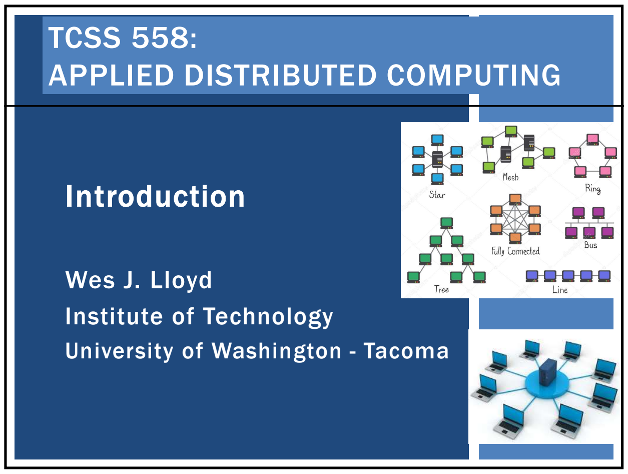
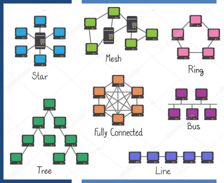


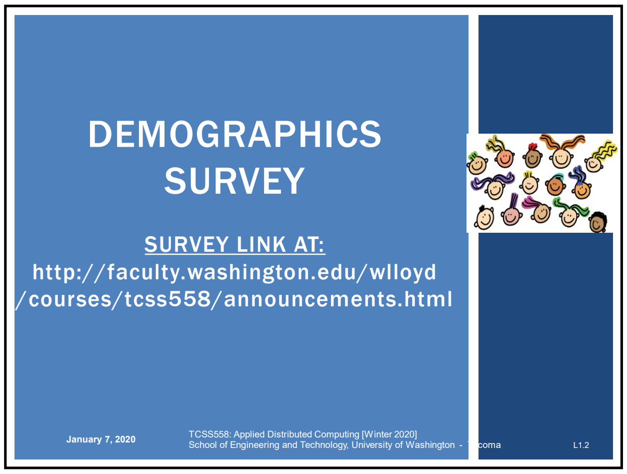
TCSS 558: APPLIED DISTRIBUTED COMPUTING

Introduction

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



1

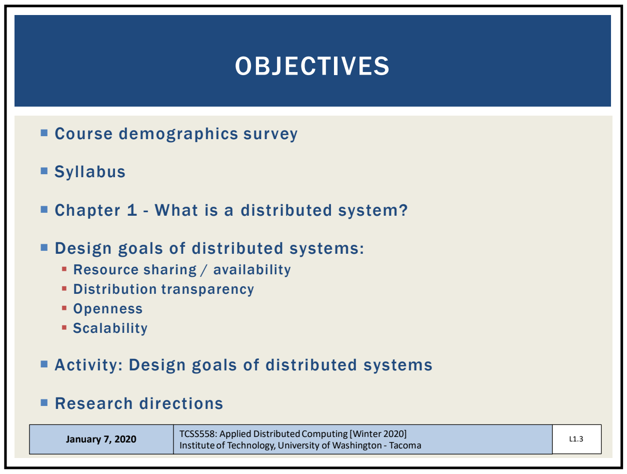


DEMOGRAPHICS SURVEY

SURVEY LINK AT:
http://faculty.washington.edu/wlloyd/courses/tcss558/announcements.html

January 7, 2020 TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma L1.2

2

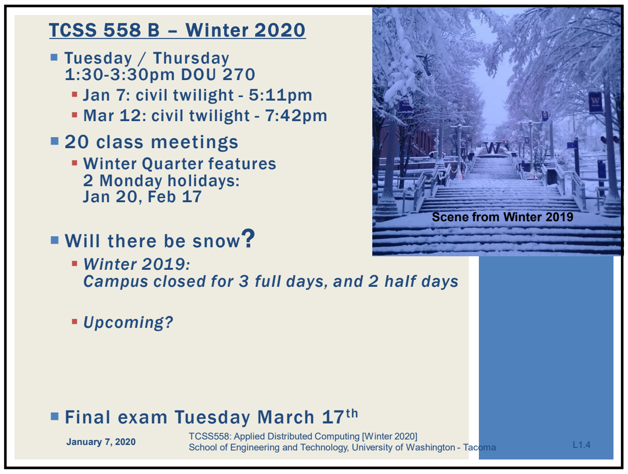


OBJECTIVES

- Course demographics survey
- Syllabus
- Chapter 1 - 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

January 7, 2020 TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma L1.3

3



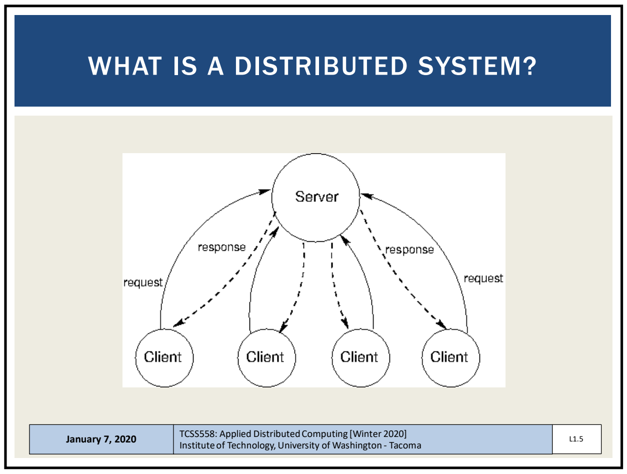
TCSS 558 B – Winter 2020

- Tuesday / Thursday 1:30-3:30pm DOU 270
  - Jan 7: civil twilight - 5:11pm
  - Mar 12: civil twilight - 7:42pm
- 20 class meetings
  - Winter Quarter features 2 Monday holidays: Jan 20, Feb 17
- Will there be snow?
  - *Winter 2019: Campus closed for 3 full days, and 2 half days*
  - *Upcoming?*
- Final exam Tuesday March 17th

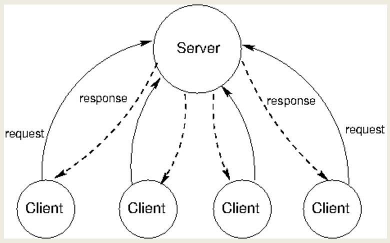
Scene from Winter 2019

January 7, 2020 TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma L1.4

4

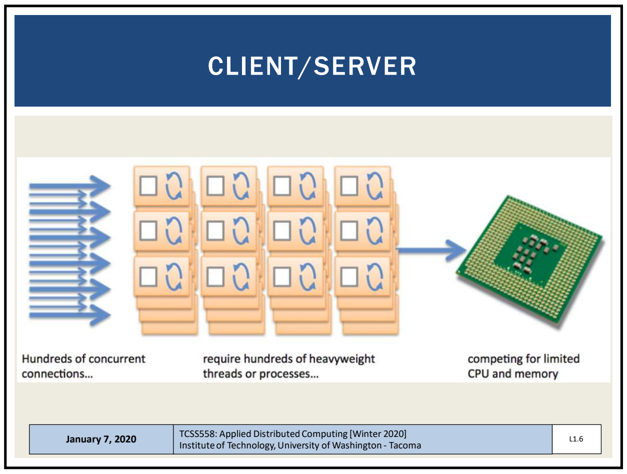


WHAT IS A DISTRIBUTED SYSTEM?

