Client Design

Srinidhi Varadarajan

Topics

- **Concurrency in client**
  - **Concepts**
  - **Approaches**
- **TCP timed echo example**

Why Use Concurrency in *Servers* ?

- **Improved response time**
- **Can be used to eliminate deadlocks**
- **Simplifies implementation of multiprotocol and multiservice servers**
- **Threads work on uniprocessors, but can take advantage of multiprocessors**

Except for multiprocessor execution, none of these reasons directly applies to *clients.*

Why Use Concurrency in *Clients* ? (1)

- **Can separate functionality into distinct components, with advantages for code design and maintenance**
  - **Requester (sends requests)**
  - **Receiver and processor**
  - **User interface**
  - **Control**
- **Client can simultaneously contact multiple servers**
  - **Distributed search**
  - **Compound documents with elements on multiple servers**

Why Use Concurrency in *Clients* ? (2)

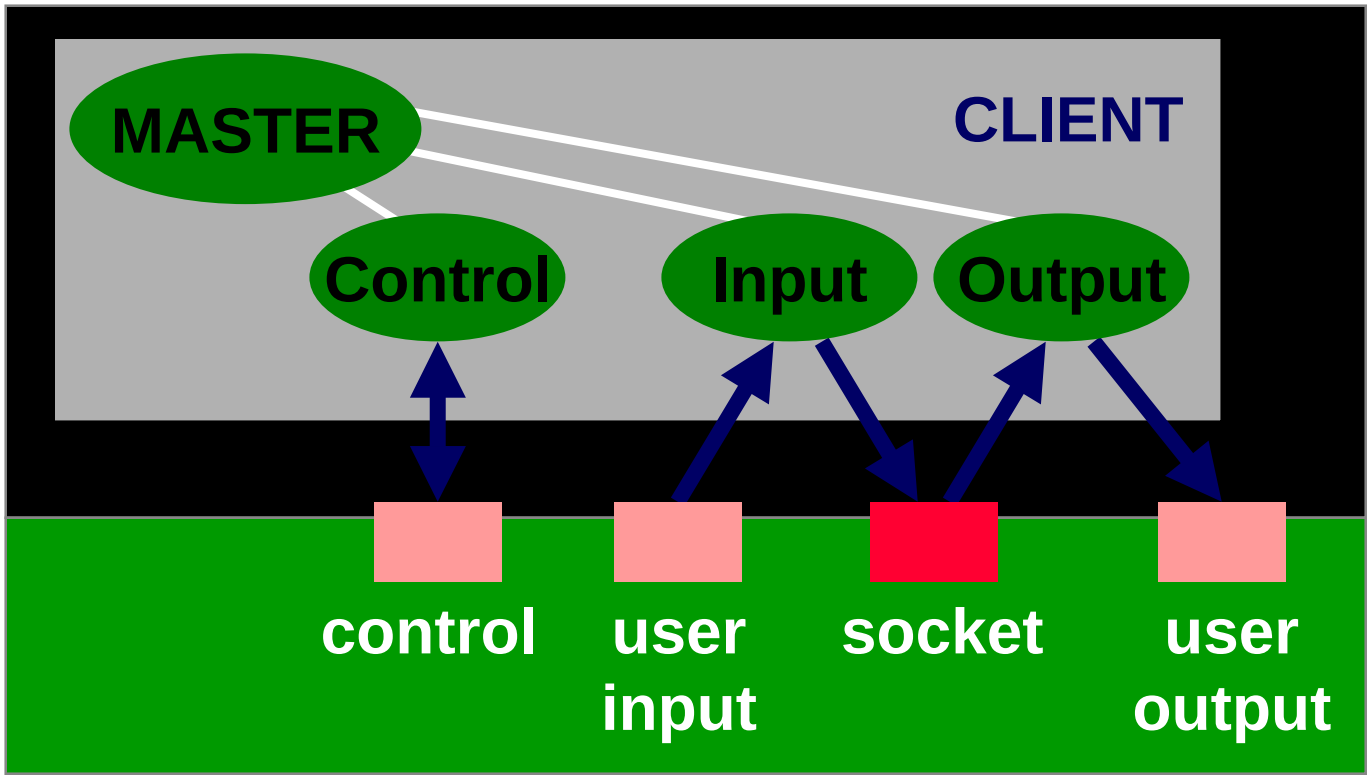
- **Allows interaction while a request is in progress**
  - **Status checks**
  - **Abort operation**
  - **Modify parameters**
- **Potential performance advantage for overlapping operations**
  - **Processing, file I/O, and network I/O**
  - **Overlap operations on multiple connections**
- **Provides asynchrony**
  - **Set of multiple tasks can be performed without the imposition of a strict ordering**

