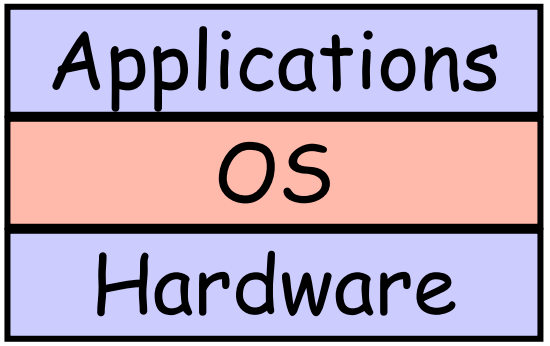
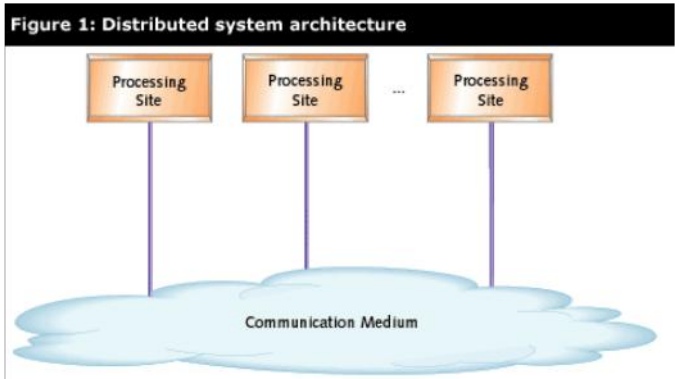
Operating Systems

- An operating system (OS) is:
  - a software layer to abstract away and manage details of hardware resources
  - a set of utilities to simplify application development



- Think about an OS across multiple computers
  - a software layer to abstract away and manage details *of multiple hardware resources*
  - a set of utilities to simplify application development on *multiple computers*