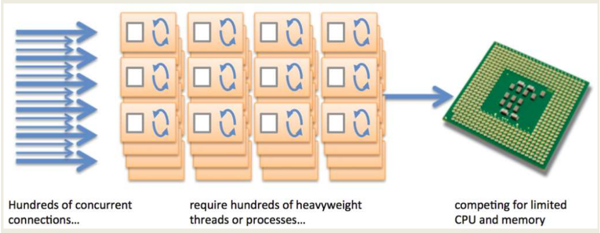


January 7, 2020 TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma L1.5

5

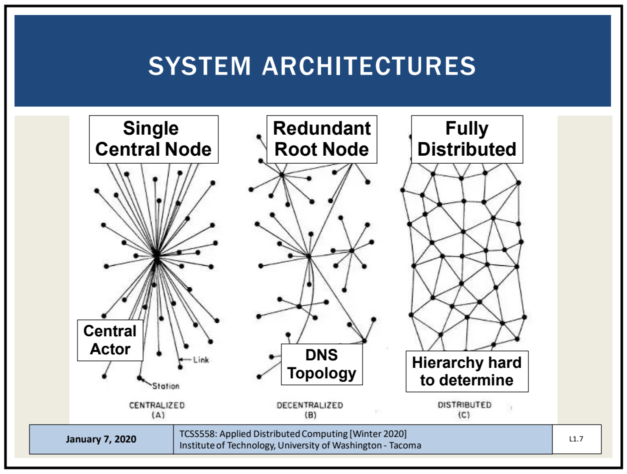


CLIENT/SERVER

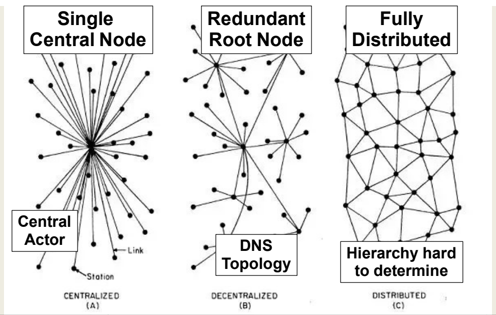

Hundreds of concurrent connections…
require hundreds of heavyweight threads or processes…
competing for limited CPU and memory

January 7, 2020 TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma L1.6

6

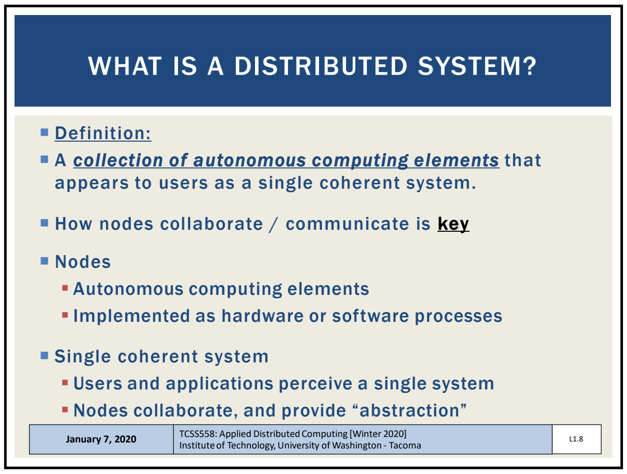


SYSTEM ARCHITECTURES



January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.7

7

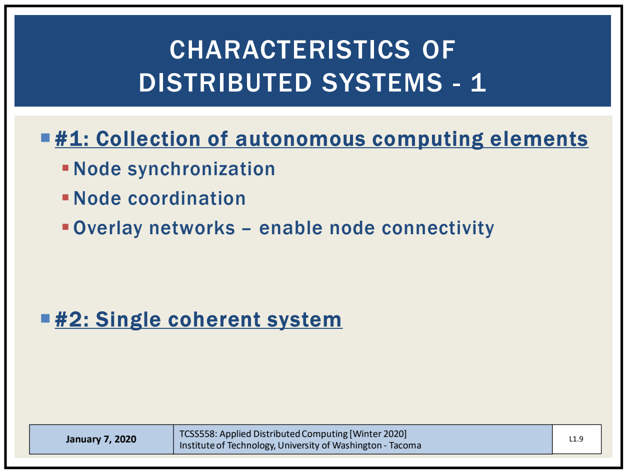


WHAT IS A DISTRIBUTED SYSTEM?

- 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"

