Distributed Systems
CS6421
Scalable Execution (continued)

Prof. Roozbeh Haghnazar

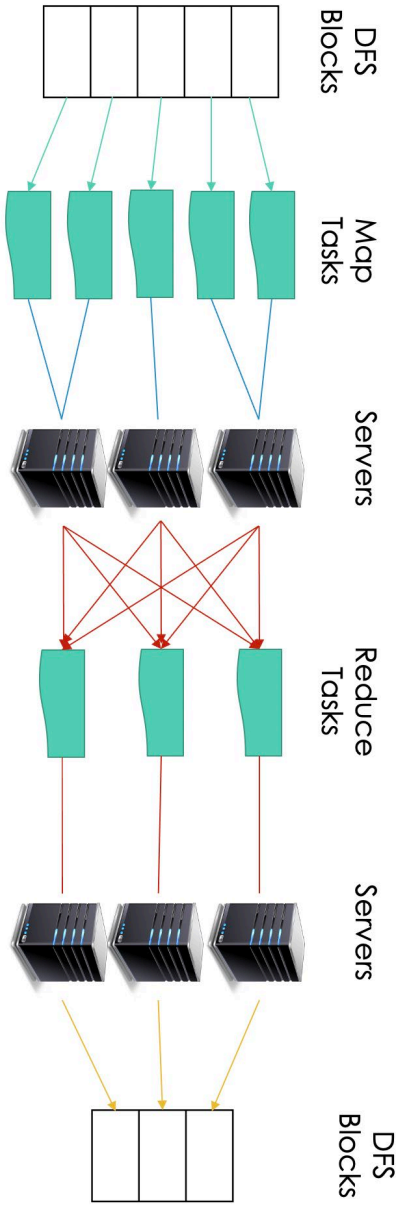
Slides Credit:

Prof. Tim Wood and Prof. Roozbeh Haghnazar

Includes material adapted from Van Steen and Tanenbaum's Distributed Systems book

Last Time… Map Reduce

- Map Phase
  - input: data element
  - Convert input data into an intermediate result
  - All map functions can be called independently in parallel
  - output: {list of keys and values}
- Shuffle and partition
  - Sort all outputs by key and combine values into a list
  - Partition keys to create new Reduce tasks
  - output: Key, {list of values}
- Reduce Phase
  - input: Key, {list of values}
  - Combine the list of values for each key to produce an output



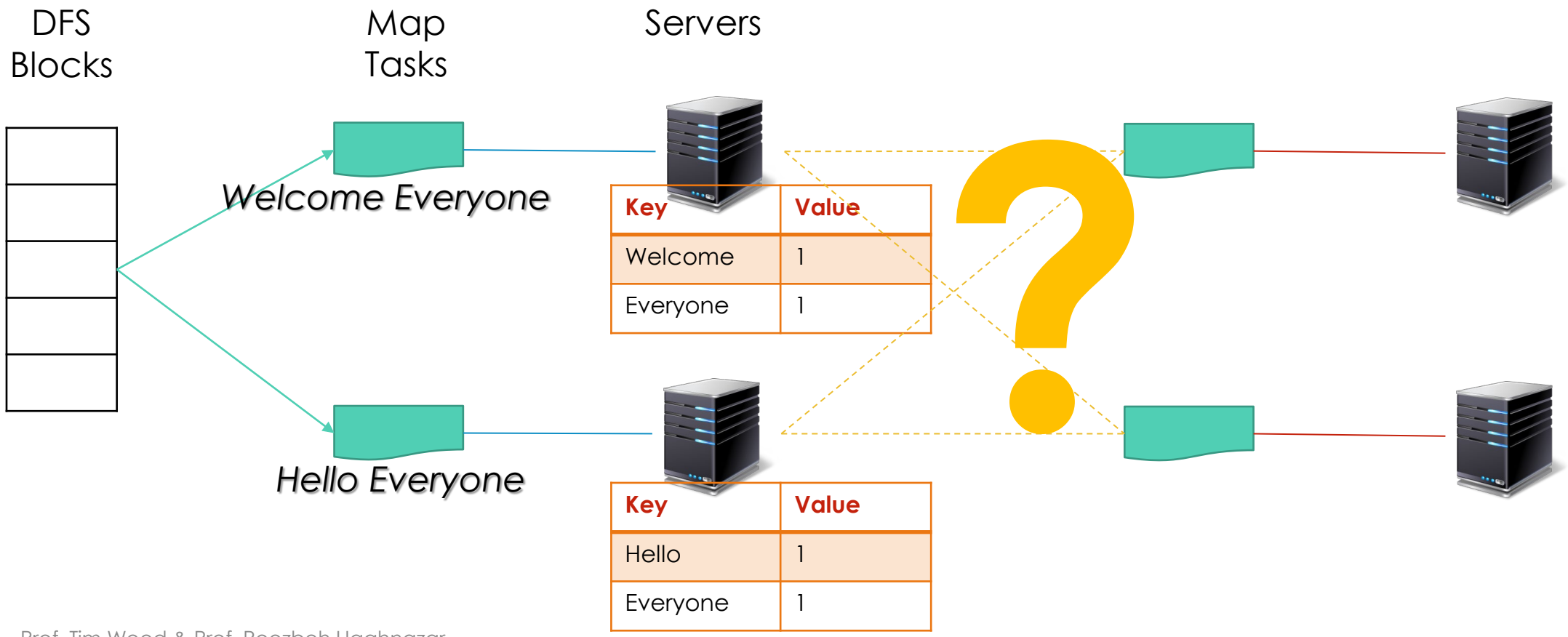
Map Reduce Parallelism?

Welcome Everyone

Hello Everyone

Key	Value
Welcome	1
Everyone	1
Hello	1
Everyone	1

Map Parallelism



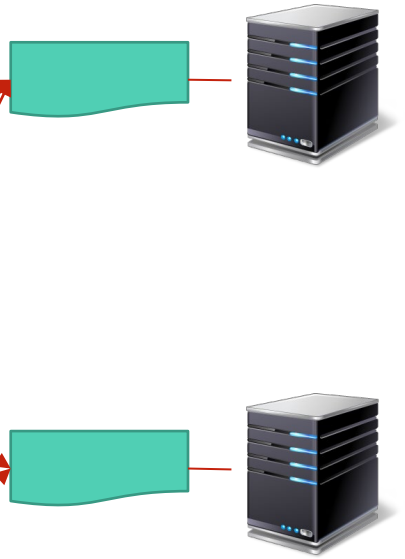
Reduce Parallelism

Reduce #=Hash(Key)%Number of Reduce processors

Servers

Key	Value	Hash (SHA-1)	%2
Welcome	1	92155f497508e89f4ce90969a330ad8c3b98b0e8	**0**
Everyone	1	c756f6af1f03c9ce381cb85934ffb274e2f54af3	**1**

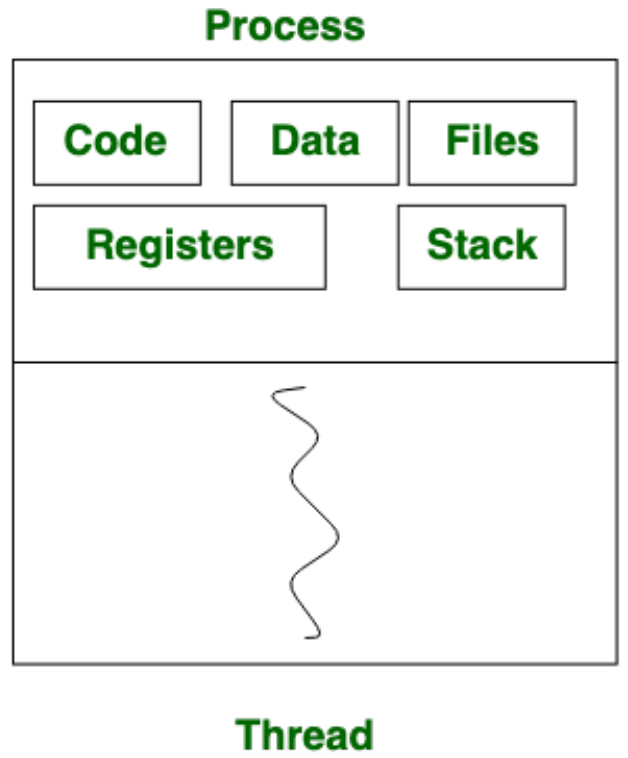
Key	Value	Hash (SHA-1)	%2
Hello	1	f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0	**0**
Everyone	1	c756f6af1f03c9ce381cb85934ffb274e2f54af3	**1**



