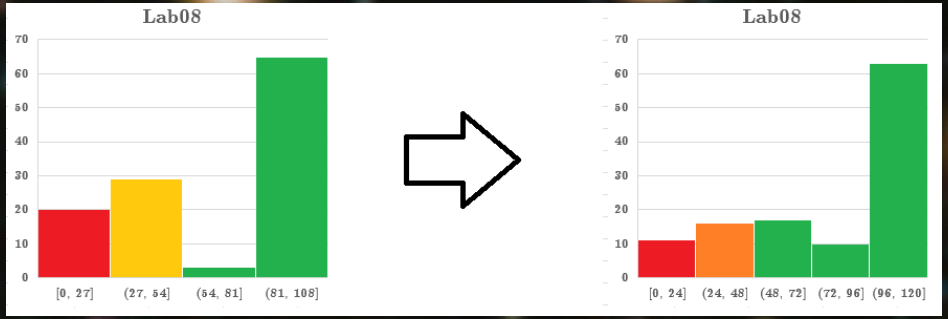
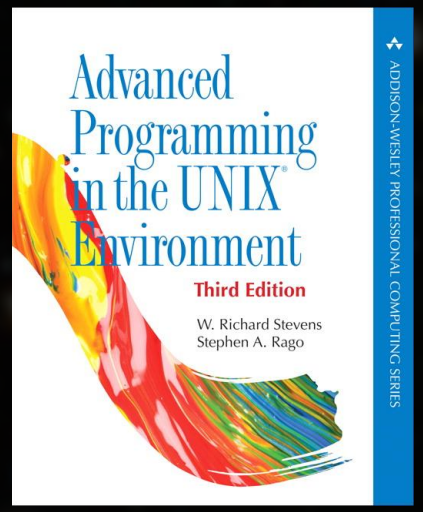
Knife in the Water (1962), Roman Polanski

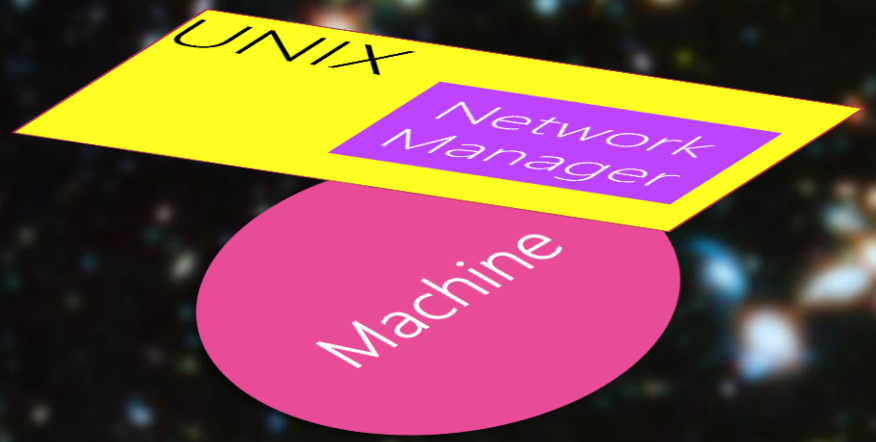
Lab11 & Lec 11 Extended by One Week

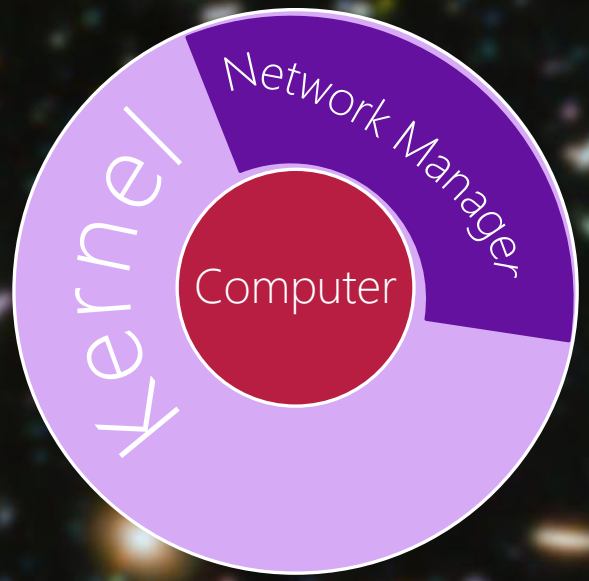
Lab08 has been remarked.

