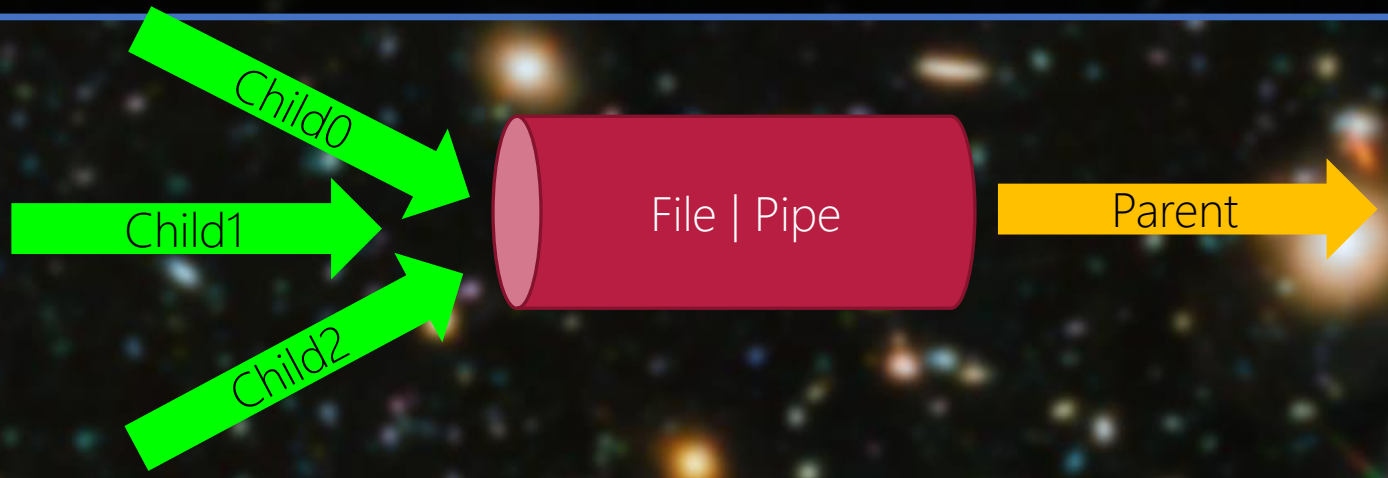
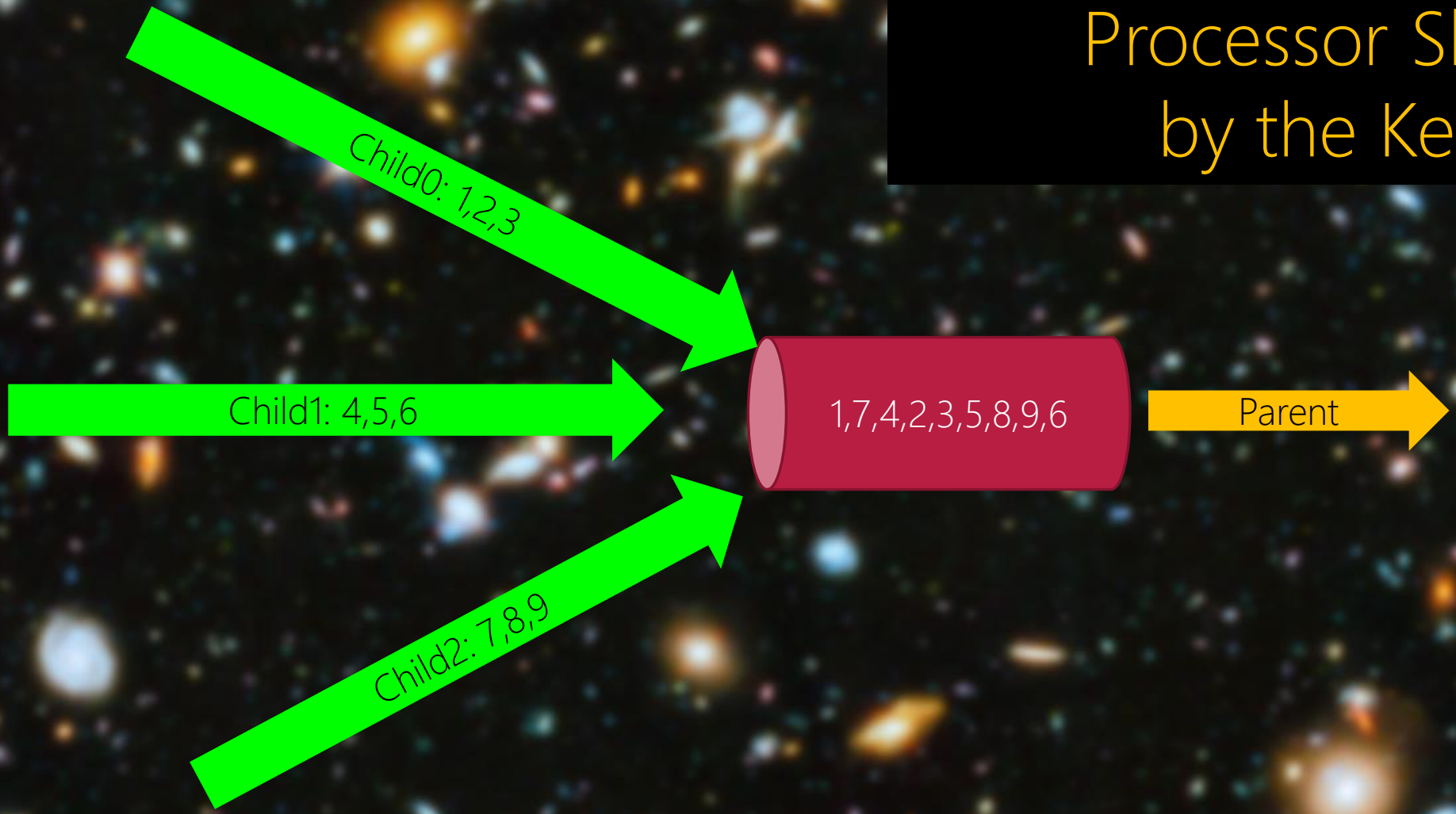
Lab09: Shared File | Pipe between Children