Implementing Concurrency in Clients

- **Two approaches (as for servers)**
  - **Multiple threads, using pthread_create()**
  - **Apparent concurrency, using select()**
- **Multiple threads**
  - **Each thread performs a distinct set of tasks, or**
  - **Each thread performs a separate request or other task, or**
  - **Some combination of the above**
- **Apparent concurrency**
  - **Single thread uses select() for asynchronous I/O**
  - **Time-outs should be included to prevent client deadlock**

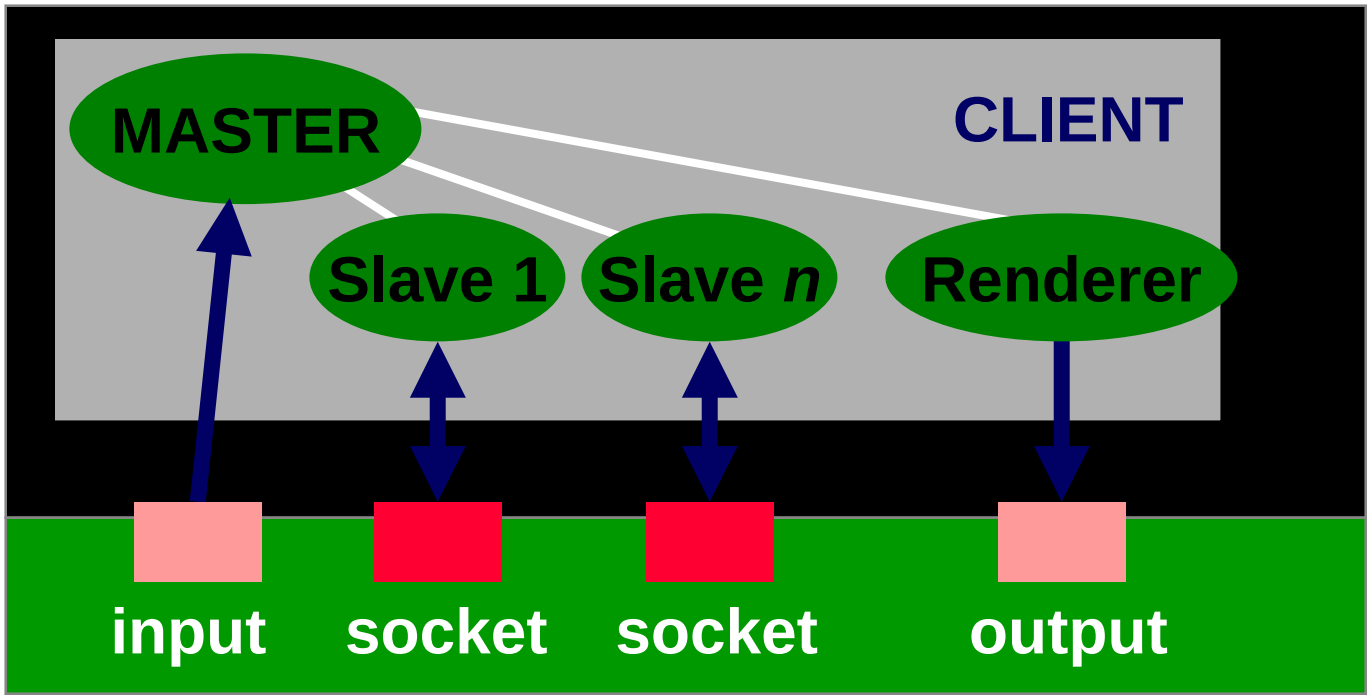
Multithreaded Client (1)