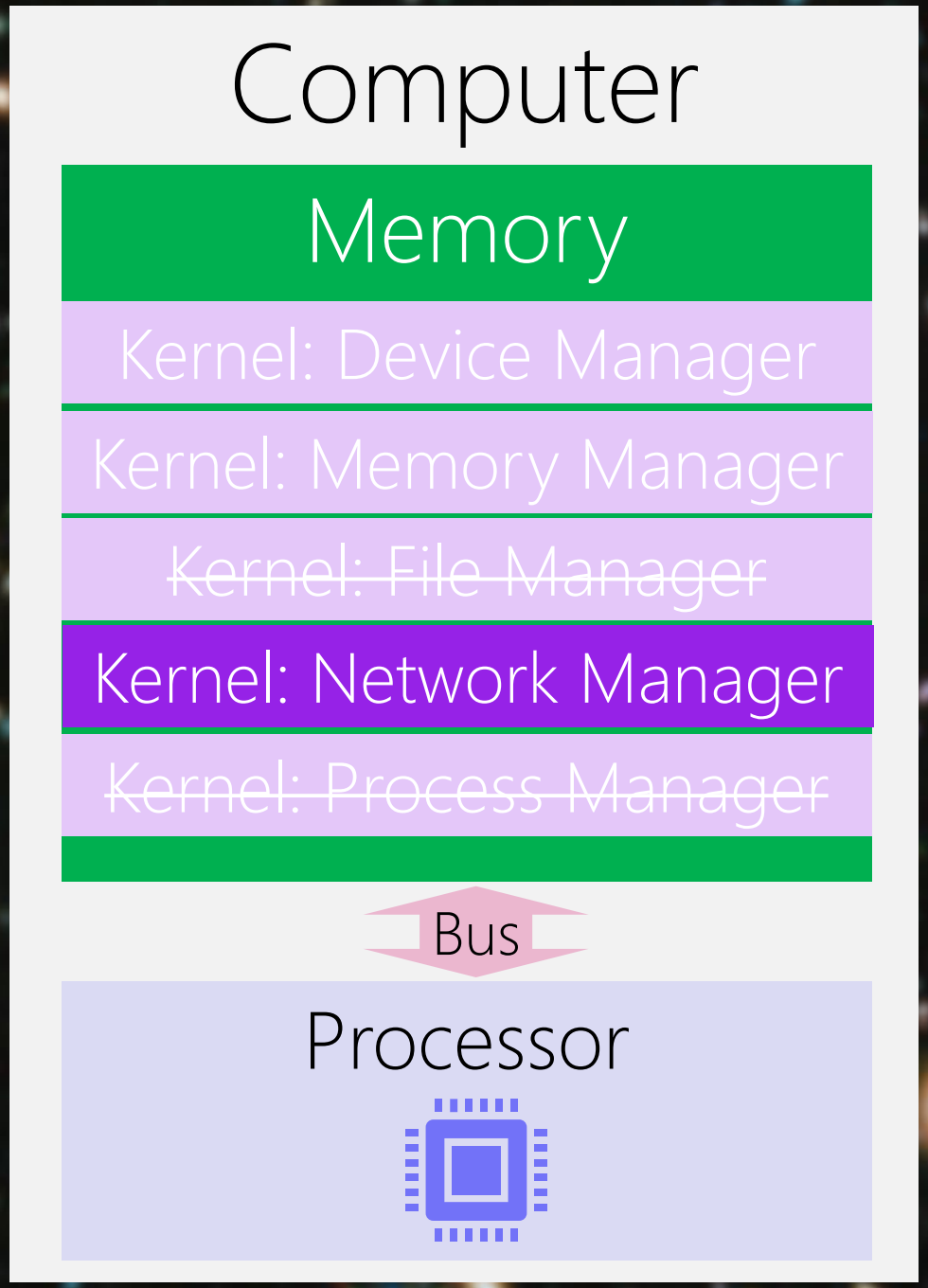


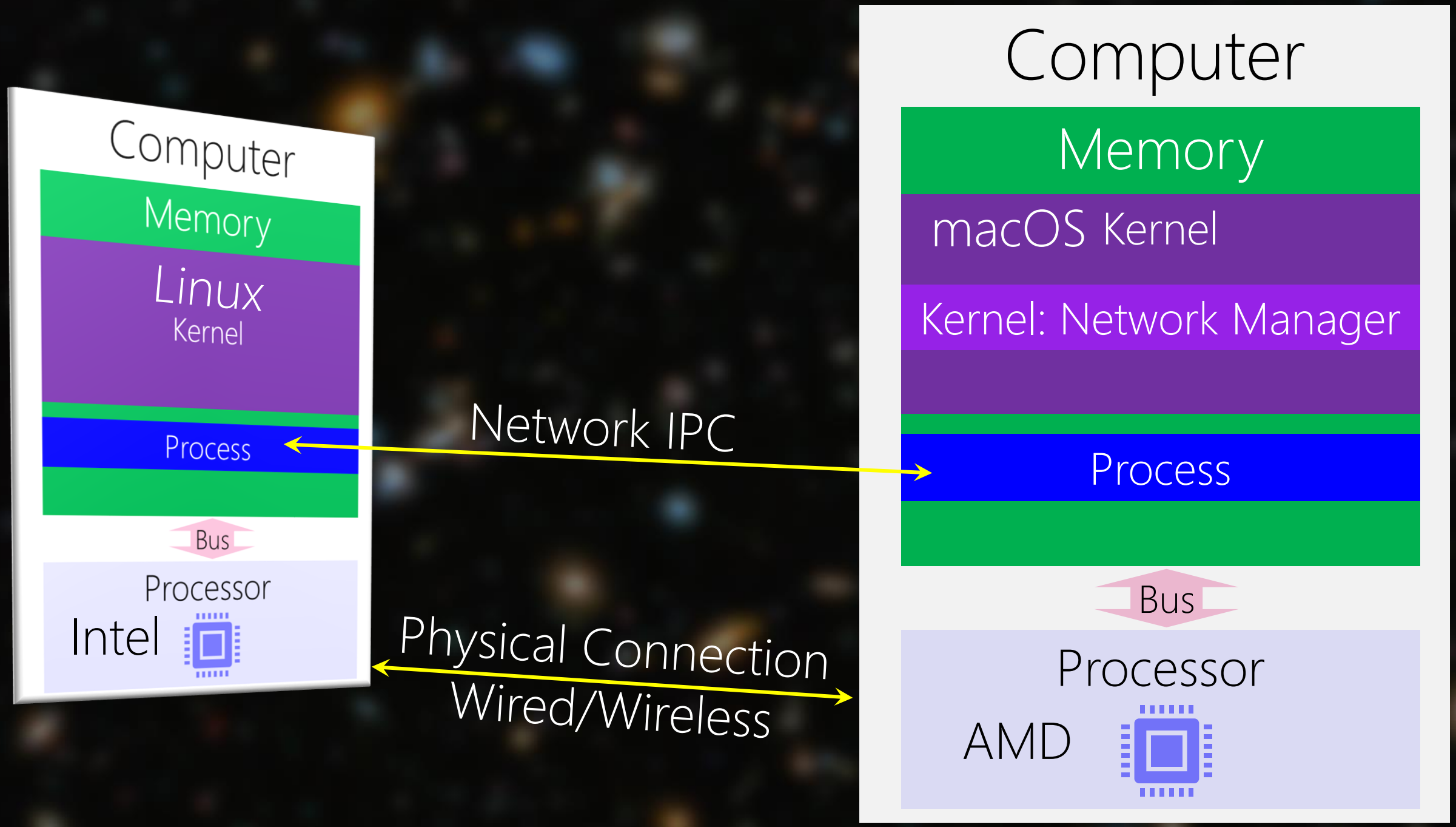


Chapter 16: Network IPC (Sockets)


UNIX
Network Manager
Machine


Kernel
Network Manager
Computer


Computer
Memory
Kernel: Device Manager
Kernel: Memory Manager
~~Kernel: File Manager~~
Kernel: Network Manager
~~Kernel: Process Manager~~
Bus
Processor


Computer
Memory
Linux
Kernel
Process
Bus
Processor
Intel
Network IPC
Physical Connection
Wired/Wireless
Computer
Memory
macOS Kernel
Kernel: Network Manager
Process
Bus
Processor
AMD

Transport Layer

Agreement on communication protocol

2) Connection-Oriented == Phone Call

Transmission Control Protocol (TCP)

- Foremost setup a connection to make sure there is a receiver ready
- Ordered (when you talk on the phone, the words are transferred in order)
- Reliability (there is an active listener)
- Each *packet* depends on previous or next packets
- Connection overhead for sender

Philosophical Debate

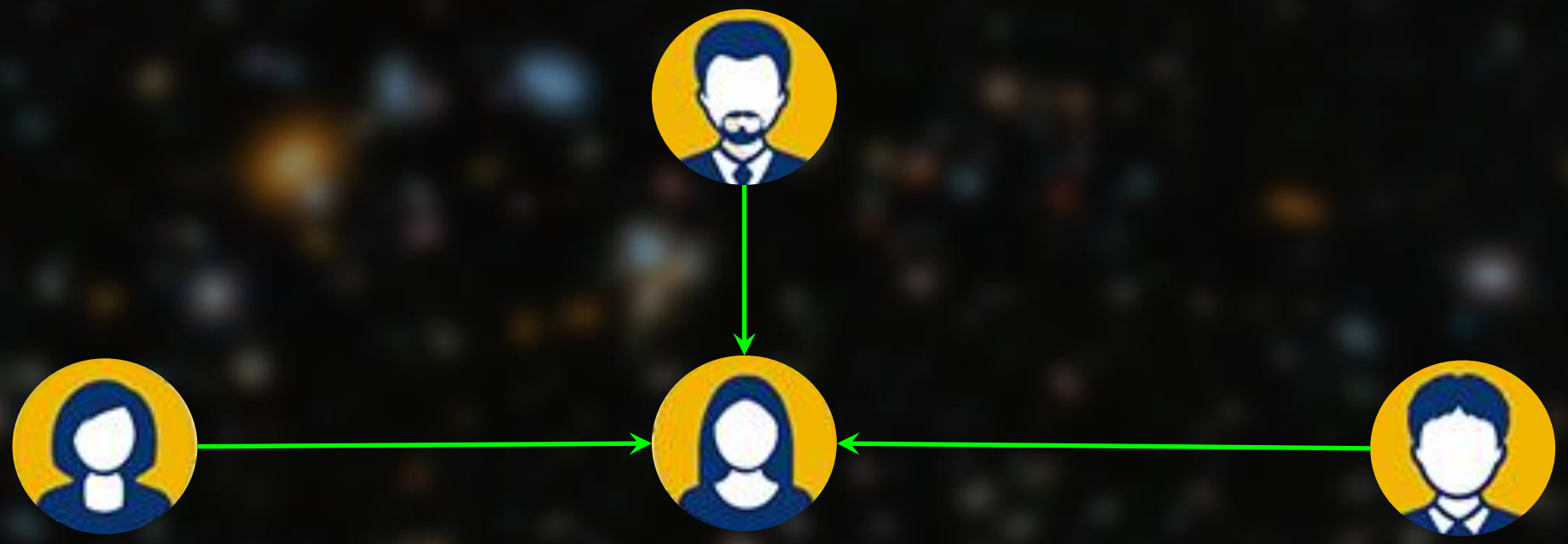
TCP/IP: TCP vs. UDP

Sender/Receiver vs. Client/Server

Any process can *initiate* a communication with other process

A contacts B

Some time later, B contacts A



A passive process! Never *initiate* a call.
Only replies if receives a call!
Never calls anybody!