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.8

8

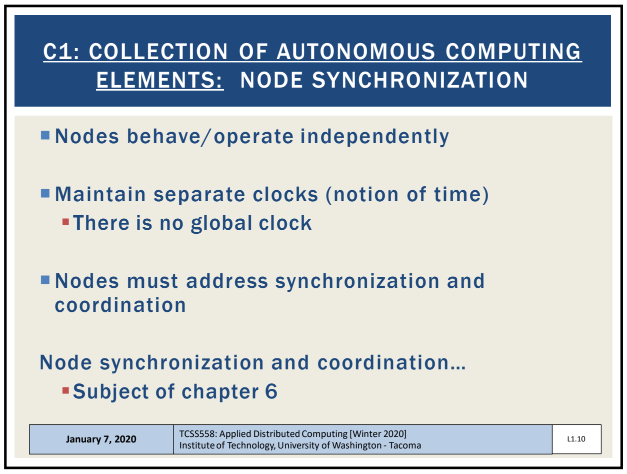


CHARACTERISTICS OF DISTRIBUTED SYSTEMS - 1

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

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.9

9



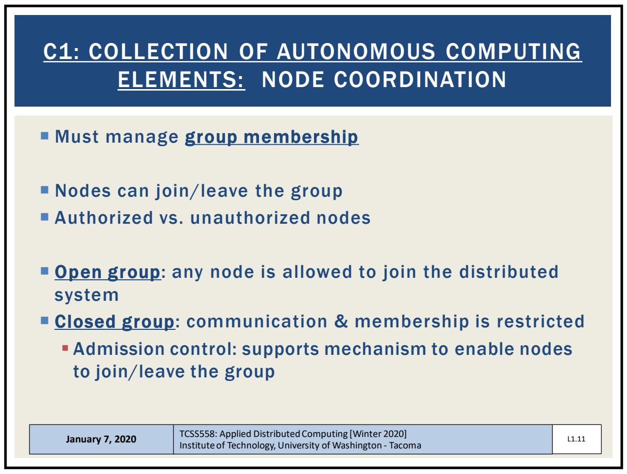
C1: COLLECTION OF AUTONOMOUS COMPUTING ELEMENTS: NODE SYNCHRONIZATION

- 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

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.10

10

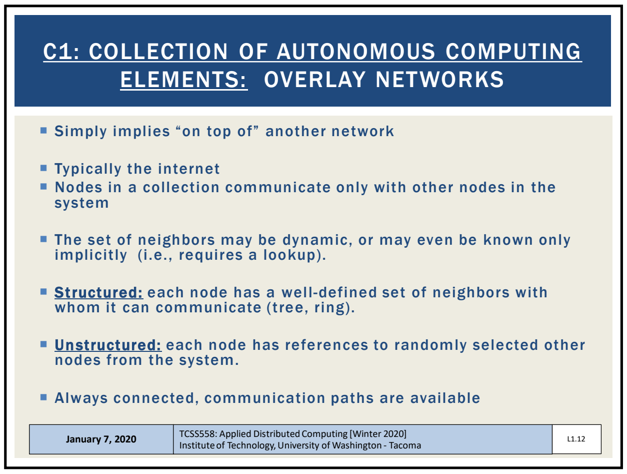


C1: COLLECTION OF AUTONOMOUS COMPUTING ELEMENTS: NODE COORDINATION

- 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 & membership is restricted
  - Admission control: supports mechanism to enable nodes to join/leave the group

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.11

11

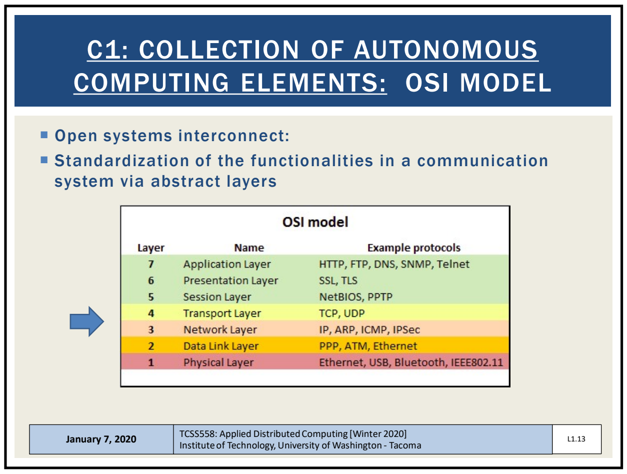


C1: COLLECTION OF AUTONOMOUS COMPUTING ELEMENTS: OVERLAY NETWORKS

- 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

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.12

12



C1: COLLECTION OF AUTONOMOUS COMPUTING ELEMENTS: OSI MODEL

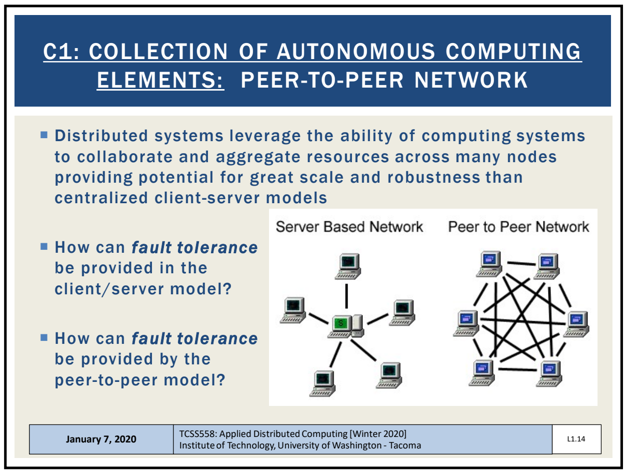
- Open systems interconnect:
- Standardization of the functionalities in a communication system via abstract layers

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

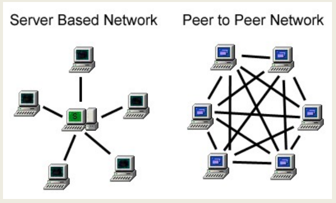
January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.13

13



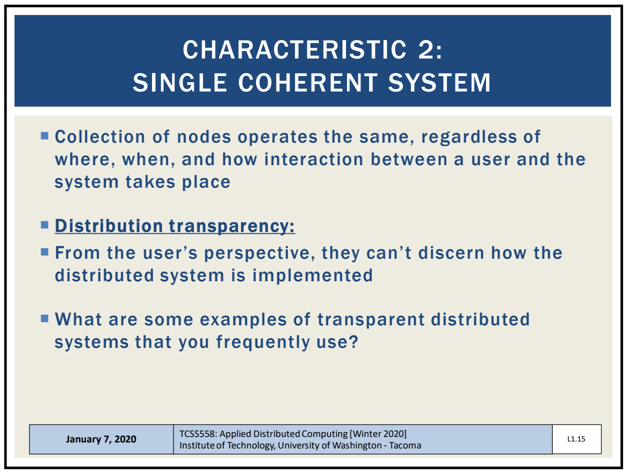
C1: COLLECTION OF AUTONOMOUS COMPUTING ELEMENTS: PEER-TO-PEER NETWORK

- 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 provided in the client/server model?
- How can ***fault tolerance*** be provided by the peer-to-peer model?



January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.14

14

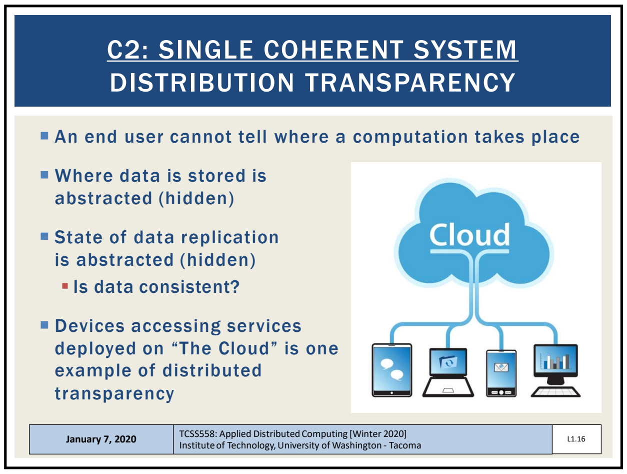


CHARACTERISTIC 2: SINGLE COHERENT SYSTEM

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

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.15

15

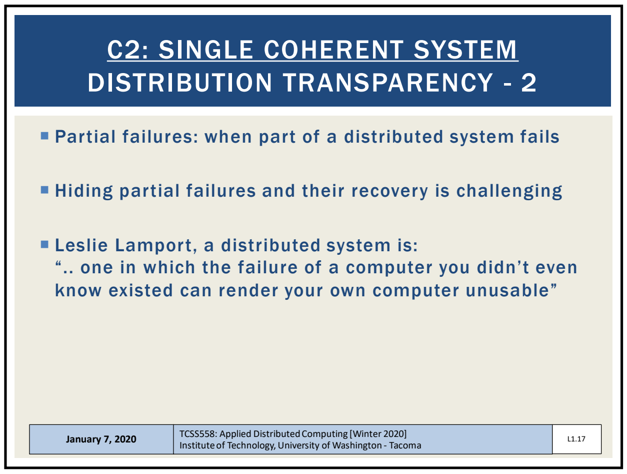


C2: SINGLE COHERENT SYSTEM DISTRIBUTION TRANSPARENCY

- An end user cannot tell where a computation takes place
- Where data is stored is abstracted (hidden)
- State of data replication is abstracted (hidden)
  - Is data consistent?
- Devices accessing services deployed on "The Cloud" is one example of distributed transparency