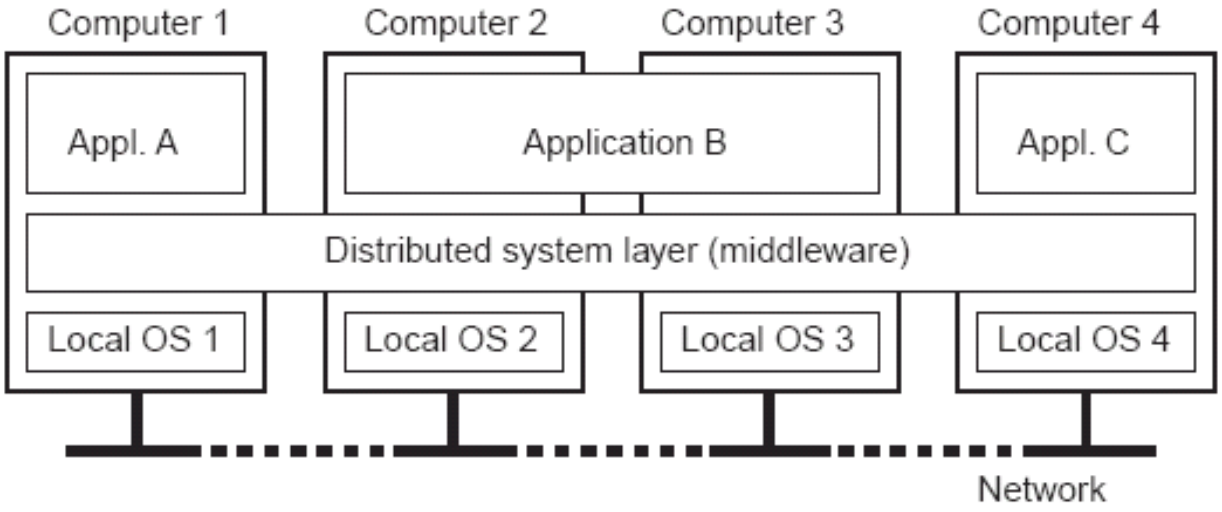


Distributed OS is an OS that works across multiple independent computers

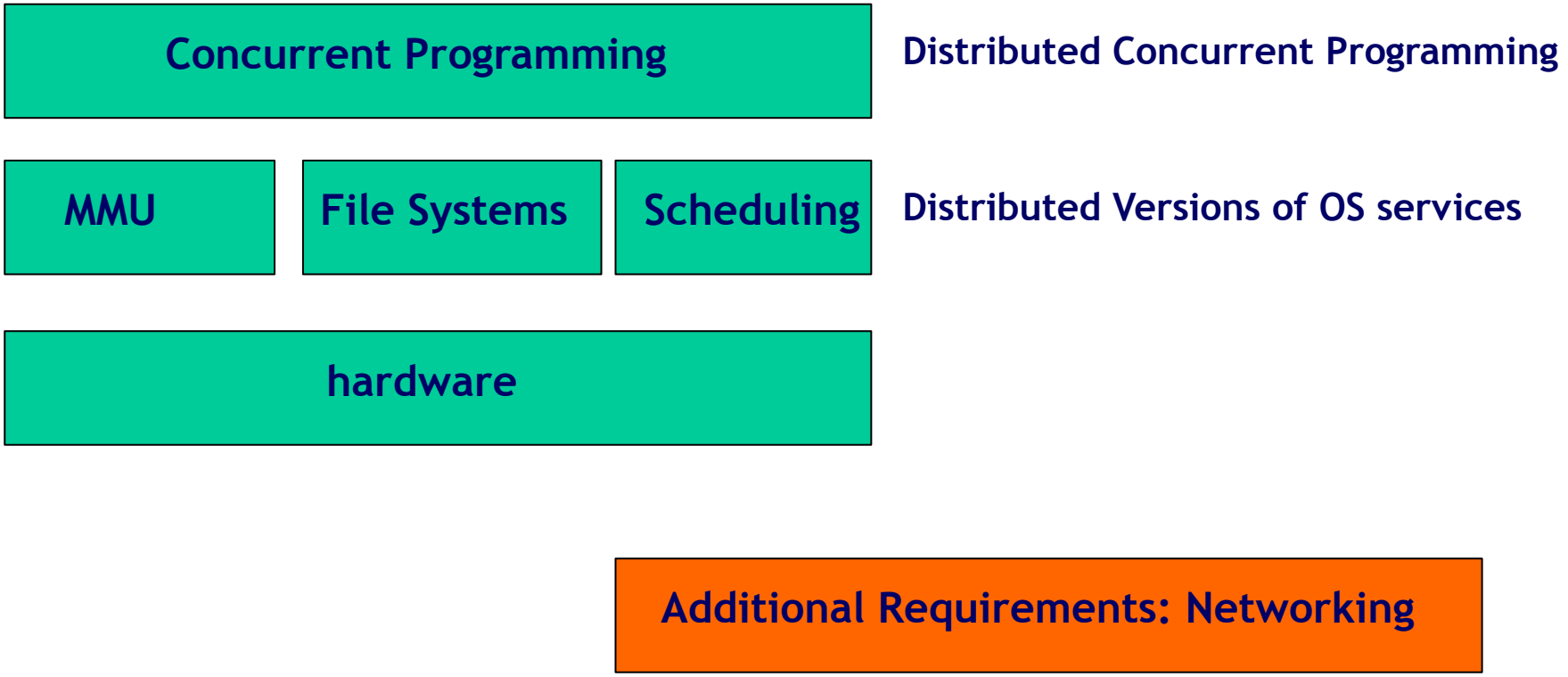
A distributed system is a piece of software that ensures that:

> *a collection of independent computers appears to its users as a single coherent system*

