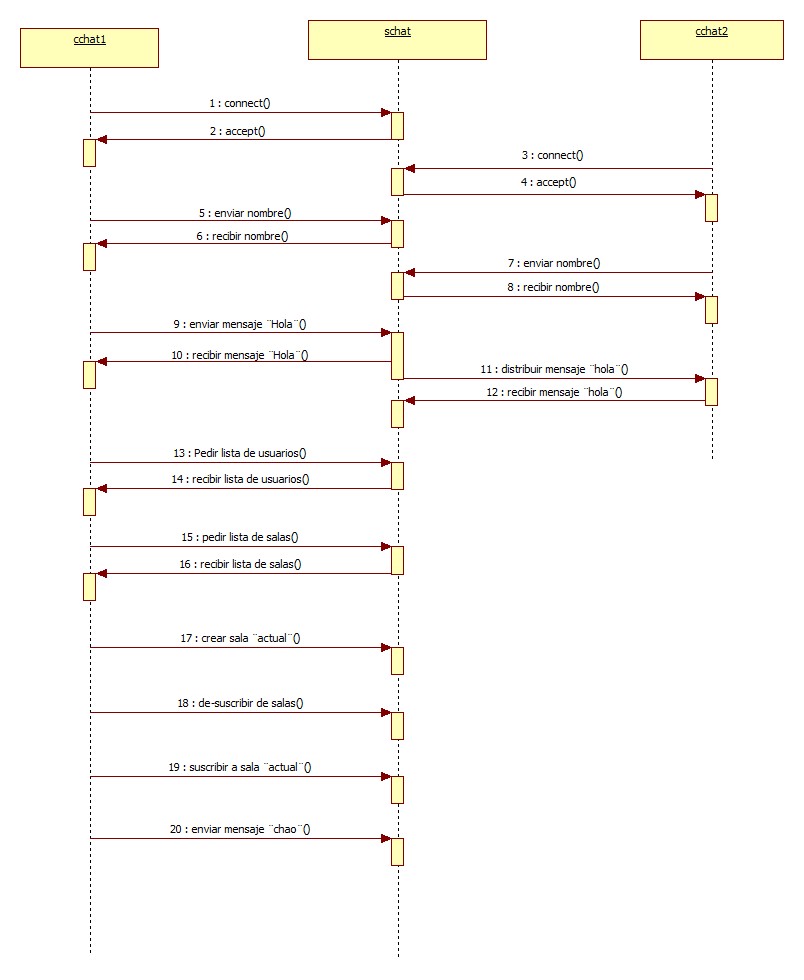
des

sus nueva

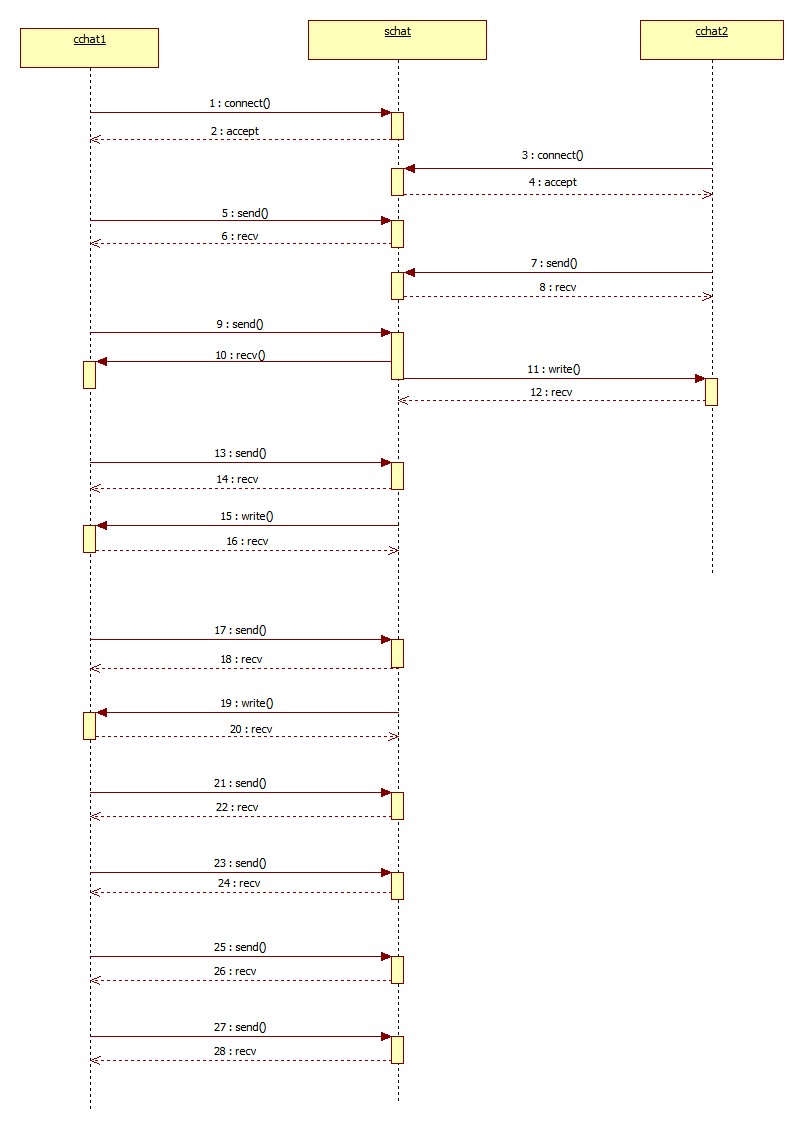
men chao

Además, se asumió que al de-suscribirse el cchat1(cliente 1) de la sala *actual y* suscribirse a la sala *nueva* el mensaje *chao* no sería distribuido por el schat (servidor) a cchat2 (cliente 2), ya que el cchat2 estaba suscrito a la sala *actual* y no a la sala creada por cchat1.

**Diagrama a alto nivel**



**Diagrama a bajo nivel**



**Descripción**

**1-2, 3-4:** conexión de los clientes (cchat1, cchat2) al servidor (schat) y aceptación por parte del servidor.

**5-6, 7-8:** envió del nombre de usuario por parte del cliente y recibimiento por parte del servidor

**9-10:** el cliente envía un mensaje con el comando *men* el servidor lo recibe y lo interpreta.

**11-12:** el servidor envía el mensaje recibido anteriormente al otro cliente y este lo recibe.

**13-14:** el cliente le envía un mensaje *usu* al servidor pidiéndole la lista de usuarios, el servidor recibe la petición.

**15-16:** el servidor le envía al cliente la lista de usuario pedida en el estado anterior. El cliente recibe la lista.

**17-18:** el cliente le envía un mensaje *sal* al servidor pidiéndole la lista de salas, el servidor recibe la petición.

**19-20:** el servidor le envía al cliente la lista de salas pedida en el estado anterior. El cliente recibe la lista de salas.

**21-22:** el cliente le envía un mensaje *cre nueva* al servidor*,*  para crear una nueva sala. El servidor recibe la petición y hace lo necesario.

**23-24:** el cliente le envía un mensaje *des* al servidor, para des-suscribirse de la sala. El servidor recibe la petición y hace lo correspondiente.

**25-26:** el cliente le envía un mensaje al servidor *sus nueva*  para suscribirse a la sala de nombre ¨nueva¨. El servidor recibe la petición.