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.16

16

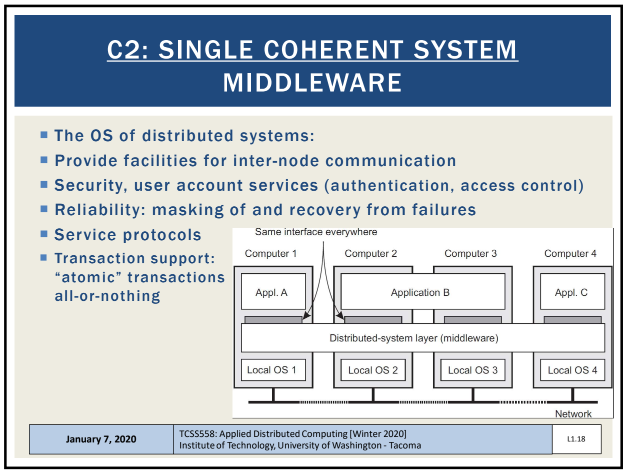


C2: SINGLE COHERENT SYSTEM DISTRIBUTION TRANSPARENCY - 2

- 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"

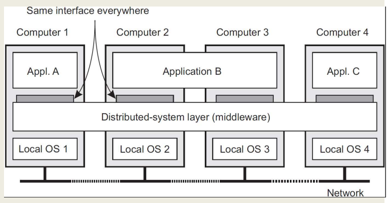
January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.17

17



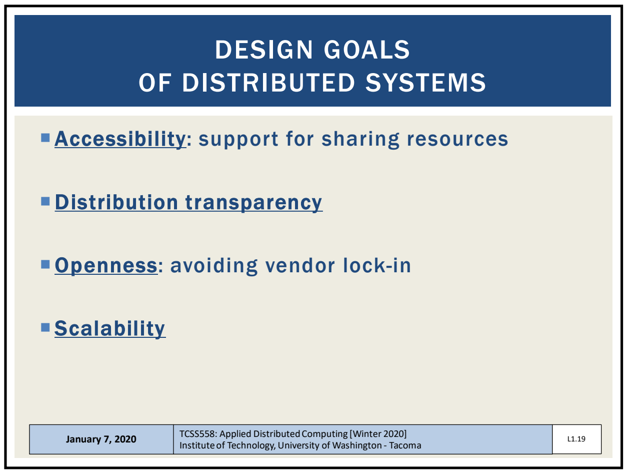
C2: SINGLE COHERENT SYSTEM MIDDLEWARE

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



January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.18

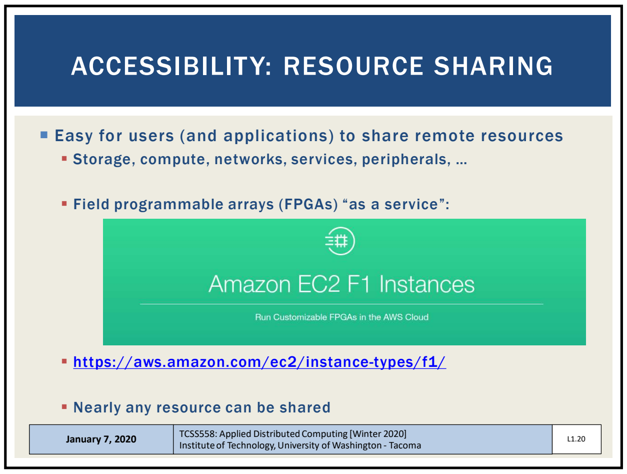
18



DESIGN GOALS OF DISTRIBUTED SYSTEMS

- **Accessibility**: support for sharing resources
- **Distribution transparency**
- **Openness**: avoiding vendor lock-in
- **Scalability**

19



ACCESSIBILITY: RESOURCE SHARING

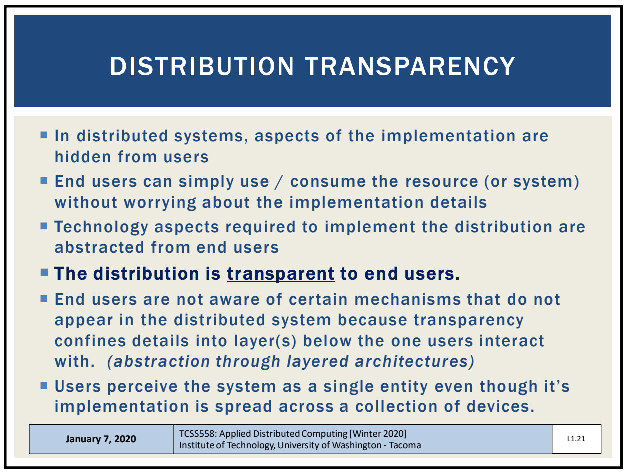
- Easy for users (and applications) to share remote resources
  - Storage, compute, networks, services, peripherals, …
  - Field programmable arrays (FPGAs) "as a service":

Amazon EC2 F1 Instances

Run Customizable FPGAs in the AWS Cloud

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

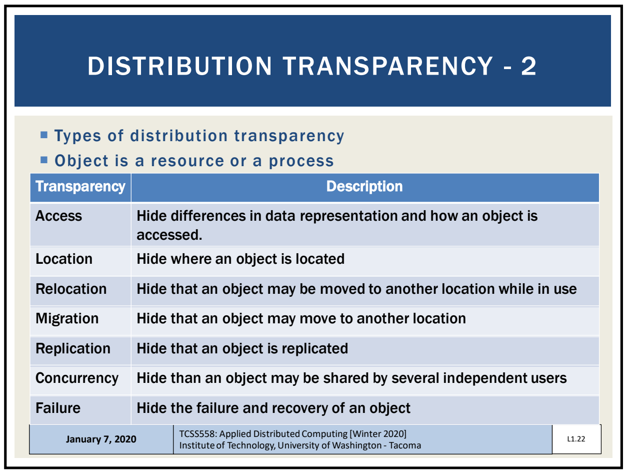
20



DISTRIBUTION TRANSPARENCY

- In distributed systems, aspects of the implementation are hidden from users
- End users can simply use / consume the resource (or system) without worrying about the implementation details
- Technology aspects required to implement the distribution are abstracted from end users
- **The distribution is transparent to end users.**
- End users are not aware of certain mechanisms that do not appear in the distributed system because transparency confines details into layer(s) below the one users interact with. *(abstraction through layered architectures)*
- Users perceive the system as a single entity even though it's implementation is spread across a collection of devices.

