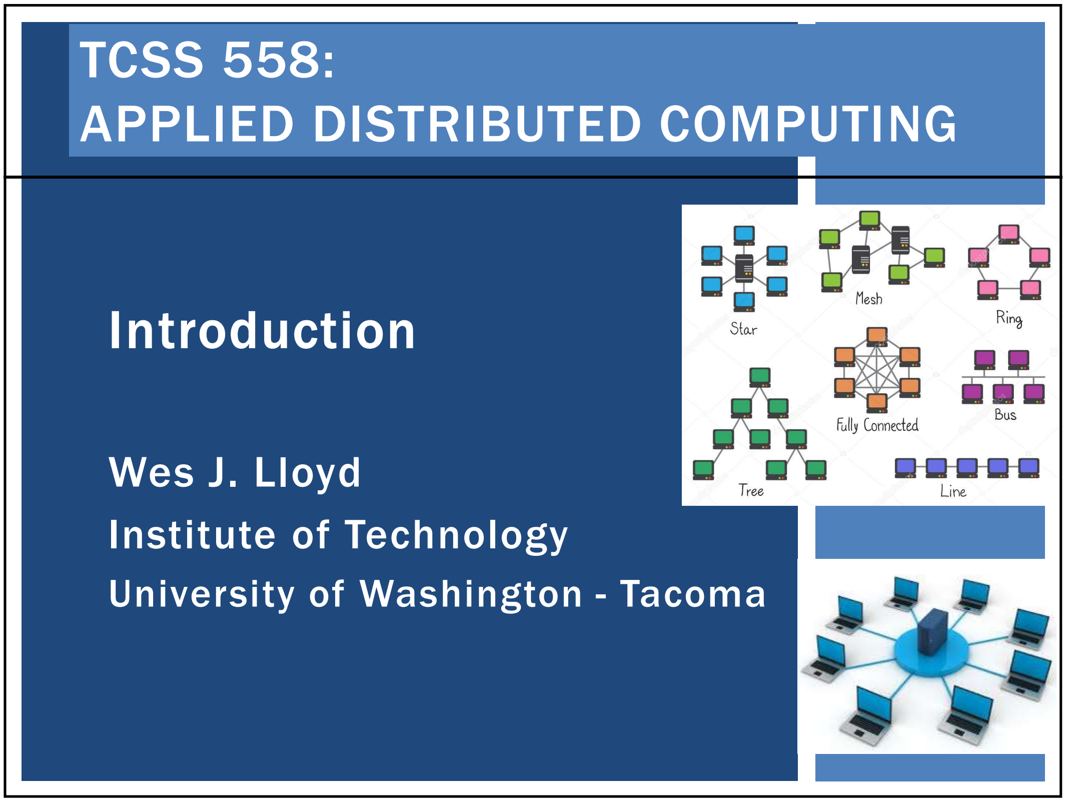
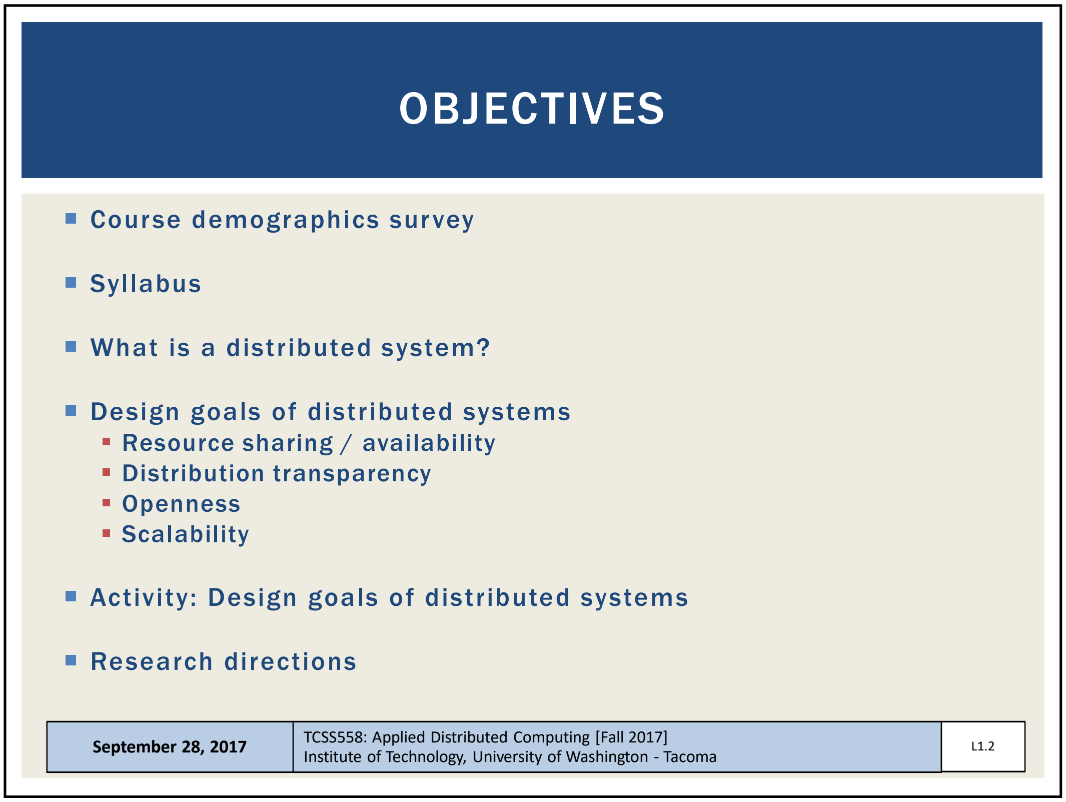


TCSS 558: APPLIED DISTRIBUTED COMPUTING

Introduction

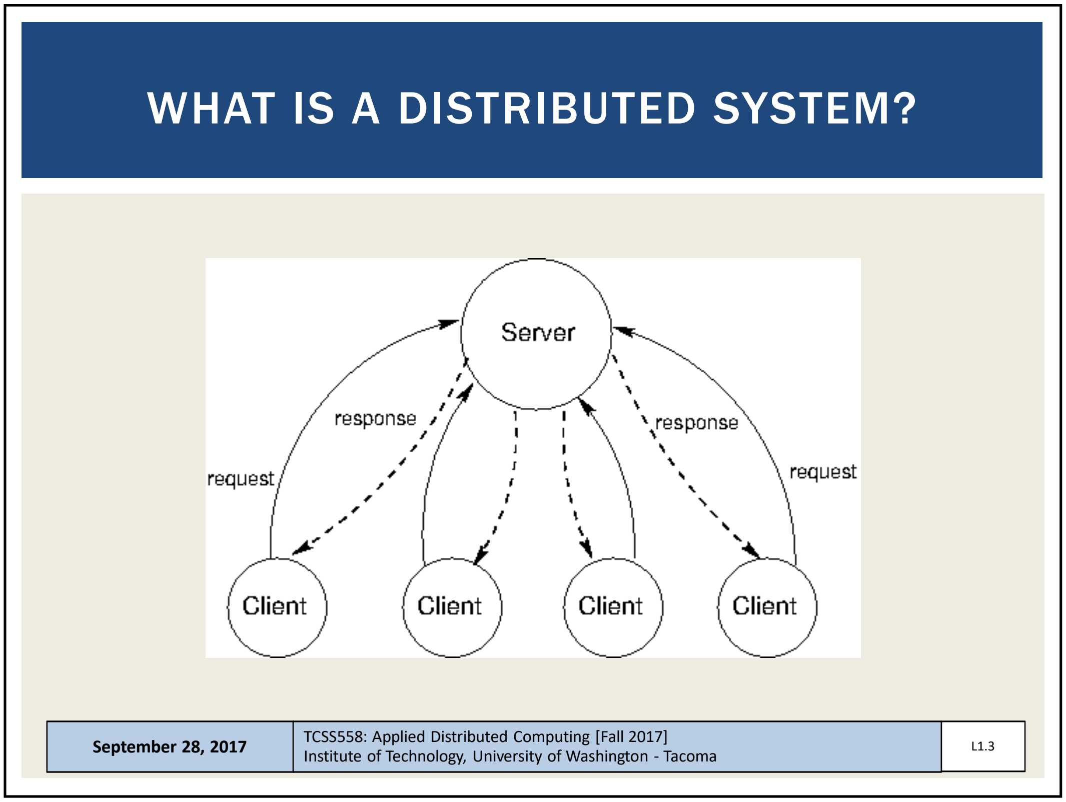
Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma

