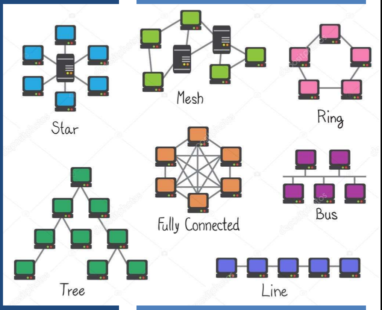
TCSS 558: APPLIED DISTRIBUTED COMPUTING



Introduction to Distributed Systems - II

Wes J. Lloyd
School of Engineering & Technology (SET)
University of Washington - Tacoma

OBJECTIVES – 1/7

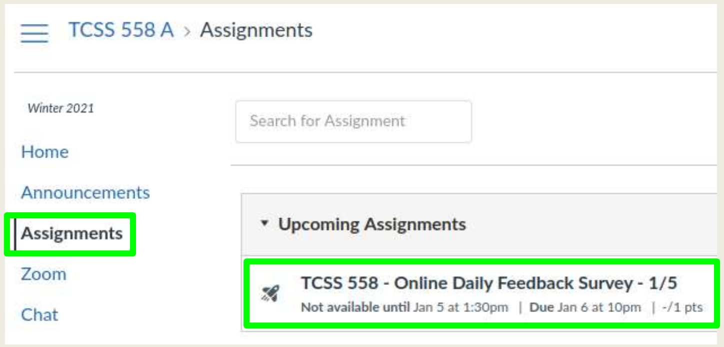
- Questions from 1/5
- Chapter 1 - What is a distributed system?
- Design goals of distributed systems:
  - Accessibility: resource sharing & availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems

TCSS 558 OFFICE HOURS – WINTER 2021

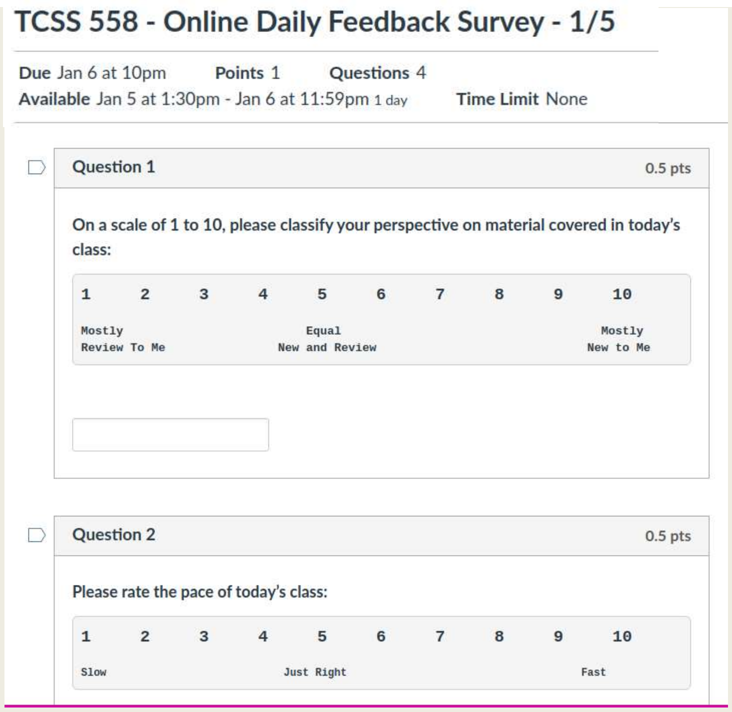
- Fridays 11:30a - 12:30p via Zoom
  - ZOOM link shared weekly via Canvas announcements
- Tuesdays 3:30p after class
- Thursdays 3:30p after class
  - Same ZOOM link as class
- By email appointment: wlloyd@uw.edu
  - Zoom link sent by email

ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Available After Each Class
- Extra credit available for completing surveys ***ON TIME***
- Tuesday surveys: due by Wed @ 10p
- Thursday surveys: due Mon @ 10p



TCSS 558 - Online Daily Feedback Survey - 1/5
Not available until Jan 5 at 1:30pm | Due Jan 6 at 10pm | -/1 pts