The Server

Socket Programming

TCP/IP: TCP

Connection-Oriented, Reliable, Ordered

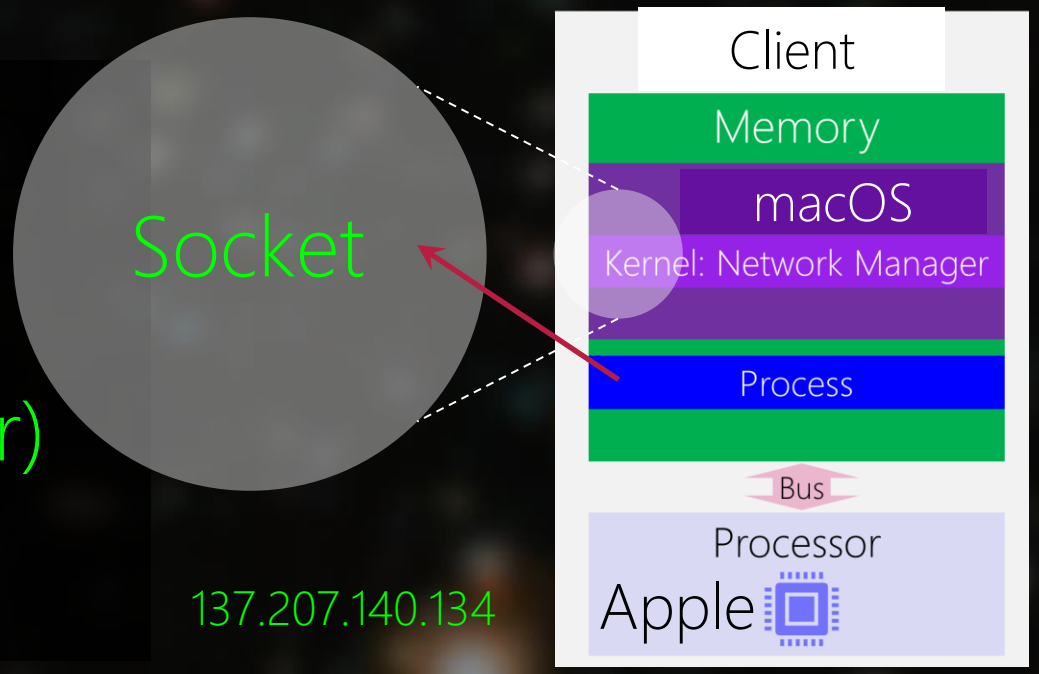
Clients call a Service Provider

TCP/IP: TCP at Clients

Connection-oriented Communication

Phone Call

1) Creating Socket
2) Binding to an Address (Optional)
3) Find The Server's Address
4) Make a Connection (Dial the Number)
5) If Connected, Communicate



137.207.140.134

```
hfani@alpha:~$ cc client.c -o client
hfani@alpha:~$ ./client
socket has created for the client with sd:3
error in connecting to The Server at address:port = 877842313:58375
```

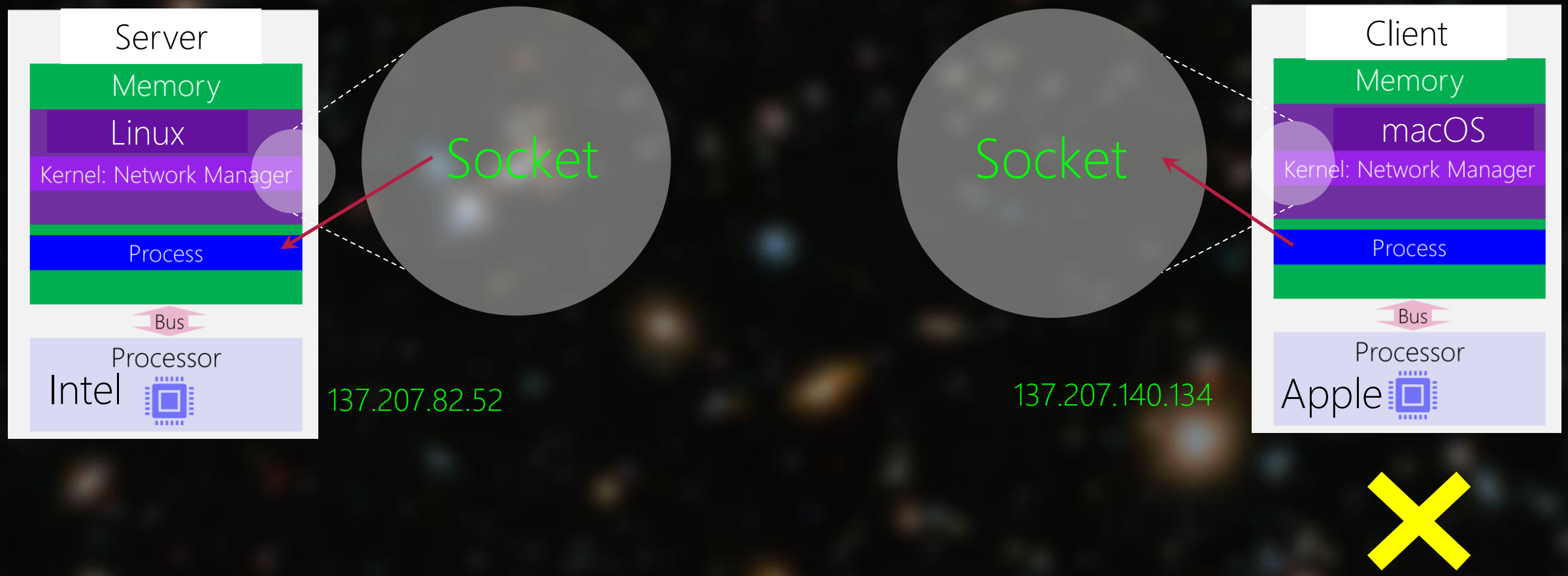
But there is no server!

If there is no connection, no communication!

We cannot move to step (5)

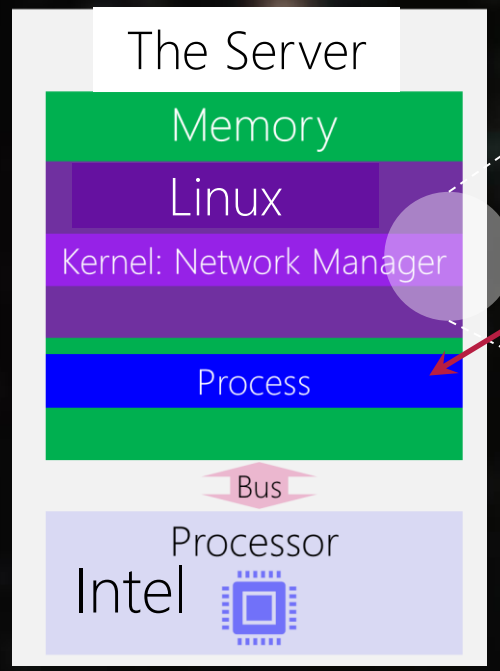
TCP/IP: TCP

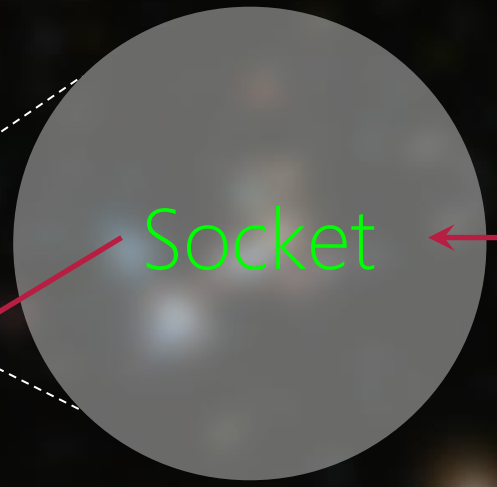
Just a name for [Link | Internet | Transport | Application] network protocol



TCP/IP: TCP at The Server

Connection-oriented Communication
Phone Call





137.207.82.52

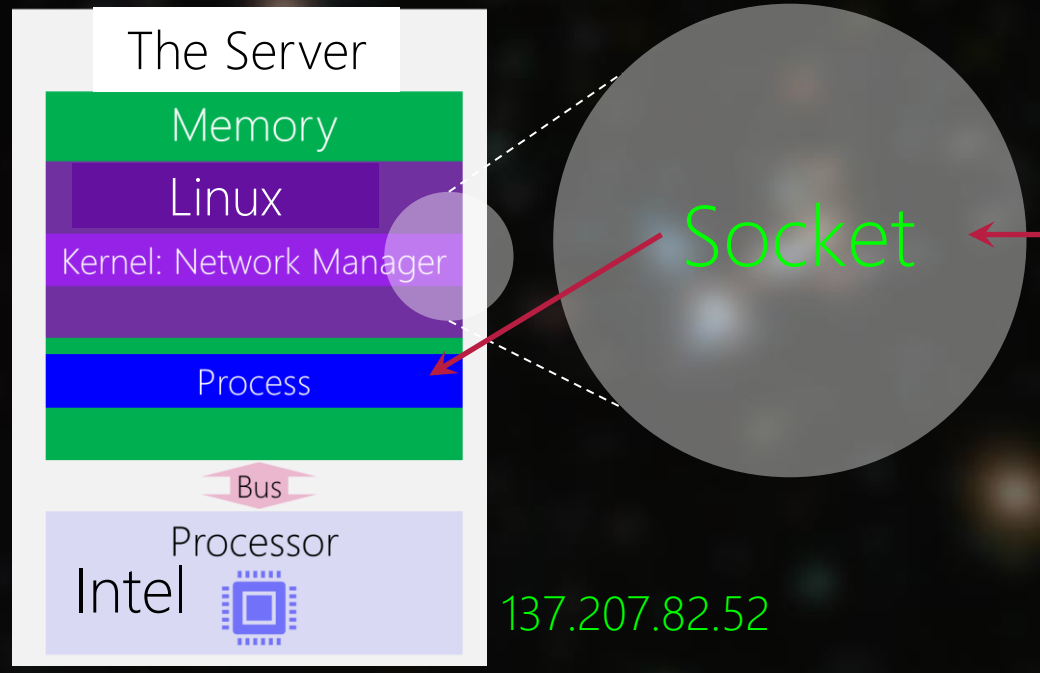
1) Creating Socket
2) Binding to an Address (MUST)
3) Wait for Clients Phone Call
4) Accept Clients' Call
5) Communicate