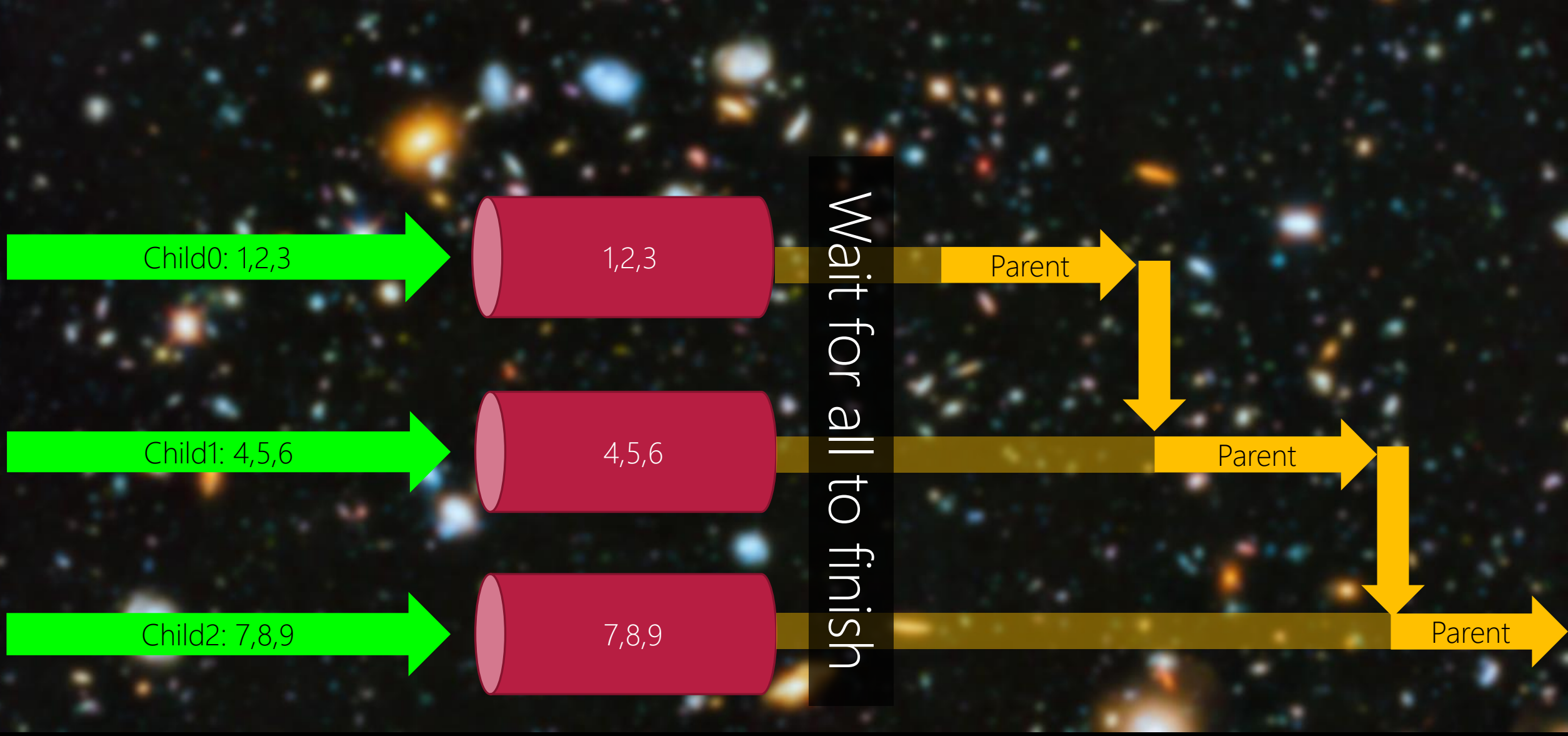


No Assumption about Processor Sharing by the Kernel

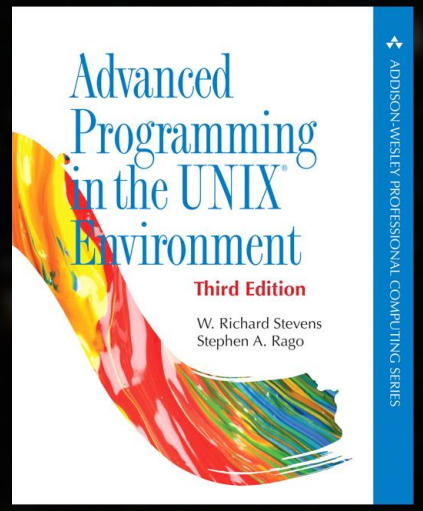


Sleep

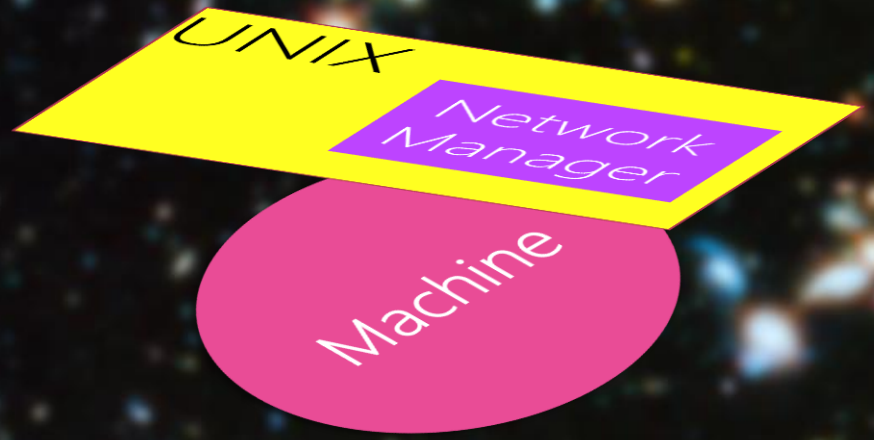
Still, There is a Problem.
Why?

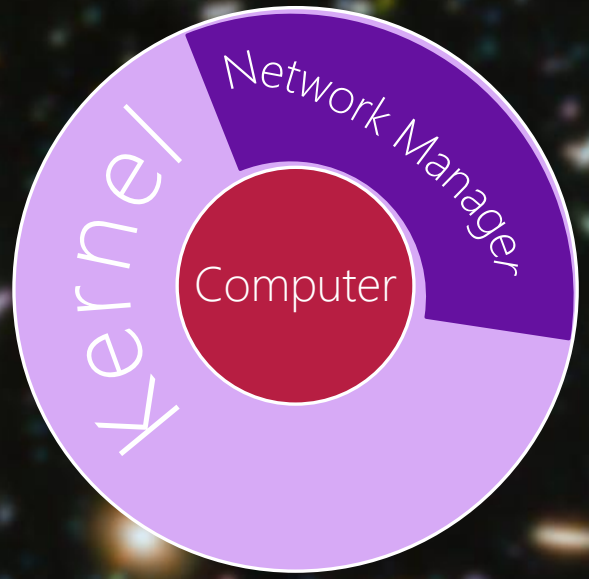


Always correct regardless of processor scheduling.



Chapter 16: Network IPC (Sockets)