TCSS 558 - Online Daily Feedback Survey - 1/5

Due Jan 6 at 10pm **Points** 1 **Questions** 4
Available Jan 5 at 1:30pm - Jan 6 at 11:59pm 1 day **Time Limit** None

Question 1 0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

```
1      2      3      4      5      6      7      8      9      10

Mostly                    Equal                              Mostly
Review To Me          New and Review                      New to Me
```

Question 2 0.5 pts

Please rate the pace of today's class:

```
1      2      3      4      5      6      7      8      9      10

Slow                   Just Right                           Fast
```

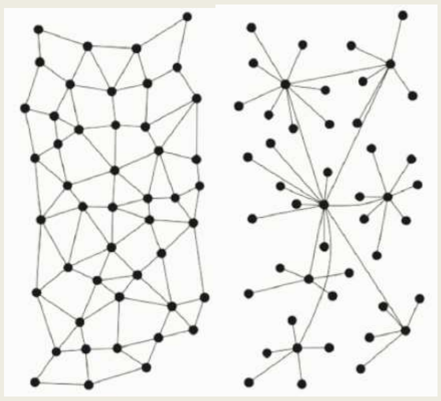
MATERIAL / PACE

- Please classify your perspective on material covered in today's class (23 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- <u>Average – 7.48</u>

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- <u>Average – 5.83</u>

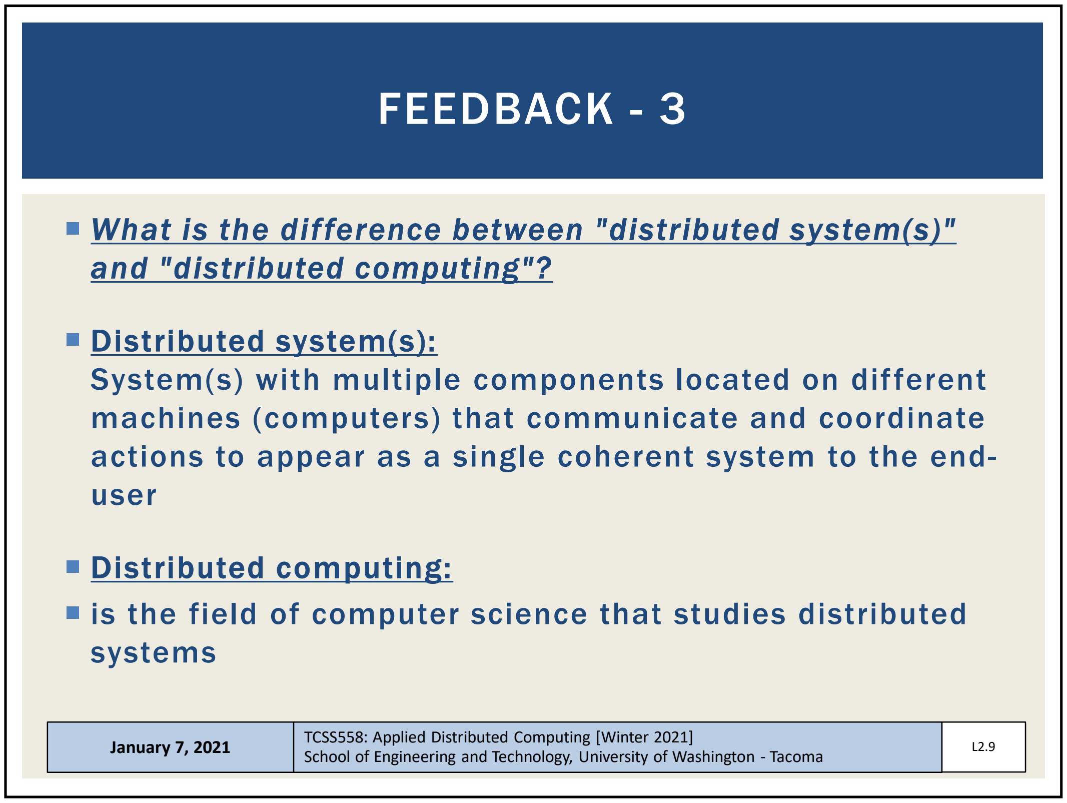
FEEDBACK FROM 1/5

- ***Early on there was a mention of the list of nodes of a neighbor possibly being dynamic. What's an example of a situation in which there would be dynamic neighbors?***
- Nodes in a distributed system may join/leave the system at anytime
- Local master/root nodes maintain membership list(s) that need to be refreshed
- In ad hoc systems, nodes must know neighbors that are available for routing messages