Like a call center :)

TCP/IP: TCP at The Server

Connection-oriented Communication

Phone Call



137.207.82.52

Socket

1) Creating Socket

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

Kernel's Socket-related System Calls

TCP/IP Network Protocol

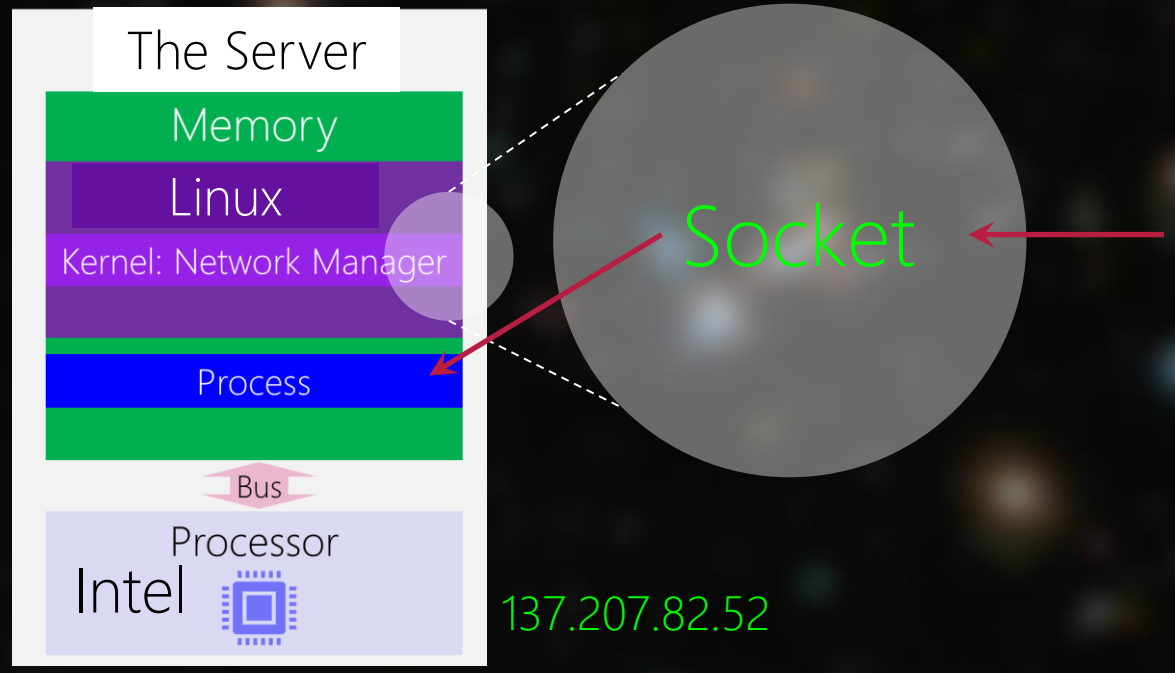
```c
#include <stdlib.h>

#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#include <stdio.h>
#include <string.h>
int main(void){
        int domain = AF_INET;//Network Protocol: TCP/IP
        int type = SOCK_STREAM;//Connection-Oriented
        int protocol = 0;//Default transport: TCP for Internet connection-oriented

        int server_sd;//socket descriptor ~= file descriptor
        server_sd = socket(domain, type, protocol);
        if (server_sd == -1){
                printf("error in creating socket for The Server!\n");
                exit(1);
        }
        else
                printf("socket has created for The Server  with sd:%d\n", server_sd);
```

Set up the type of network communication

TCP/IP: TCP at The Server

Connection-oriented Communication

Phone Call



1) Creating Socket
2) Binding to an Address (MUST)

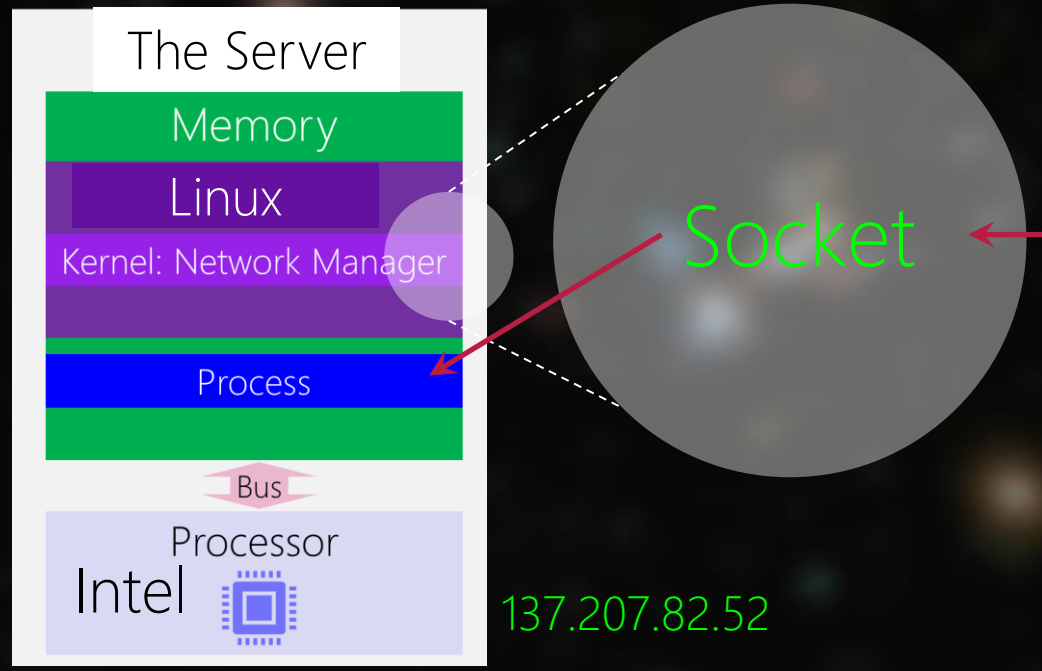
```
struct in_addr server_sin_address;
server_sin_address.s_addr = inet_addr("137.207.82.52");//nslookup `hostname`
int server_sin_port = htons(2021);//larger than 1024

struct sockaddr_in server_sin;
server_sin.sin_family = domain;
server_sin.sin_addr = server_sin_address;
server_sin.sin_port = server_sin_port;
int result = bind(server_sd, (struct sockaddr *) &server_sin, sizeof(server_sin));
if (result == -1){
        printf("error in binding The Server to the address:port = %d:%d\n", server_sin.sin_addr, server_
        exit(1);
}
else
        printf("The Server bound to the address:port = %d:%d\n", server_sin.sin_addr, server_sin.sin_por
```

IP:PORT for The Server

Binding the socket to IP:PORT

TCP/IP: TCP at The Server

Connection-oriented Communication
Phone Call



Socket

1) Creating Socket
2) Binding to an Address (MUST)
3) Wait for Clients Calls

```
#include <sys/socket.h>
int listen(int sockfd, , int backlog);
```

Returns 0 if OK, −1 on error

```
//The Server ready to receive calls (up to 5 calls. More are rejected!)
if (listen(server_sd, 5) < 0) {
        perror("The Server's listening failed!\n");
        exit(1);
}
```

What is the queue size?

Old: only 1 call at a time, others receive busy signals

Now: Waiting Queue!