Computer

Memory

Kernel: Device Manager

Kernel: Memory Manager

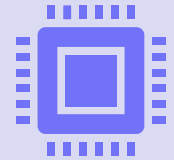
~~Kernel: File Manager~~

Kernel: Network Manager

~~Kernel: Process Manager~~

Bus

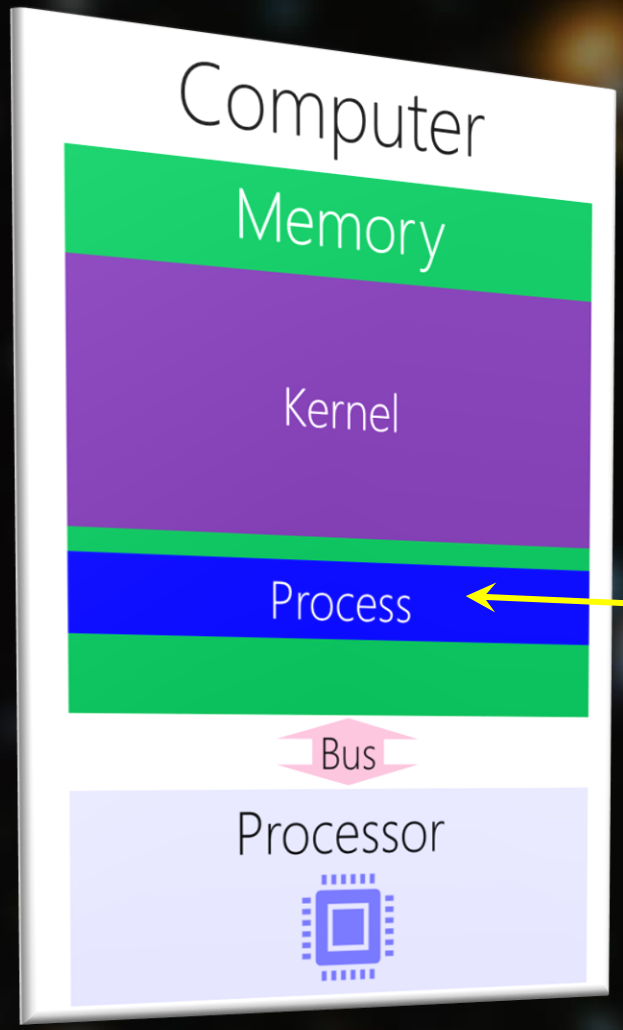
Processor



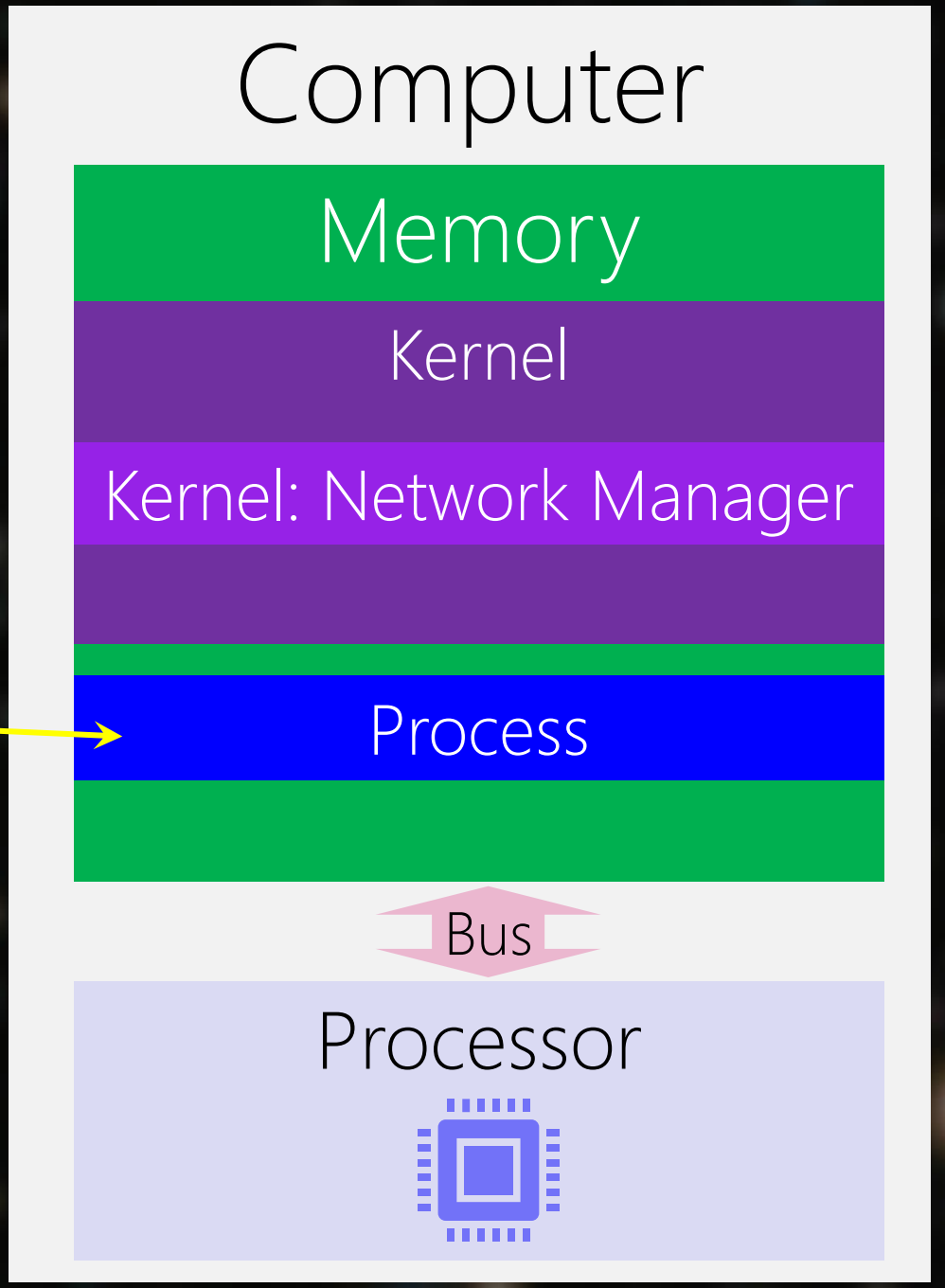
Multi~~processing~~ Computers

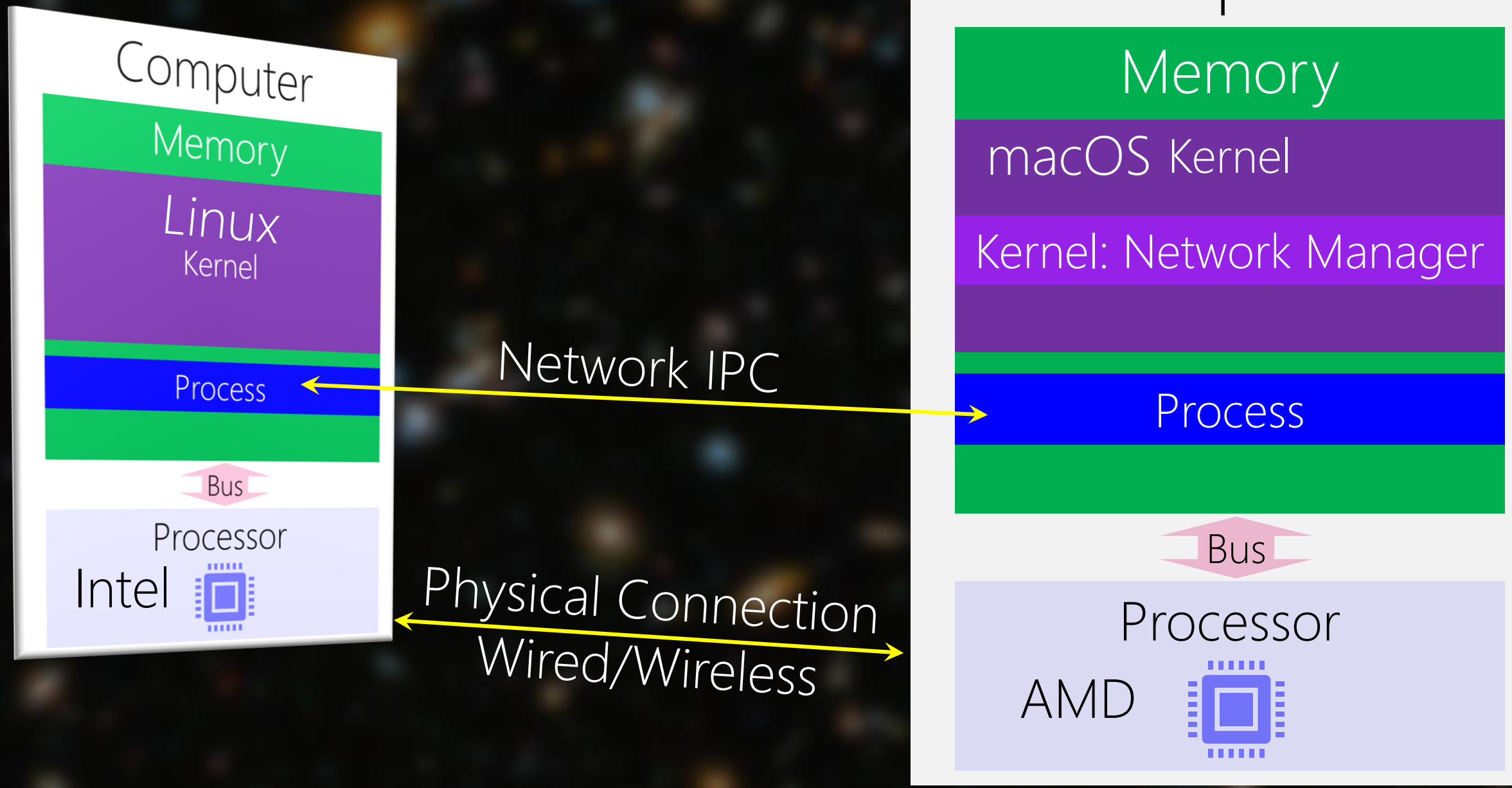
aka Computer Network

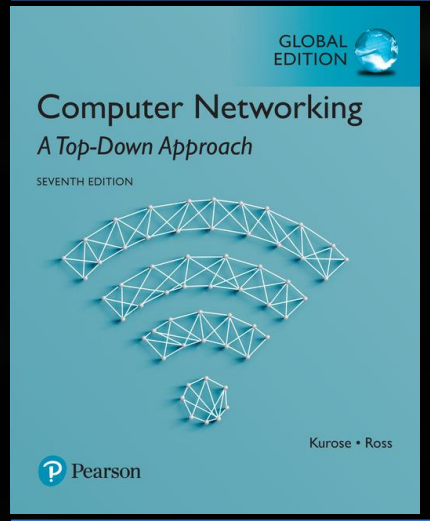
Multiple Single Processor ~~Multiprocessor~~


Computer
Memory
Kernel
Process
Bus
Processor

Network IPC


Computer
Memory
Kernel
Kernel: Network Manager
Process
Bus
Processor

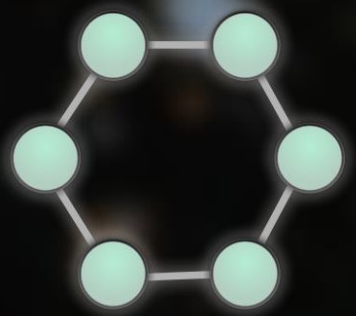

Computer
Memory
Linux
Kernel
Process
Bus
Processor
Intel
Network IPC
Physical Connection
Wired/Wireless
Computer
Memory
macOS Kernel
Kernel: Network Manager
Process
Bus
Processor
AMD



COMP3670: Computer Networks

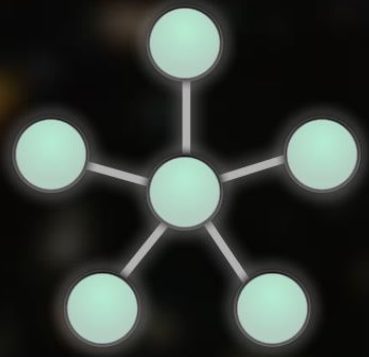
Network Topology

The way computer systems are connected.
By software control, we can convert one to another.

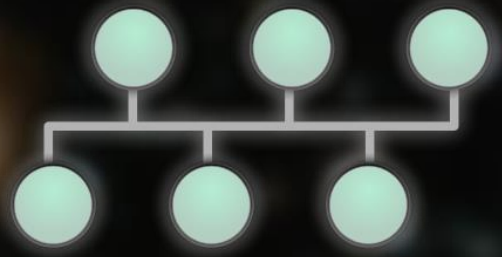


Ring

Line



Star



Bus

Network Protocol

Conversation Protocol between Computers
Language, Order (Who Talks, Who Listens), Addressing (Finding Each Other), …

1975: 2-network communications between Stanford and University College London
1977: 3-network between sites in the US, the UK, and Norway

TCP/IP

There are other network protocols!

