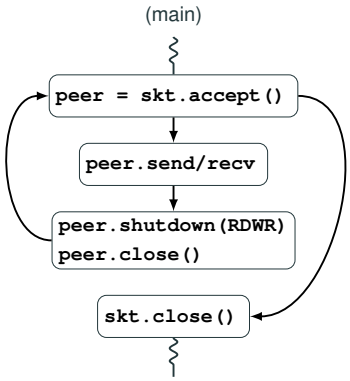
Servidores multi-cliente (draft)

Di Paola Martín
martinp.dipaola <at> gmail.com

Facultad de Ingeniería
Universidad de Buenos Aires

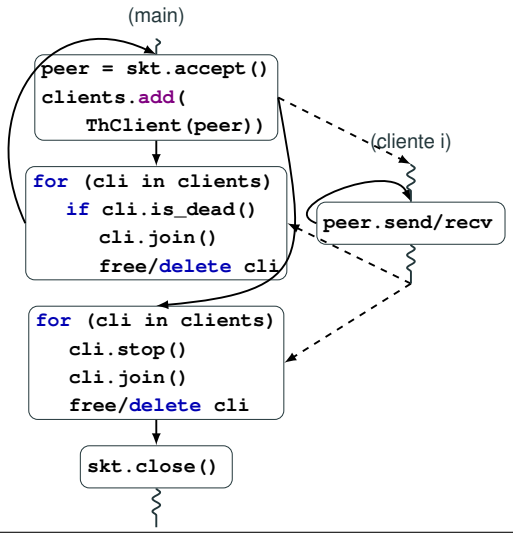
Servidores multi-cliente (draft)

Simple: servidor para un solo cliente a la vez



- El programa es un bucle simple, se espera una conexión remota, se la procesa y se repite el proceso
- Solo cuando `accept` falla por recibir una señal, el bucle finaliza.
- Este esquema solo soporta un cliente a la vez y no permite hacer cosas en paralelo mientras se espera a un nuevo cliente ni mientras se habla con uno ya conectado.
- Es muy simple. Usado en servidores RPC y servidores Web simples (o dummies).

Servidor multi-cliente (draft)



Frenar un hilo

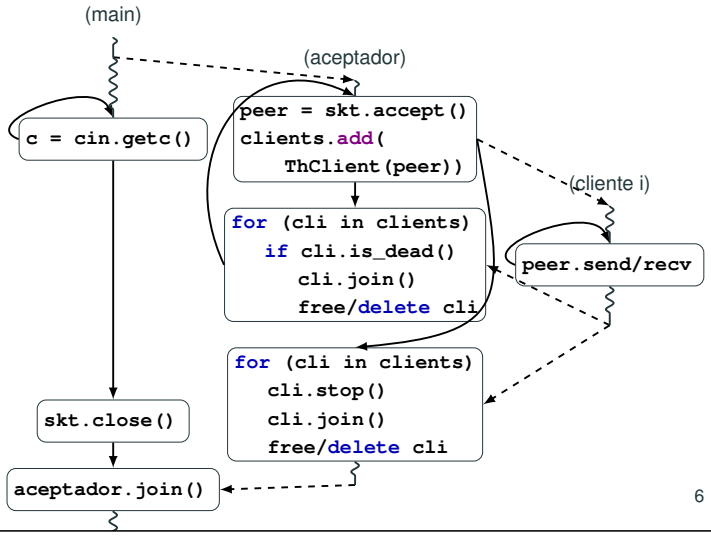
```
1  class ThClient:public Thread {
2     bool keep_talking;
3     Socket peer;
4
5     public:
6     virtual void run() {
7        while (keep_talking) {
8           ...
9           peer.send(...);
10          ...
11          peer.recv(...);
12          ...
13       }
14    }
15
16    void stop();
17 };
```

```
1  // Violento pero efectivo
2  void ThClient::stop() {
3     keep_talking = false;
4     peer.shutdown();
5     peer.close();
6  };
```

```
1  // Polite pero peligroso
2  void ThClient::stop() {
3     keep_talking = false;
4  };
```

- Como frenar un hilo? La librería **`pthread`** ofrece una manera genérica de frenar o matar a un hilo (stop/kill) pero deja los recursos sin finalizar. No usar.
- Frenar un hilo correctamente dependerá de la naturaleza del hilo (depende de la aplicación en cuestión, no hay una solución general).
- Hay dos variantes posibles: forzar un cierre o decirle al hilo que cuando pueda él mismo finalize.
- Si el hilo esta bloqueado en una operaciíon de sockets, se puede hacer un **`shutdown/close`** del socket para forzar un cierre. Obviamente si el hilo estaba en el medio de una comunicación, el trabajo puedo quedar trunco o corrupto.
- Usar un **`bool`** para decirle al hilo que finalize. El hilo puede terminar su conversación y cerrar ordenamdamente: es más seguro pero si el hilo está bloqueado no se desbloqueará.
- Y si el hilo esta bloqueado en otra operación? Como en un **`queue_safe.pull()`**? No hay una solución genérica.

Servidor multi-cliente (final): cierre ordenado



Hilos de comunicación

- Un socket puede ser leido (**`recv`**) por un hilo y escrito (**`send`**) por otro. La lectura no entra en conflicto con la escritura.
- Pero hacer un **`send`** (o un **`recv`**) sobre un mismo socket desde múltiples hilos trae problemas de concurrencia!

Mantra

- Detectar que acciones son bloqueantes o potencialmente lentas:
  accept, send, recv, join, fwrite, fread
- Por cada acción bloqueante o potencialmente lenta preguntarse,
  si el hilo se bloqueara o tardara mucho, que podría hacer en simultaneo?
  - hacer un request (send) a http://fi.uba.ar y esperar (recv) a recibir la pagina web para renderizarla.
  - mientras se hace se procesa un request, el usuario interactúa con la interfaz gráfica.

Ejemplo: Sala de Chat

Aplicación por consola con multiples clientes conectandose a una sala de chat.

- Draft inicial? Cuáles son los features a implementar?
- Puedo recibir mensajes sin yo tener que enviar? Ejemplo de un send independiente de un recv.
- Objetos compartidos en el cliente? y en el servidor?
- Thread Safe Queue como alternativa para la comunicación entre hilos.
  - hacer un request (send) a http://fi.uba.ar y esperar (recv) a recibir la pagina web para renderizarla.
  - mientras se hace se procesa un request, el usuario interactúa con la interfaz gráfica.

Proxies

```
1  ChatSystem cs (/*...*/);
2
3  Room r = cs.join("#taller");
4  r.write("hi everyone!");
5
6  r.leave();
```

```
1  struct ChatSystem {
2      Room open(roomname) {
3          skt.send("JOIN");
4          skt.send(roomname);
5
6          id = skt.recv(1);
7          return Room(id);
8      }
9
10 struct Room {
11     write(txt) {
12         skt.send("WRITE");
13         skt.send(id);
14         skt.send(txt.sz());
15         skt.send(txt.str());
16     }
```