Key	Value
Everyone	2
Welcome	1
Hello	1

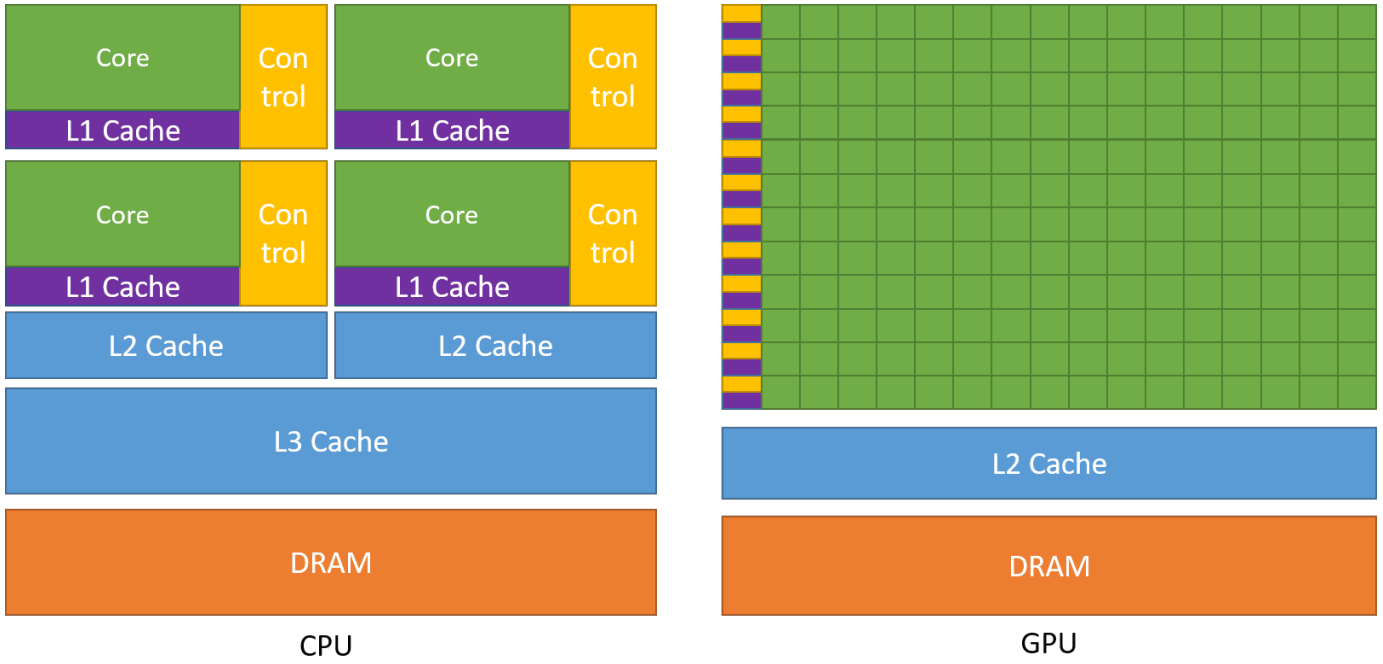
Last Time… Processes & Threads

- Processes
- Process means any program is in execution. Process control block controls the operation of any process. Process control block contains the information about processes for example: Process priority, process id, process state, CPU, register etc. A process can creates other processes which are known as Child Processes. Process takes more time to terminate and it is isolated means it does not share memory with any other process.
  - Costly and heavy creation, modification and distortion procedure
  - Process means any program is in execution.
  - Process can block the other related processes
- Threads
- Thread is the segment of a process means a process can have multiple threads and these multiple threads are contained within a process. A thread have 3 states: running, ready, and blocked. Thread takes less time to terminate as compared to process and like process threads do not isolate.
  - Easy and cheap creation and management
  - Thread means segment of a process.
  - Second thread in the same task could run, while one server thread is blocked.



CUDA PROGRAMMING

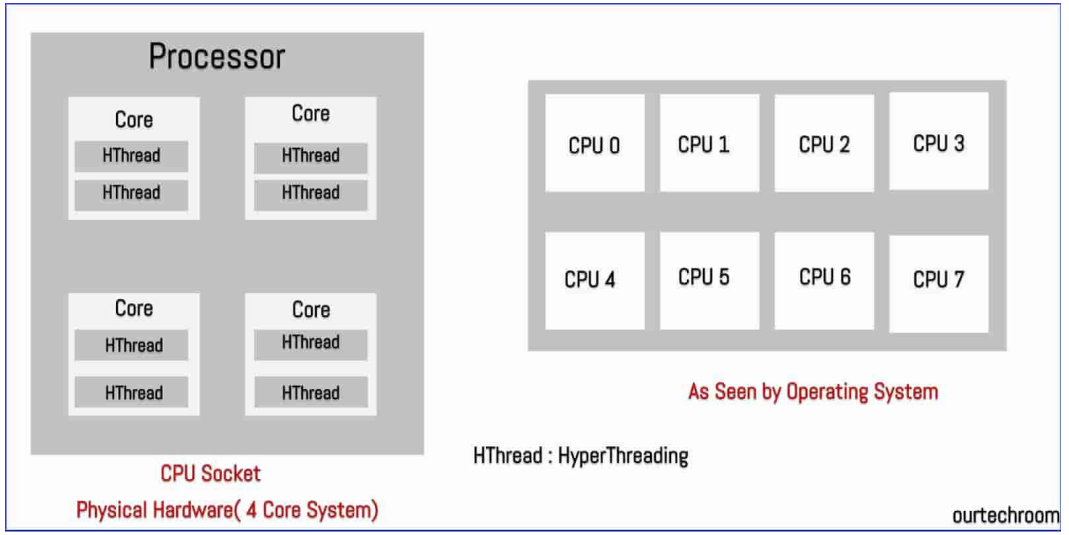
- In November 2006, NVIDIA® introduced CUDA®, a general purpose parallel computing platform and programming model that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU.
- CUDA comes with a software environment that allows developers to use C++ as a high-level programming language. As illustrated by Figure 2, other languages, application programming interfaces, or directives-based approaches are supported, such as FORTRAN, DirectCompute, OpenACC.



https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Hyper Threading and Multi-threading Security Threat: PortSmash

- Intel processors are impacted by a new vulnerability that can allow attackers to leak encrypted data from the CPU's internal processes.
- The new vulnerability, which has received the codename of PortSmash, has been discovered by a team of five academics from the Tampere University of Technology in Finland and Technical University of Havana, Cuba.
- Researchers have classified PortSmash as a side-channel attack. In computer security terms, a side-channel attack describes a technique used for leaking encrypted data from a computer's memory or CPU, which works by recording and analyzing discrepancies in operation times, power consumption, electromagnetic leaks, or even sound to gain additional info that may help break encryption algorithms and recovering the CPU's processed data.
- The researchers team also published proof-of-concept (PoC) code on GitHub that demonstrates a PortSmash attack on Intel Skylake and Kaby Lake CPUs.



https://ourtechroom.com/tech/secret-of-cpu-hyperthreading/

Prof. Tim Wood & Prof. Roozbeh Haghnazar

Where Does Code Actually Run?

Layers and Layers…

- In a "cloud"