21

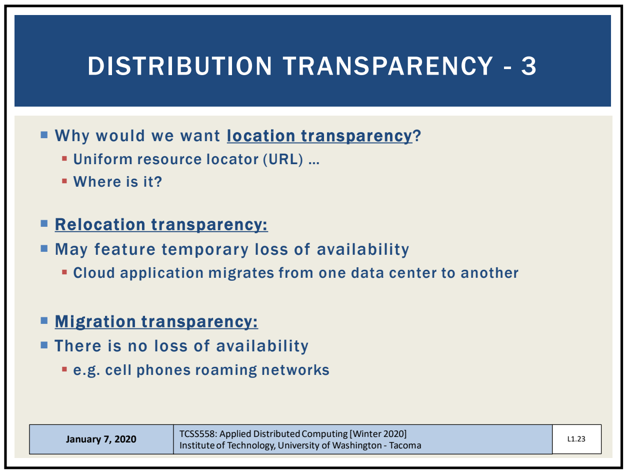


DISTRIBUTION TRANSPARENCY - 2

- Types of distribution 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

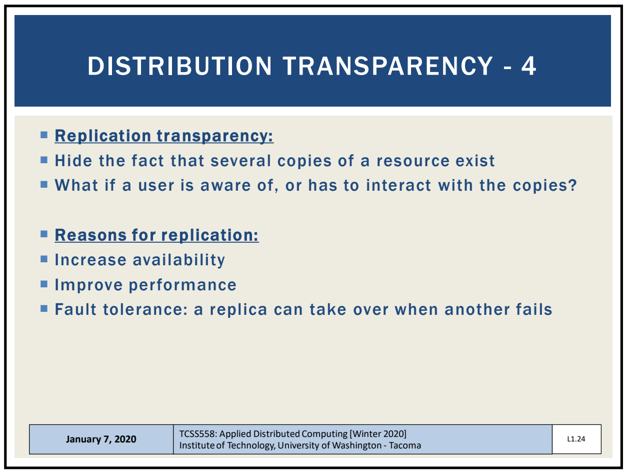
22



DISTRIBUTION TRANSPARENCY - 3

- 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

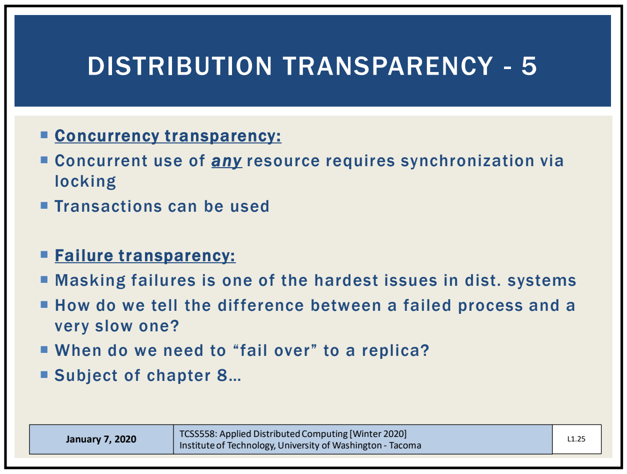
23



DISTRIBUTION TRANSPARENCY - 4

- **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

24



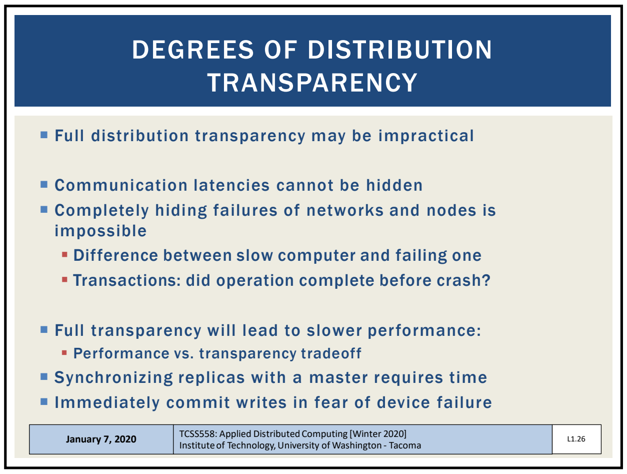
DISTRIBUTION TRANSPARENCY - 5

- **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…

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.25

25



DEGREES OF DISTRIBUTION TRANSPARENCY

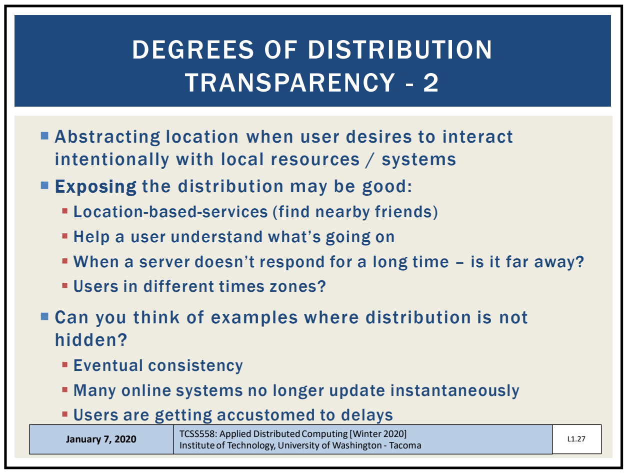
- Full distribution transparency may be impractical

- 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

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.26

26

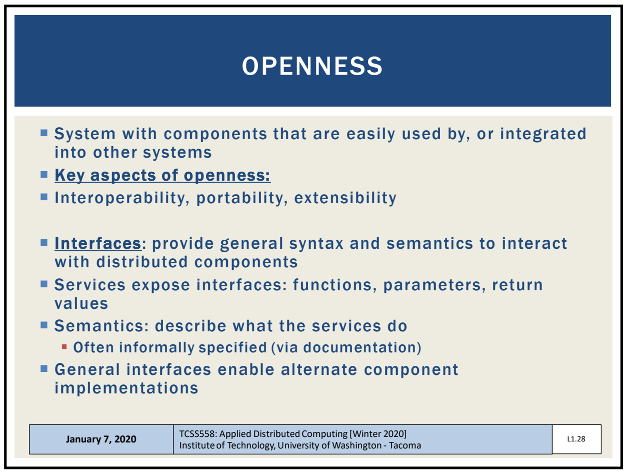


DEGREES OF DISTRIBUTION TRANSPARENCY - 2

- Abstracting location when user desires to interact intentionally with local resources / systems
- **Exposing** the distribution 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

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.27

27



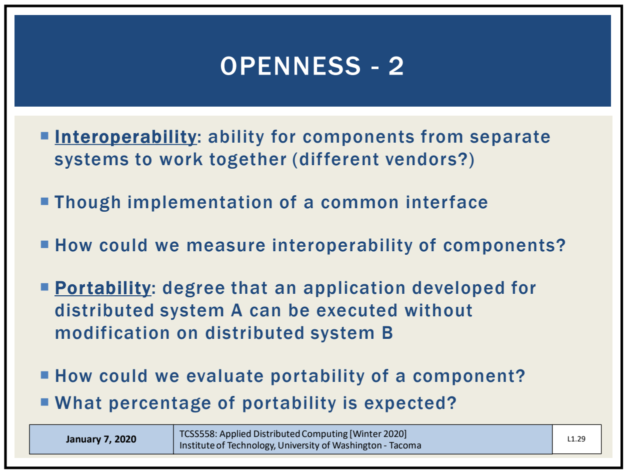
OPENNESS

- System with components that are easily used by, or integrated into other systems
- **Key aspects of openness:**
- Interoperability, portability, extensibility

- **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

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.28

28



OPENNESS - 2

- **Interoperability**: ability for components from separate systems to work together (different vendors?)