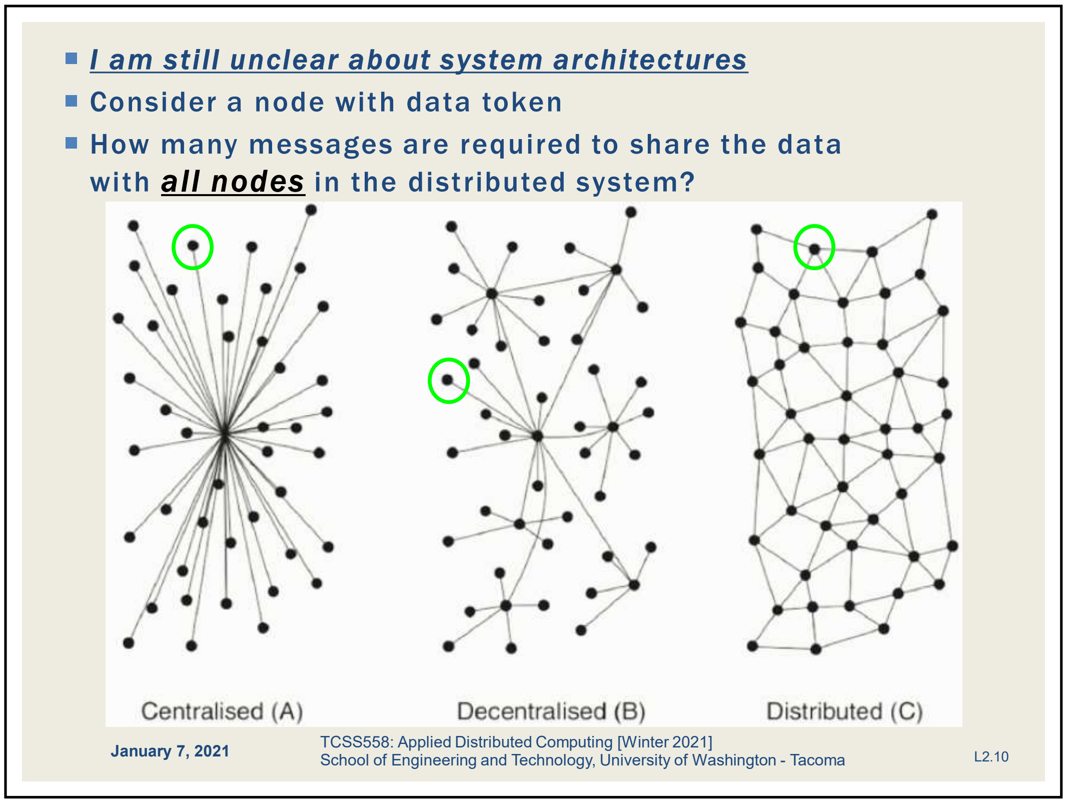
FEEDBACK - 2

- ***One concept that remains unclear is how distributed systems keep track of membership although I am sure we will go over that soon***
- When nodes of a distributed system require a membership list for a specific algorithm/function, the membership list can be stored on a **centralized server** to synchronize Reads/Writes,
  or **replicated** across all the nodes of the system
- Membership lists can be replicated using transactional approaches to replicate data when changes are made (see two-phase commit in chapter 8.5)

