CSC552 – Advanced UNIX Programming

Sockets

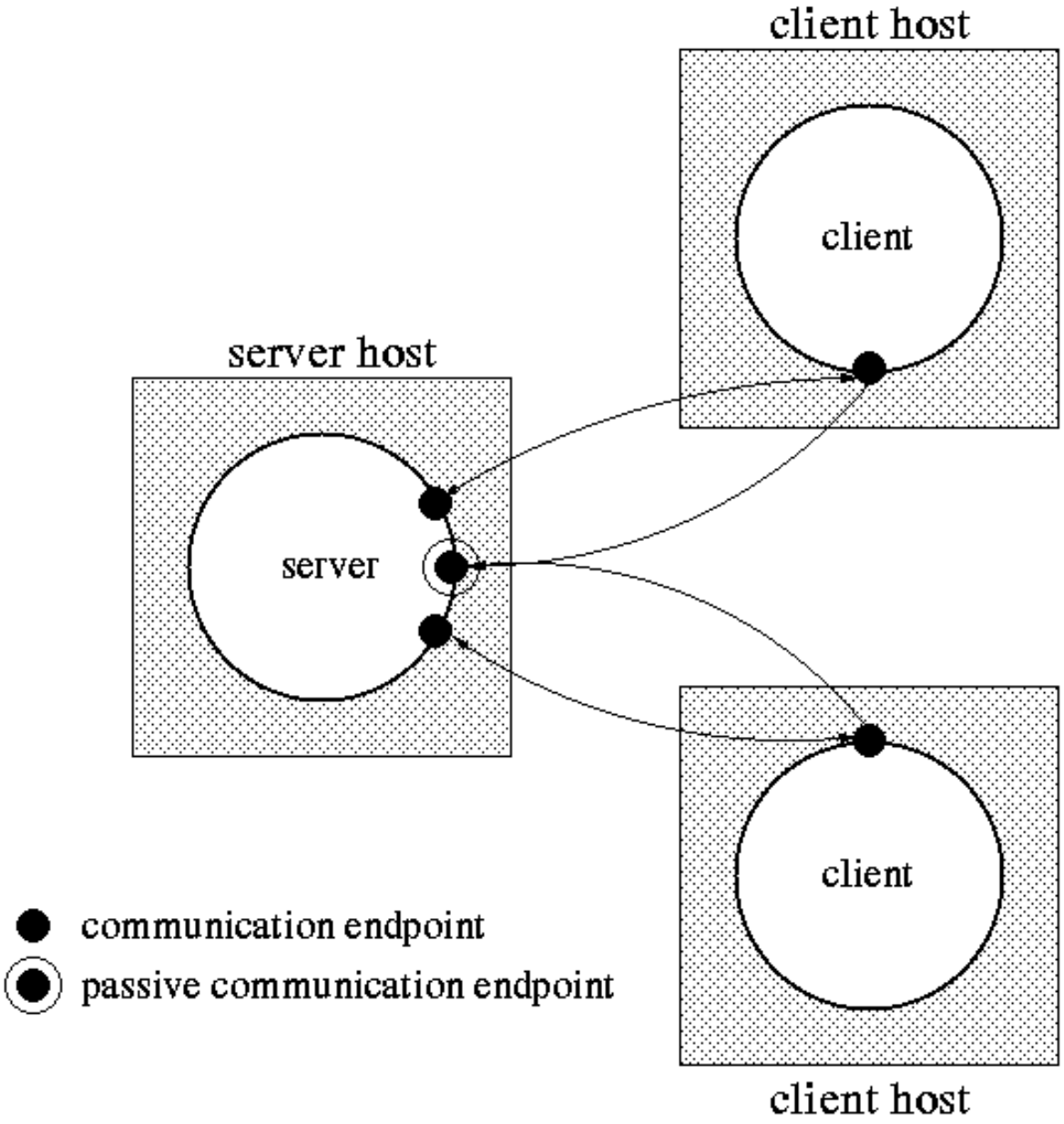
Dr. L. Frye
Kutztown University

Client/Server Communication

- How can processes on the same machine communicate with each other?

