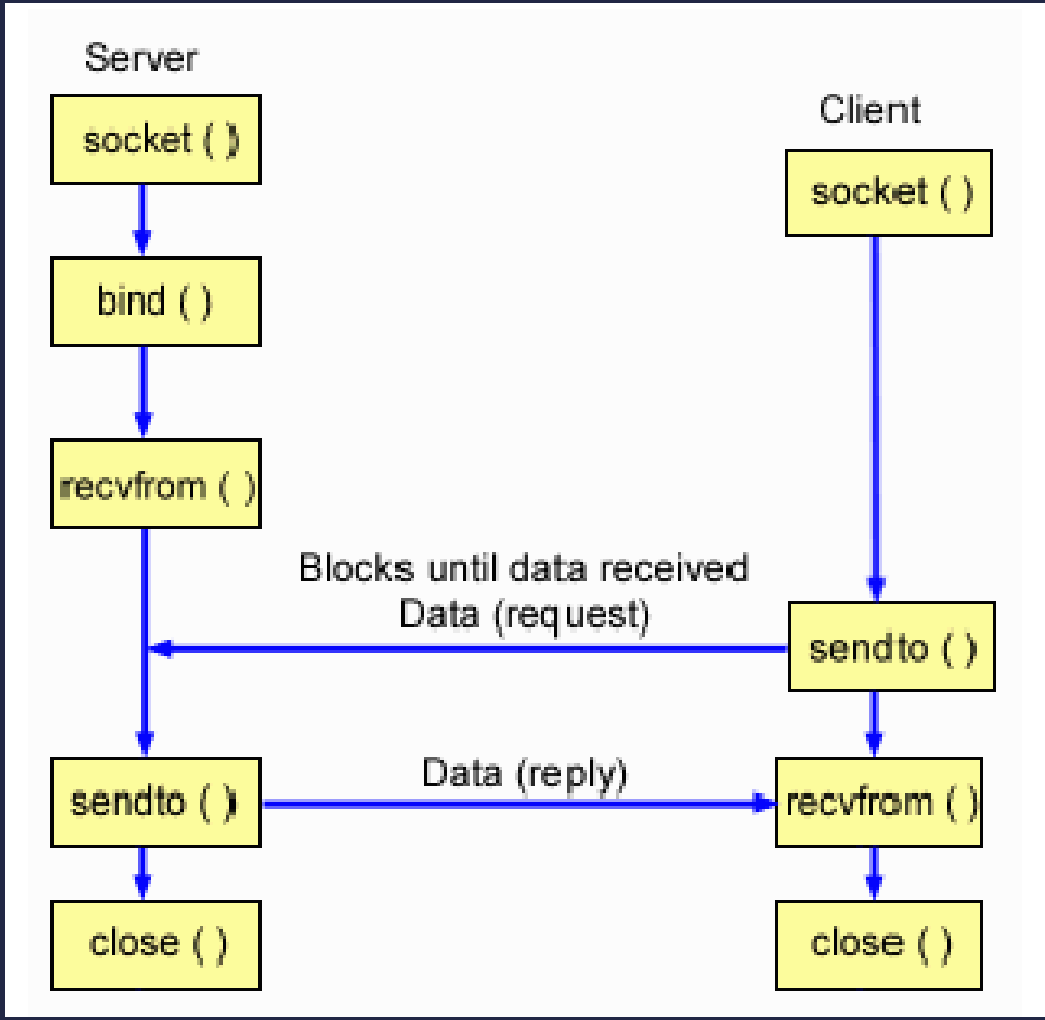
Elememtary UDP Sockets

Lecture 7

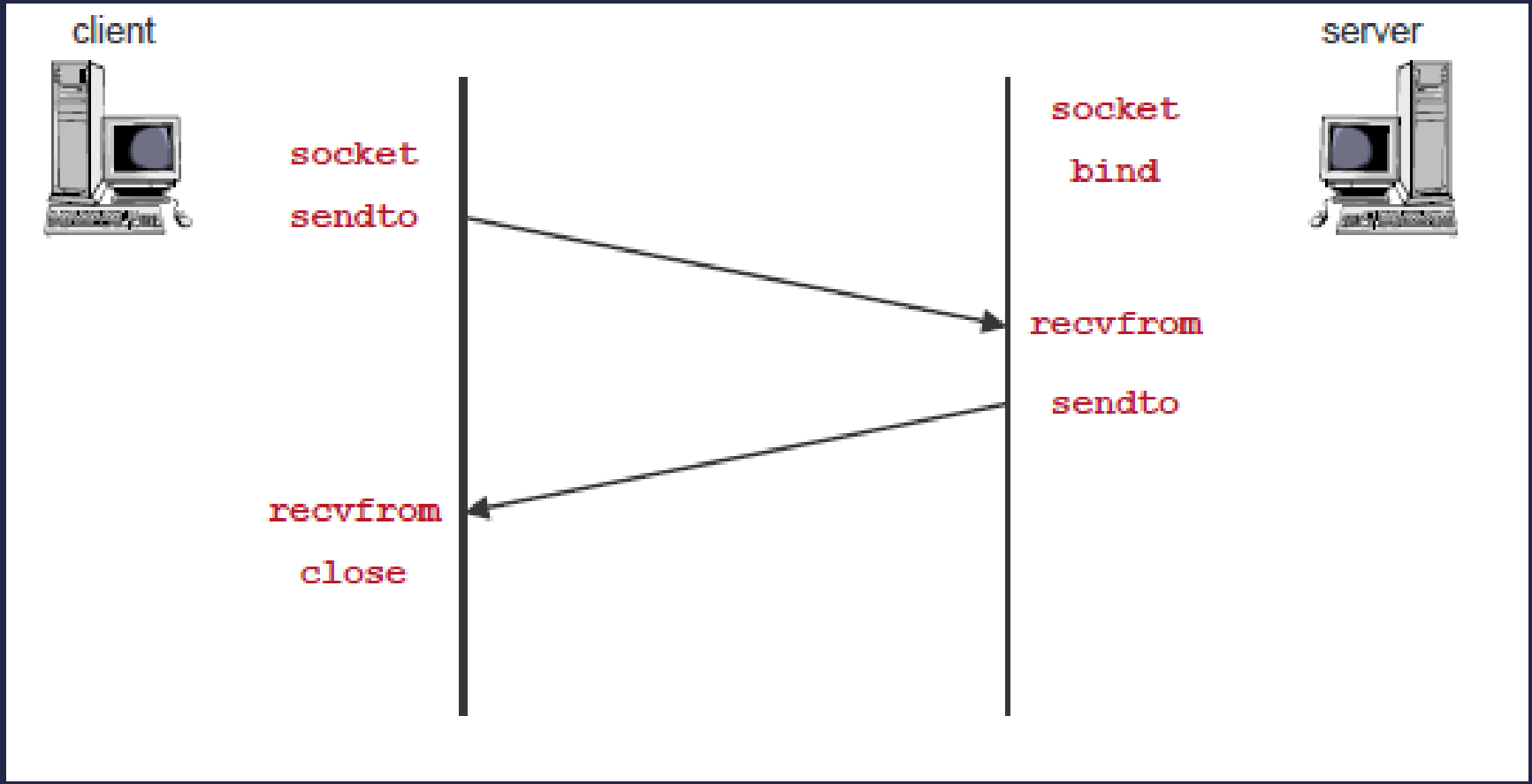
Introduction

- Connectionless
- Unreliable
- Datagram protocol

Connectionless socket – 1/3



Connectionless socket – 2/3



Connectionless socket – 3/3

- Client does not establish connection with the server
- Client just sends a datagram to the server using sendto() which requires the address of the destination (server) as a parameter
- Server does not accept a connection from a client
- The server just calls recvfrom() which waits until data arrives from some client
- The recvfrom() returns protocol address of the client, along with datagram, so server can response to correct client

recvfrom() and sendto() – 1/6

- Syntax

```
#include <sys/socket.h>

ssize_t recvfrom(int sockfd, void *buff, size_t nbytes, int flags,
struct sockaddr *from, socklen_t *addrlen);

ssize_t sendto(int sockfd, const void *buff, size_t nbytes, int
flags, struct sockaddr *to, socklen_t addrlen);
```