Star
Mesh
Ring
Fully Connected
Bus
Tree
Line

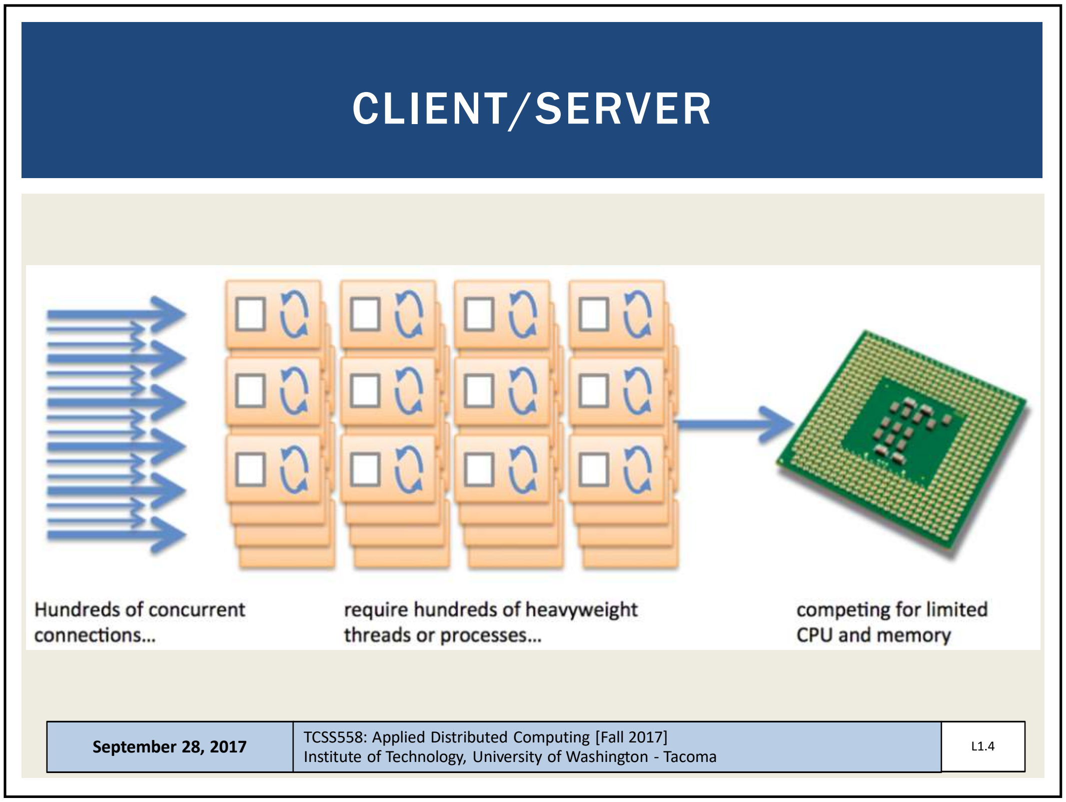


OBJECTIVES

- Course demographics survey
- Syllabus
- What is a distributed system?
- Design goals of distributed systems
  - Resource sharing / availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems
- Research directions



WHAT IS A DISTRIBUTED SYSTEM?



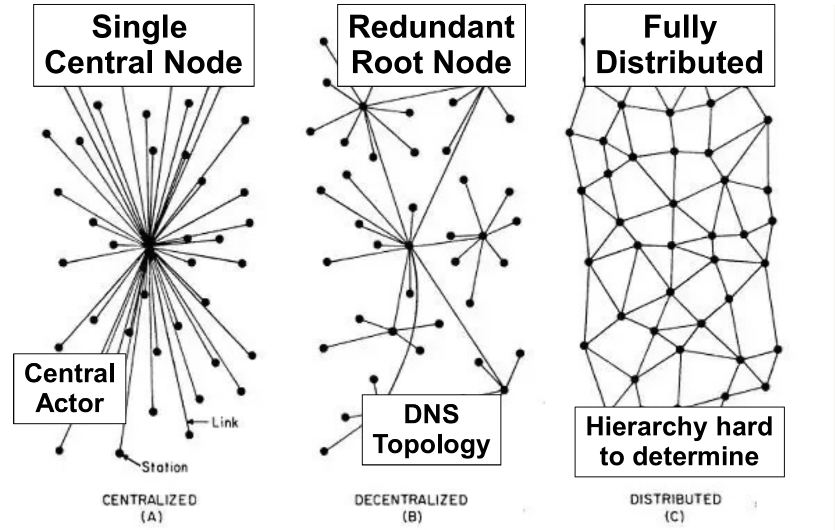
CLIENT/SERVER

Hundreds of concurrent connections…

require hundreds of heavyweight threads or processes…

competing for limited CPU and memory

SYSTEM ARCHITECTURES



Single Central Node

Redundant Root Node

Fully Distributed

Central Actor

DNS Topology

Hierarchy hard to determine

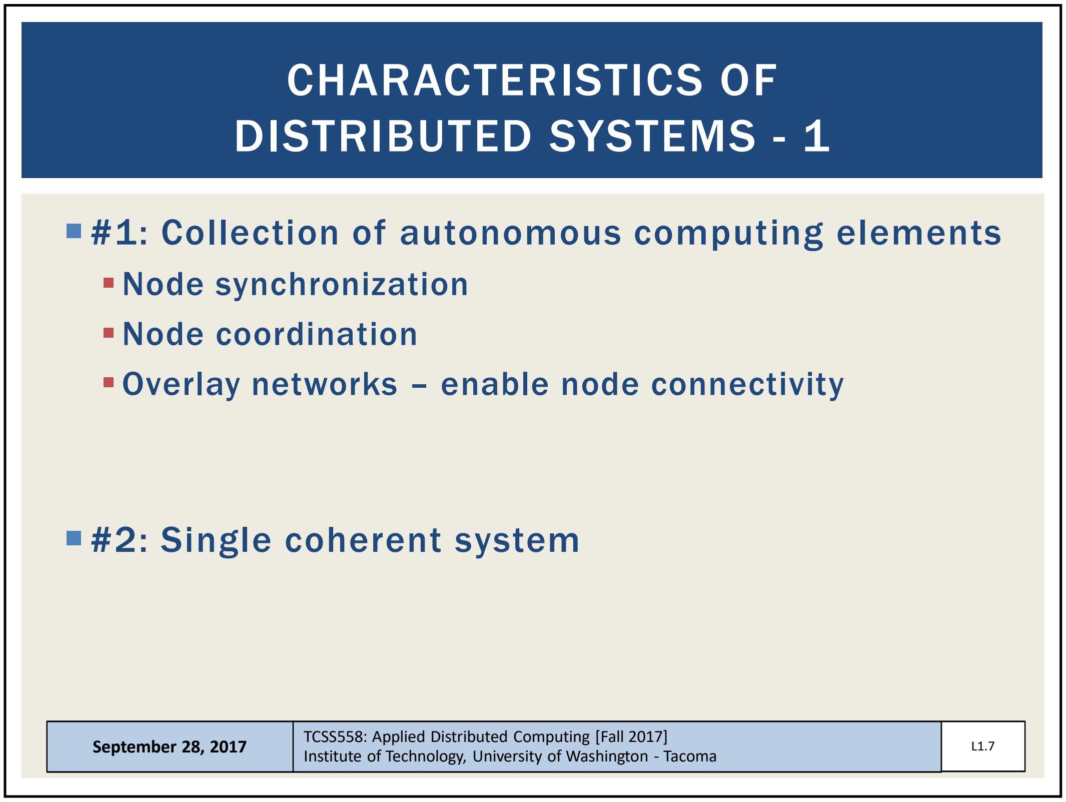
WHAT IS A DISTRIBUTED SYSTEM?

- Definition:
- A ***collection of autonomous computing elements*** that appears to users as a single coherent system.