- How can processes on different machines communicate with each other?

Connection-Oriented Communication



Sockets

- Bidirectional
- Client/Server model
- IP address – port number

Server Pseudocode

```
for ( ; ; )  {
    wait for client request on listening file des
    create private 2-way channel to client
    while (no error on channel)   {
        read from client
        process request
        respond to client
    }
    close file desc
}
```

- What is wrong with this code?
- How can it be improved?

Modified Server and Client

- for (; ;) {
  wait for client request on listening file des
  create private 2-way channel to client
  fork child to handle client
  close channel
  clean up zombie children
  }
  - close listening file desc
    handle client
    close communication for channel
    exit

Questions

- What happens if the parent doesn't close the file descriptor?
- What is a zombie process?
- What happens if the parent doesn't clean up after zombie children?

Thread Approach

- What are the advantages of this approach?
- Disadvantages?
- for (; ;) {

```
    wait for client request on listening file des
    create private 2-way channel to client
    create detached thread to handle client
}
```

- Why doesn't the parent close the file descriptor in this case?
- Why create a detached thread?

Other Options

- Fixed number of children
- Pool of work threads

- Universal Internet Communication Interface (UICI) library

Socket Address Structures

- sockaddr
- sockaddr_in
  - in_addr
- sockaddr_un

- Passed by reference
- Generic address → cast specific one

Endian

- Gulliver's Travels
- How values are represented
- Little-endian vs. Big-endian
- int num = 91329;
- Hex value?

- Big-endian: 00 01 64 C1
- Little-endian: C1 64 01 00

Network Representation

- Big-endian
- *Network byte order*

- Functions
  - htonl
  - htons
  - ntohl
  - ntohs

Socket Connection

- Passive open (server)
  - socket()
  - bind()
  - listen()
- Active open (client)
  - connect()

- Close
  - Active
  - Passive
  - close()

Server Actions

- socket()
- bind()
- listen()
- accept()

Address and Host Functions

- gethostname()
- gethostbyname()
  - gethostbyname_r()
  - getnameinfo()
- char *hostn = “usp.cs.utsa.edu”;

```
struct hostent *hp;
struct sockaddr_in server;
if ((hp = gethostbyname(hstn)) == NULL)
        fprintf(stderr, “Failed to resolve host name\n”);
else
        memcpy((char *)&server.sin_addr_s.addr,
                                hp->h_addr_list[0], hp->h_length);
```

Address and Host Functions

- addrinfo structure
- gethostbyaddr
  - gethostbyaddr_r()
  - getaddrinfo()
- struct hostent *hp;

```
struct sockaddr_in net;
int sock;
if ((hp = gethostbyaddr(&net.sin_addr,
                                        4, AF_INET))
    printf("Host name is %s\n", hp->h_name);
```