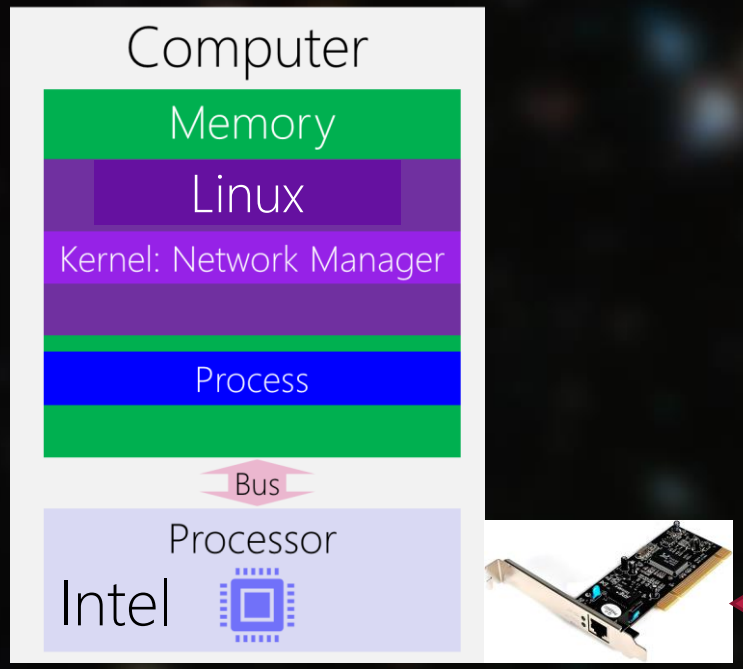
TCP/IP is just a name. It does not represent all this protocol offers!

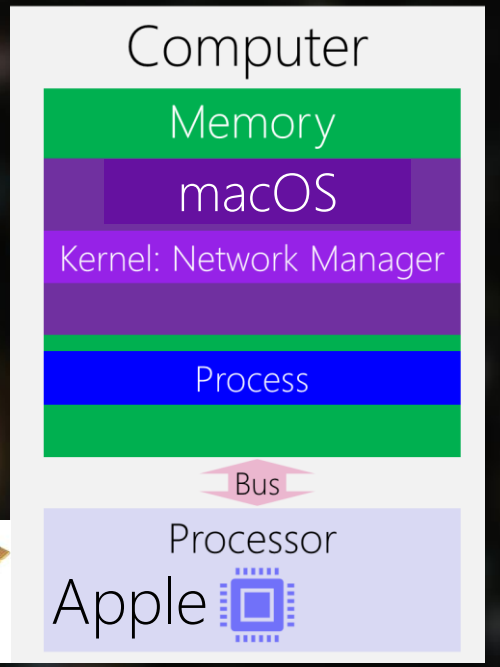
Link Layer

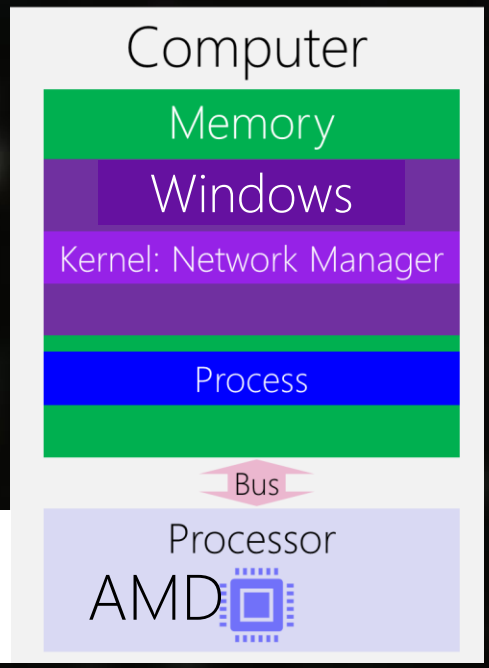
Type of physical connection

Wired (Ethernet), Wireless (Wifi, Bluetooth, Infrared, ...)

We don't care in this course!

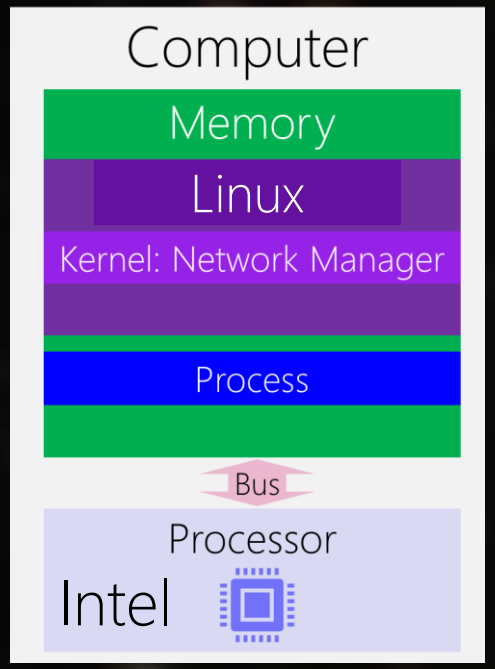




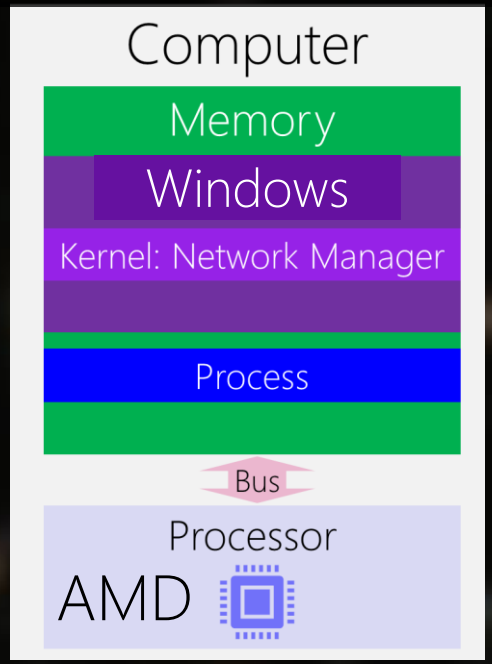


Inter-Network → Internet (Network) Layer → Internet Protocol (IP)

Computers' Address, Names, ...
We use the addresses in this course.
We don't care about the rest.
Why the format is like this?
Who assigns the addresses?
...

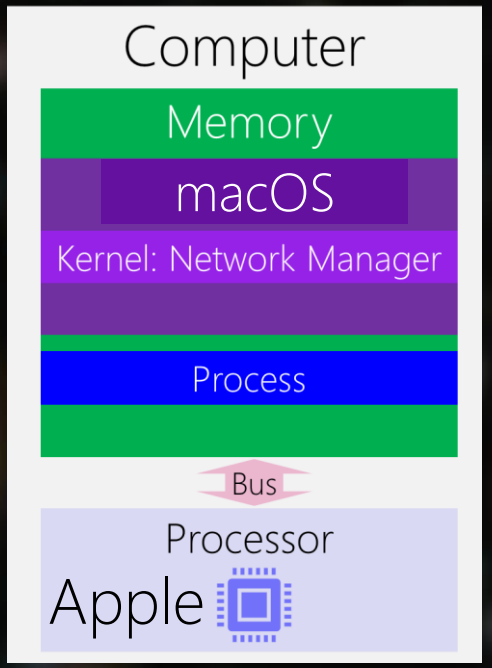


137.207.82.52



4.2.2.2

137.207.140.134



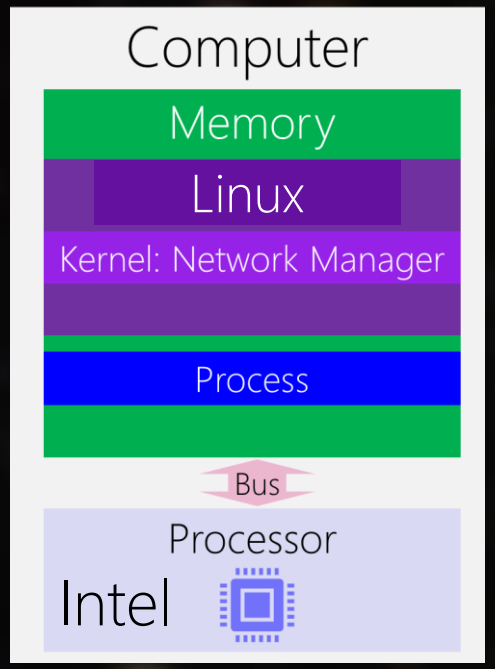
Transport Layer

Agreement on communication protocol

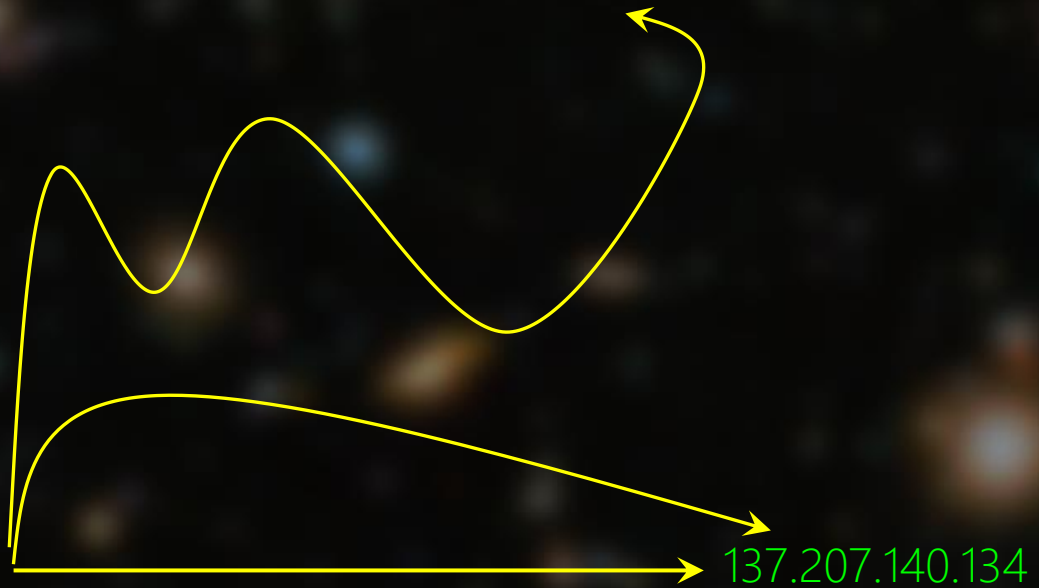
1) Connectionless == Sending a mail

User Datagram Protocol (UDP)