Layers and Layers…

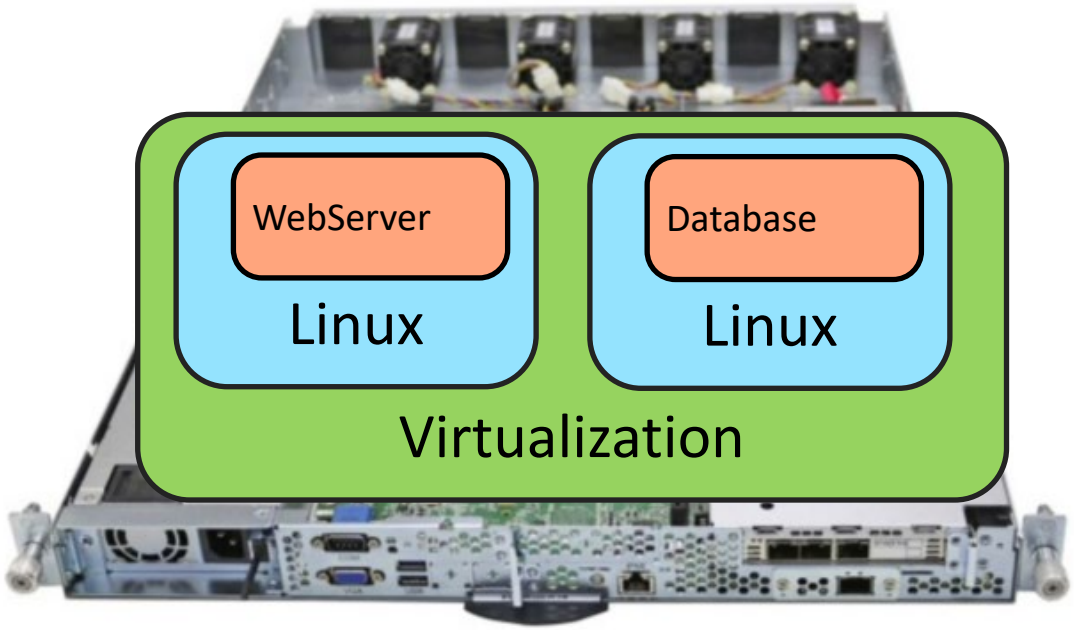
- In a "cloud"
  - In a data center

Layers and Layers…

- In a "cloud"
  - In a data center
    - In a server

Layers and Layers…

- In a "cloud"
  - In a data center
    - In a server
      - In a Virtual Machine
        - In an OS
          - In a process
            - In a thread



CHALLENGES???

1. Heterogeneity
2. Openness
3. Security
4. Failure Handling
5. Concurrency
6. Quality of Service
7. Scalability
8. Transparency

- In a "cloud"
  - In a data center
    - In a server
      - In a Virtual Machine
        - In an OS
          - In a process
            - In a thread

CHALLENGES???

1. Heterogeneity
2. Openness
3. Security
4. Failure Handling
5. Concurrency
6. Quality of Service
7. Scalability
8. **Transparency**

- In a "cloud"
  - In a data center
    - In a server
      - In a Virtual Machine
        - In an OS
          - In a process
            - In a thread

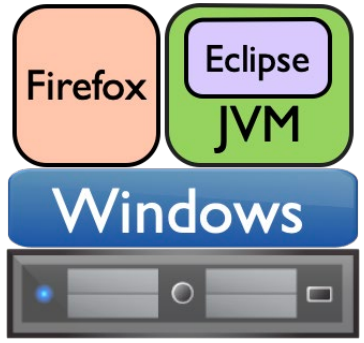
What is virtualization?

- An extra **interface** that mimics the behavior of a lower layer
- Used since 1970s so new mainframes could support legacy applications

OS ⟷ SW Interface

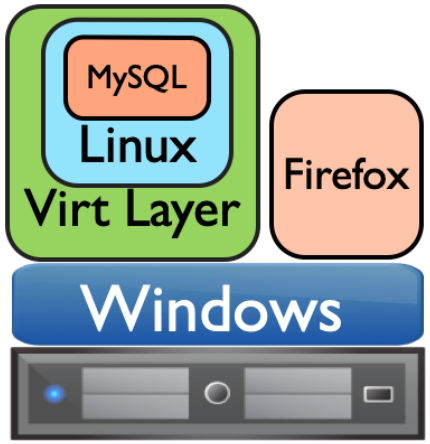
OS ⟷ HW Interface

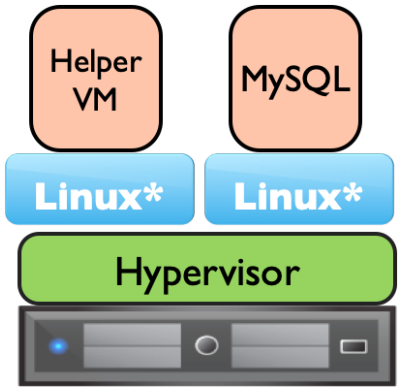
Types of Virtual Machines



1. **Application Virtualization**
   - Runs application code
   - *Java VM, WINE*

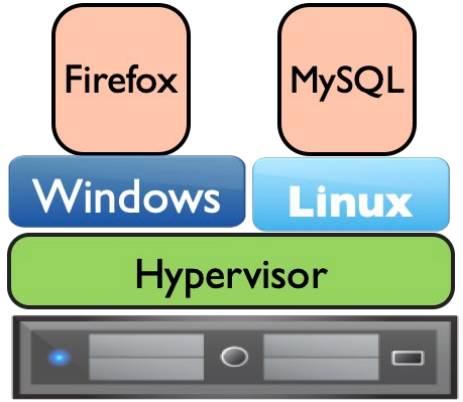
2. **Hosted Virtualization**
   - Virtualizes a full OS and apps
   - *VMware Player, VirtualBox*





3. **Paravirtualization**
   - Modify OS to simplify hypervisor
   - *Xen*

4. **Full Virtualization**
   - Runs directly on HW
   - *VMware ESXi*



How to Virtualize?

- Virtualization layer replaces an interface
- Must intercept calls and translate them
  - Java - interpret/compile code to match host
  - Hosted VM - translate system calls for host OS
  - Full Virtualization - trap on sensitive instructions
- How to allocate resources?
  - VMs must share memory and CPU time
- How to handle I/O?
  - Abstraction layer separates VM from physical hardware

Why Virtualize?

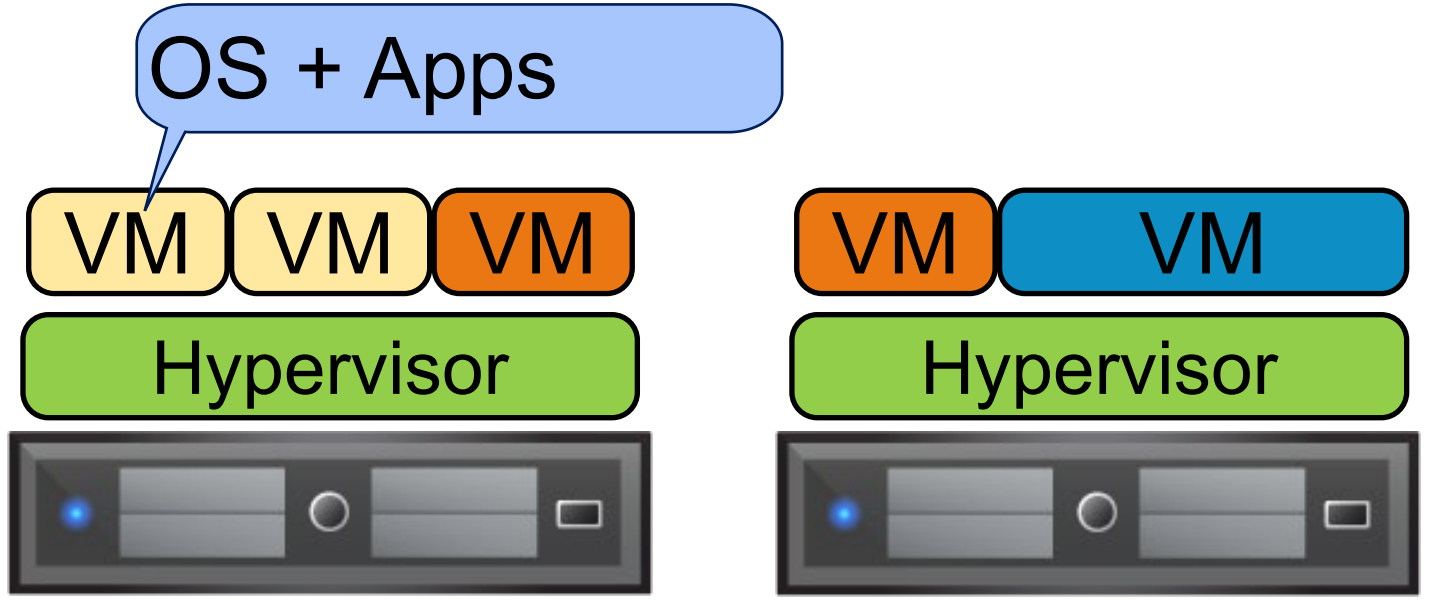
- Consolidation
  - Can split a physical server into many smaller servers
- Security
  - VMs are isolated from one another
- Resource management
  - Can dynamically adjust a VM's CPU and memory share
- Convenience
  - VM is abstracted away from physical hardware
  - Great for development

More later!