- Though implementation of a common interface

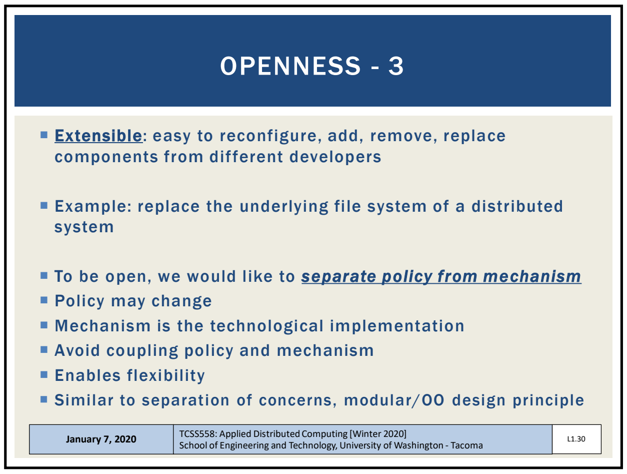
- How could we measure interoperability of components?

- **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?
- What percentage of portability is expected?

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] Institute of Technology, University of Washington - Tacoma | L1.29

29



OPENNESS - 3

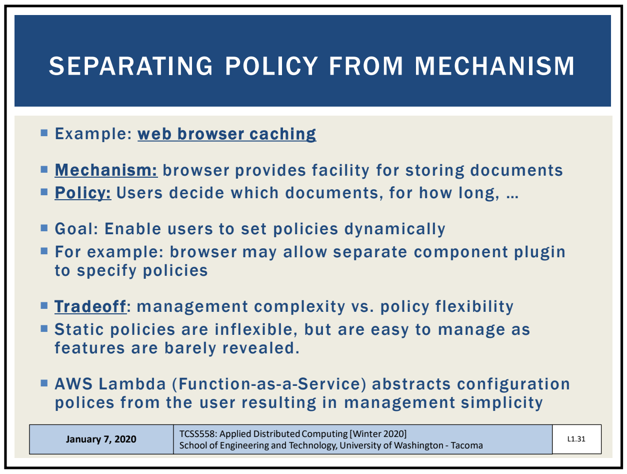
- **Extensible**: easy to reconfigure, add, remove, replace components from different developers

- 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
- Similar to separation of concerns, modular/OO design principle

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.30

30

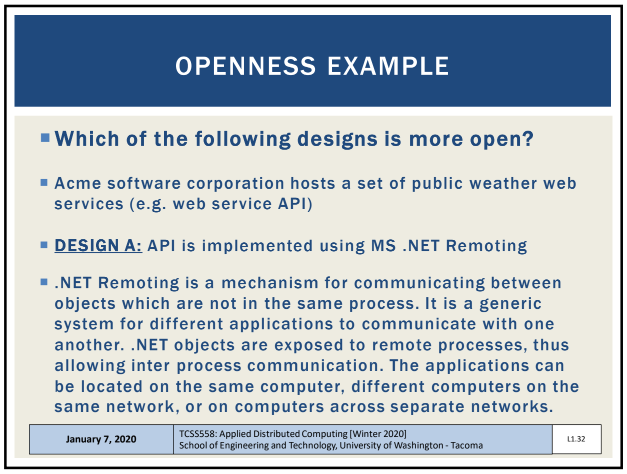


SEPARATING POLICY FROM MECHANISM

- Example: **web browser caching**
- **Mechanism:** browser provides facility for storing documents
- **Policy:** Users decide which documents, for how long, …
- Goal: Enable users to set policies dynamically
- For example: browser may allow separate component plugin to specify policies
- **Tradeoff**: management complexity vs. policy flexibility
- Static policies are inflexible, but are easy to manage as features are barely revealed.
- AWS Lambda (Function-as-a-Service) abstracts configuration polices from the user resulting in management simplicity

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.31

31

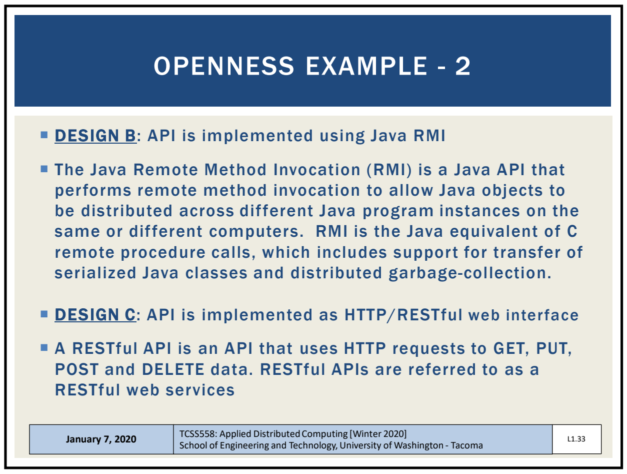


OPENNESS EXAMPLE

- **Which of the following designs is more open?**
- Acme software corporation hosts a set of public weather web services (e.g. web service API)
- **DESIGN A:** API is implemented using MS .NET Remoting
- .NET Remoting is a mechanism for communicating between objects which are not in the same process. It is a generic system for different applications to communicate with one another. .NET objects are exposed to remote processes, thus allowing inter process communication. The applications can be located on the same computer, different computers on the same network, or on computers across separate networks.

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.32

32

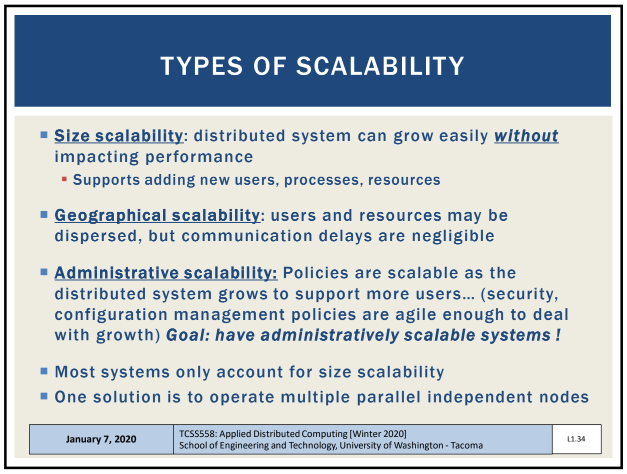


OPENNESS EXAMPLE - 2

- **DESIGN B**: API is implemented using Java RMI
- The Java Remote Method Invocation (RMI) is a Java API that performs remote method invocation to allow Java objects to be distributed across different Java program instances on the same or different computers. RMI is the Java equivalent of C remote procedure calls, which includes support for transfer of serialized Java classes and distributed garbage-collection.
- **DESIGN C**: API is implemented as HTTP/RESTful web interface
- A RESTful API is an API that uses HTTP requests to GET, PUT, POST and DELETE data. RESTful APIs are referred to as a RESTful web services