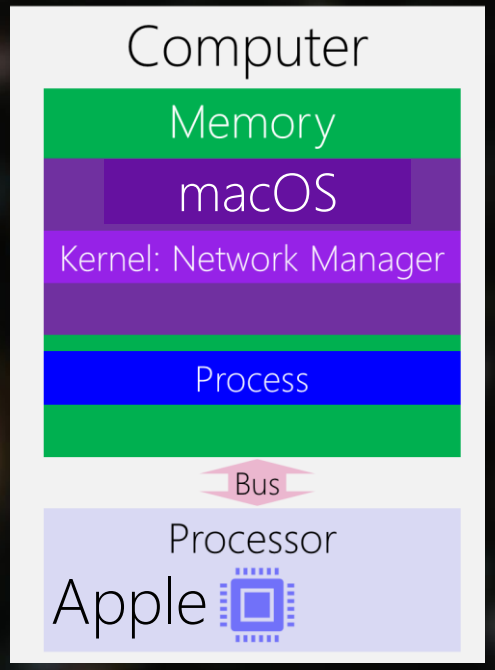
- No order (a mail may be sent sooner, but received later)
- No reliability (non-tracking mail.) Cannot see whether it is received or lost
- Each message is self-contained (Does not depends on previous or next mails)
- Simple and light (no overhead for sender, like PR card by government of Canada)



137.207.82.52



137.207.140.134



Transport Layer

Agreement on communication protocol

2) Connection-Oriented == Phone Call

Transmission Control Protocol (TCP)

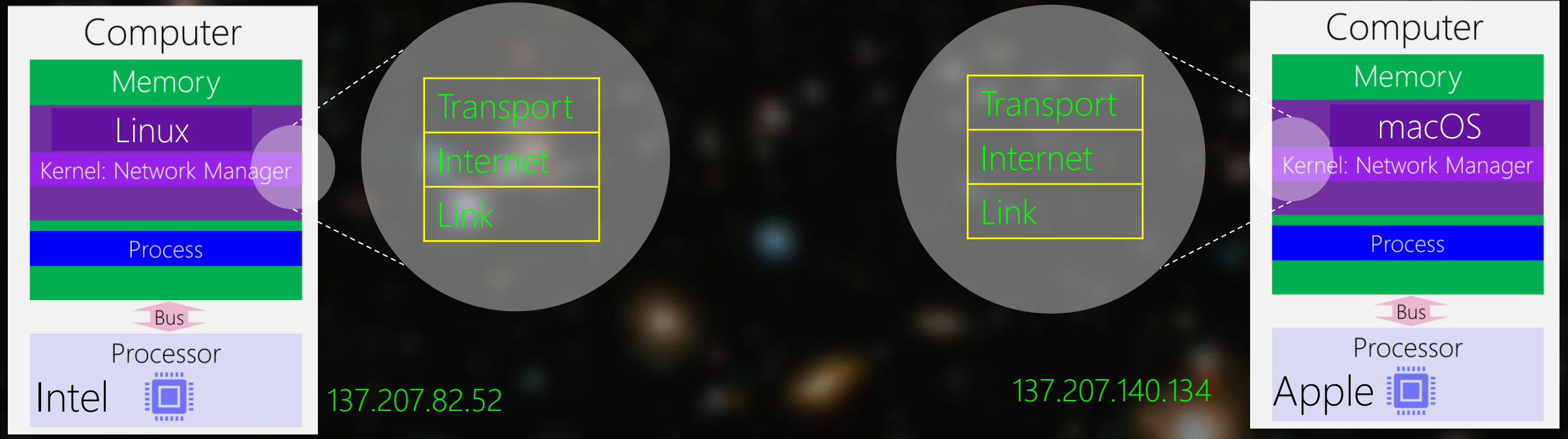
- Foremost setup a connection to make sure there is a receiver ready
- Ordered (when you talk on the phone, the words are transferred in order)
- Reliability (there is an active listener)
- Each *packet* depends on previous or next packets
- Connection overhead for sender

Application Layer

Any process that wants to communicate via the network

TCP/IP

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



TCP/IP

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

TCP/IP

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

Socket Programming

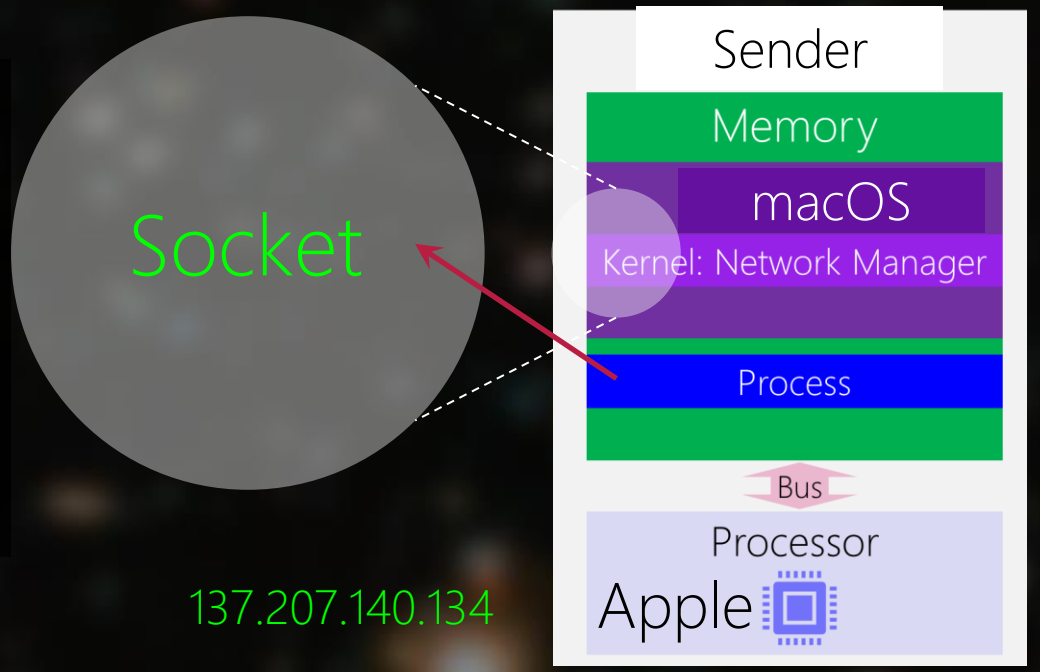
TCP/IP: UDP

TCP/IP: UDP at Sender

Connectionless Communication

Sending a mail

1) Creating Socket
2) Binding to an Address (Optional)
3) Find the Receiver's Address
4) Send the Mail to the Receiver