**27-28:** el cliente envía un mensaje al servidor *men chao. E*l servidor lo recibe.(el servidor distribuiría el mensaje a otras clientes en un paso próximo, si estuvieran conectados a la misma sala)

**Aspectos funcionales**

**cchat.c:** corresponde al cliente del modelo cliente-servidor

Funciones:

* **recibir:** recibe las respuestas del servidor para mostrarle al usuario
* **enviar:** se encarga de enviar al servidor las líneas de comandos leídas por consola
* **main:** se encarga de
* validación de parámetros: se valida el orden y completitud de los parámetros.
* conexiones con el servidor: se realiza todo el manejo pertinente de sockets, para la conexión de un cliente a un servidor.
* manejo del archivo de entrada: se hace el manejo del archivo de entrada, se envía al servidor las líneas de comandos leídas de dicho archivo.
* creación y asignación de hilos: se crean dos hilos, uno para enviar información al servidor y otro para recibir las respuestas del servidor.

**schat.c:** corresponde al servidor del modelo cliente-servidor

Funciones:

* **comandos:** posee el protocolo para manejar los comandos:
* fue: el usuario envía el comando al servidor, este lo recibe y envía de vuelta al usuario un mensaje “salida” que el cliente recibe con la función “recibir”, actúa de señal para hacer pthread\_exit del hilo actual.
* sal: el usuario envía el comando al servidor, este lo recibe y envía de vuelta al usuario una lista de las salas existentes en el servidor. Si no existen salas, envía un mensaje “El servidor no posee salas en este momento”. Se maneja una lista de las salas existentes en el servidor.
* \*usu: el usuario envía el comando al servidor, este lo recibe y envía de vuelta una lista de todos los usuarios conectados al servidor. Se maneja una lista de todos los usuarios.
* men: el usuario envía el comando acompañando de un mensaje, el servidor lo recibe y distribuye el mensaje a todos los usuarios que están suscritos a la misma sala, excepto al mismo. Se indica quien envió el mensaje para que el intercambio de mensajes sea más comprensible.
* des: el usuario envía el comando, el servidor lo recibe y elimina la lista de salas que posee dicho usuario.
* cre: el usuario envía el comando acompañado del nombre de la sala, el servidor lo recibe y la agrega a la lista de salas de dicho usuario y se le notifica que se ha creado exitosamente. Si el argumento es vacío, se envía un mensaje al usuario notificándole el error.
* eli: el usuario envía el comando acompañado del nombre de la sala que desea eliminar, si la sala esta en las listas de salas de los usuarios se elimina de todas y se le informa al usuario que la sala ha sido eliminada exitosamente, si no se notifica que la sala no existe.
* sus: el usuario envía el comando acompañando del nombre de la sala, el servidor lo recibe y busca la sala en las listas, si la encontró la agrega a la lista de salas del usuario, si no le notifica al usuario que la sala no fue encontrada.
* connection\_handler: maneja las conexiones entrantes de cada cliente, se encarga de recibir los mensajes que estos quieren enviar y de despacharlo a la función “comandos” donde se lleva a cabo el protocolo para comando. También se encarga de cerrar el socket perteneciente a cada cliente, al terminar.
* main:
* validación de parámetros: valida la completitud y correctitud de los argumentos de entrada. El parámetro “sala” es opcional, por defecto la sala es “actual”.
* Conexiones con los clientes: se maneja y administra las conexiones entrantes de clientes al servidor, luego de hacer bind y listen, el servidor queda en un estado de espera por las conexiones de los clientes, por cada cliente crea un hilo para manejar el proceso.

**listas.c, listas.h:** este archivo contiene todas las funciones necesarias para el manejo de la estructura lista, la cual es usada en la implementación de nuestro proyecto.

Funciones:

* crearLista
* insertarNodo
* eliminarNodo
* iterador
* getElem
* getUsuario
* imprimir
* eliminarLista
* insertarSala
* buscarSala
* borrarSala
* borrarSalas
* imprimirSalas
* buscarUsuario
* getSala
* buscarEnSalas

**makefile:** archivo requerido para la compilación.

**Aspectos no funcionales**

\*Comando “usu”: no tiene el comportamiento esperado, algunas veces muestra los usuarios en el servidor y otras no. Tambien pudimos observar que en algunos casos imprimia basura. Creemos que es manejo inapropiado de memoria.