- Both return number of bytes read or written if successful
- Return -1 on error

recvfrom() and sendto() – 2/6

- First 3 args : *sockfd*, *buff*, and *nbytes* are identical to first 3 args for read() and write() i.e. descriptor, pointer to buffer to read into or write from, and no of bytes to read or write
- *flags* will be set to 0
- The *to* arg for sendto() is socket address structure containing protocol address (IP address and port no) of where data is to be sent
- Size of socket address structure is specified by *addrlen*

recvfrom() and sendto() – 3/6

- The recvfrom() fills in the socket address structure pointed to by *from* with the protocol address of who sent the datagram
- Number of bytes stored in this socket address structure is also returned to the caller in the integer pointed by *addrlen*
- *Note:The final argument in sendto() is int, while in recvfrom() is pointer to int (value-result)*

recvfrom() and sendto() – 4/6

- The final 2 arguments of recvfrom() are similar to final 2 arguments of accept()
  - Contents of socket address structure upon return tells us who sent the datagram (UDP) or who initiated the connection (TCP)

- Final 2 arguments of sendto() similar to connect()
  - Fill in the socket address structure with protocol address of where to send the datagram (UDP) or with whom to establish connection (TCP)

recvfrom() and sendto() – 5/6

- In typical use of recvfrom(), with datagram protocol, return value is amount of user data in the datagram received
- **Writing datagram of length 0 is acceptable**
- In case of UDP, this results in IP datagram containing IP header, UDP header, and no data
- **Return value of 0 is acceptable for datagram protocol, does not mean that peer has closed connection**
- Since UDP is connectionless, there is no such thing as closing UDP connection

recvfrom() and sendto() – 6/6

- If *from* argument to recvfrom() is NULL, then corresponding length argument (*addrlen*) must also be NULL, indicating that we are not interested in knowing the protocol address of who send data

UDP echo server and client – 1/3

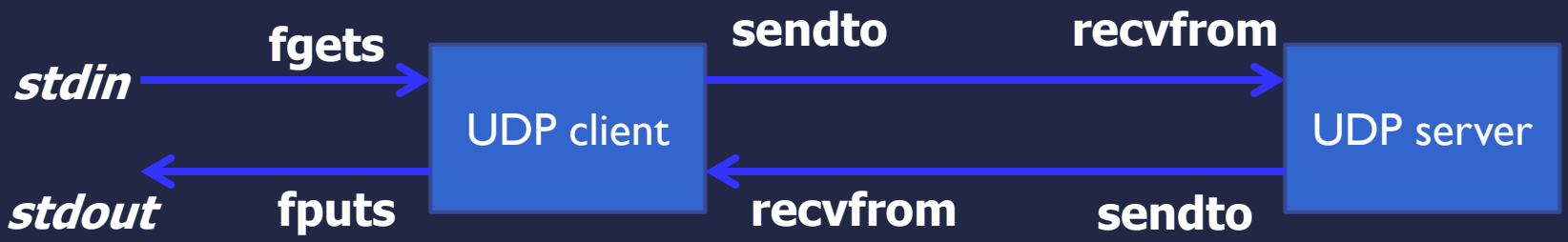


Fig: A simple client/server using UDP

UDP echo server and client – 2/3

- In the server program : dg_server()
  - Never terminates
  - Provides iterative server
  - No call to fork()
- Single server process handles any and all clients
- Most UDP servers are iterative

UDP echo server and client – 3/3

- Each UDP socket has a receive buffer
- Each datagram that arrives for this socket is placed in that socket receive buffer
- Buffer has limited size, but could be increased with SO_RCVBUF socket option

Example – UDP echo client/server program