TCP/IP: UDP at Sender

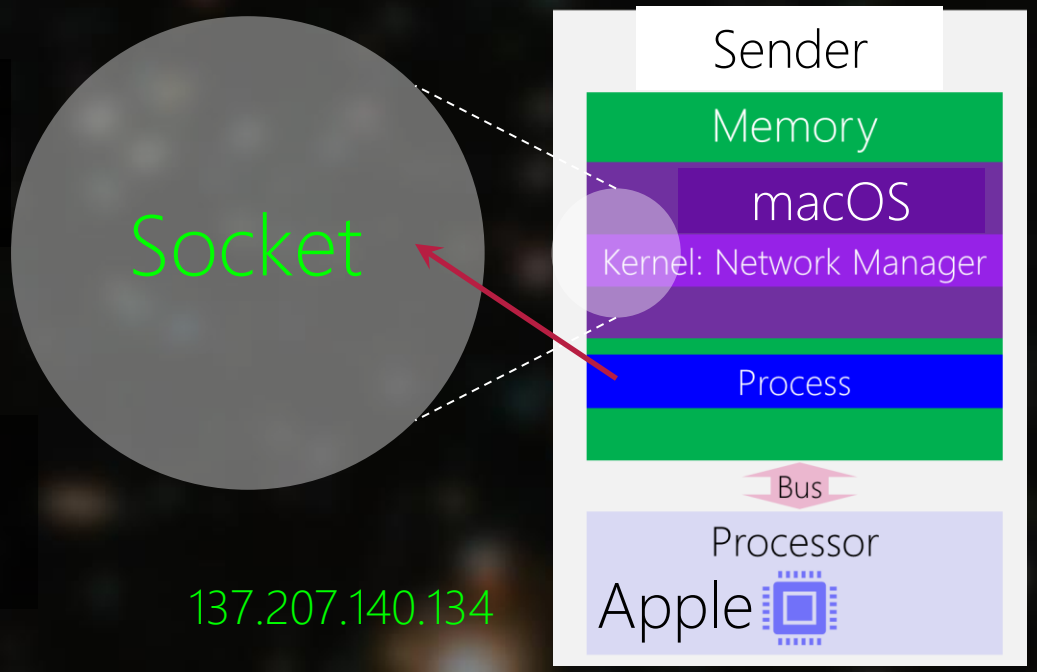
Connectionless Communication

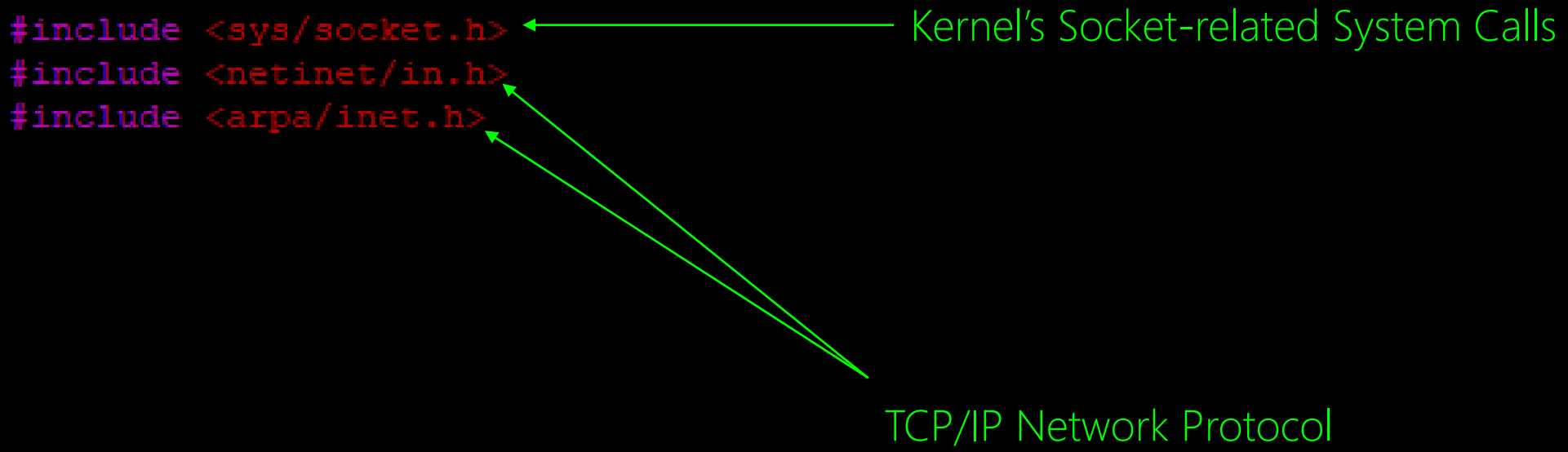
Sending a mail

1) Creating Socket

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

Returns socket descriptor if OK, –1 on error




#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
Kernel's Socket-related System Calls
TCP/IP Network Protocol

```c
#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_DGRAM;//Connectionless
        int protocol = 0;//Default transport: UDP for Internet connectionless
```

Set up the type of network communication

Domain	Description
AF_INET	IPv4 Internet domain
AF_INET6	IPv6 Internet domain (optional in POSIX.1)
AF_UNIX	UNIX domain
AF_UNSPEC	unspecified

Figure 16.1 Socket communication domains

Type	Description
SOCK_DGRAM	fixed-length, connectionless, unreliable messages
SOCK_RAW	datagram interface to IP (optional in POSIX.1)
SOCK_SEQPACKET	fixed-length, sequenced, reliable, connection-oriented messages
SOCK_STREAM	sequenced, reliable, bidirectional, connection-oriented byte streams

Figure 16.2 Socket types

Protocol	Description
IPPROTO_IP	IPv4 Internet Protocol
IPPROTO_IPV6	IPv6 Internet Protocol (optional in POSIX.1)
IPPROTO_ICMP	Internet Control Message Protocol
IPPROTO_RAW	Raw IP packets protocol (optional in POSIX.1)
IPPROTO_TCP	Transmission Control Protocol
IPPROTO_UDP	User Datagram Protocol

Figure 16.3 Protocols defined for Internet domain sockets

```c
#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_DGRAM;//Connectionless
        int protocol = 0;//Default transport: UDP for Internet connectionless

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

Open a socket and receive a socket descriptor

Very similar to `open()` a file and file descriptor

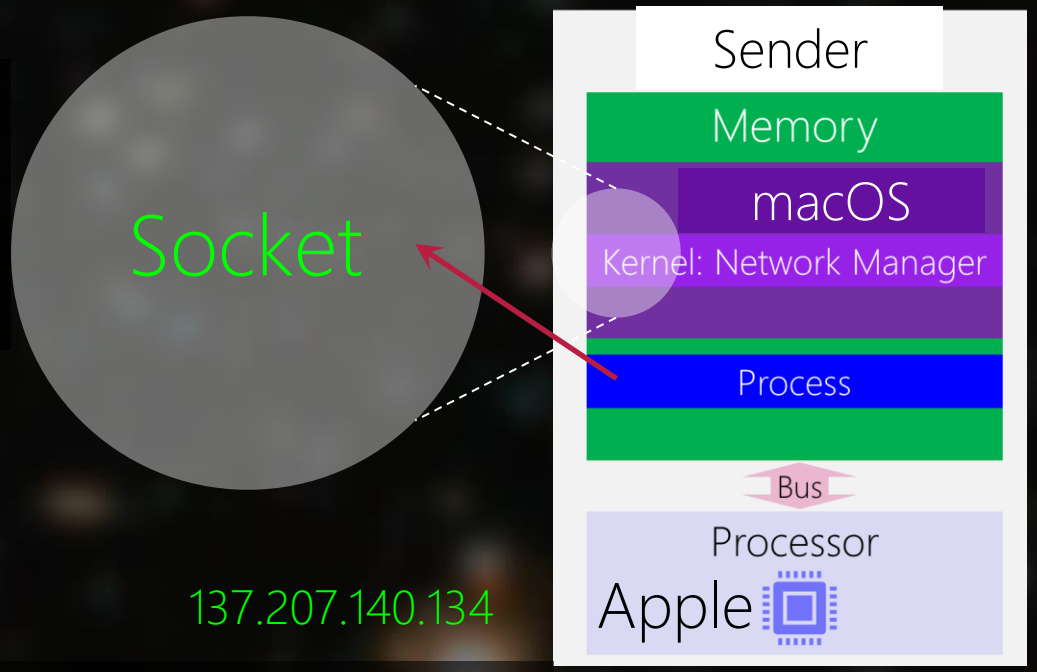
Indeed, behind the scene, there are implemented very similar!

TCP/IP: UDP at Sender

Connectionless Communication

Sending a mail

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




```
#include <sys/socket.h>
int bind(int sockfd, const struct sockaddr *addr, socklen_t len);
```

Returns 0 if OK, −1 on error

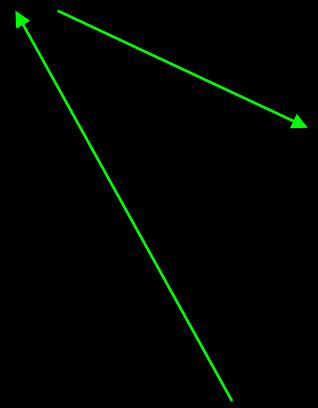
```
//Binding to an address is optional for sender!
struct in_addr sender_sin_address;
sender_sin_address.s_addr = inet_addr("137.207.140.134 ");//nslookup `hostname`
```

IP Address of the computer system
[[0-255][0-255][0-255][0-255]]
4 bytes

There are many ways to know the IP of a computer, e.g.,