- How nodes collaborate / communicate is **key**

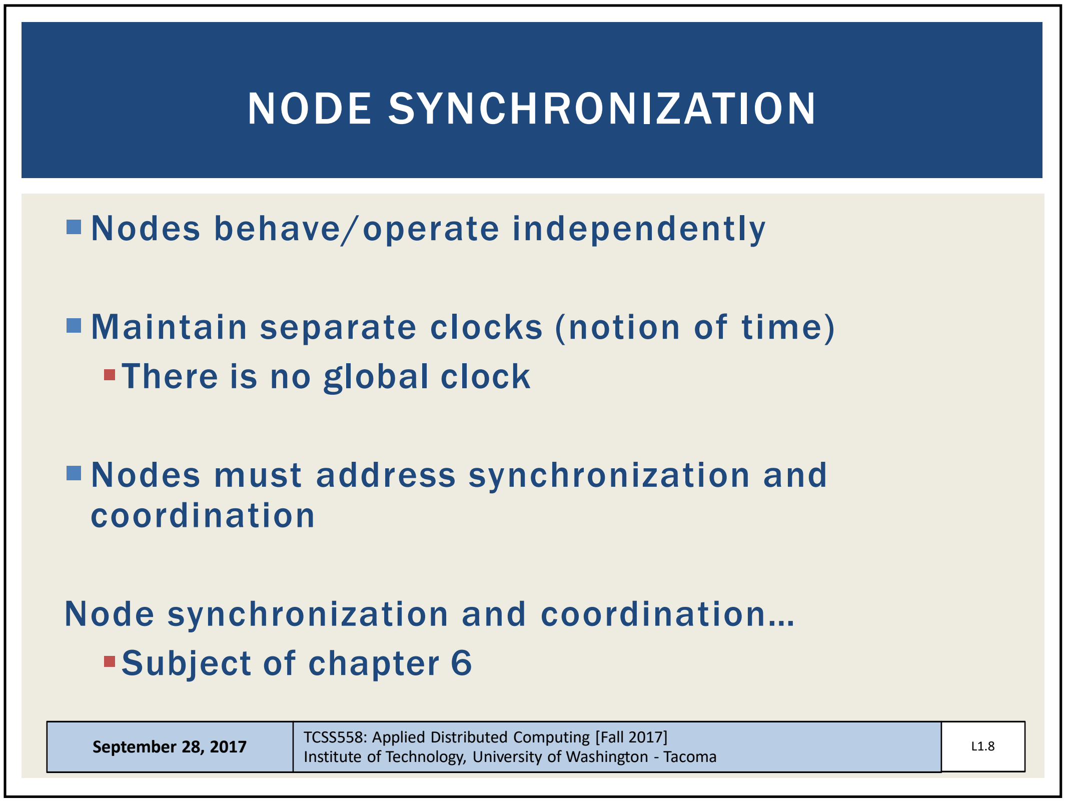
- Nodes
  - Autonomous computing elements
  - Implemented as hardware or software processes

- Single coherent system
  - Users and applications perceive a single system
  - Nodes collaborate, and provide “abstraction”



CHARACTERISTICS OF DISTRIBUTED SYSTEMS - 1

- #1: Collection of autonomous computing elements
  - Node synchronization
  - Node coordination
  - Overlay networks – enable node connectivity
- #2: Single coherent system

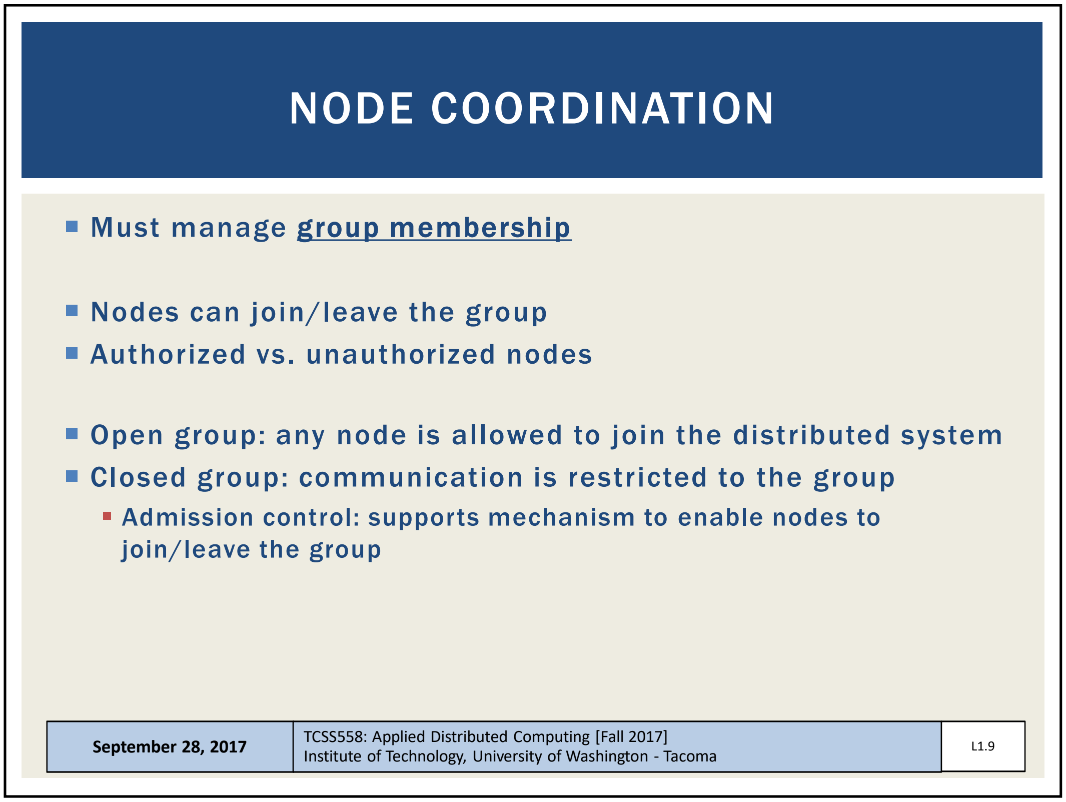


NODE SYNCHRONIZATION

- Nodes behave/operate independently
- Maintain separate clocks (notion of time)
  - There is no global clock
- Nodes must address synchronization and coordination

Node synchronization and coordination…

- Subject of chapter 6

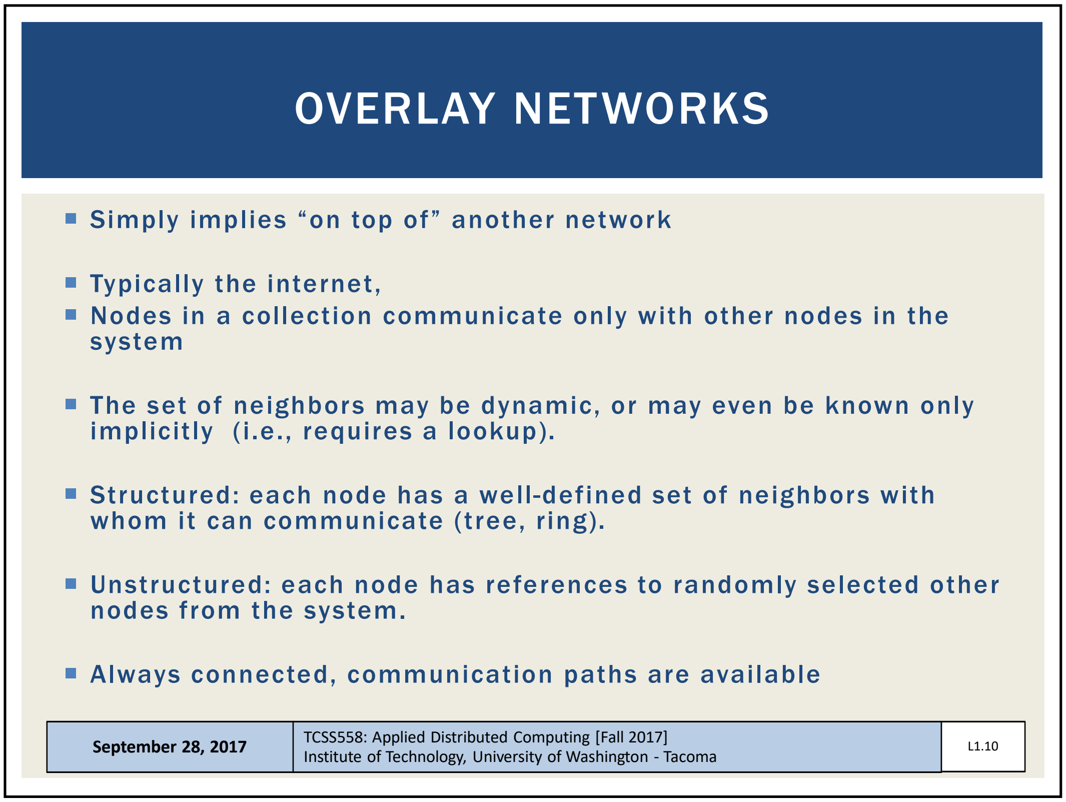


NODE COORDINATION

- Must manage group membership

- Nodes can join/leave the group
- Authorized vs. unauthorized nodes

- Open group: any node is allowed to join the distributed system
- Closed group: communication is restricted to the group
  - Admission control: supports mechanism to enable nodes to join/leave the group



OVERLAY NETWORKS

- Simply implies "on top of" another network

- Typically the internet,
- Nodes in a collection communicate only with other nodes in the system

- The set of neighbors may be dynamic, or may even be known only implicitly (i.e., requires a lookup).