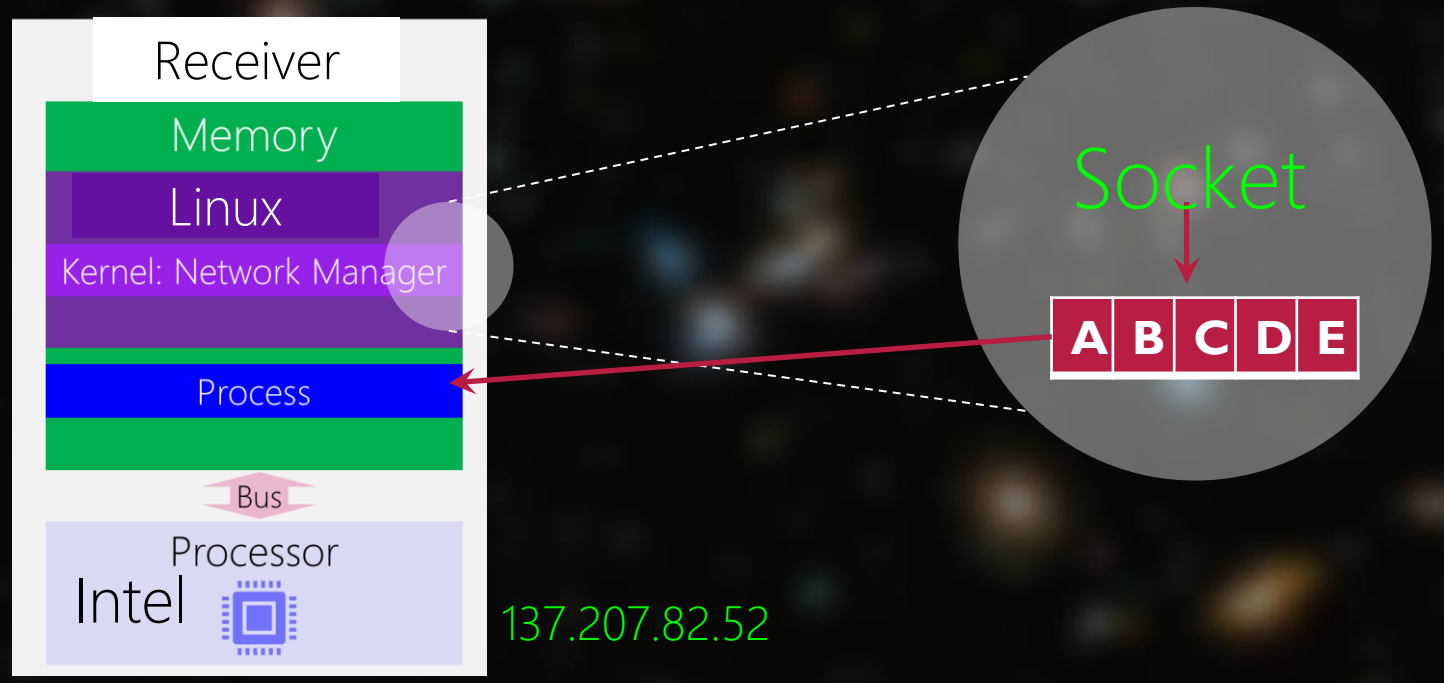
But even the queue is full, reject more connections!

backlog: an accumulation of uncompleted work or matters that need to be dealt with

TCP/IP: TCP at The Server

Connection-oriented Communication

Phone Call

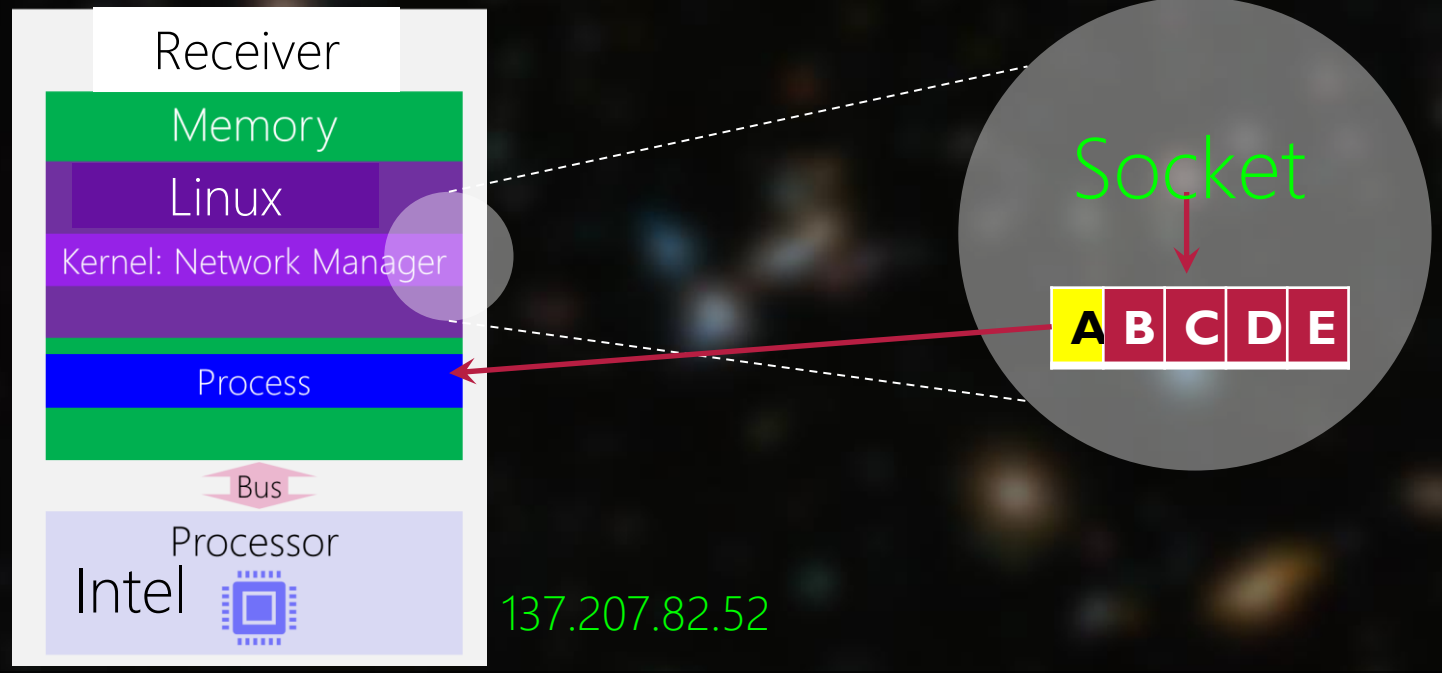


1) Creating Socket
2) Binding to an Address (MU
3) Wait for Clients Calls

TCP/IP: TCP at The Server

Connection-oriented Communication

Phone Call



1) Creating Socket
2) Binding to an Address (MU
3) Wait for Clients Calls
4) Accept Clients' Call

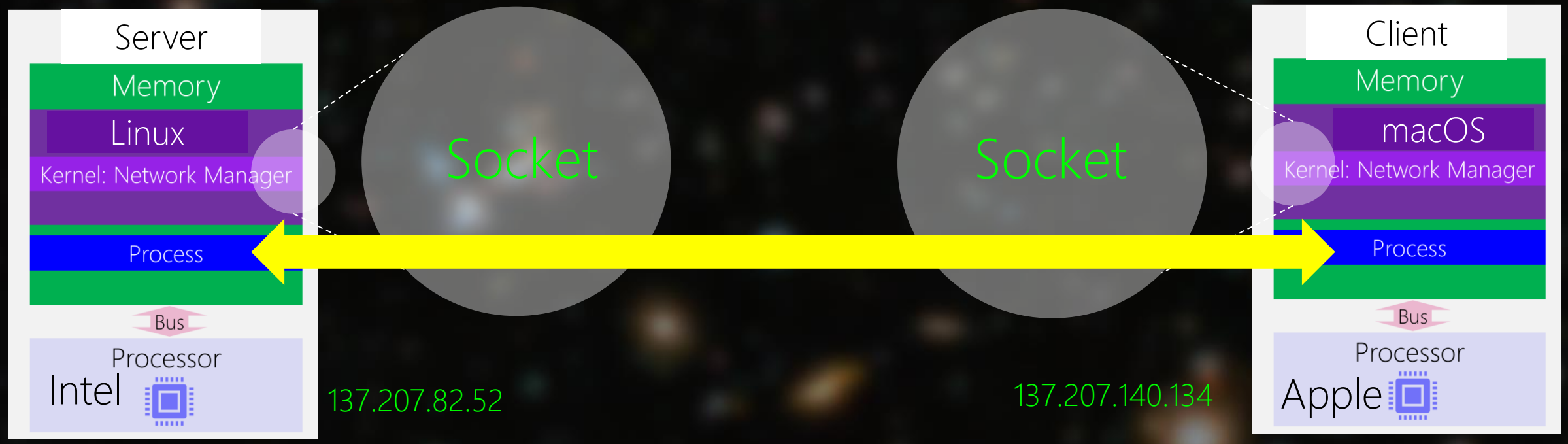
```
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *restrict addr, socklen_t *restrict len);
Returns file (socket) descriptor if OK, -1 on error
```

```
struct sockaddr_in client_sin;//I want to know who send the message
int client_sin_len;
while(1)
{
        result = accept(server_sd, (struct sockaddr *) &client_sin, &client_sin_len);
        if (result == -1){
                printf("error in opening the request from client %d:%d !\n", client_sin.si
                //exit(1);Do not exit. Go for the next client call
        }
        else
                printf("The Server opened the request from client %d:%d\n", client_sin.sin
}
```