VMs in Cloud Data Centers

- Virtualization is used to **multiplex** a physical server
  - Allows multiple customers to share one machine
  - Simplifies management since VMs are not strictly tied to HW
  - Provides isolation between cloud users

Cloud Data Center



Amazon EC2

Infrastructure as a Service

VM Type	Description	Cost
t3.Micro	1GB RAM, up to 1 core	$0.01 / hour
t3.Large	8GB RAM, ~2 cores	$0.08 / hour
c5.18xlarge	144GB RAM, 72 cores	$3.06 / hour

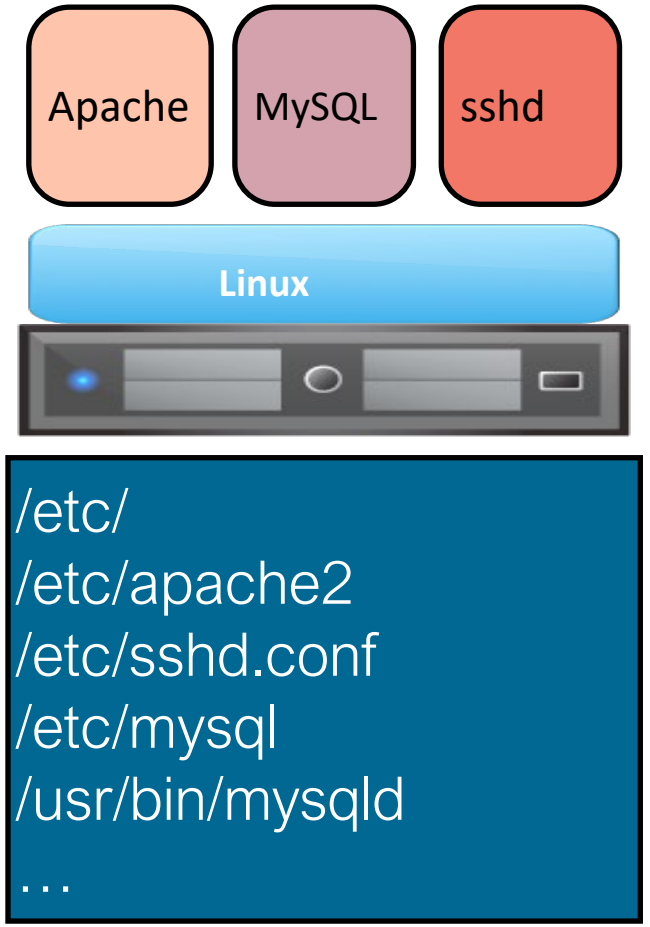
What are the Downsides of Virtual Machines?

VMs are "Heavy"

- Each Virtual Machine needs to be allocated resources
  - RAM – 512MB for smallest VM on AWS
  - CPU – switching between VMs is an expensive context switch
  - Full operating system may be redundant if many VMs are similar

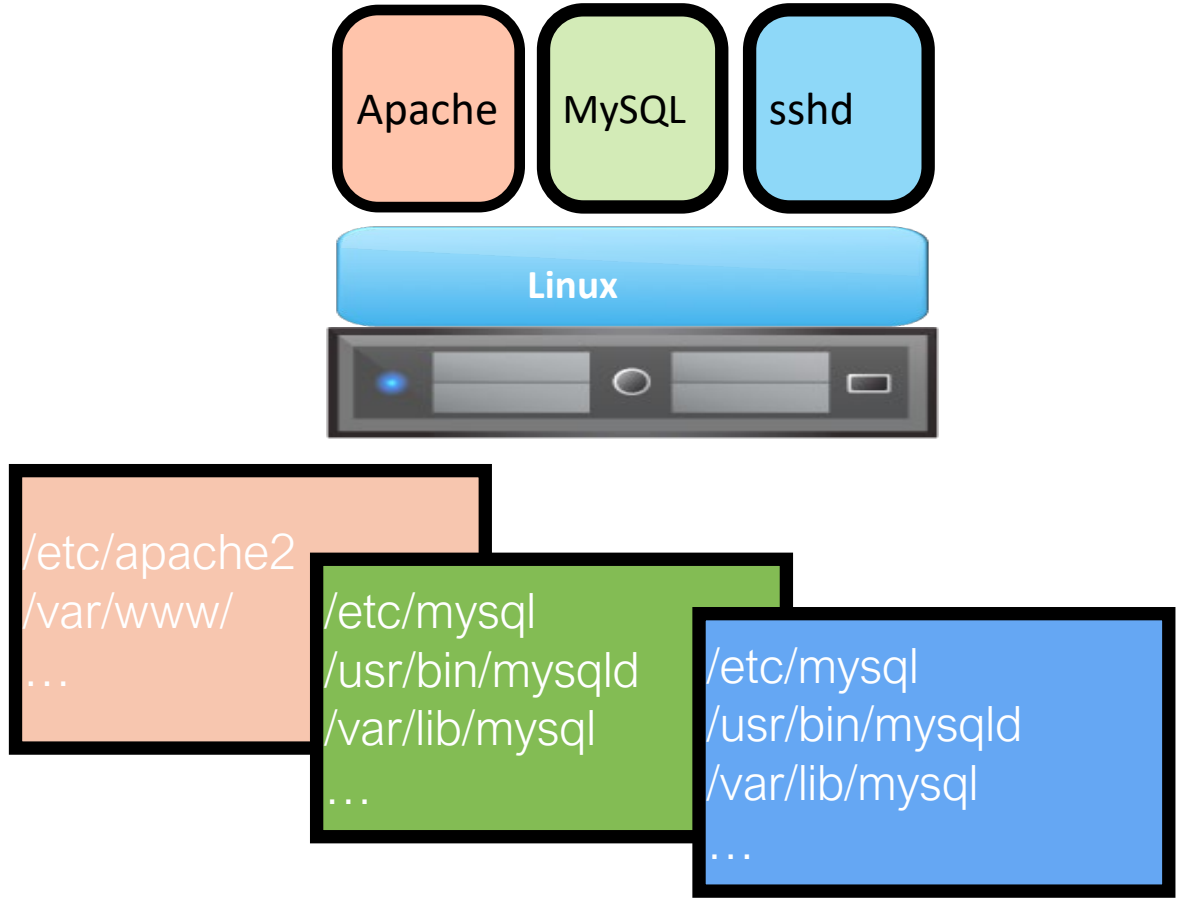
Are Processes Better?

- Processes
  - OS provides isolation
- Isolated:
  - Memory
- Shared:
  - File system
  - Network
  - Devices
  - OS Kernel



Containers

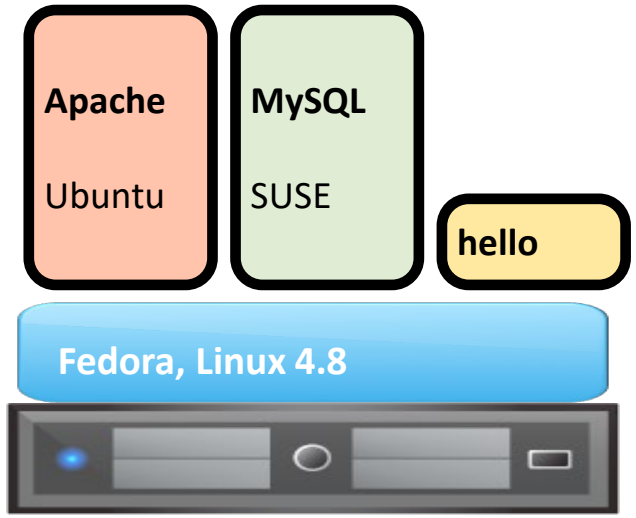
- Containers
  - Namespace-based isolation using LXC and cgroups
- Isolated:
  - Memory
  - File system
  - Network
  - Devices
- Shared:
  - OS Kernel



Container Packaging

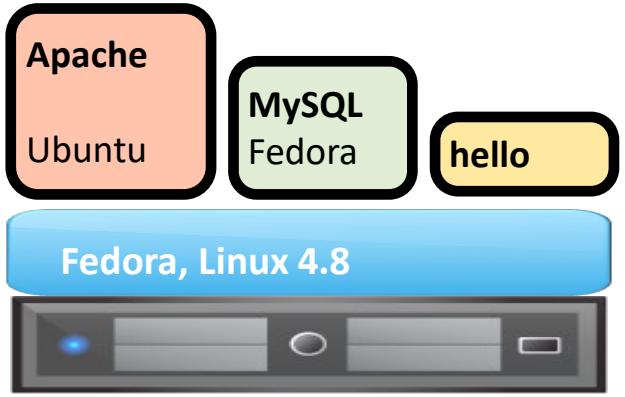
- Deployment - big benefit of containers/virtualization
  - Lets you package up an application and all of its requirements
  - Even the distribution and 3rd party utilities!
  - Very helpful for system administrators