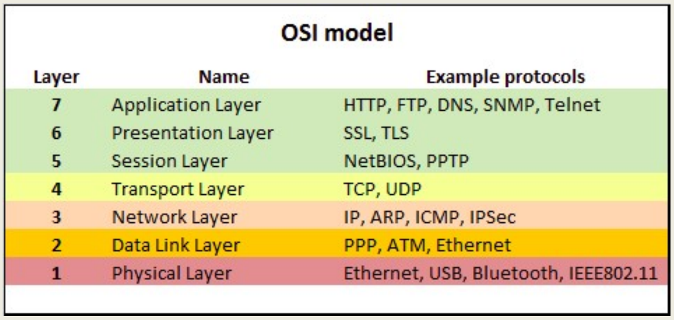
- Structured: each node has a well-defined set of neighbors with whom it can communicate (tree, ring).

- Unstructured: each node has references to randomly selected other nodes from the system.

- Always connected, communication paths are available

OSI MODEL

- Open systems interconnect:
- Standardization of the functionalities in a communication system via abstract layers



OSI model

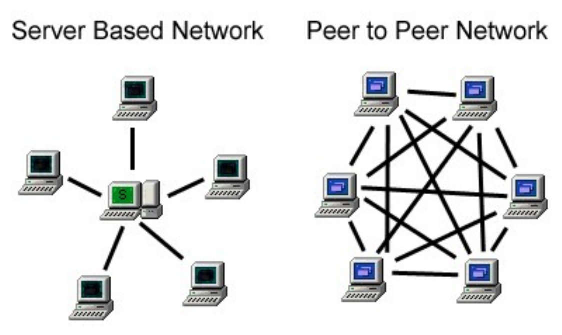
Layer	Name	Example protocols
7	Application Layer	HTTP, FTP, DNS, SNMP, Telnet
6	Presentation Layer	SSL, TLS
5	Session Layer	NetBIOS, PPTP
4	Transport Layer	TCP, UDP
3	Network Layer	IP, ARP, ICMP, IPSec
2	Data Link Layer	PPP, ATM, Ethernet
1	Physical Layer	Ethernet, USB, Bluetooth, IEEE802.11

PEER-TO-PEER NETWORK

- Distributed systems leverage the ability of computing systems to collaborate and aggregate resources across many nodes providing potential for great scale and robustness than centralized client-server models

- How can fault tolerance be replicated in the client/server model?

- How can fault tolerance be provided by the peer-to-peer model?

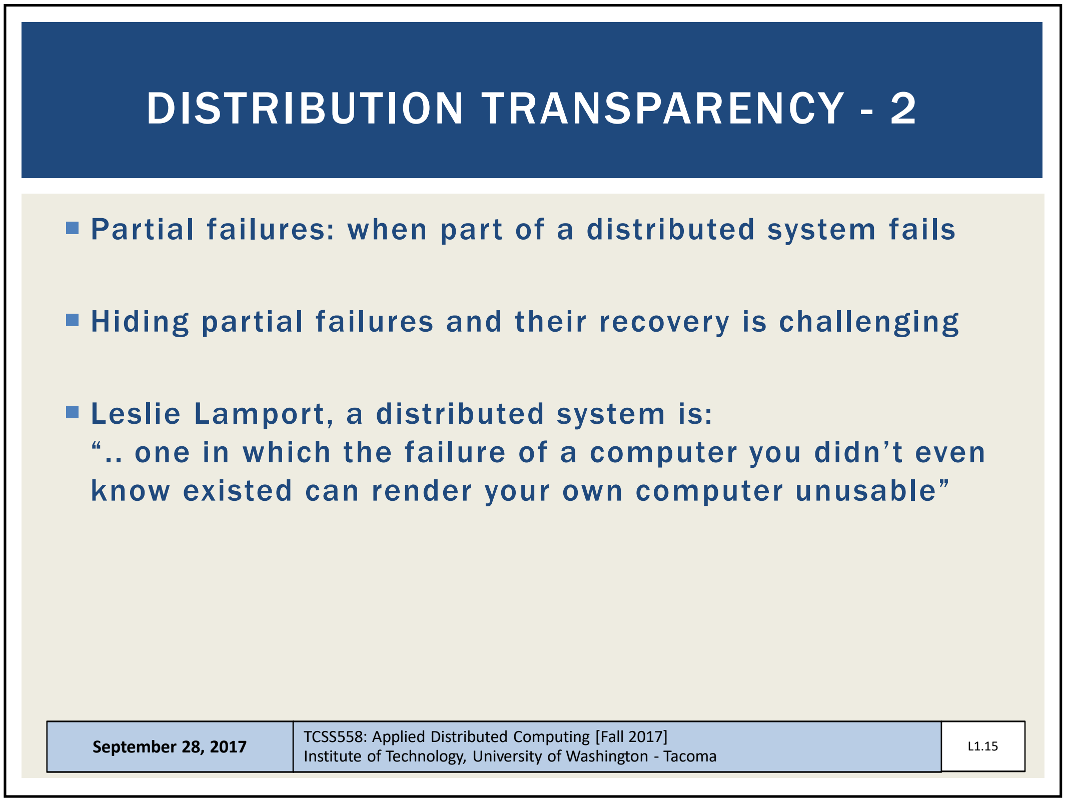


COHERENT SYSTEM

- Collection of nodes operates the same, regardless of where, when, and how interaction between a user and the system takes places
- Distribution transparency
- From the user's perspective, they can't discern how the system is implemented
- What are some examples of transparent distributed systems that you frequently use?

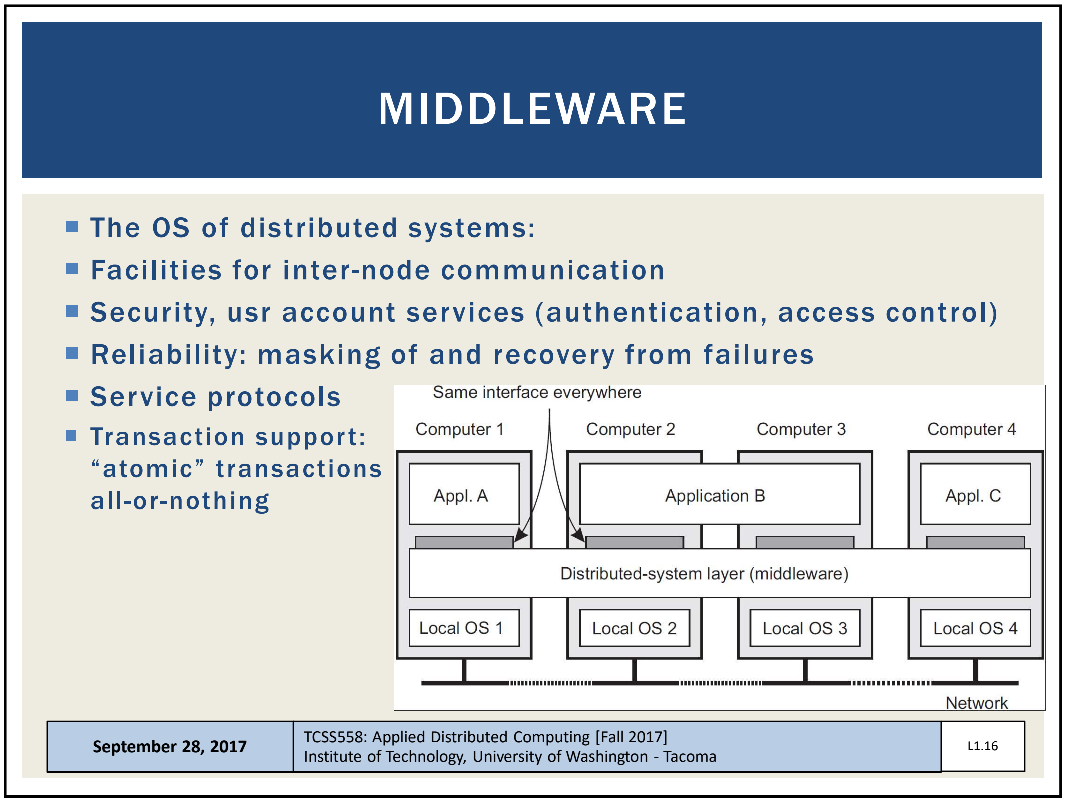
DISTRIBUTION TRANSPARENCY

- An end user cannot tell where a computation takes place
- Where data is stored is abstracted (hidden)
- State of data replication is abstracted (hidden)
- "The Cloud" provides one example of distributed transparency