Address and Host Functions

- inet_addr
- inet_aton
- inet_ntoa

Client Actions

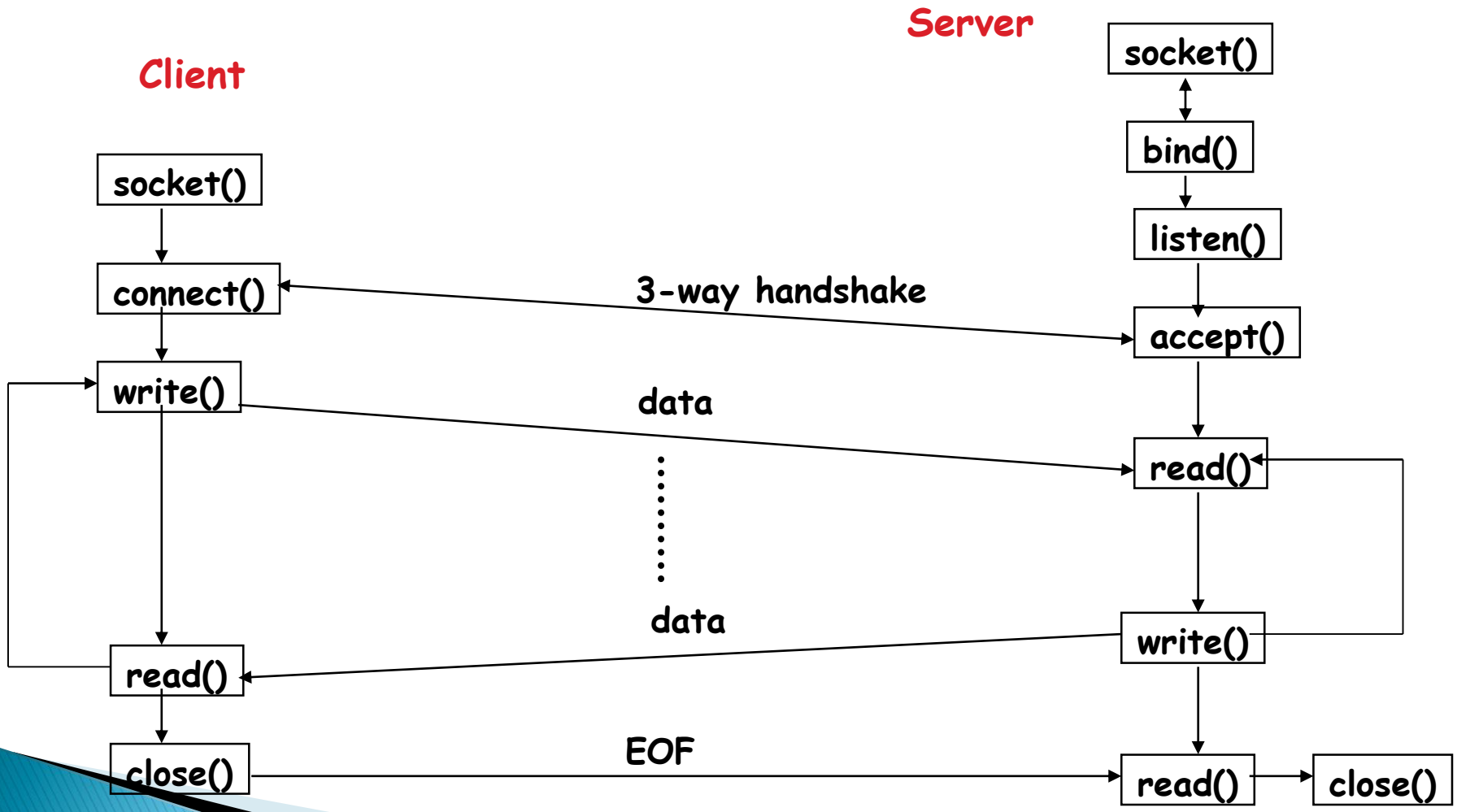
- socket()
- connect()

Transfer Data

- read()
- write()
- recv()
- send()

- Library functions
  - readn()
  - writen()
  - readline()