- Container "image" includes:
  - Linux distribution base files
  - Dependency libs/utils
  - Configuration files
  - Application to run



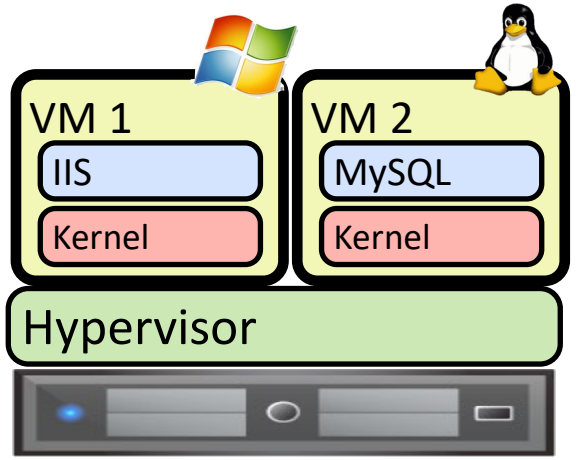
- **Can inherit files/libraries from host to reduce size of the container package!**

Container vs VMs



Containers

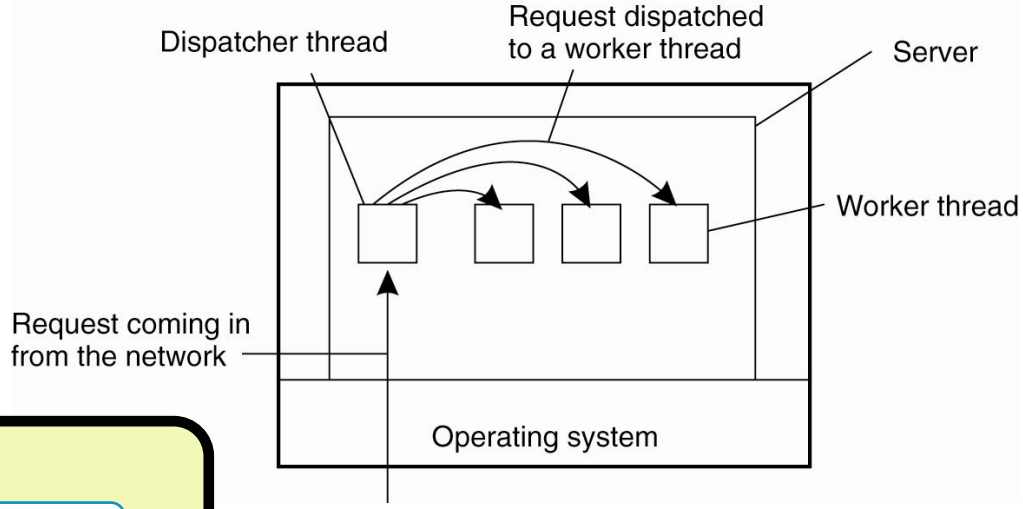
- ???

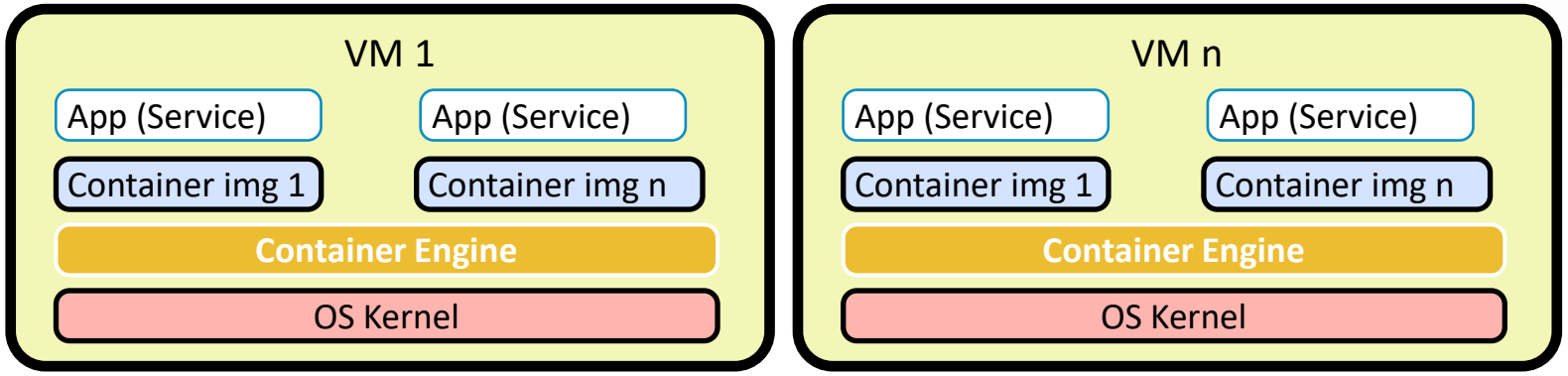


VMs

- ???

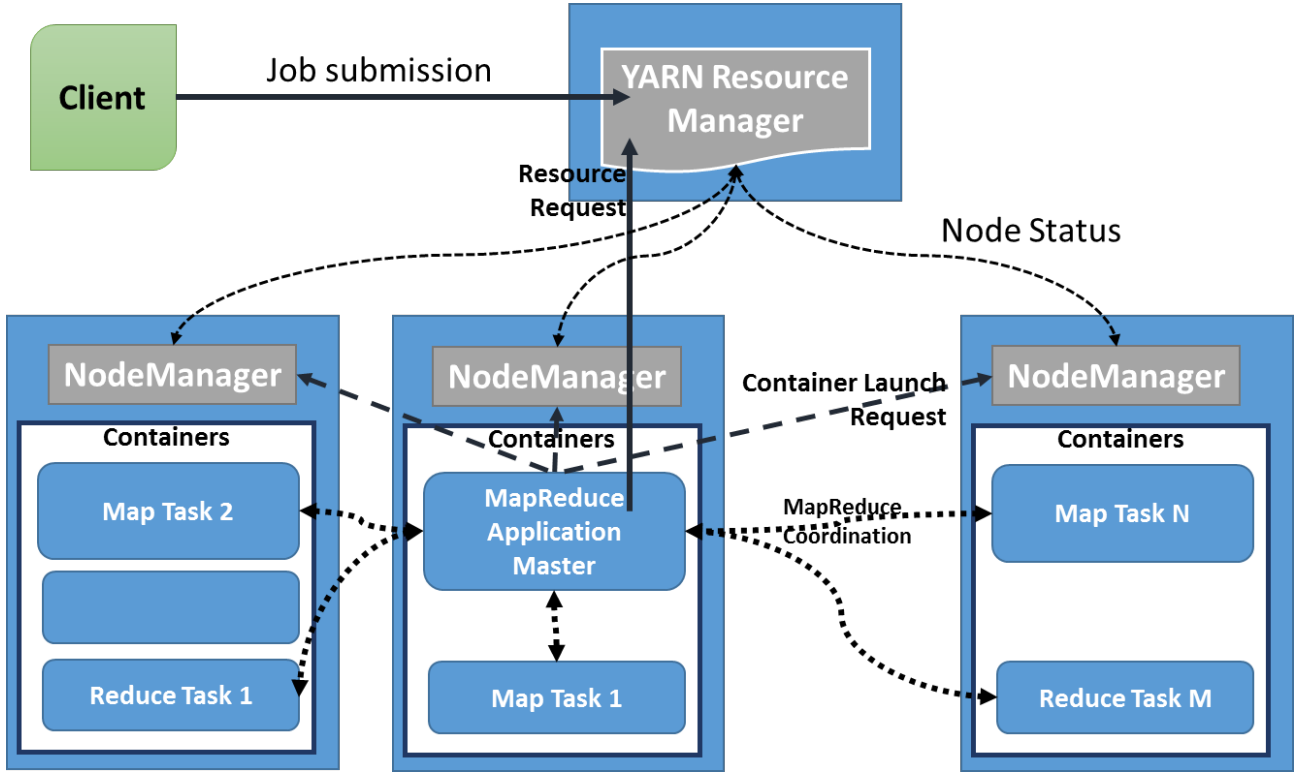
REGULAR ORGANIZATIONS





YARN Scheduler

- There are some servers
- Containers run the jobs
- Application Masters control and execute the tasks
- Node Manager can inform the Resource Manager that hey one task is done.
- Then RM, inform the AM who has a task to get done.
- AM negotiate with NM to get the task to the container to get done.