DISTRIBUTION TRANSPARENCY - 2

- Partial failures: when part of a distributed system fails
- Hiding partial failures and their recovery is challenging
- Leslie Lamport, a distributed system is:
  ".. one in which the failure of a computer you didn't even know existed can render your own computer unusable"



MIDDLEWARE

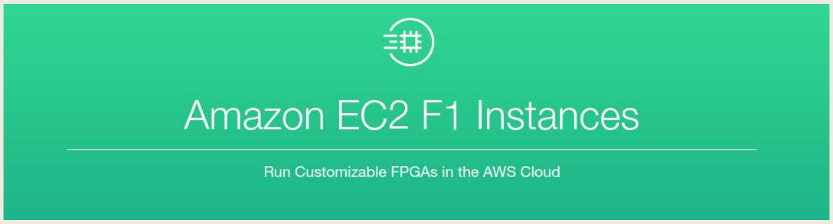
- The OS of distributed systems:
- Facilities for inter-node communication
- Security, usr account services (authentication, access control)
- Reliability: masking of and recovery from failures
- Service protocols
- Transaction support:
  "atomic" transactions
  all-or-nothing

DESIGN GOALS OF DISTRIBUTED SYSTEMS

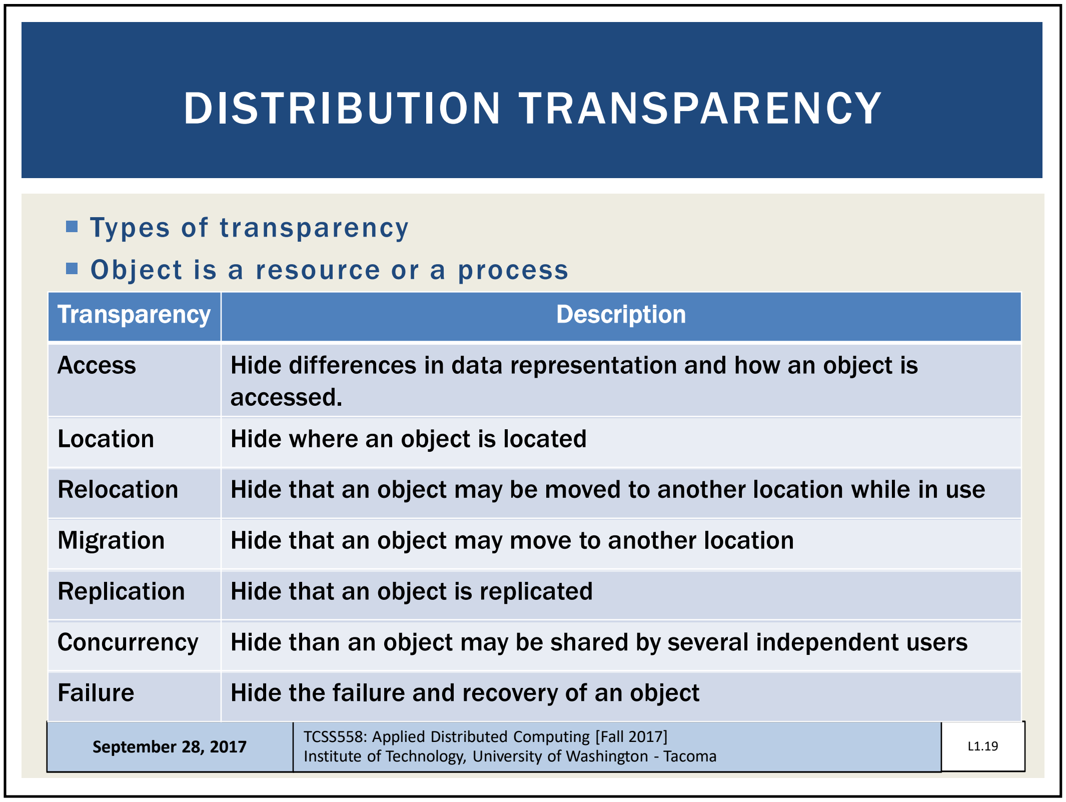
- Support for sharing resources (accessibility)
- Distribution transparency
- Openness (avoiding vendor lock-in)
- Scalability

ACCESSIBILITY: RESOURCE SHARING

- Easy for users (and applications) to share remote resources
  - Storage, compute, networks, services, peripherals, …
  - Field programmable arrays (FPGAs) "as a service":



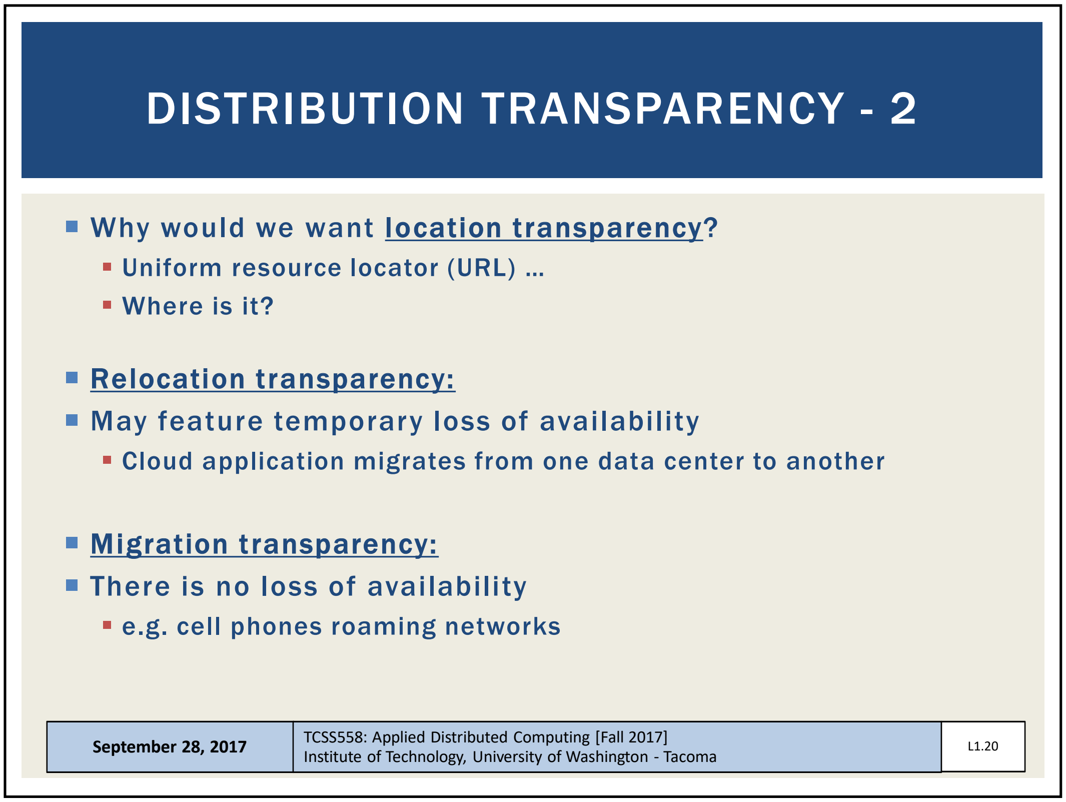
  - https://aws.amazon.com/ec2/instance-types/f1/
  - Nearly any resource can be shared