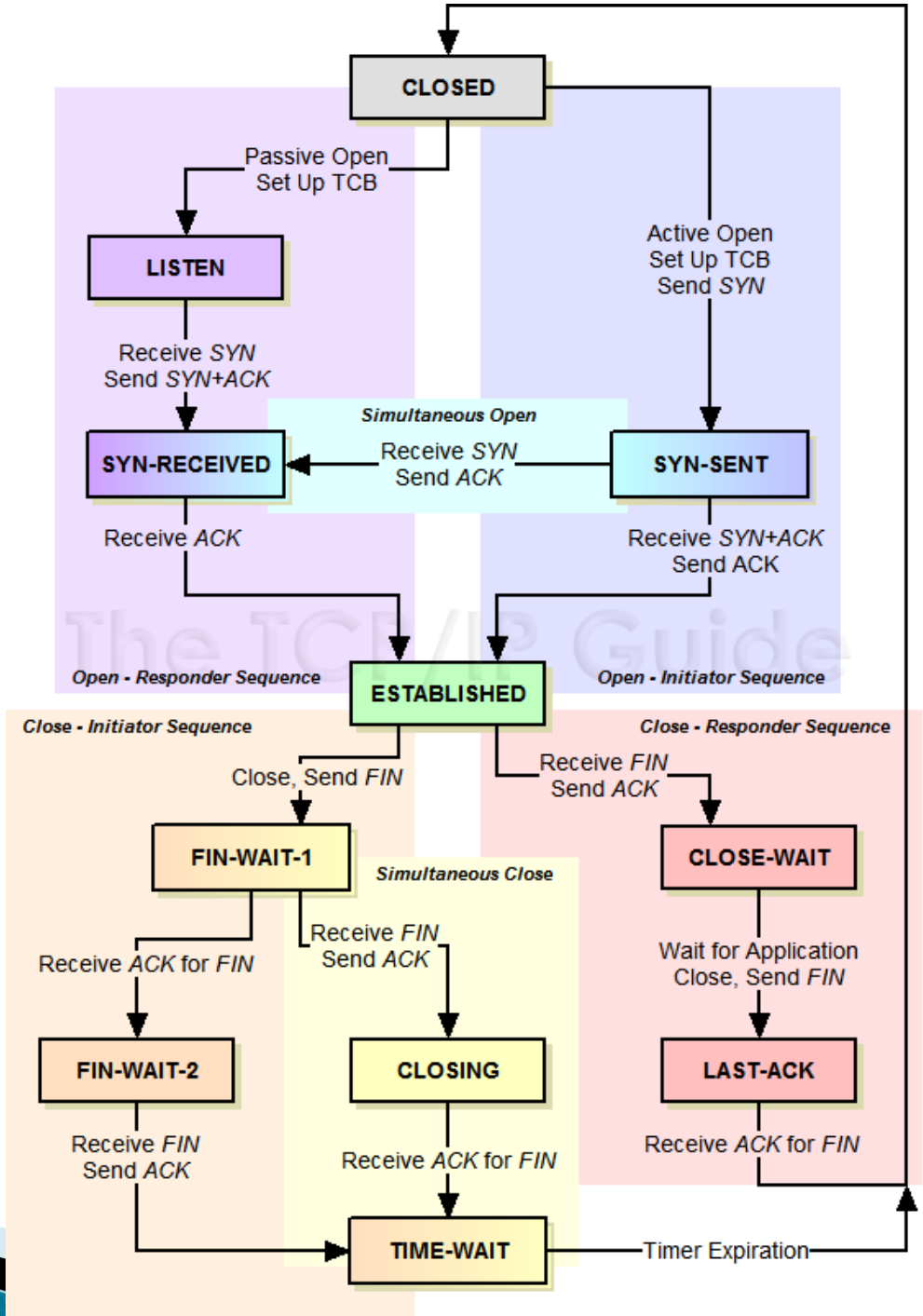
Flowchart


Server
Client
socket()
bind()
listen()
socket()
connect()
3-way handshake
accept()
write()
data
read()
data
write()
read()
close()
EOF
read()
close()

Socket Examples

- sockets/TCPsockets/tcpcliserv
- sockets/TCPsockets/tcpcliserv2
- sockets/TCPsockets/sum
- sockets/TCPsockets/tcpserv_signals

TCP Finite State Machine



Concurrent Server

- Server must handle multiple requests

UDP Sockets

- Datagram
- How is socket programming over TCP different from socket programming over UDP?
- What are the advantages of connectionless sockets?
- Disadvantages?

- Simple request protocol
- Request-reply protocol

Programming UDP Sockets

- No calls for
  - listen()
  - accept()
  - connect()
- Do need
  - socket()
  - bind()
- Read and Write
  - sendto()
  - recvfrom()

- sockets/connectionless