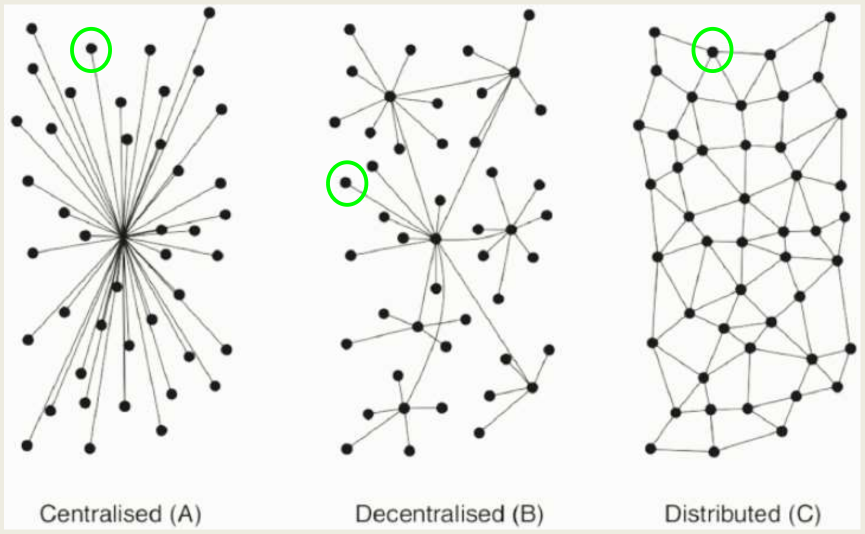


FEEDBACK - 3

- ***What is the difference between "distributed system(s)" and "distributed computing"?***
- **Distributed system(s):**
  System(s) with multiple components located on different machines (computers) that communicate and coordinate actions to appear as a single coherent system to the end-user
- **Distributed computing:**
- is the field of computer science that studies distributed systems

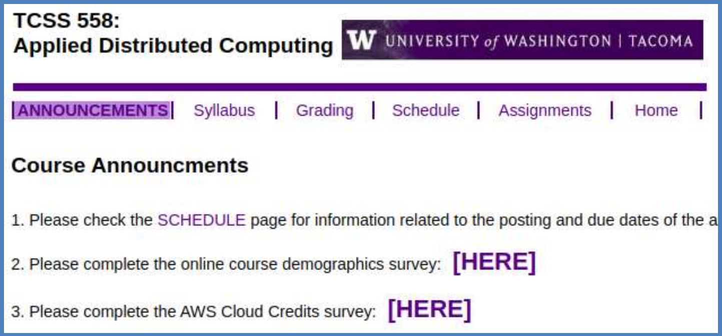


- ***I am still unclear about system architectures***
- Consider a node with data token
- How many messages are required to share the data with ***all nodes*** in the distributed system?



SURVEY LINKS

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


TCSS 558:
Applied Distributed Computing

UNIVERSITY of WASHINGTON | TACOMA

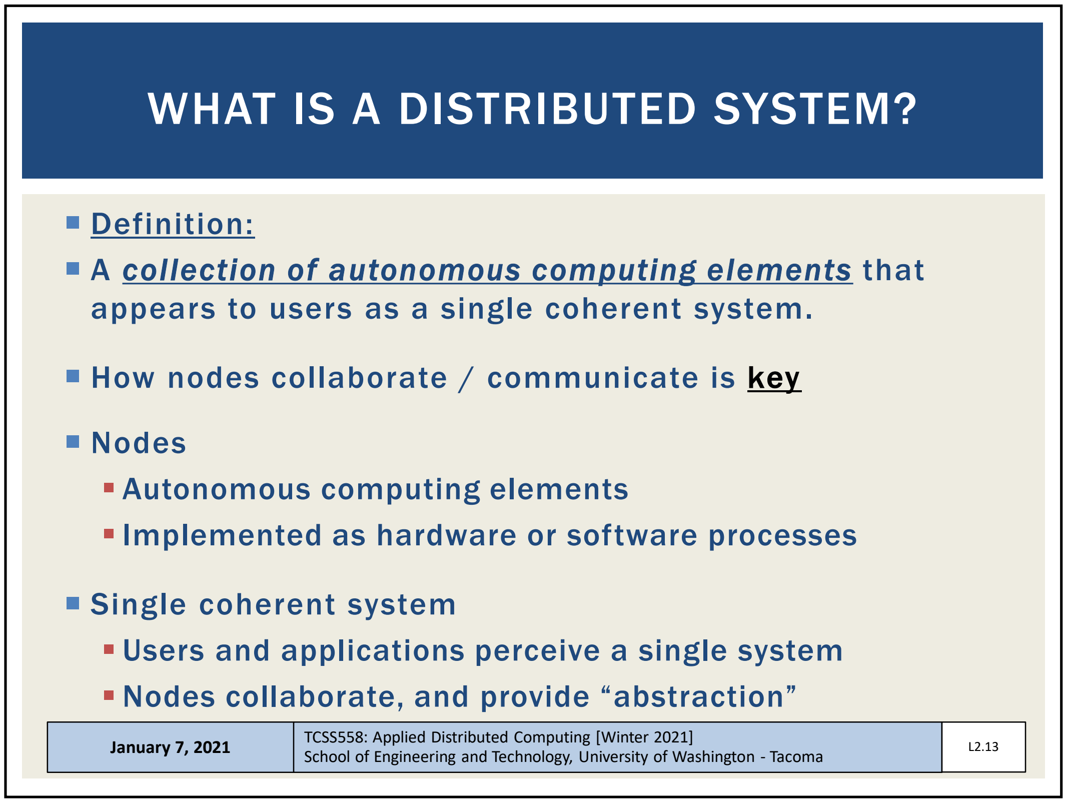
|ANNOUNCEMENTS| Syllabus | Grading | Schedule | Assignments | Home |