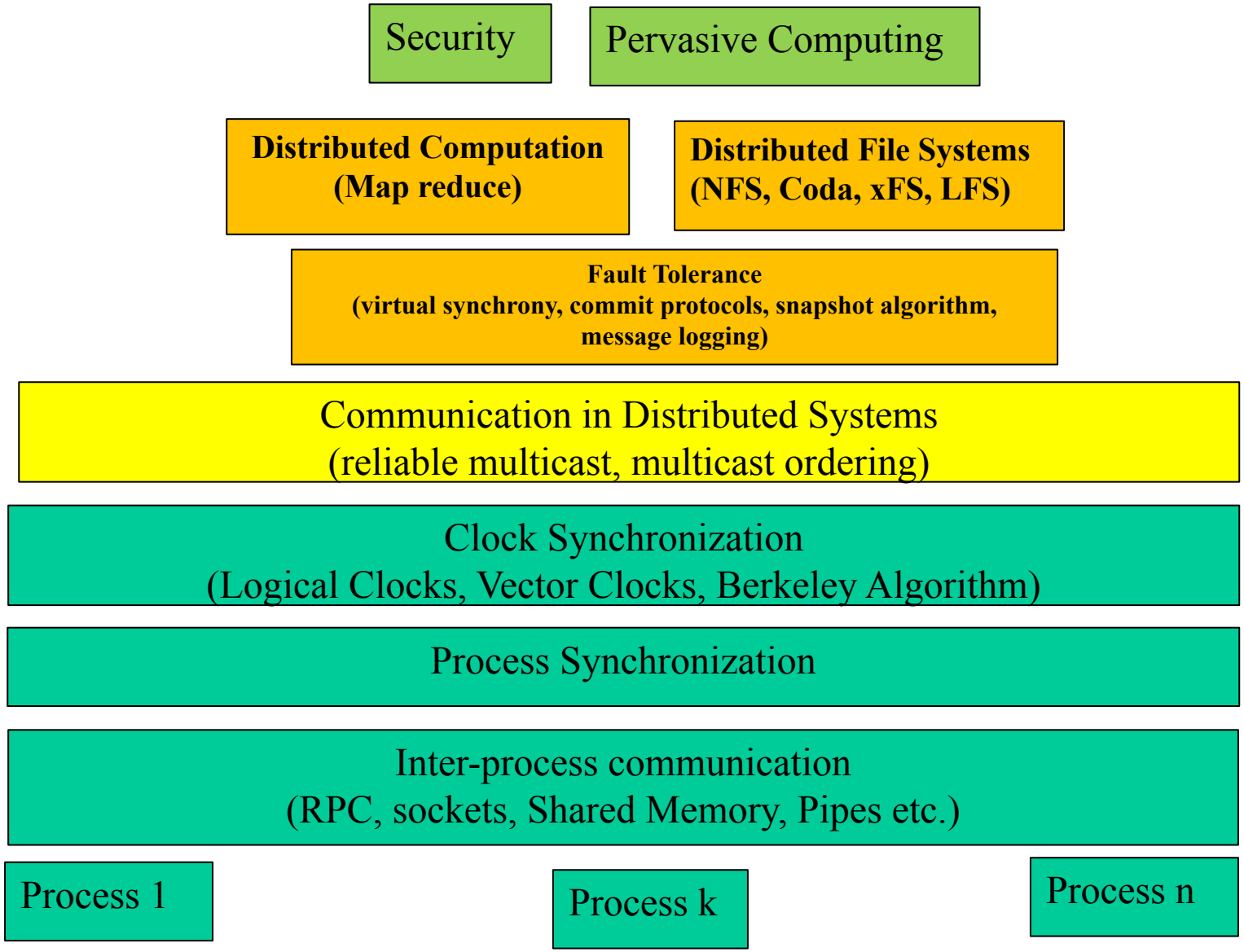
Two aspects: (1) independent computers and
(2) single system ⇒ middleware.



Designing distributed operating system requires understanding of basic operating systems



Overview of Topics



Designing Distributed Systems is challenging!