DISTRIBUTION TRANSPARENCY

- Types of transparency
- Object is a resource or a process

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

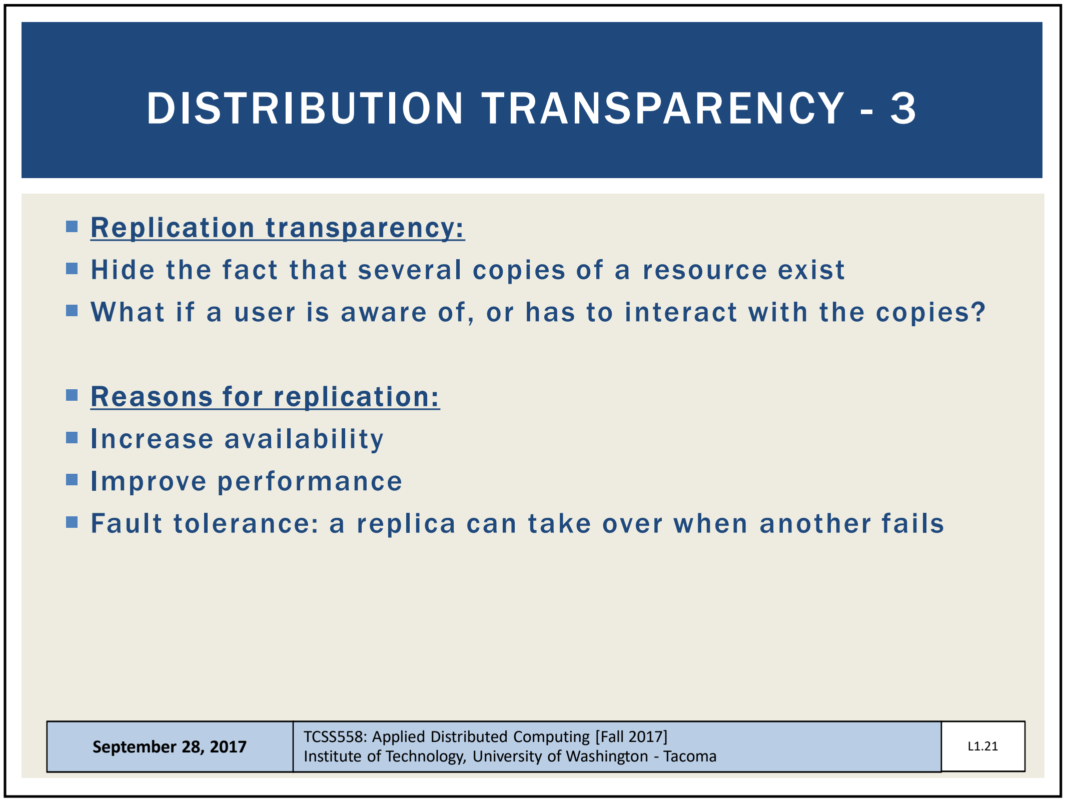


DISTRIBUTION TRANSPARENCY - 2

- Why would we want location transparency?
  - Uniform resource locator (URL) …
  - Where is it?

- Relocation transparency:
- May feature temporary loss of availability
  - Cloud application migrates from one data center to another

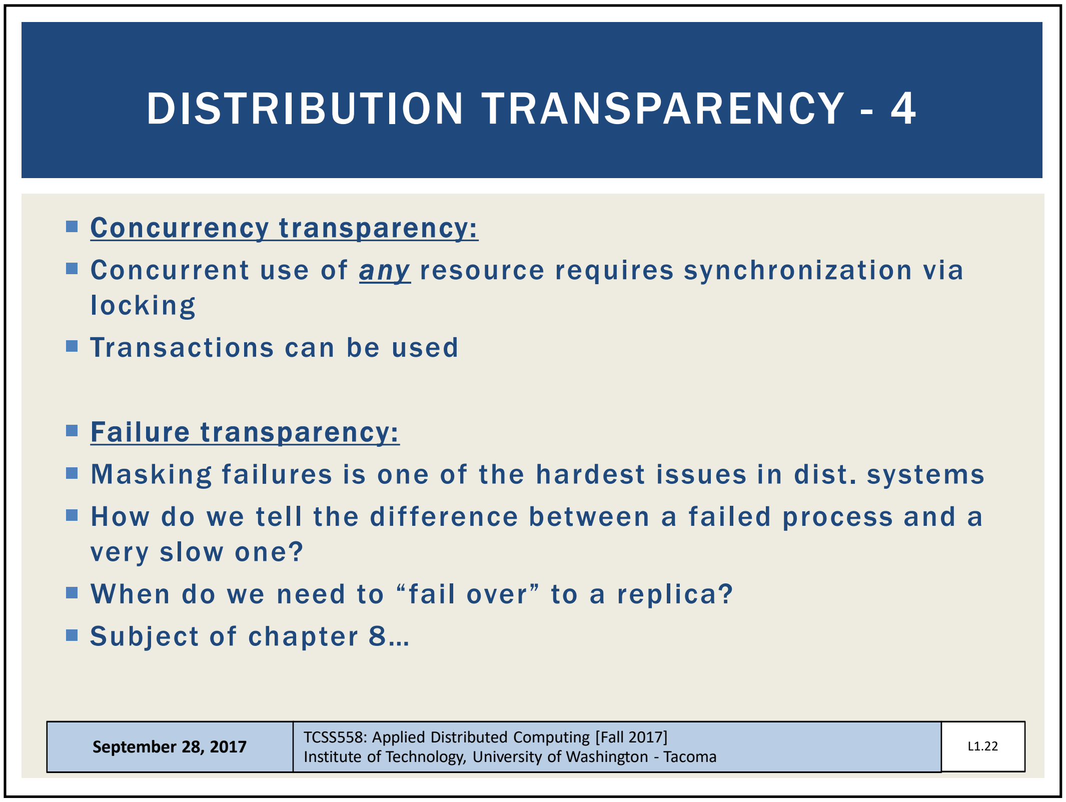
- Migration transparency:
- There is no loss of availability
  - e.g. cell phones roaming networks



DISTRIBUTION TRANSPARENCY - 3

- **Replication transparency:**
- Hide the fact that several copies of a resource exist
- What if a user is aware of, or has to interact with the copies?

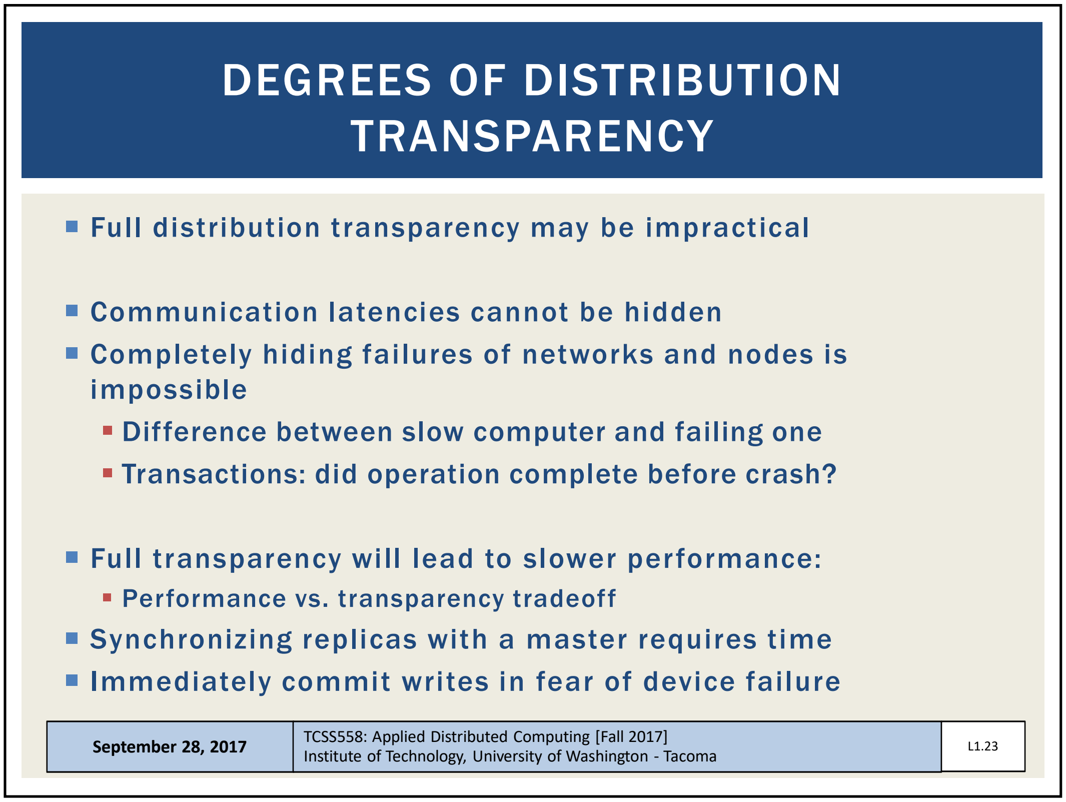
- **Reasons for replication:**
- Increase availability
- Improve performance
- Fault tolerance: a replica can take over when another fails