Course Announcments

1. Please check the SCHEDULE page for information related to the posting and due dates of the a
2. Please complete the online course demographics survey: [HERE]
3. Please complete the AWS Cloud Credits survey: [HERE]

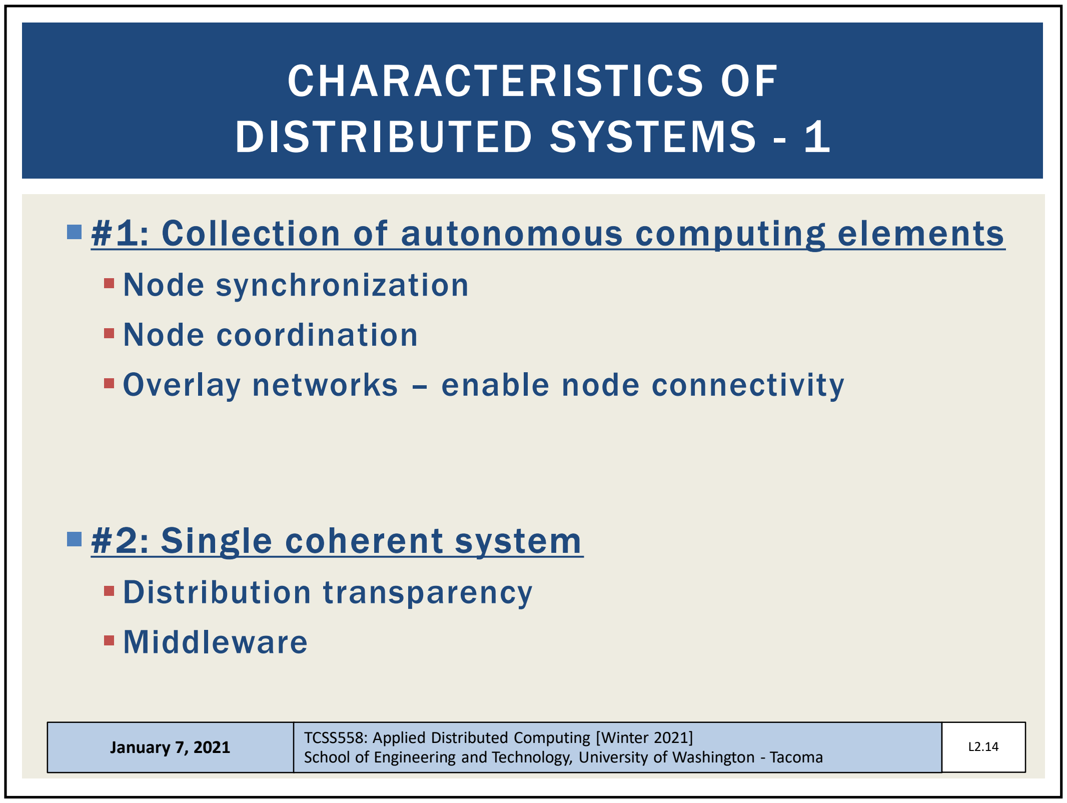
OBJECTIVES – 1/7

- Questions from 1/5
- Chapter 1 - What is a distributed system?
- Design goals of distributed systems:
  - Accessibility: resource sharing & availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems



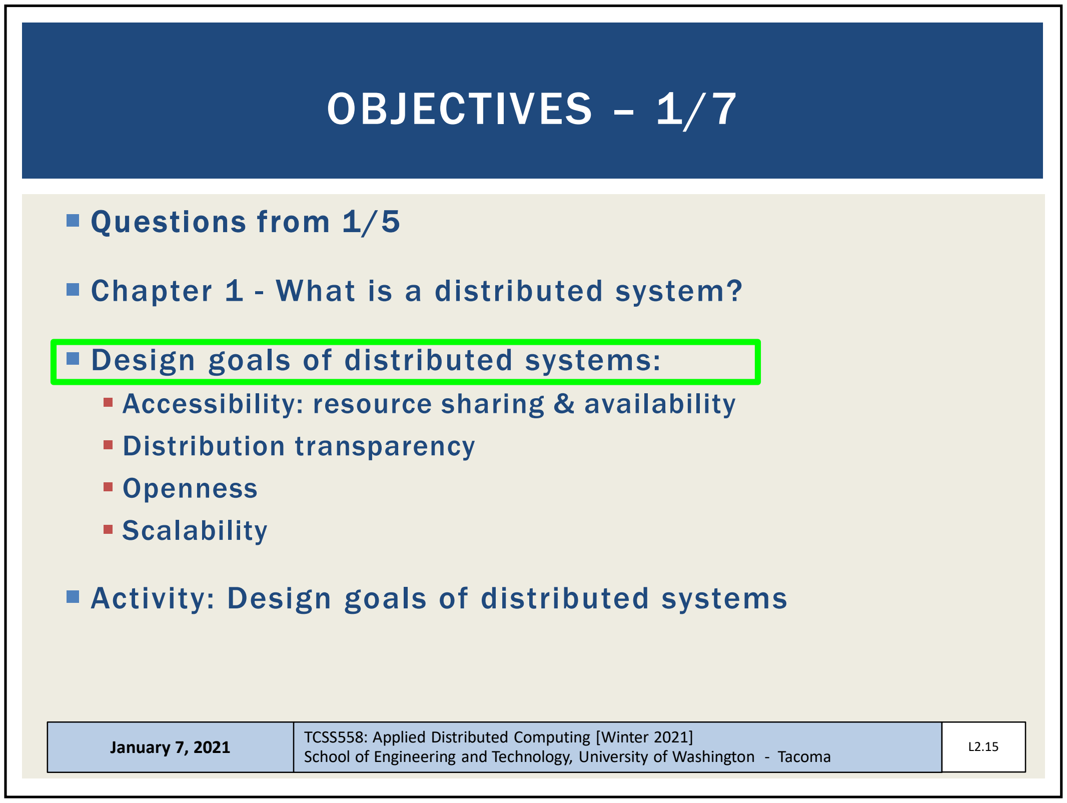
WHAT IS A DISTRIBUTED SYSTEM?

- <u>Definition:</u>
- A ***<u>collection of autonomous computing elements</u>*** that appears to users as a single coherent system.
- How nodes collaborate / communicate is **<u>key</u>**
- 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”