Distributed Systems
CS6421
Communication

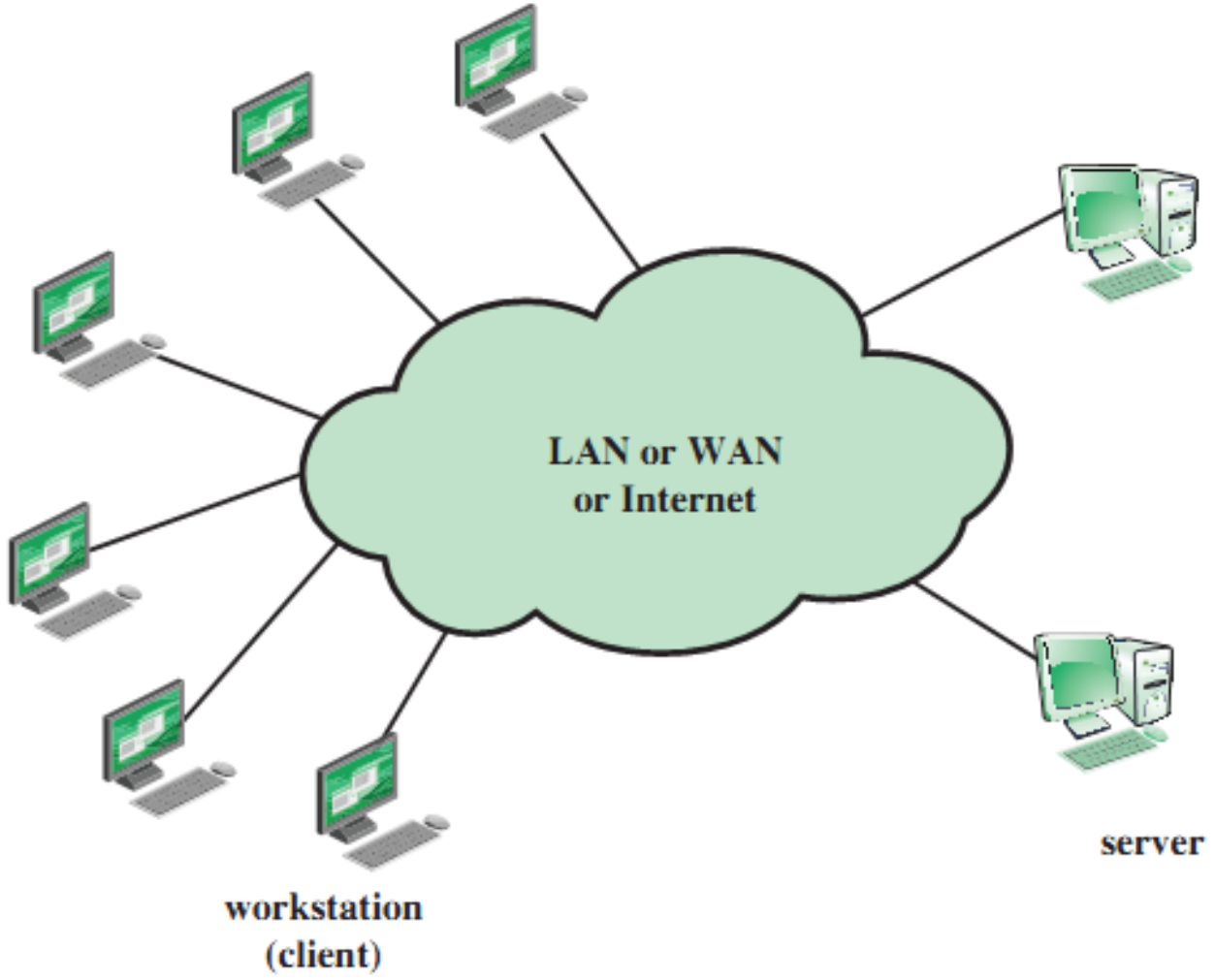
Prof. Tim Wood and Prof. Roozbeh Haghnazar

Includes material adapted from Van Steen and Tanenbaum's Distributed Systems book

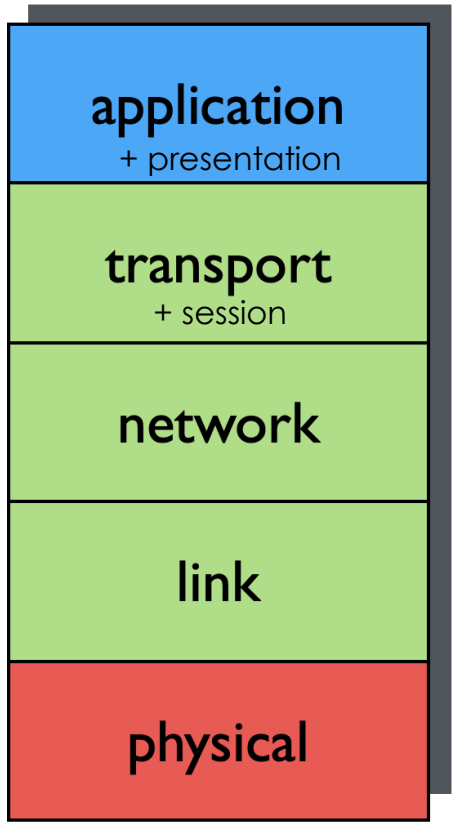
Client/Server Model

- Need a way to send data
- Need a way to find destinations

- Need to share the network
- Need to handle failure or loss

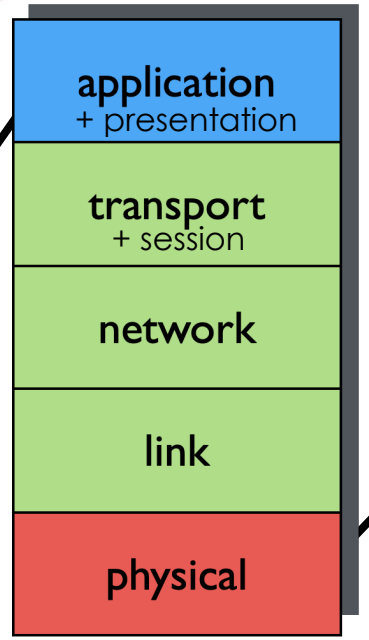


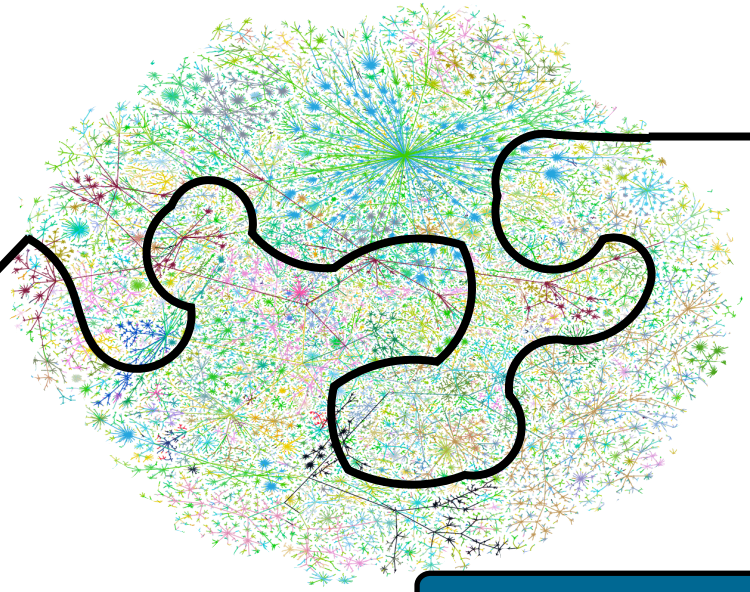
Layered OSI Model



- **Application (+ presentation)**: <your code here>
  - FTP, SMTP, HTTP
- **Transport (+ session)**: data transfer
  - TCP, UDP
- **network**: routing protocols
  - IP
- **link**: adjacent nodes
  - Ethernet, 802.111 (WiFi), PPP
- **physical**:
  - bits on the wire or in the air

How to speak *website*?





