Chapter 8
Server Software Issues
Iterative vs. Concurrent

Principle: one long request should not block short requests.

Principle: a long interactive connection (a telnet login) should not prohibit other connections.

Observation: iterative servers are simpler.

Design choice: use an iterative server when all requests are short (like daytime).

Design choice: use a concurrent server when you have interaction with a user.

Design choice: use a concurrent server when the request takes a long time, (file transfer protocol).

Conection-Oriented vs. Connectionless

Problems: missing data, duplication of data, order of data

TCP: protocol handles the problems

UDP: client and/or server must handle the problems

TCP advantage: server is simpler to write

TCP disadvantage: slower, connection usually closes on error

UDP advantages: low protocol overhead, flexible handling of problems

UDP disadvantage: server is more complex to write (correctly)

TCP: point-to-point communication. Client-server format.

UDP: many-to-many communication. Allows broadcast, multicast and peer-to-peer.

TCP: stream, no boundary preservation

UDP: message (datagram), one send-one recv.

Stateless vs. Intelligent

Observation: clients and servers crash

Principle: Don't trust the other program (it may be gone)

Stateless: don't rely on past transactions

Client must repeat a full command
ex: give me block 3 of `/etc/termcap`

Server must sent a full reply
ex: here is block 4 of `/etc/passwd`

Note: past information may be used for caching
(this does not violate stateless)

Of course, the program still needs to handle what happens if the program itself (or the machine it is on) crashes.

Robustness

Observation: What can go wrong will go wrong

Examples:
-file system full
-process table full
-out of memory (swap space)
-client or server involved in a crash

Principle: Plan to survive

Examples of "rules"

One client crashing shouldn't lock the entire server
-Server should still handle other/future clients.

Server crashing should cause the client to exit.
-Do not hang or infinite loop.
-OK to sleep, then retry, but after a while–give up.

The server should carefully allocate or write.
-Either report the error to client or retry.

TCP: Master-Slave Paradigm

In a TCP server there are two types of sockets.

Master Socket:
The result of the socket.
Used only for the accept.
Cannot be used to talk to a client.

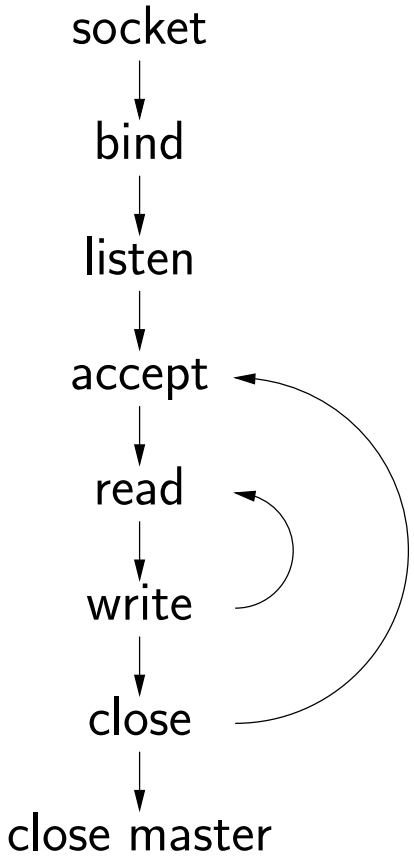
Slave Socket:
The result of an accept.
Used only to talk to a client.
Cannot be used for the accept.

UDP server

There is only one type of socket.
Think post-office box.
There is no accept.
This socket is used for send and receive.