- **Single network socket (TCP or UDP)**
- **Functional decomposition**



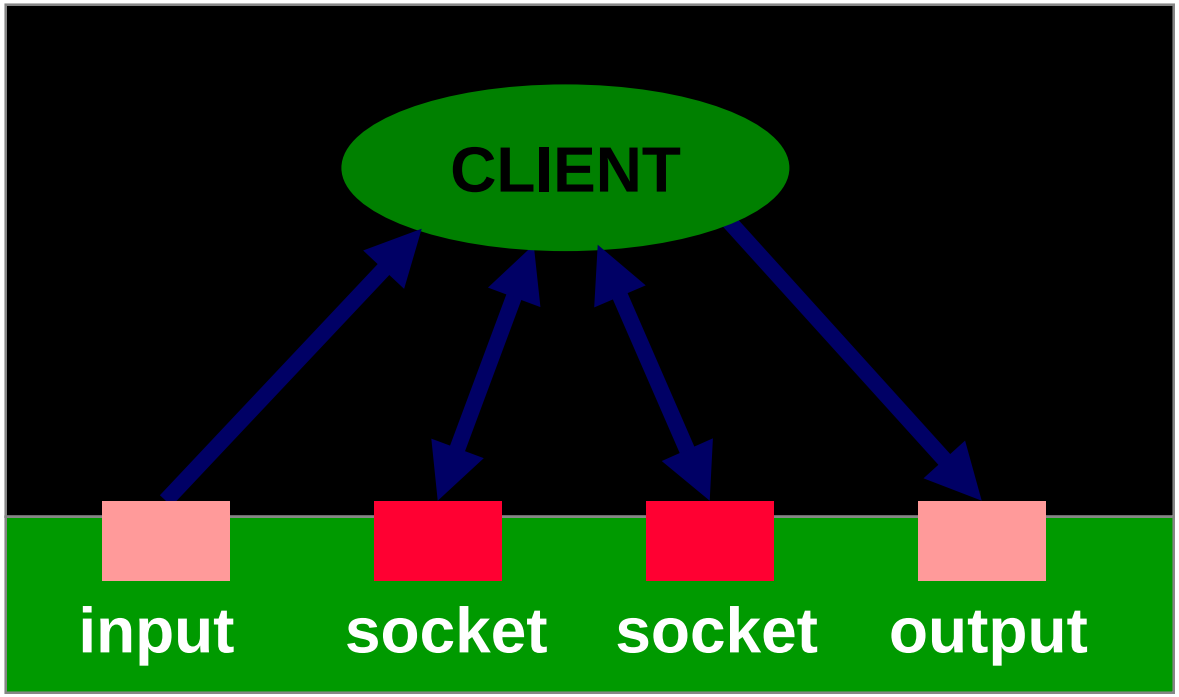
Multithreaded Client (2)

- **Multiple network sockets**
- **Hybrid approach, since there is also functional decomposition**



Single-Threaded Concurrent Client

- **Single thread uses select() call to find active socket and file descriptors**
- **Decomposition by socket and functions**



TCPtecho Example (1)

- **TCPtecho**
  - **Single client that accesses multiple servers (in this case, ECHO servers)**
  - **Utility is to simultaneously measure network throughput between the client and multiple servers**
- **Basic tasks**
  - **Make connections to each server -- main()**
  - **Send data until all data is sent -- writer()**
  - **Receive data until all data is received -- reader()**

TCPtecho Example (2)

- **writer()**
  - **For a given host …**
    - **Send as much data as possible up to total amount to send**
    - **Reduce amount left to send by amount actually sent**
    - **If all is sent, shutdown connection for send with shutdown()**
  - **writer() called when a socket is ready for send()**
  - **Since data to be sent may be larger than what can be sent, sockets are set to “non-blocking” to ensure that send() won’t block**
    - **ioctl(fd, FIONBIO, &one)**

TCPtecho Example (3)

- **reader()**
  - **For a given host …**
    - **Receive as much data as possible, up to buffer size**
    - **Reduce amount received from amount to receive**
    - **If all is received close the connection with close()**

ioctl()

- **ioctl(socket, command, arg_ptr)**
- **Commands**
  - **FIONBIO: enable non-blocking mode**
  - **FIONREAD: determine amount of data pending in the network's input buffer**
  - **SIOCATMARK: determine whether or not all out of band data has been read**
- **In TCPtecho**
  - **u_long one = 1**
  - **ioctl(fd, FIONBIO, &one)**

Getsockopt() and Setsockopt()

- **setsockopt() and getsockopt() also used to monitor and control socket operation**
- **For example, to force TCP to immediately send data**

```
int optval = 1;
setsockopt( sock, IPPROTO_TCP,
  TCP_NODELAY, (const char *)
  &optval, sizeof(int));
```

You should now be able to …

- **Describe the need for concurrency in a client**
- **Describe approaches to making a client concurrent**
- **Analyze and design a simple concurrent client**
- **Use ioctlsocket() to control socket options**