DISTRIBUTION TRANSPARENCY - 4

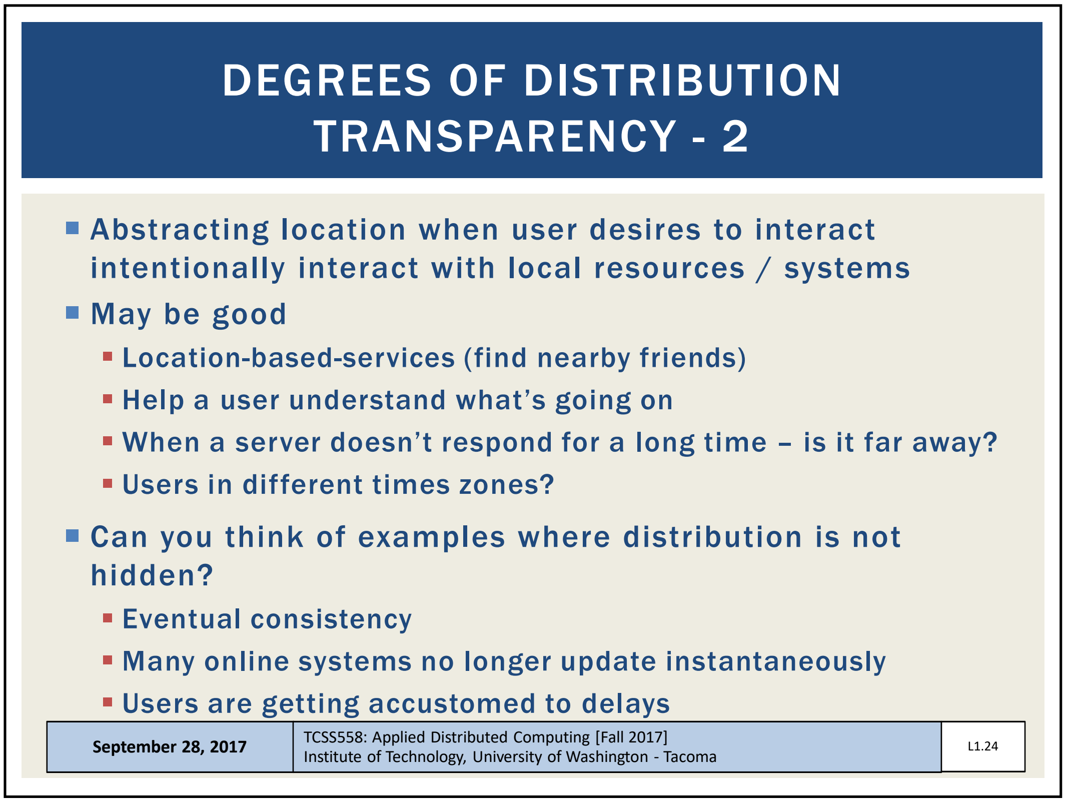
- **Concurrency transparency:**
- Concurrent use of ***any*** resource requires synchronization via locking
- Transactions can be used

- **Failure transparency:**
- Masking failures is one of the hardest issues in dist. systems
- How do we tell the difference between a failed process and a very slow one?
- When do we need to “fail over” to a replica?
- Subject of chapter 8…



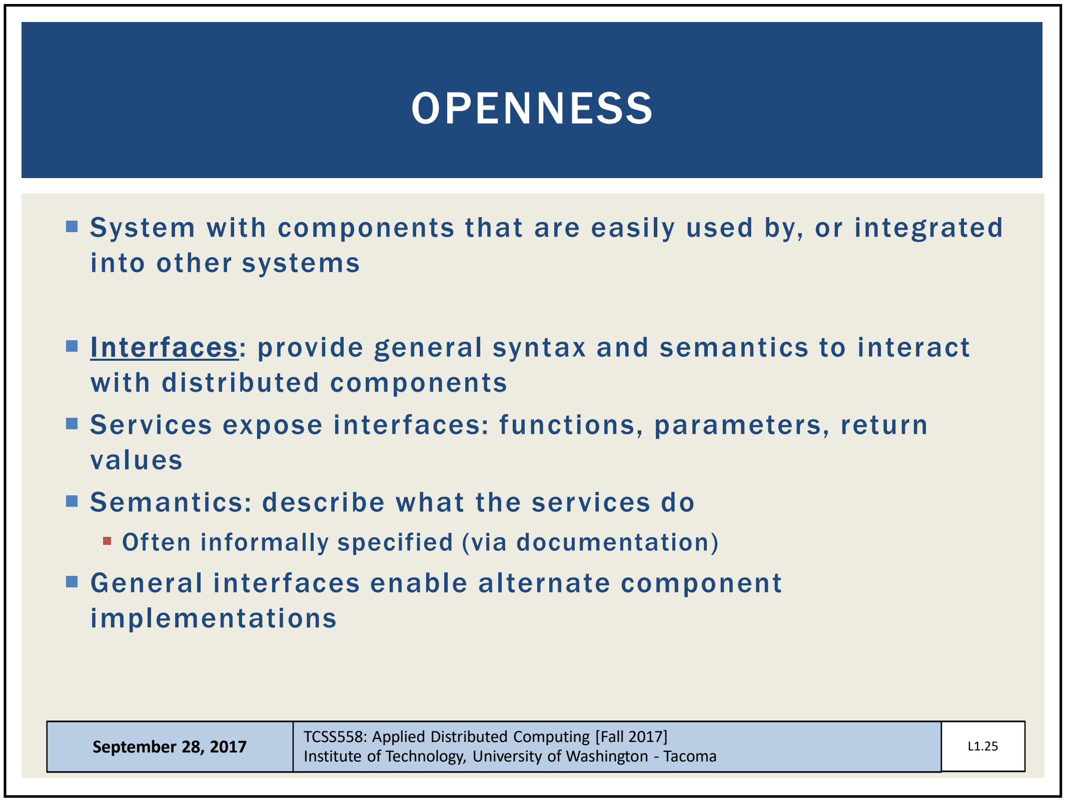
DEGREES OF DISTRIBUTION TRANSPARENCY

- Full distribution transparency may be impractical
- Communication latencies cannot be hidden
- Completely hiding failures of networks and nodes is impossible
  - Difference between slow computer and failing one
  - Transactions: did operation complete before crash?
- Full transparency will lead to slower performance:
  - Performance vs. transparency tradeoff
- Synchronizing replicas with a master requires time
- Immediately commit writes in fear of device failure