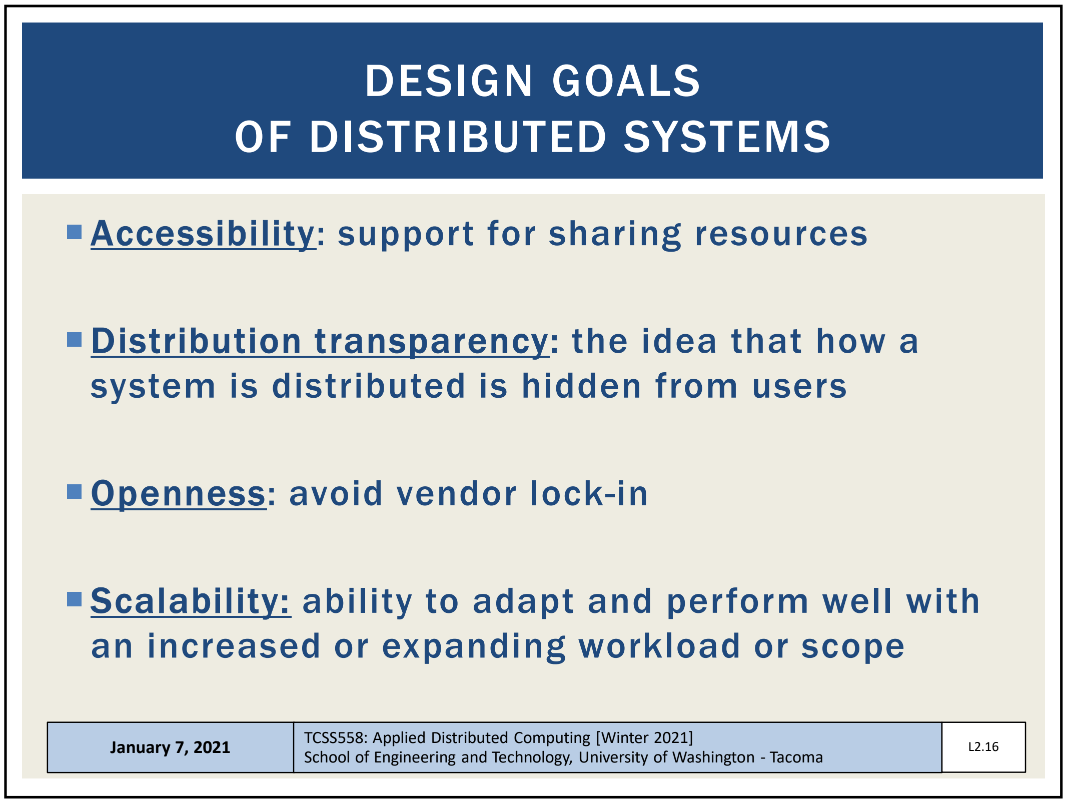
CHARACTERISTICS OF DISTRIBUTED SYSTEMS - 1

- **<u>#1: Collection of autonomous computing elements</u>**
  - Node synchronization
  - Node coordination
  - Overlay networks – enable node connectivity
- **<u>#2: Single coherent system</u>**
  - Distribution transparency
  - Middleware



OBJECTIVES – 1/7

- Questions from 1/5
- Chapter 1 - What is a distributed system?
- Design goals of distributed systems:
  - Accessibility: resource sharing & availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems



DESIGN GOALS OF DISTRIBUTED SYSTEMS

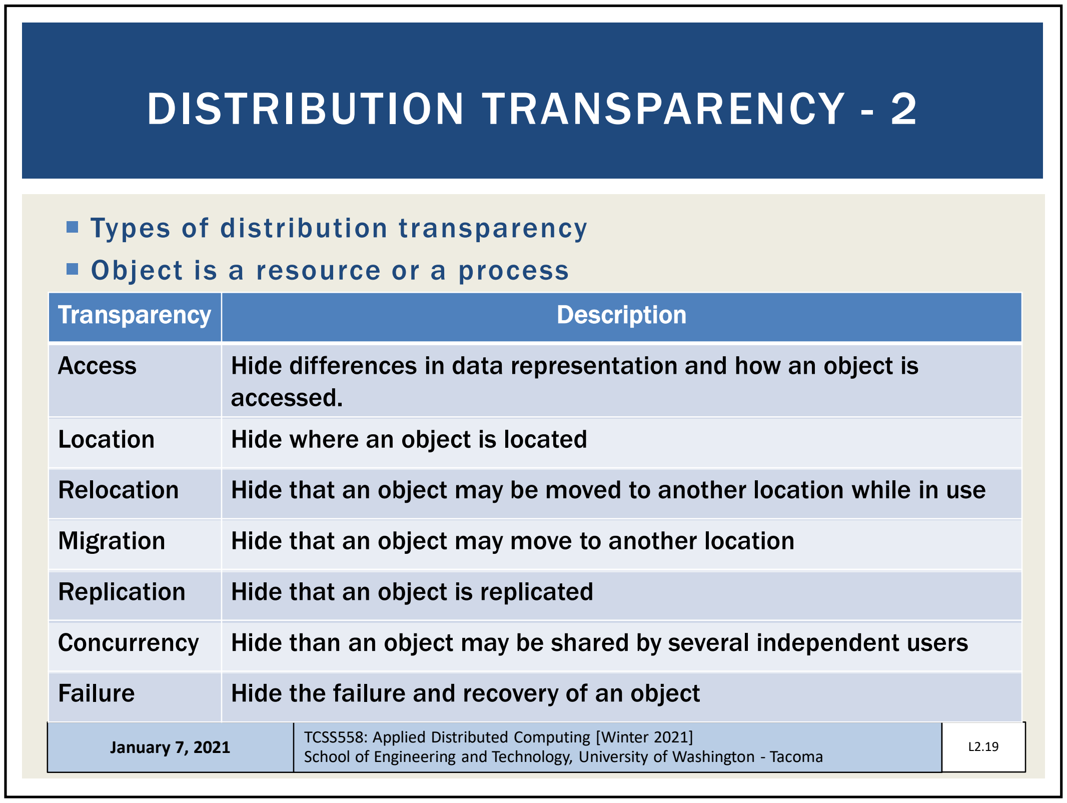
- Accessibility: support for sharing resources
- Distribution transparency: the idea that how a system is distributed is hidden from users
- Openness: avoid vendor lock-in
- Scalability: ability to adapt and perform well with an increased or expanding workload or scope