`faculty.cs.gwu.edu`

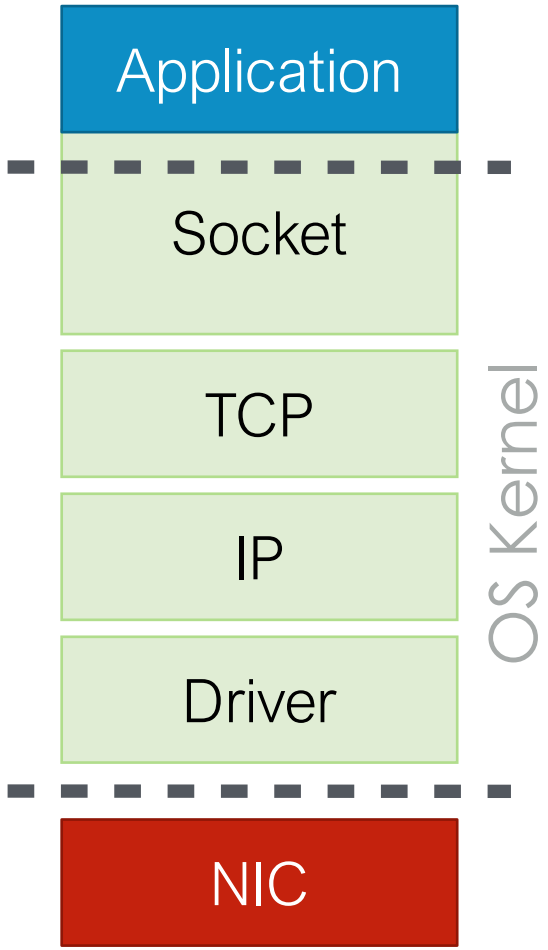
```
HTTP/1.1 200 OK
Server: GitHub.com
Content-Type: text/html; charset=utf-8
Last-Modified: Thu, 17 Sep 2020 19:26:33 GMT
ETag: "5f63b869-b6"
Access-Control-Allow-Origin: *
Expires: Thu, 17 Sep 2020 19:56:47 GMT
…
```

```
telnet faculty.cs.gwu.edu 80
```

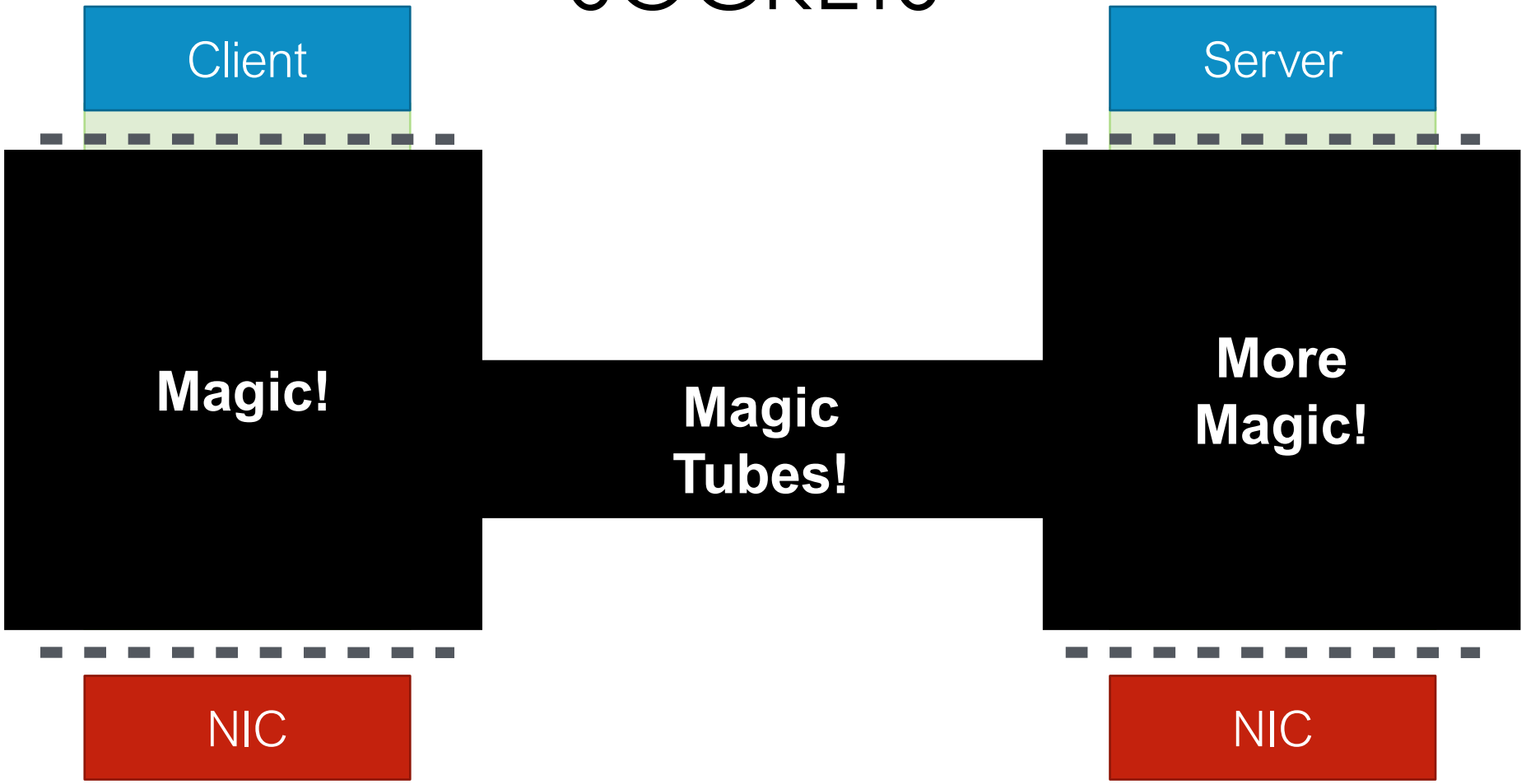
```
GET /timwood/simple.html HTTP/1.1
Host: faculty.cs.gwu.edu
(blank line)
```

SOFTWARE LAYERS

- Network Interface Card (NIC)
  - Reads "bytes on wire"
- Driver
  - Moves data from NIC to main memory
- Internet Protocol (IP)
  - Handles addressing and routing
- Transmission Control Protocol (TCP)
  - Ensures reliable, ordered transmission of packets and manages congestion
- Socket
  - Provides interface between OS and App