You can ignore but you can know who is the sender and decide Client's IP:PORT

TCP/IP: TCP

Just a name for [Link | Internet | Transport | Application] network protocol



hfani@bravo: ~

```
hfani@bravo:~$ ./server
socket has created for The Server  with sd:3
The Server bound to the address:port = 877842313:58631
The Server opened the request from client 21899:32204
The Server opened the request from client 877842313:51899
The Server opened the request from client 877842313:53947
The Server opened the request from client 877842313:65211
The Server opened the request from client 877842313:188
```

hfani@bravo: ~

```
hfani@bravo:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port
 = 877842313:58631
hfani@bravo:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port
 = 877842313:58631
hfani@bravo:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port
 = 877842313:58631
hfani@bravo:~$
```

hfani@bravo: ~

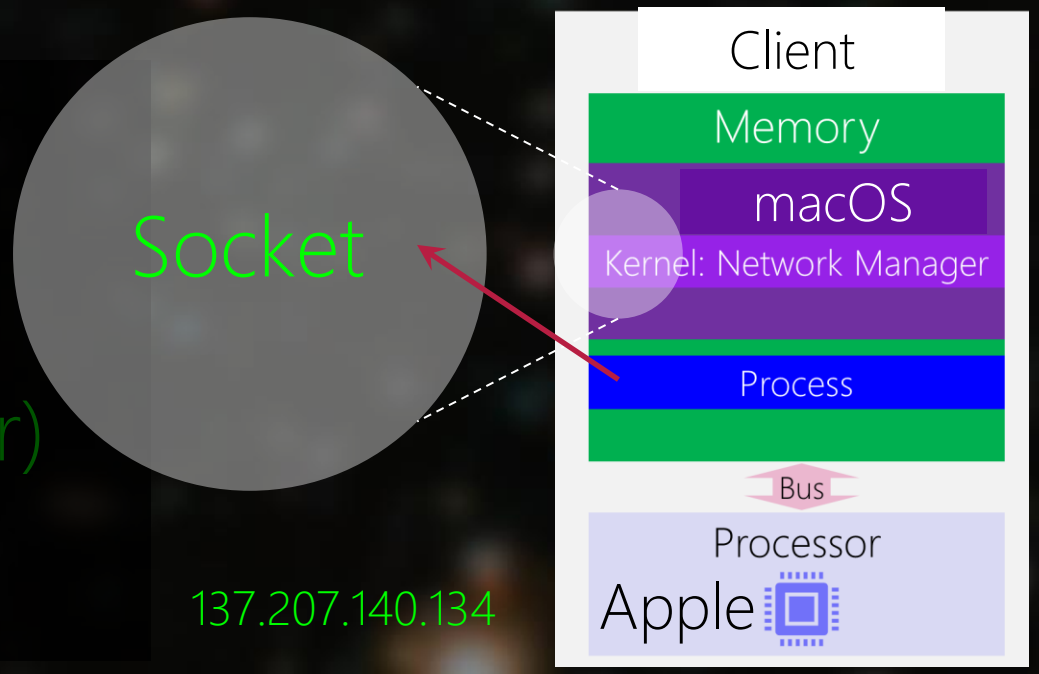
```
hfani@bravo:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port
 = 877842313:58631
hfani@bravo:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port
 = 877842313:58631
hfani@bravo:~$
```

TCP/IP: TCP at Clients

Connection-oriented Communication

Phone Call

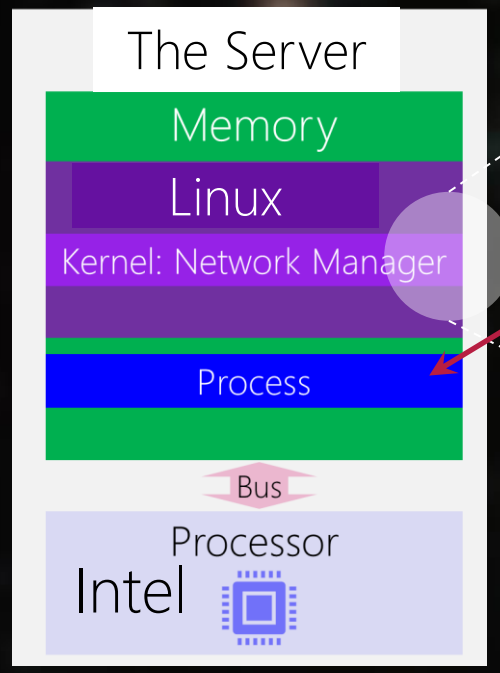
1) Creating Socket
2) Binding to an Address (Optional)
3) Find The Server's Address
4) Make a Connection (Dial the Number)
5) If Connected, Communicate

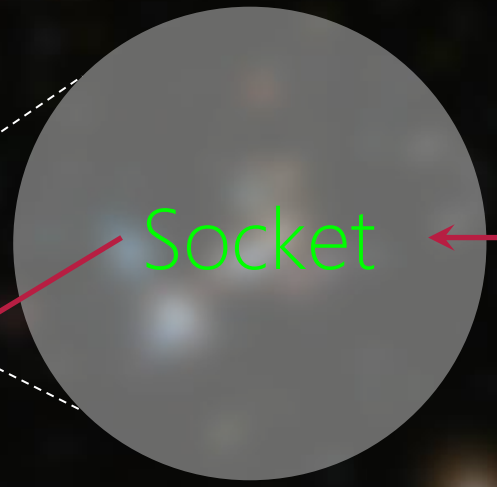


137.207.140.134

TCP/IP: TCP at The Server

Connection-oriented Communication
Phone Call





137.207.82.52

1) Creating Socket
2) Binding to an Address (MUST)
3) Wait for Clients Phone Call
4) Accept Clients' Call
5) Communicate

Like a call center :)

```
send(sd) == write(sd)
recv(sd) == read(sd)
```

The Server

```
struct sockaddr_in client_sin;//I want to know who send the message
int client_sin_len;
while(1)
{
        result = accept(server_sd, (struct sockaddr *) &client_sin, &client_sin_len);
        if (result == -1){
                printf("error in opening the request from client %d:%d !\n", client_si
                //exit(1);Do not exit. Go for the next client call
        }
        else
                printf("The Server opened the request from client %d:%d\n", client_sin

        char msg[10];
        recv(result, msg, 10, 0);
        printf("%s\n", msg);
        send(result, "hello back", 11, 0);

}
```

The Client

```
int result = connect(client_sd, (struct sockaddr *) &server_sin, sizeof(server_sin));
if (result == -1){
        printf("error in connecting to The Server at address:port = %d:%d\n", server_sin.sin_addr
        exit(1);
}
else
        printf("client is connected to The Server at address:port = %d:%d\n", server_sin.sin_addr

char response[10];
send(client_sd, "hello", 6, 0);
recv(client_sd, response, 10, 0);
printf("%s\n", response);
```

hfani@charlie: ~

```
hfani@charlie:~$ vi server.c
hfani@charlie:~$ cc receiver.c -o receiver
hfani@charlie:~$ ./server
socket has created for The Server  with sd:3
error in binding The Server to the address:port = 861065097:51486
hfani@charlie:~$ vi server.c
hfani@charlie:~$ cc receiver.c -o receiver
hfani@charlie:~$ vi server.c
hfani@charlie:~$ ./server
socket has created for The Server  with sd:3
error in binding The Server to the address:port = 861065097:51486
hfani@charlie:~$ cc receiver.c -o receiver
hfani@charlie:~$ cc server.c -o server
hfani@charlie:~$ ./server
socket has created for The Server  with sd:3
The Server bound to the address:port = 894619529:51742
The Server opened the request from client 21902:29652
hello
The Server opened the request from client 894619529:52946
hello
The Server opened the request from client 894619529:53458
hello
```