```
hfani@bravo:~$ nslookup `hostname`
```

```
//Binding to an address is optional for sender!
struct in_addr sender_sin_address;
sender_sin_address.s_addr = inet_addr("137.207.140.134 ");//nslookup `hostname`
int sender_sin_port = htons(2000);//larger than 1024
```



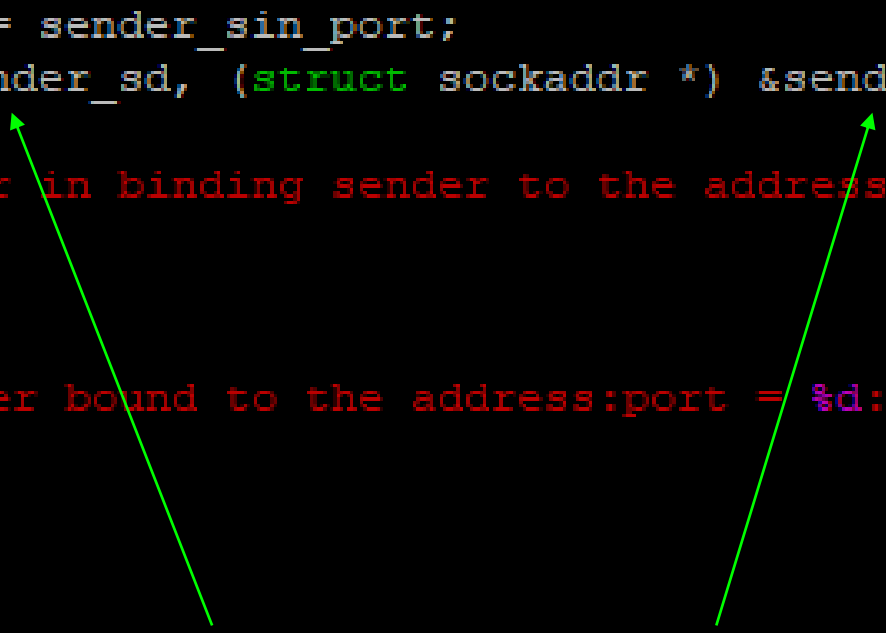
`htons(0);` → random port by the kernel

In a computer system, there are multiple processes
To distinguish which process is the sender → PORT

End Point (Full Address) → IP:PORT

It is unique all around the world! Why?

```
struct sockaddr_in sender_sin;
sender_sin.sin_family = domain;
sender_sin.sin_addr = sender_sin_address;
sender_sin.sin_port = sender_sin_port;
int result = bind(sender_sd, (struct sockaddr *) &sender_sin, sizeof(sender_sin));
if (result == -1){
        printf("error in binding sender to the address:port = %d:%d\n", sender_sin.sin_
        exit(1);
}
else
        printf("sender bound to the address:port = %d:%d\n", sender_sin.sin_addr, sende
```

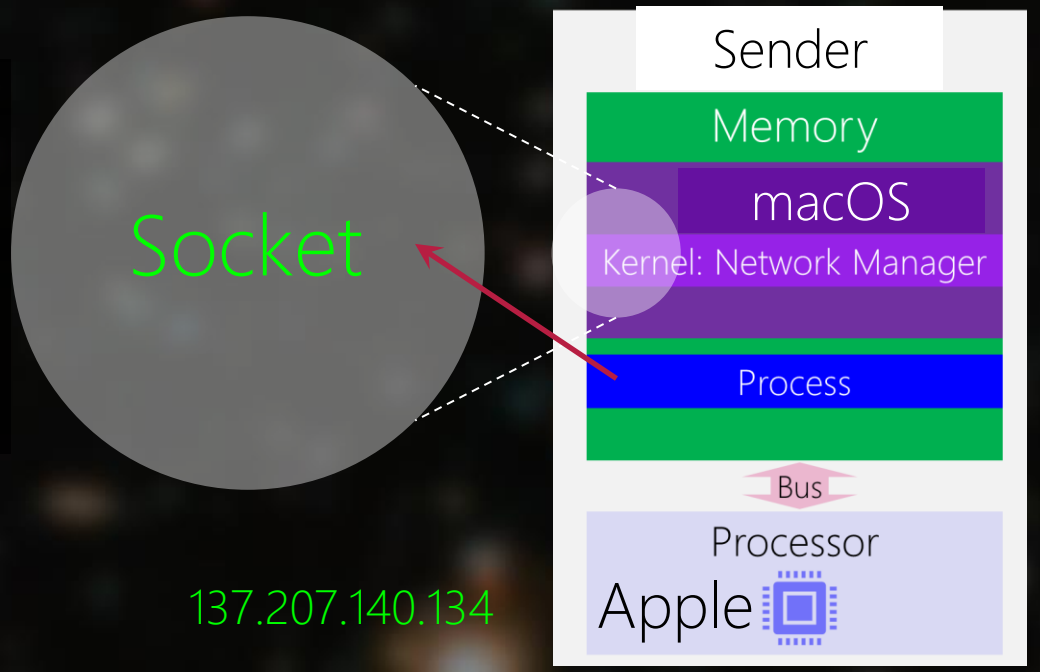


TCP/IP: UDP at Sender

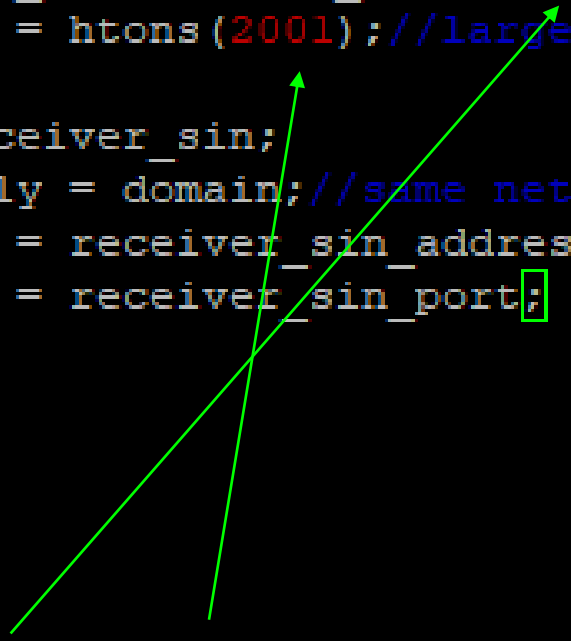
Connectionless Communication

Sending a mail

1) Creating Socket
2) Binding to an Address (Optional)
3) Find the Receiver's Address



```
//Sending a message to the receiver at 137.207.82.52:2001
struct in_addr receiver_sin_address;
receiver_sin_address.s_addr = inet_addr("137.207.82.52");//nslookup `hostname` at the ta
int receiver_sin_port = htons(2001);//larger than 1024

struct sockaddr_in receiver_sin;
receiver_sin.sin_family = domain;//same network protocol
receiver_sin.sin_addr = receiver_sin_address;
receiver_sin.sin_port = receiver_sin_port;
```



IP:PORT of the Receiver

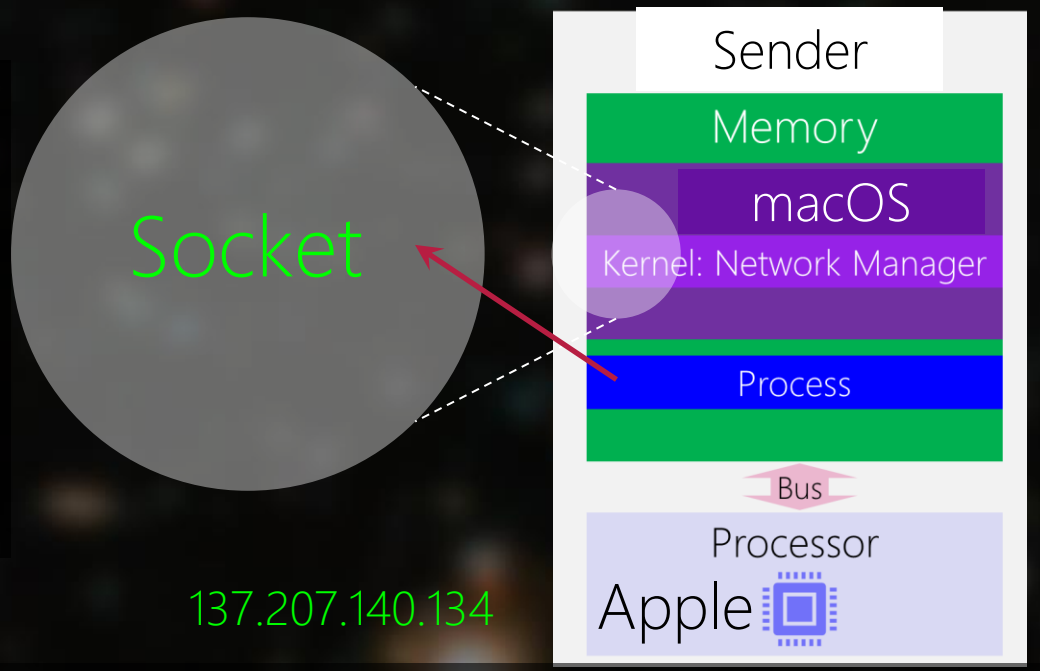
It is unique all around the world! Why?

TCP/IP: UDP at Sender

Connectionless Communication

Sending a mail

1) Creating Socket
2) Binding to an Address (Optional)
3) Find the Receiver's Address
4) Send the Mail to the Receiver