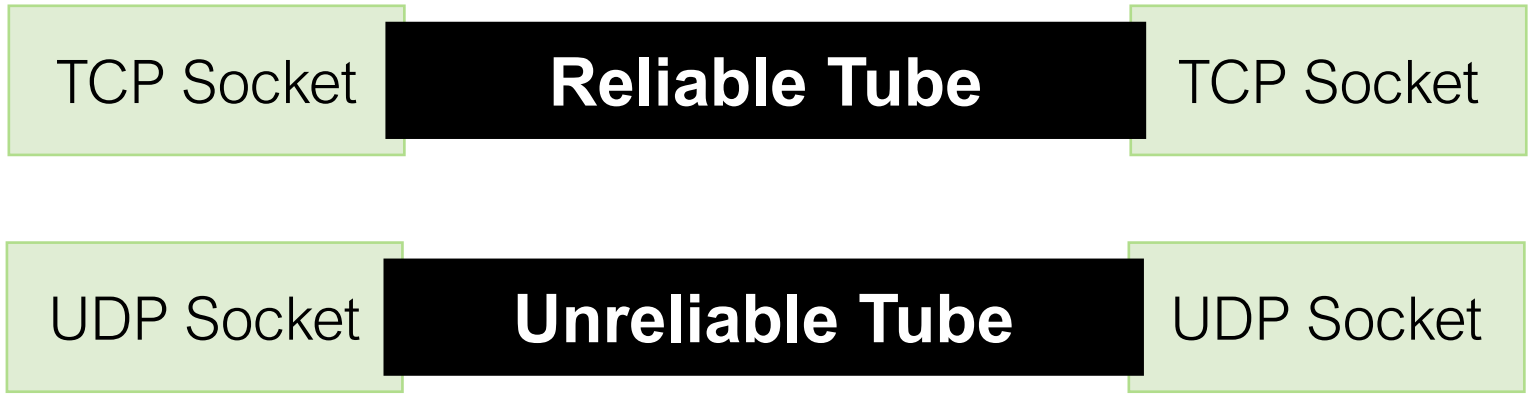
SOCKETS



ABSTRACTIONS

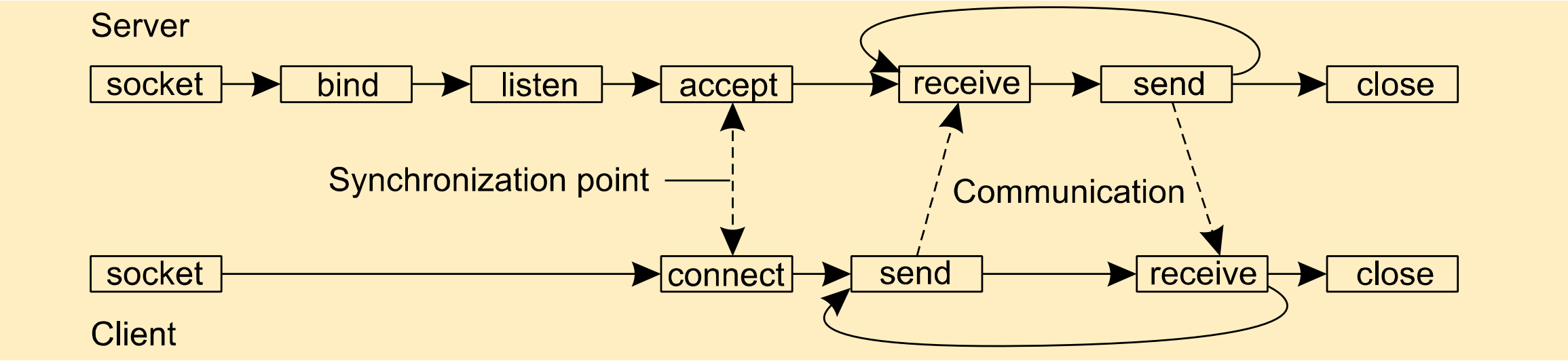
- Networking (and all CS) is about abstraction layers!
- We don't need to know how something works if we understand its inputs and outputs

- …but we do need to understand the guarantees that lower abstraction layers are providing!



SOCKETS

- Socket
- Connect
- Bind, Listen, Accept
- Send, Receive
- Close



Message Based Protocols

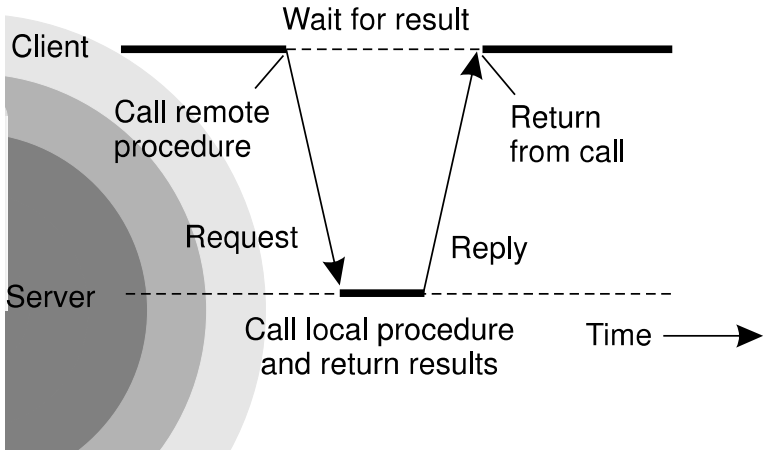
- Server waits for client to send a message
- Client sends a message
- Server unpacks message, processes it, packs a new message, returns it

- Many message protocols are text based
  - HTTP, FTP, SMTP, IRC
- Others are binary-based
  - HTTP/2,

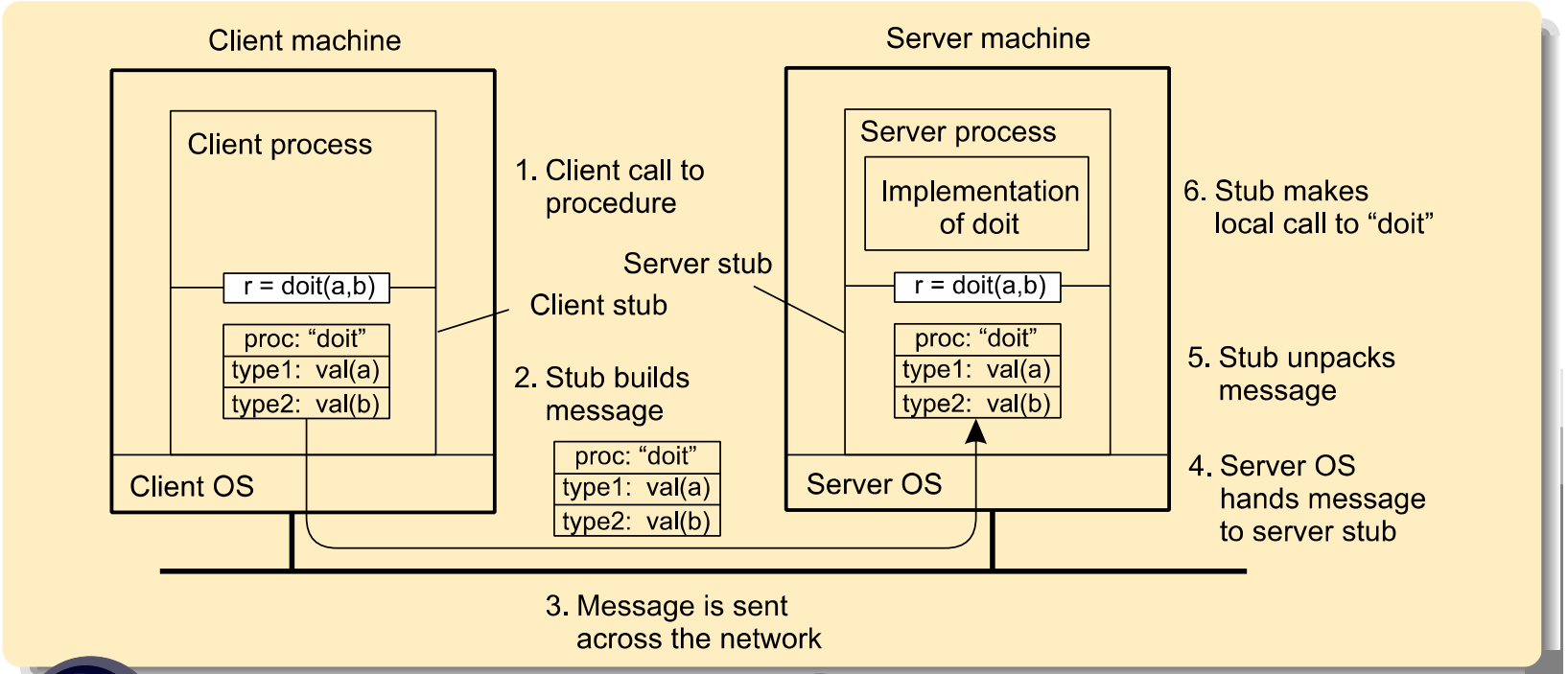
- Benefit? Drawback?

Remote Procedure Calls

- RPC is a layer of abstraction built on top of sockets exchanging messages
- Synchronous
  - Client waits for result
  - Often run in a thread/goroutine!
  - (Async variants exist in some languages)
- RPC layer handles packaging and sending data needed to process function



RPC Implementation



1. Client procedure calls client stub.
2. Stub builds message; calls local OS.
3. OS sends message to remote OS.
4. Remote OS gives message to stub.
5. Stub unpacks parameters; calls server.

6. Server does local call; returns result to stub.
7. Stub builds message; calls OS.
8. OS sends message to client's OS.
9. Client's OS gives message to stub.
10. Client stub unpacks result; returns to client.

RPC Challenges

- Client and server machines may have **different data representations** (think of byte ordering)
- Wrapping a parameter means **transforming a value into a sequence of bytes**
- Client and server have to **agree on the same encoding**:
  - How are basic data values represented (integers, floats, characters)
  - How are complex data values represented (arrays, unions)

- Client and server need to properly interpret messages, transforming them into machine-dependent representations.

RPC IN GO

- Go RPC package: https://golang.org/pkg/net/rpc/

> Package rpc provides access to the exported methods of an object across a network or other I/O connection. A server registers an object, making it visible as a service with the name of the type of the object. After registration, exported methods of the object will be accessible remotely. A server may register multiple objects (services) of different types but it is an error to register multiple objects of the same type.

- Exported methods must follow this template:

```
func (t *T) MethodName(args T1, reply *T2) error
```

- T1 is a struct for input arguments, T2 is for reply. Both must support **marshalling**

Communication + Execution

- Execution in…
  - Processes, threads, VMs, containers
- Communicating via…
  - Sockets, RPC, message queues, publish/subscribe, multicast

- These combine to form Distributed Systems

> "A distributed system is a collection of independent computers that appears to its users as a single coherent system."