hfani@charlie: ~

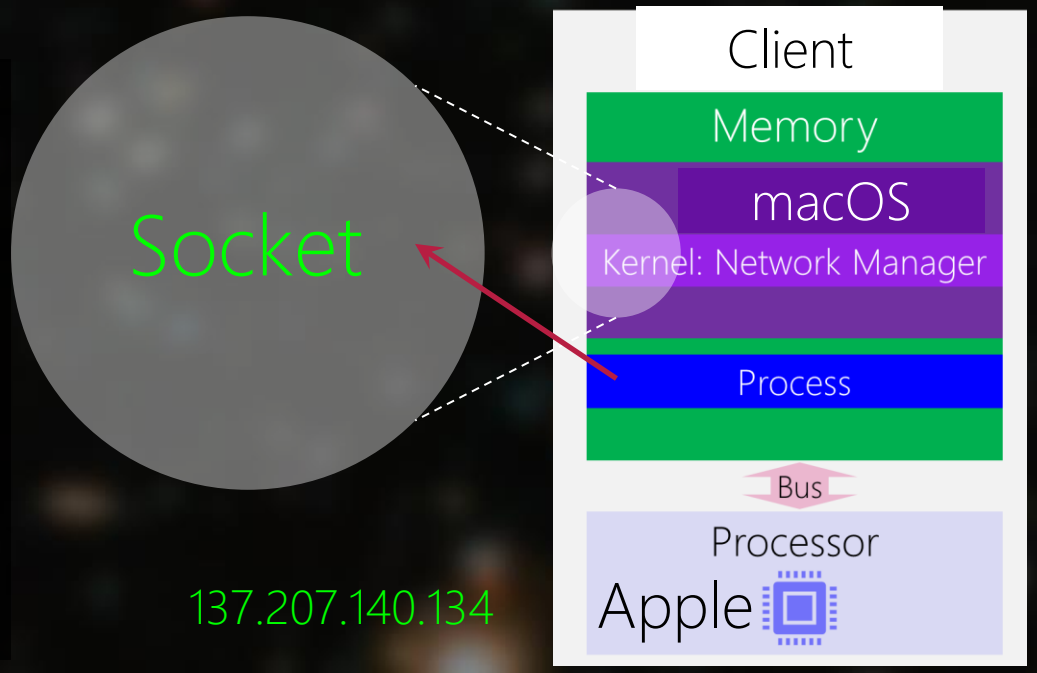
```
+-----------------------------------------------------------------+

"mesg n" has been appended to /etc/profile to prevent users from broa
ssages to your terminal using the wall/write command.

Last login: Mon Nov 29 11:00:43 2021 from 137.207.140.134
hfani@charlie:~$ vi client.c

[1]+  Stopped                 vi client.c
hfani@charlie:~$ vi client.c
hfani@charlie:~$ cc client.c -o client
hfani@charlie:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port = 894619529:51742
hello back
hfani@charlie:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port = 894619529:51742
hello back
hfani@charlie:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port = 894619529:51742
hello back
hfani@charlie:~$
```

TCP/IP: TCP at Clients

Connection-oriented Communication

Phone Call

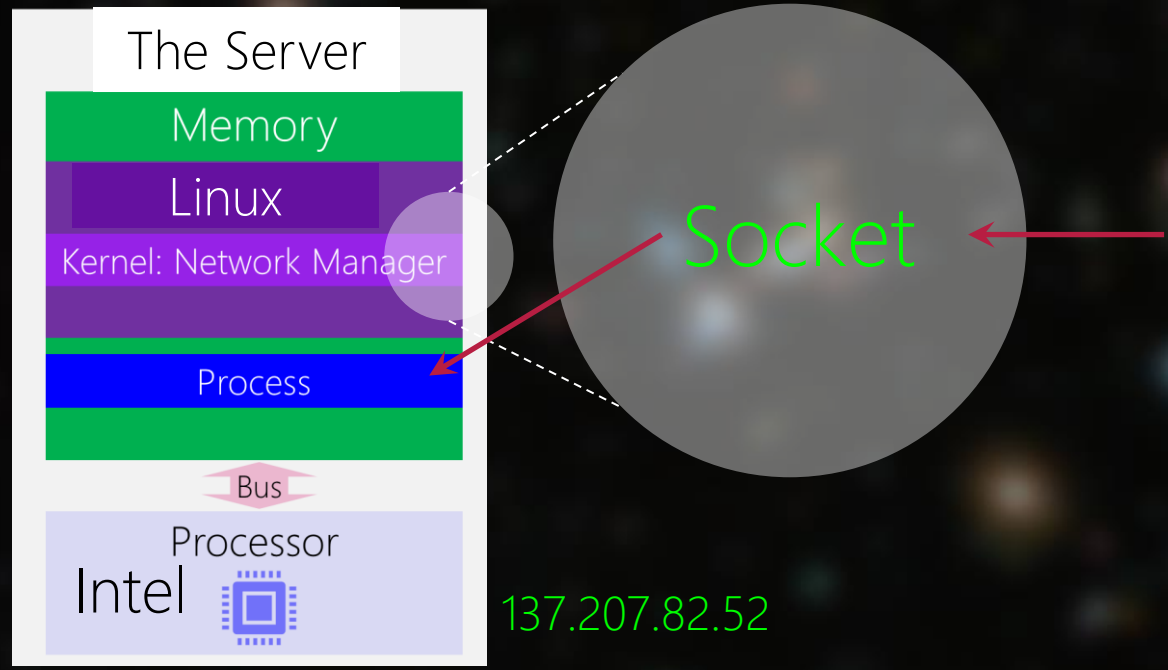
1) `socket()`
2) `bind()`
3) Receiver's Address
4) `connect()`
5) `recv()` or `send()`



TCP/IP: TCP at The Server

Connection-oriented Communication

Phone Call



1) `socket()`
2) `bind()`
3) `listen()`
4) `accept()`
5) Find Client's Address
6) `recv()` or `send()`

`accept()` returns a new socket descriptor. What?!

The Server

```
struct sockaddr_in client_sin;//I want to know who send the message
int client_sin_len;
while(1)
{
        result = accept(server_sd, (struct sockaddr *) &client_sin, &client_sin_len);
        if (result == -1){
                printf("error in opening the request from client %d:%d !\n", client_si
                //exit(1);Do not exit. Go for the next client call
        }
        else
                printf("The Server opened the request from client %d:%d\n", client_sin

        char msg[10];
        recv(result, msg, 10, 0);
        printf("%s\n", msg);
        send(result, "hello back", 11, 0);
}
```

Each connection has its own sd!