- Concurrency Control
  - Threads, Parallelism, Ordering, Race Condition
  - Communication in Distributed Systems
  - Clock Synchronization
  - Transactions
- Fault Tolerance
- Distributed Storage
- Security

Goals of Distributed Systems

- Distribution transparency
- Openness
- Available resources
- Scalability

Transparency in a Distributed System

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Openness in Distributed Systems

- Offers services according to standard rules that describe syntax and semantics of the services
- Can interact with services from other open systems, irrespective of the underlying environment
- In computer networks, standard rules govern the format, contents and meaning of messages sent and received
- In distributed systems, services are specified through interface description language (IDL)

Concurrency

- Multithreaded capabilities necessary for distributed computations
  - pthreads
- Communication allows systems to know status of other processes
  - IPC
- Clock synchronization ensures systems are in the same phase of computations
- Locking
  - Mutexes and semaphores

Pthreads

- UNIX thread library
- Allows for the creation and management of threads
- Quick pthread demo

IPC in Distributed Computing

- Shared memory
  - Processes share memory to read/write
- Memory mapped files
  - A file is mapped to memory
  - Processes interact with file
  - File persists after restart to restore data
- Sockets
  - Communicate over a network

Fault Tolerance

- Distributed operating systems should be able to continue functioning in the presence of faults
- Related to dependability

Dependability

- Availability
- Reliability
- Safety
- Maintainability

Availability & Reliability

- **Availability**: A measurement of whether a system is *ready to be used immediately*
  - System is up and running at any given moment
  - A system goes down 1ms/hr has an availability of more than 99.99%, but is unreliable

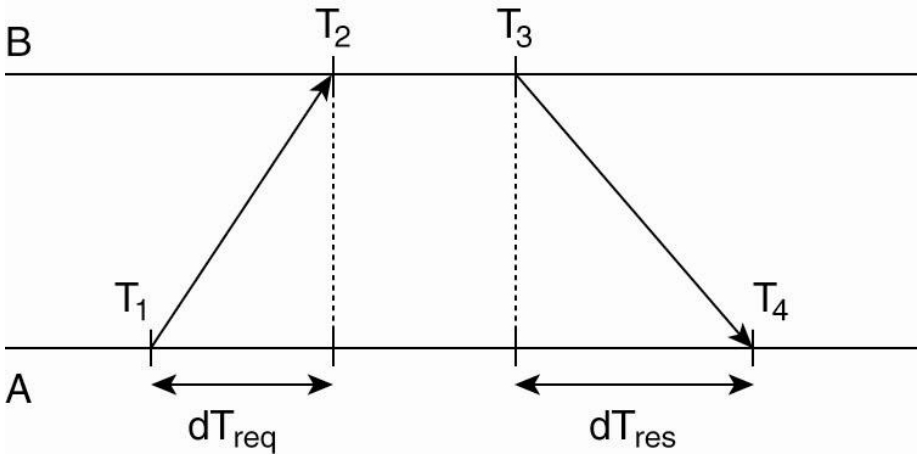
- **Reliability**: A measurement of whether a system can *run continuously without failure*
  - System continues to function for a long period of time
  - A system that never crashes but is shut down for a week once every year is 100% reliable but only 98% available

Safety & Maintainability

- **Safety**: A measurement of *how safe failures are*
  - System fails, nothing serious happens
  - For instance, high degree of safety is required for systems controlling nuclear power plants
- **Maintainability**: A measurement of *how easy it is to repair a system*
  - A highly maintainable system may also show a high degree of availability
  - Failures can be detected and repaired automatically? Self-healing systems?