Iterative, Connection-Oriented Server

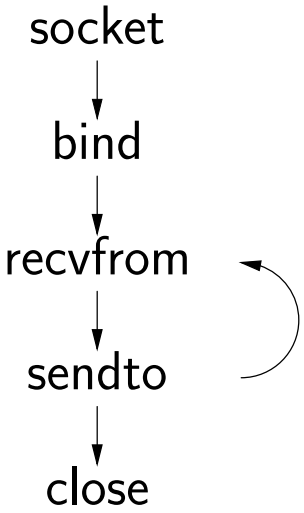
Algorithm 8.1



Chapter 10 gives detailed code

Iterative, Connectionless Server

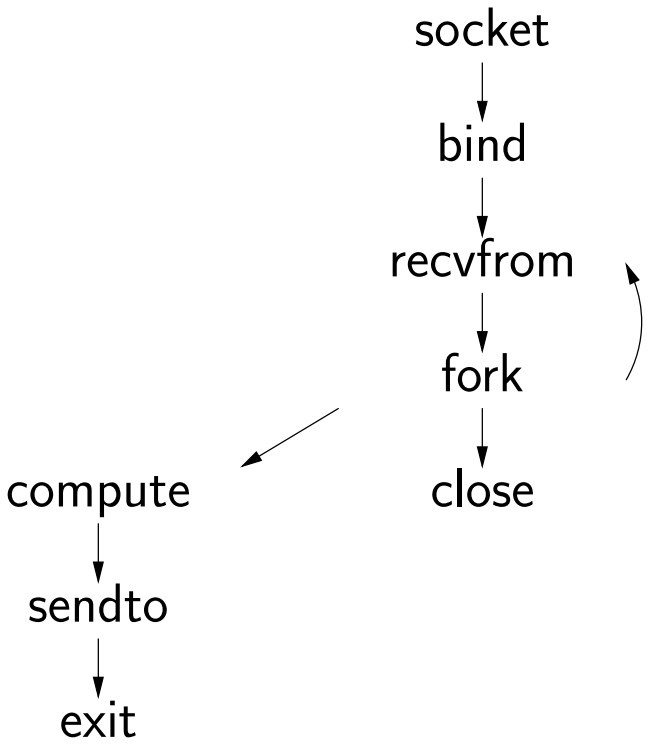
Algorithm 8.2



Chapter 9 gives detailed code

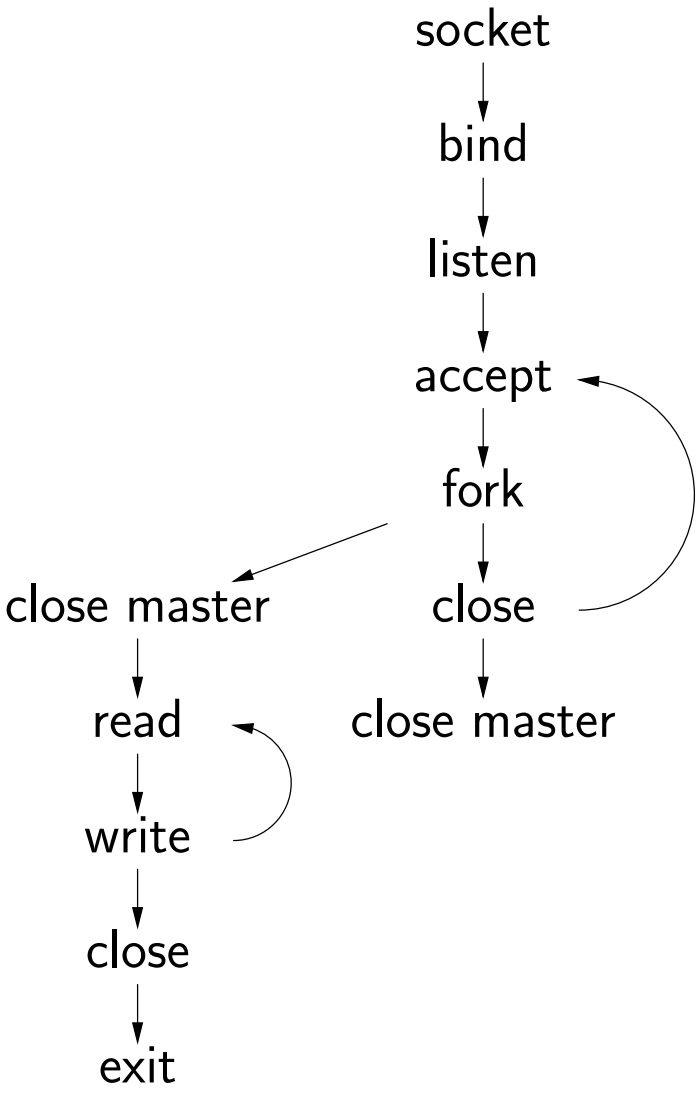
Concurrent, Connectionless Server

Algorithm 8.3



Concurrent, Connection-Oriented Server

Algorithm 8.4



Chapter 11 gives detailed code

TCP Deadlock

You have a limited amount of network buffer. If you exceed your buffer; UDP drops messages.

TCP is not allowed to drop data, it must buffer. If the receiver runs out of buffer, the sender is unable to write. When this happens `write` call on the sender block. If both the client and the server write before reading, deadlock can occur.

Suppose both the client and server write a 32Mb message, then read. With the default TCP options, this usually deadlocks.

Multihomed and Machine Portable Code

When the server calls `bind` it specifies an address, including a port and an internet address.
This port and address must be available.
Usually `INADDR_ANY` is specified which allows the program to listen on all network interfaces of the server.

Alternate: specify an internet number such as `134.139.248.17`.
The program the only listens on the interface with that number.
Used in firewalls. Allows connection only from the inside or the outside of the firewall.