The Client

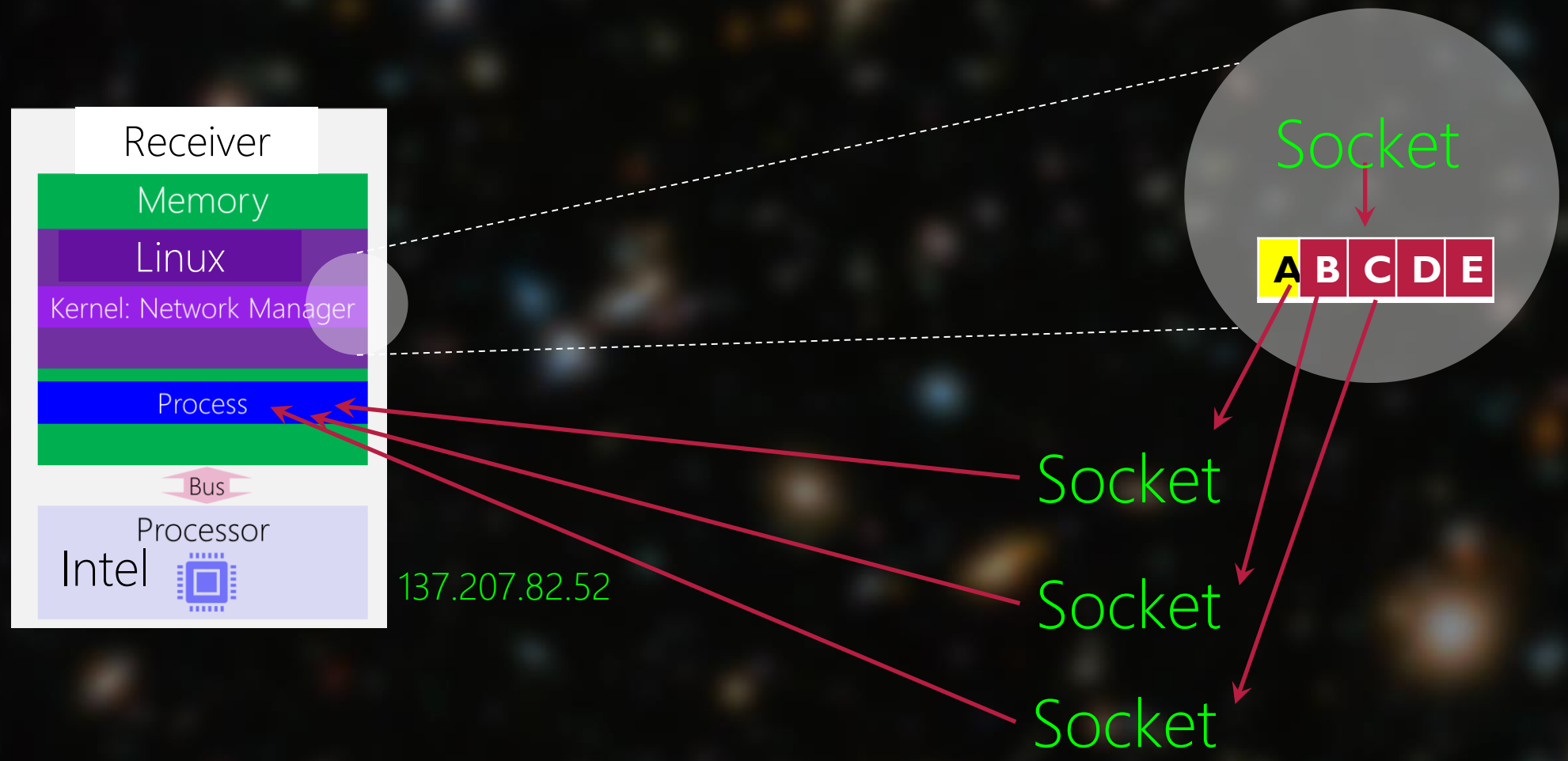
```
int result = connect(client_sd, (struct sockaddr *) &server_sin, sizeof(server_sin));
if (result == -1){
        printf("error in connecting to The Server at address:port = %d:%d\n", server_sin.sin_addr
        exit(1);
}
else
        printf("client is connected to The Server at address:port = %d:%d\n", server_sin.sin_addr

char response[10];
send(client_sd, "hello", 6, 0);
recv(client_sd, response, 10, 0);
printf("%s\n", response);
```

The same sd is used

TCP/IP: TCP at The Server

Connection-oriented Communication

Phone Call



`accept()` returns a new socket descriptor. Why?
We can use The Server main socket, can't we?

Hint: `fork()`, children, parallel, ...

The Server w/ `fork()`

The Server

```
while(1)
{
        result = accept(server_sd, (struct sockaddr *) &client_sin, &client_sin_len);
        if (result == -1){
                printf("error in opening the request from client %d:%d !\n", client_sin.sin_addr, client_sin.sin_port);
                //exit(1);Do not exit. Go for the next client call
        }
        else{
                printf("The Server opened the request from client %d:%d\n", client_sin.sin_addr, client_sin.sin_port);

                int child_pid = fork();
                if(child_pid == 0){//child
                        printf("Child: I The Server's child to handle the communication with the client %d:%d\n", client_

                        char msg[10];
                        recv(result, msg, 10, 0);
                        printf("%s\n", msg);
                        for(int i=0; i<100; ++i){
                                send(result, "hello back", 11, 0);
                                sleep(1);
                        }
                }
        }
}
```

The Server:

Hey child, take care of this client. I go ahead and taking another call

The Server

```
hfani@alpha:~$ ./server_fork
socket has created for The Server  with sd:3
The Server bound to the address:port = 861065097:51742
```

Client 0

```
hfani@alpha: ~
hfani@alpha:~$ ./client_fork
```

Client 1

```
hfani@alpha: ~
hfani@alpha:~$ ./client_fork
```

The Server

```
hfani@alpha:~$ ./server_fork
socket has created for The Server  with sd:3
The Server bound to the address:port = 861065097:51742
The Server opened the request from client 0:0
Child: I The Server's child to handle the communication with the client 0:0
hello
The Server opened the request from client 861065097:15084
Child: I The Server's child to handle the communication with the client 861065097:15084
hello
```

Client 0

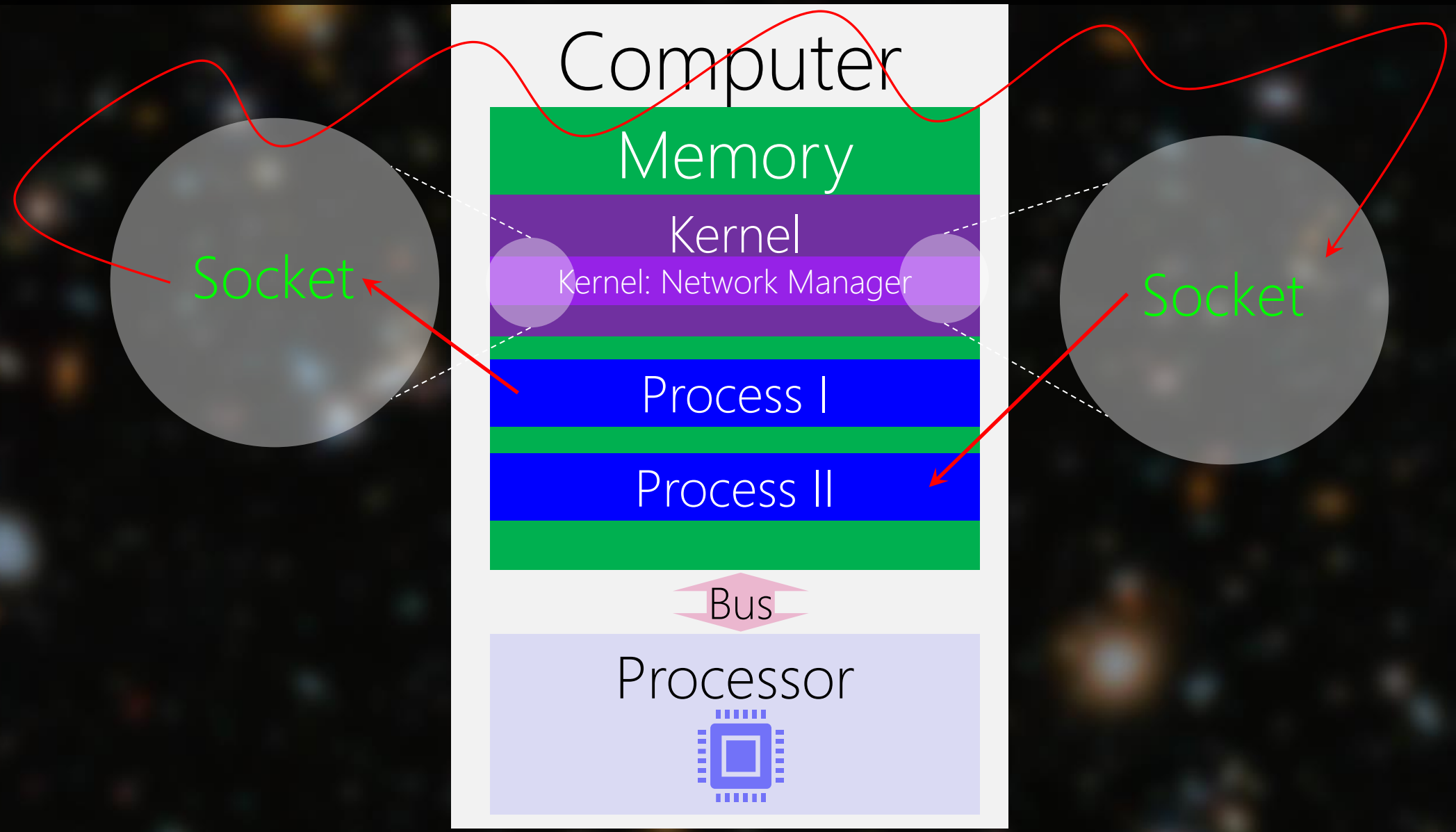
```
hfani@alpha: ~
hfani@alpha:~$ ./client_fork
socket has created for the client with sd:3
client is connected to The Server at address:port = 861065097:51742
hello back

hello back

hello back

hello back

hello back
```

Client 1

```
hfani@alpha: ~
hfani@alpha:~$ ./client_fork
socket has created for the client with sd:3
client is connected to The Server at address:port = 861065097:51742
hello back

hello back

hello back

hello back
```

Network IPC in the Same Computer
An Alternative for IPC

Network IPC on the Same Computer → IPC



Do we need example?!

All we showed so far was in the same School's computer!