```
#include <sys/socket.h>
ssize_t sendto(int sockfd, const void *buf, size_t nbytes, int flags, const struct sockaddr *destaddr, socklen_t destlen);
```

Returns number of bytes sent if OK, –1 on error

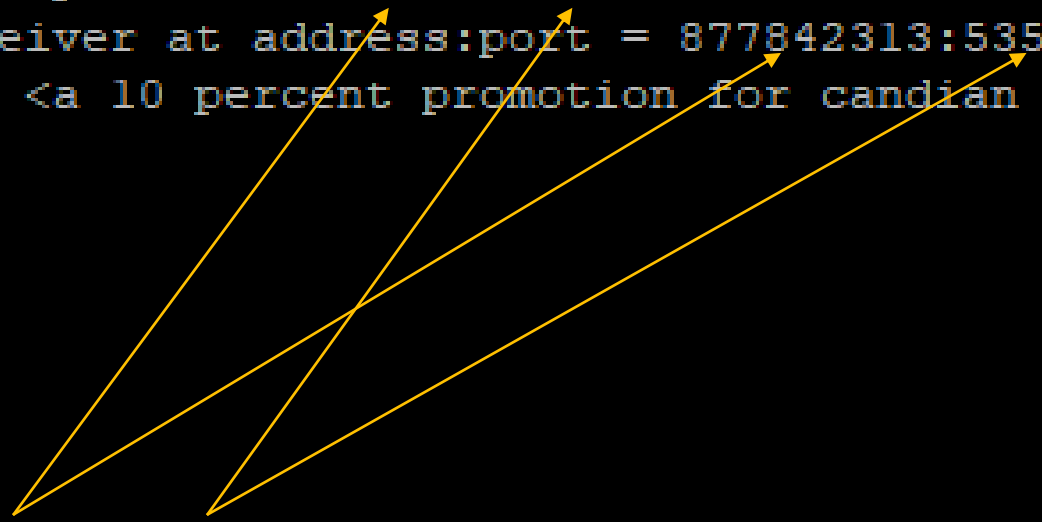
```
char *mail = "a 10 percent promotion for candian tire!";

result = sendto(sender_sd, mail, strlen(mail), 0, (struct sockaddr *)&receiver_sin, size
if (result == -1){
        printf("error in sending message to the receiver!\n");
        exit(1);
}
else
{
        printf("a mail has sent to the receiver at address:port = %d:%d\n", receiver_sin
        printf("the content of the mail is <%s>\n", mail);
}
```

Message

Very similar to `write()` to a file

Receiver's IP:PORT

```
hfani@bravo:~$ ./sender
socket has created for sender with sd:3
sender bound to the address:port = -2037592183 :53255
a mail has sent to the receiver at address:port = 877842313:53511
the content of the mail is <a 10 percent promotion for candian tire!>
```



Why IP:PORT does not look like familiar?

Sender's IP:PORT =137.207.140.134:2000

Receiver's IP:PORT=137.207.82.52:2001

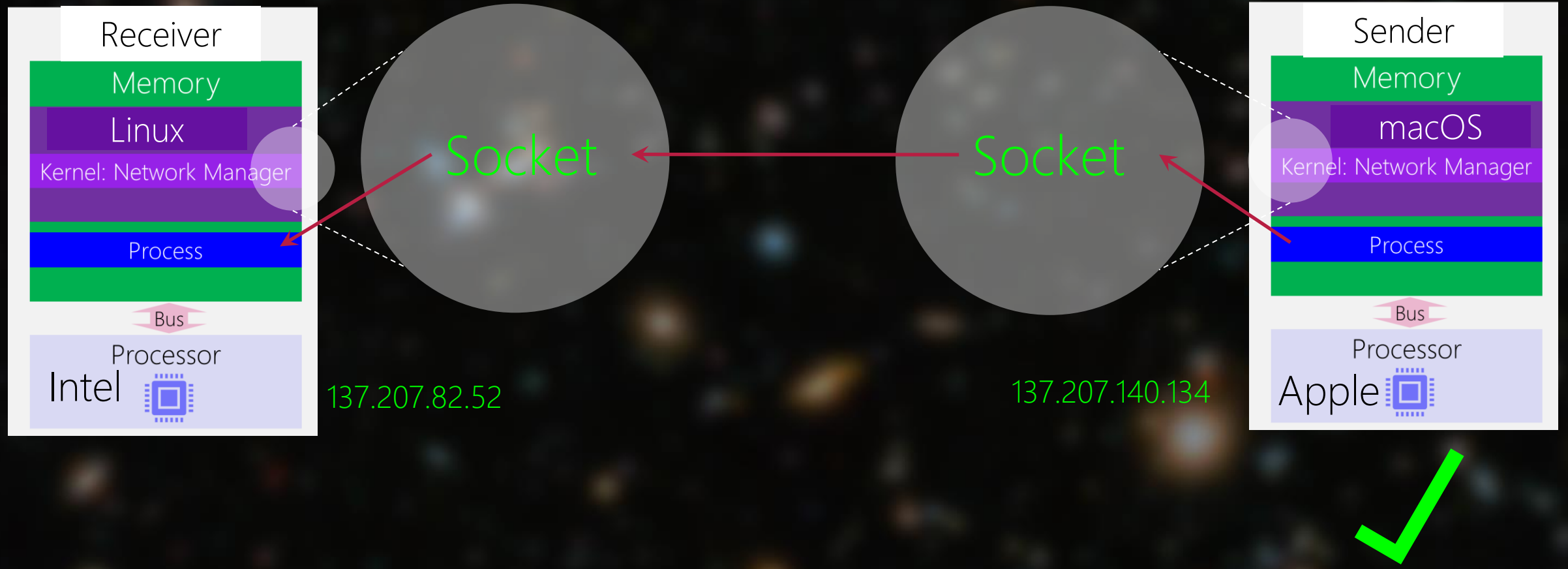
```
hfani@bravo:~$ ./sender
socket has created for sender with sd:3
sender bound to the address:port = -2037592183 :53255
a mail has sent to the receiver at address:port = 877842313:53511
the content of the mail is <a 10 percent promotion for candian tire!>
```

But there no receiver!

What happen to the mail?!

TCP/IP

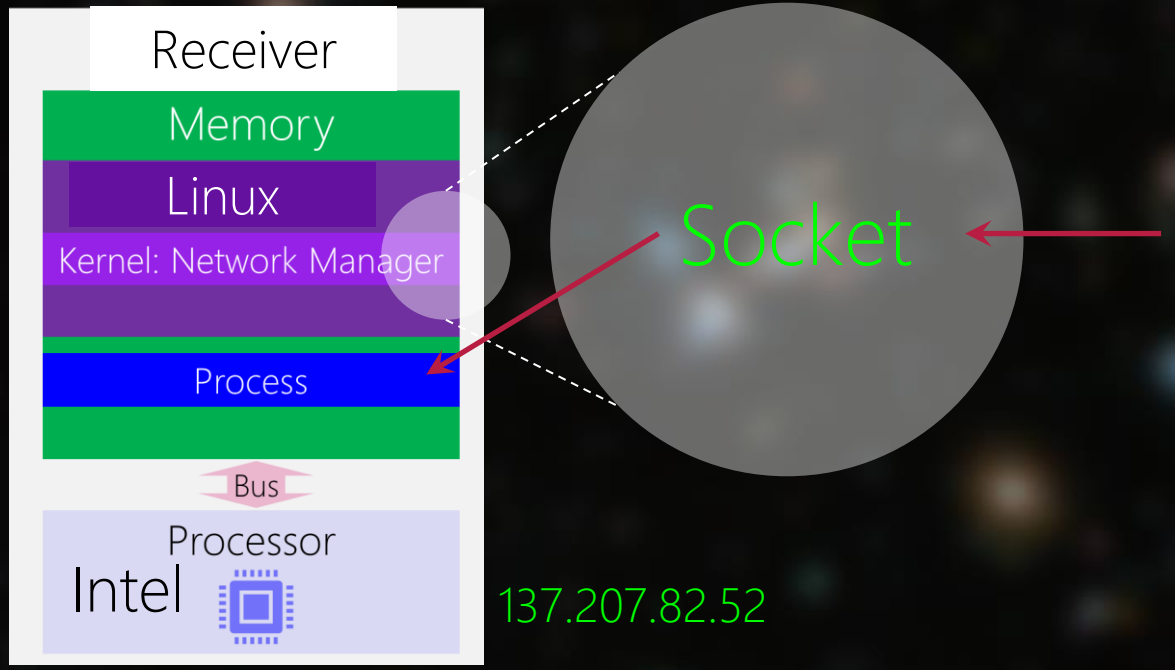
Just a name for [Link | Internet | Transport | Application] network protocol



TCP/IP: UDP at Receiver

Connectionless Communication

Sending a mail



1) Creating Socket
2) Binding to an Address (MUST)
3) Wait to receive a mail
4) Find the Sender's Address (Optional)
5) Read the Mail from the Sender