DEGREES OF DISTRIBUTION TRANSPARENCY - 2

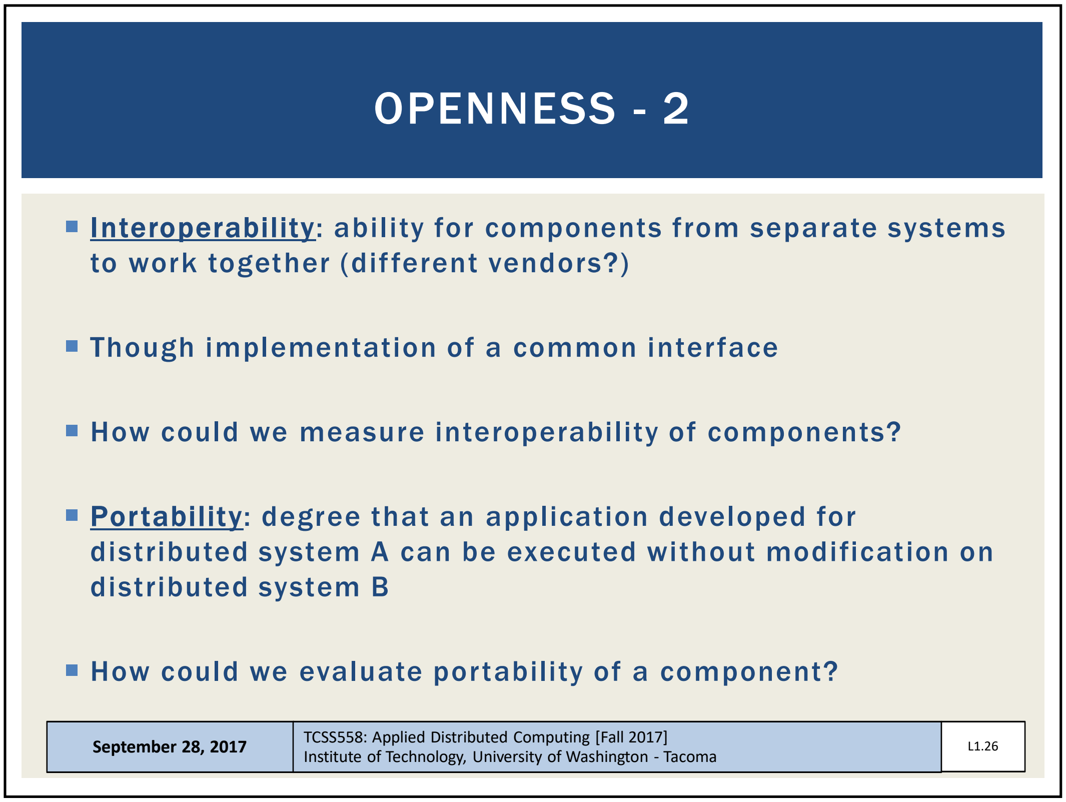
- Abstracting location when user desires to interact intentionally interact with local resources / systems
- May be good
  - Location-based-services (find nearby friends)
  - Help a user understand what's going on
  - When a server doesn't respond for a long time – is it far away?
  - Users in different times zones?
- Can you think of examples where distribution is not hidden?
  - Eventual consistency
  - Many online systems no longer update instantaneously
  - Users are getting accustomed to delays



OPENNESS

- System with components that are easily used by, or integrated into other systems
- **Interfaces**: provide general syntax and semantics to interact with distributed components
- Services expose interfaces: functions, parameters, return values
- Semantics: describe what the services do
  - Often informally specified (via documentation)
- General interfaces enable alternate component implementations

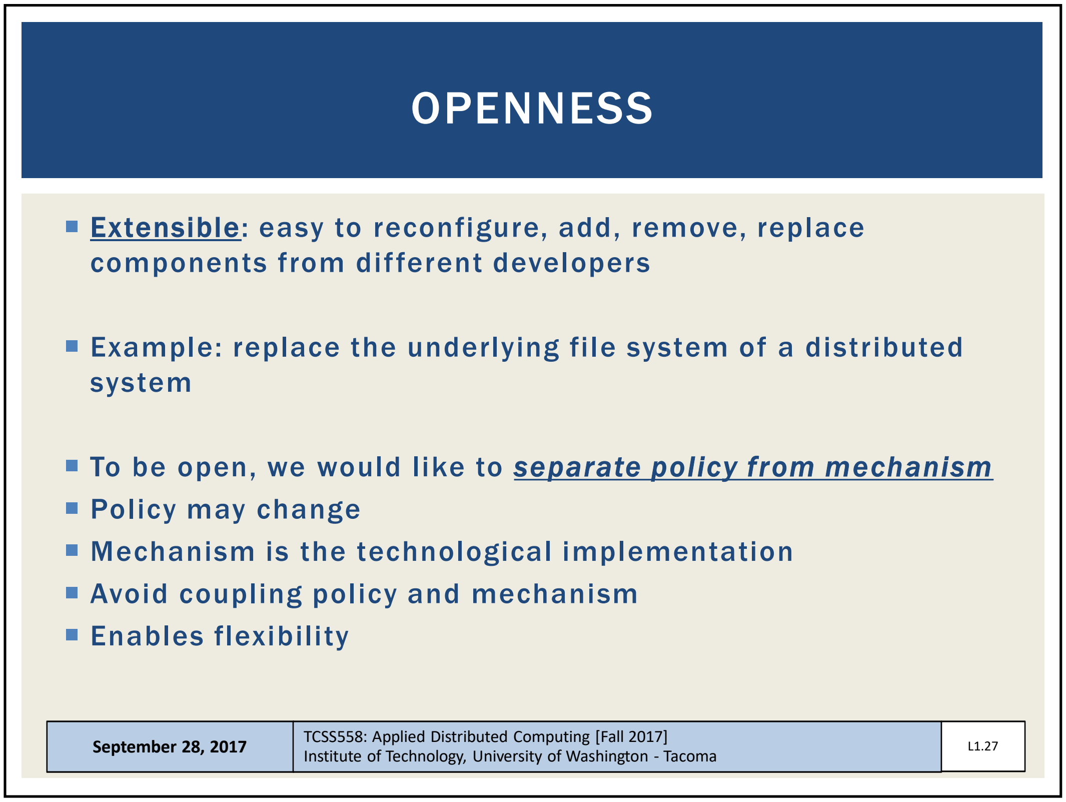
September 28, 2017 | TCSS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma | L1.25



OPENNESS - 2

- **Interoperability**: ability for components from separate systems to work together (different vendors?)
- Though implementation of a common interface
- How could we measure interoperability of components?
- **Portability**: degree that an application developed for distributed system A can be executed without modification on distributed system B
- How could we evaluate portability of a component?

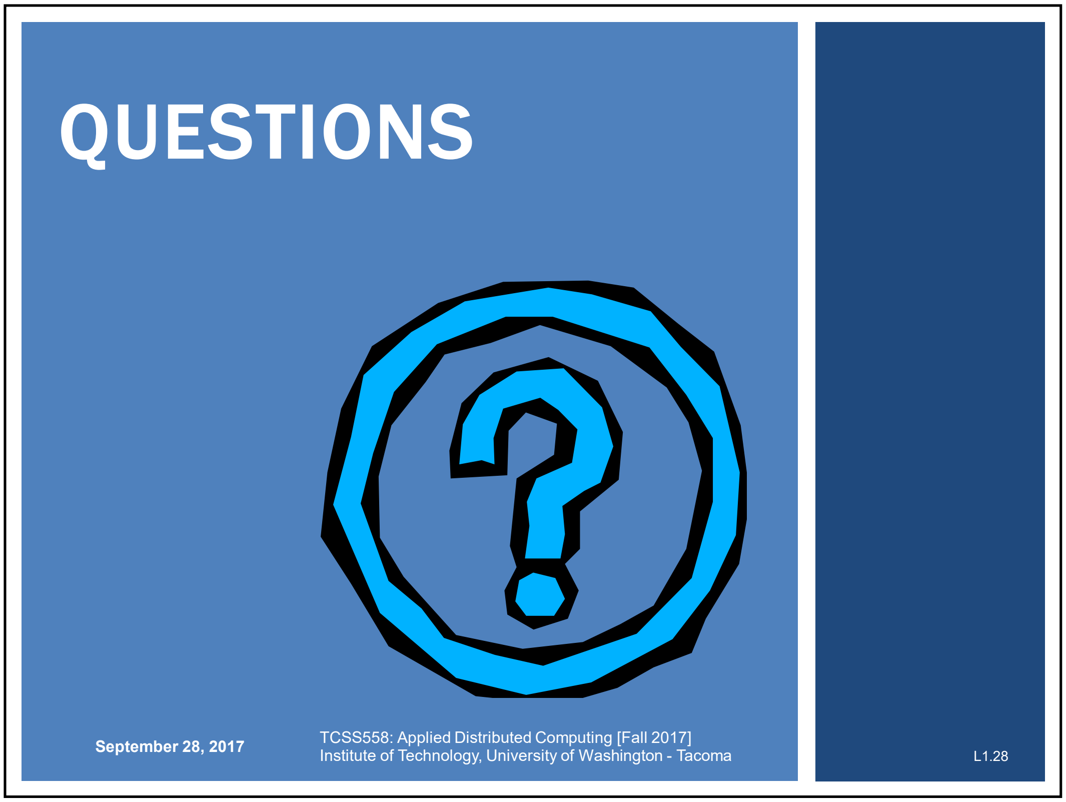
September 28, 2017 | TCSS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma | L1.26



OPENNESS

- **Extensible**: easy to reconfigure, add, remove, replace components from different developers
- Example: replace the underlying file system of a distributed system
- To be open, we would like to ***separate policy from mechanism***
- Policy may change
- Mechanism is the technological implementation
- Avoid coupling policy and mechanism
- Enables flexibility

September 28, 2017 TCSS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma L1.27



QUESTIONS

September 28, 2017 TCSS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma L1.28