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.33

33

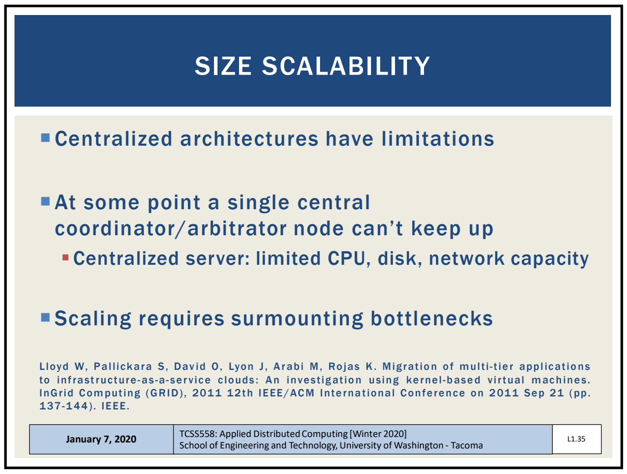


TYPES OF SCALABILITY

- **Size scalability**: distributed system can grow easily ***without*** impacting performance
  - Supports adding new users, processes, resources
- **Geographical scalability**: users and resources may be dispersed, but communication delays are negligible
- **Administrative scalability:** Policies are scalable as the distributed system grows to support more users… (security, configuration management policies are agile enough to deal with growth) ***Goal: have administratively scalable systems !***
- Most systems only account for size scalability
- One solution is to operate multiple parallel independent nodes

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.34

34



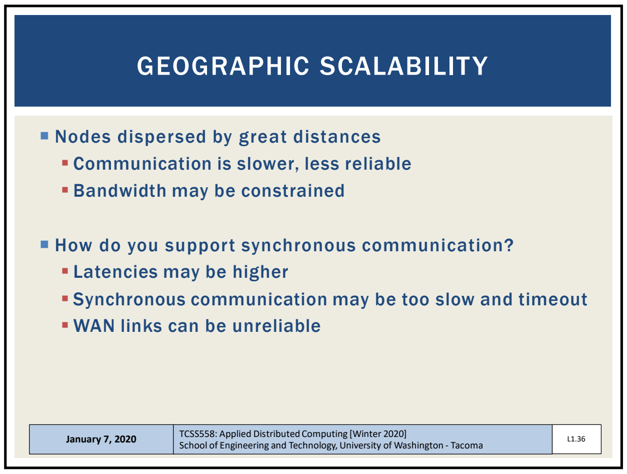
SIZE SCALABILITY

- Centralized architectures have limitations
- At some point a single central coordinator/arbitrator node can't keep up
  - Centralized server: limited CPU, disk, network capacity
- Scaling requires surmounting bottlenecks

Lloyd W, Pallickara S, David O, Lyon J, Arabi M, Rojas K. Migration of multi-tier applications to infrastructure-as-a-service clouds: An investigation using kernel-based virtual machines. InGrid Computing (GRID), 2011 12th IEEE/ACM International Conference on 2011 Sep 21 (pp. 137-144). IEEE.

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.35

35

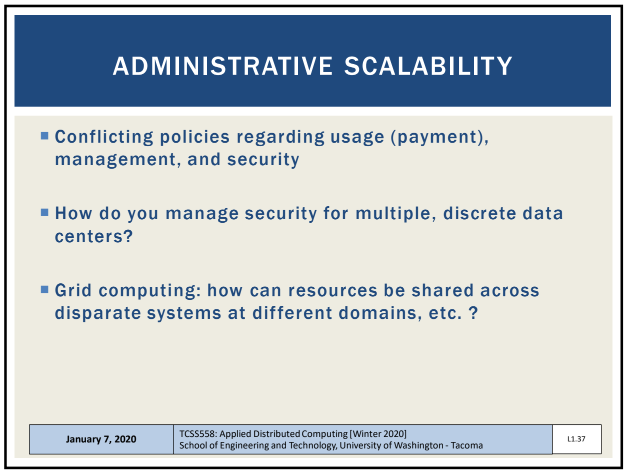


GEOGRAPHIC SCALABILITY

- Nodes dispersed by great distances
  - Communication is slower, less reliable
  - Bandwidth may be constrained
- How do you support synchronous communication?
  - Latencies may be higher
  - Synchronous communication may be too slow and timeout
  - WAN links can be unreliable

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.36

36

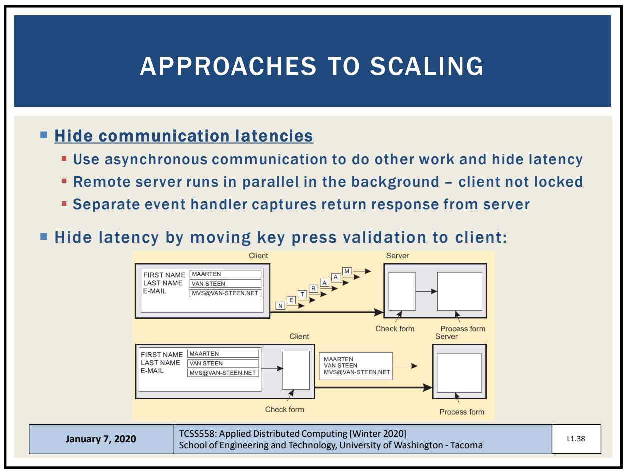


ADMINISTRATIVE SCALABILITY

- Conflicting policies regarding usage (payment), management, and security
- How do you manage security for multiple, discrete data centers?
- Grid computing: how can resources be shared across disparate systems at different domains, etc. ?

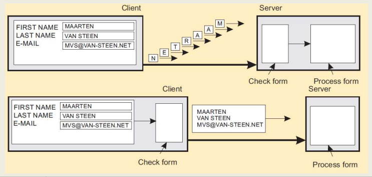
January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.37

37



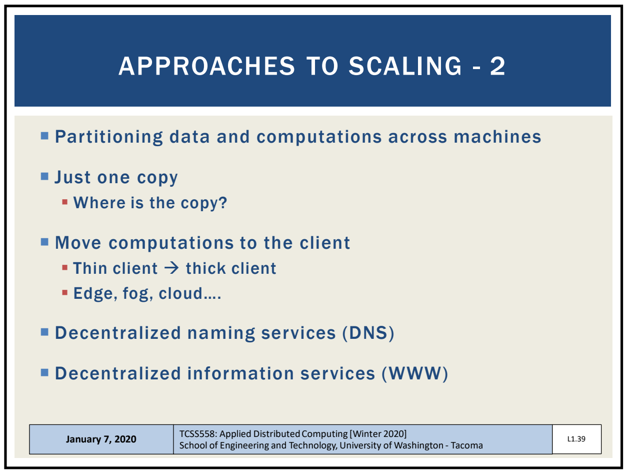
APPROACHES TO SCALING

- **Hide communication latencies**
  - Use asynchronous communication to do other work and hide latency
  - Remote server runs in parallel in the background – client not locked
  - Separate event handler captures return response from server
- Hide latency by moving key press validation to client:



January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.38

38

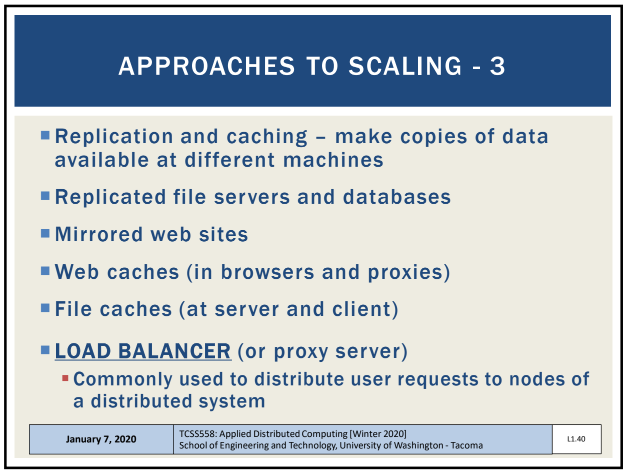


APPROACHES TO SCALING - 2

- Partitioning data and computations across machines
- Just one copy
  - Where is the copy?
- Move computations to the client
  - Thin client → thick client
  - Edge, fog, cloud….
- Decentralized naming services (DNS)
- Decentralized information services (WWW)

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.39

39

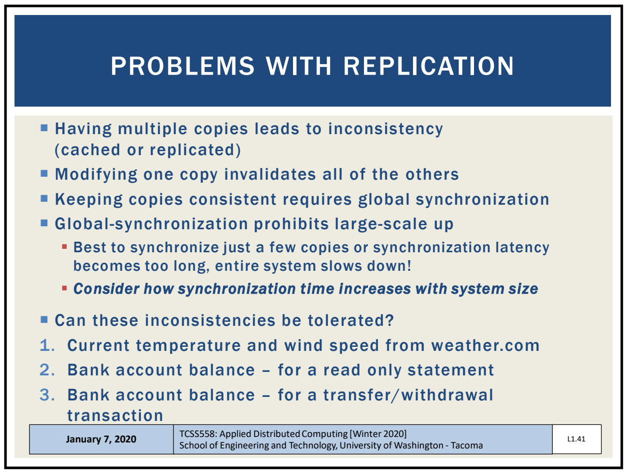


APPROACHES TO SCALING - 3

- Replication and caching – make copies of data available at different machines
- Replicated file servers and databases
- Mirrored web sites
- Web caches (in browsers and proxies)
- File caches (at server and client)
- **LOAD BALANCER** (or proxy server)
  - Commonly used to distribute user requests to nodes of a distributed system

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.40

40



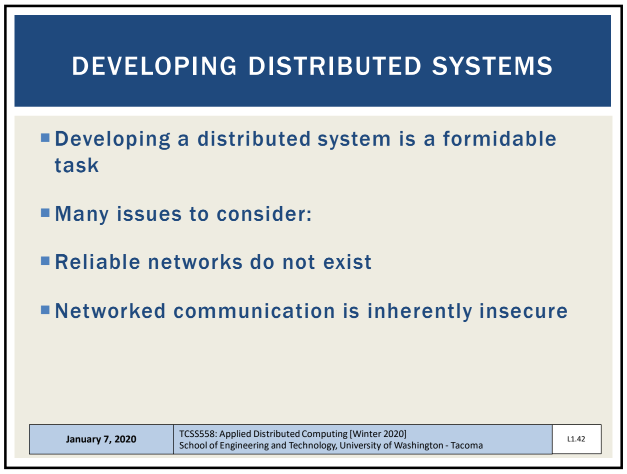
PROBLEMS WITH REPLICATION

- Having multiple copies leads to inconsistency (cached or replicated)
- Modifying one copy invalidates all of the others
- Keeping copies consistent requires global synchronization
- Global-synchronization prohibits large-scale up
  - Best to synchronize just a few copies or synchronization latency becomes too long, entire system slows down!
  - ***Consider how synchronization time increases with system size***
- Can these inconsistencies be tolerated?

1. Current temperature and wind speed from weather.com
2. Bank account balance – for a read only statement
3. Bank account balance – for a transfer/withdrawal transaction

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.41

41

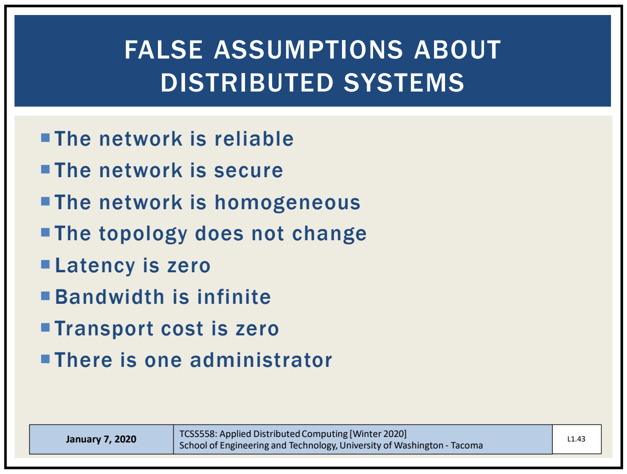


DEVELOPING DISTRIBUTED SYSTEMS

- Developing a distributed system is a formidable task
- Many issues to consider:
- Reliable networks do not exist
- Networked communication is inherently insecure

January 7, 2020 | TCSS558: Applied Distributed Computing [Winter 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.42

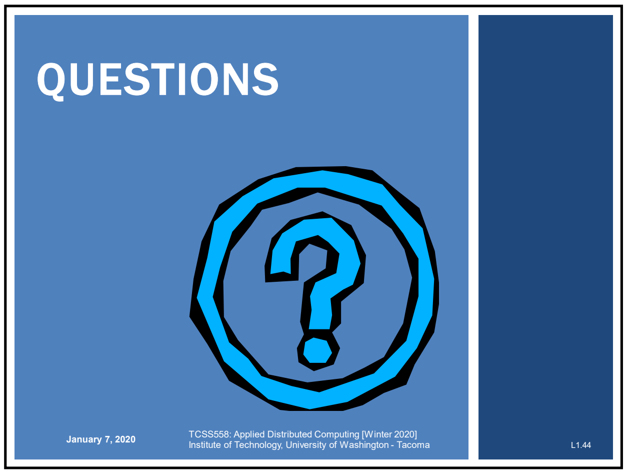
42



FALSE ASSUMPTIONS ABOUT DISTRIBUTED SYSTEMS

- The network is reliable
- The network is secure
- The network is homogeneous
- The topology does not change
- Latency is zero
- Bandwidth is infinite
- Transport cost is zero
- There is one administrator

43



QUESTIONS

44