Clock Synchronization

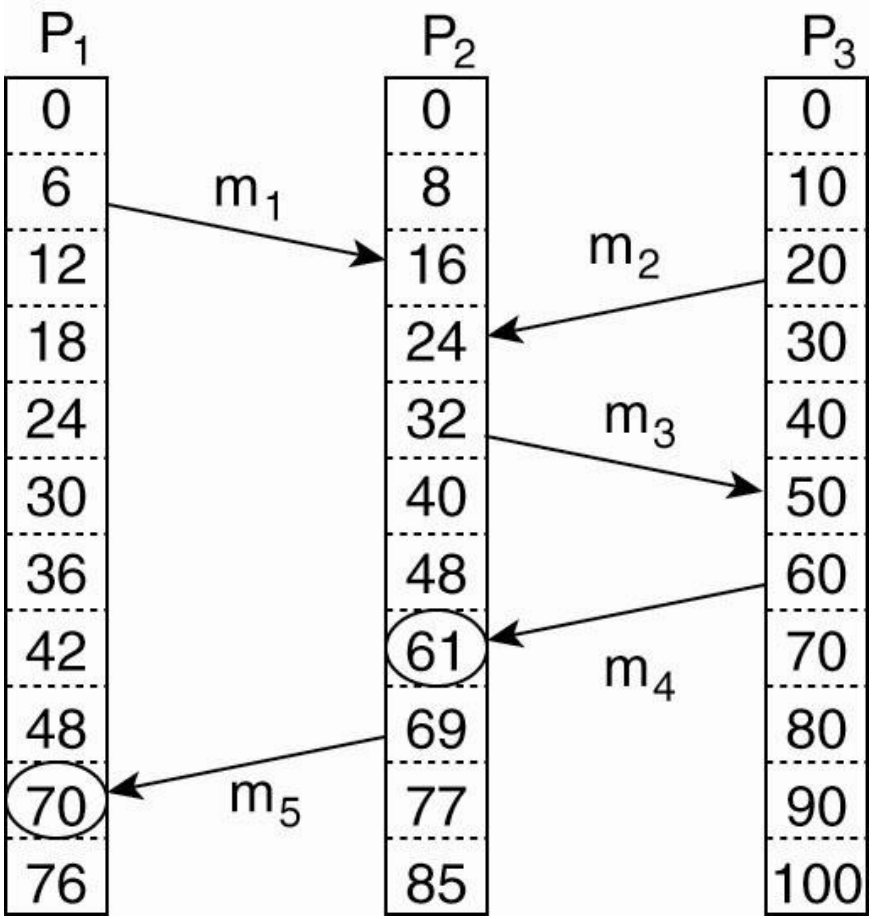
- Ensure processes are in the same phase of computation
  - GPS
- Cristhian's Algorithm



- Berkeley Algorithm
  - Time server asks for all clock values
  - Participants respond
  - Server sends out new time based on average

Clock Synchronization

- Lamport's Logical Clocks
  - The "happens-before" relation $\rightarrow$ can be observed directly in two situations:
  - If *a* and *b* are events in the same process, and *a* occurs before *b*, then $a \rightarrow b$ is true.
  - If a is the event of a message being sent by one process, and *b* is the event of the message being received by another process, then $a \rightarrow b$



Distributed Storage

- File system spread over multiple machines
- Caching
  - Cache data on machines
  - Reduces retrieval time
  - Data consistency a concern
- Mounting
  - Construct hierarchy of file system over network
  - File systems mounted on individual nodes
  - OS stores mount table: map of mount points over network
- Transparency
  - Hide where file is physically located

Distributed Computing: Map Reduce

- Allows processing of large dataset spread across multiple machines

- Two phases
  - Map stage: Takes in a value and outputs (key, value) pairs
  - Reduce stage: Takes in pairs and performs reduction function
- Parallelizable across nodes
  - Distribute different sets of data to separate nodes

MapReduce Example

- Facebook friends
  - Friends are stored as Person: [List of friends]
  - A -> [B C D], B -> [A C D E], C -> [A B D E]
  - Pass each to map function
    - Output (key, value) pairs in alphabetical order
    - [A B] -> [B C D], [A C] -> [B C D], [A D] -> [B C D]
    - [A B] -> [A C D E], [B C] -> [A C D E], [B D] -> [A C D E], etc
  - Group values sharing keys
    - [A B] -> [B C D], [A C D E]
  - Pass to reduce function
    - Take intersection of each keys values
    - [A B] -> [C D] – C & D are mutual friends of A & B

Slide Credit

- Dr. Nilanjan Banerjee, CSEE UMBC