OBJECTIVES – 1/7

- Questions from 1/5
- Chapter 1 - What is a distributed system?
- Design goals of distributed systems:
  - Accessibility: resource sharing & availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems

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 <u>transparent</u> 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.

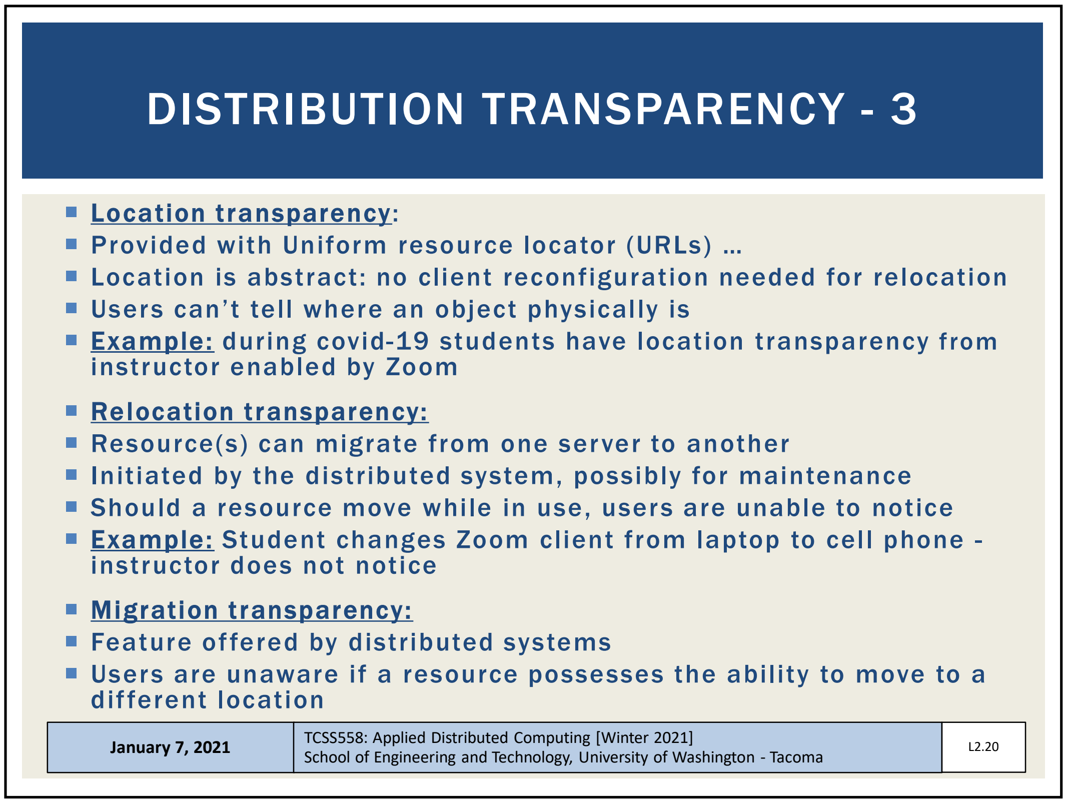


DISTRIBUTION TRANSPARENCY - 2

- 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

January 7, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L2.19



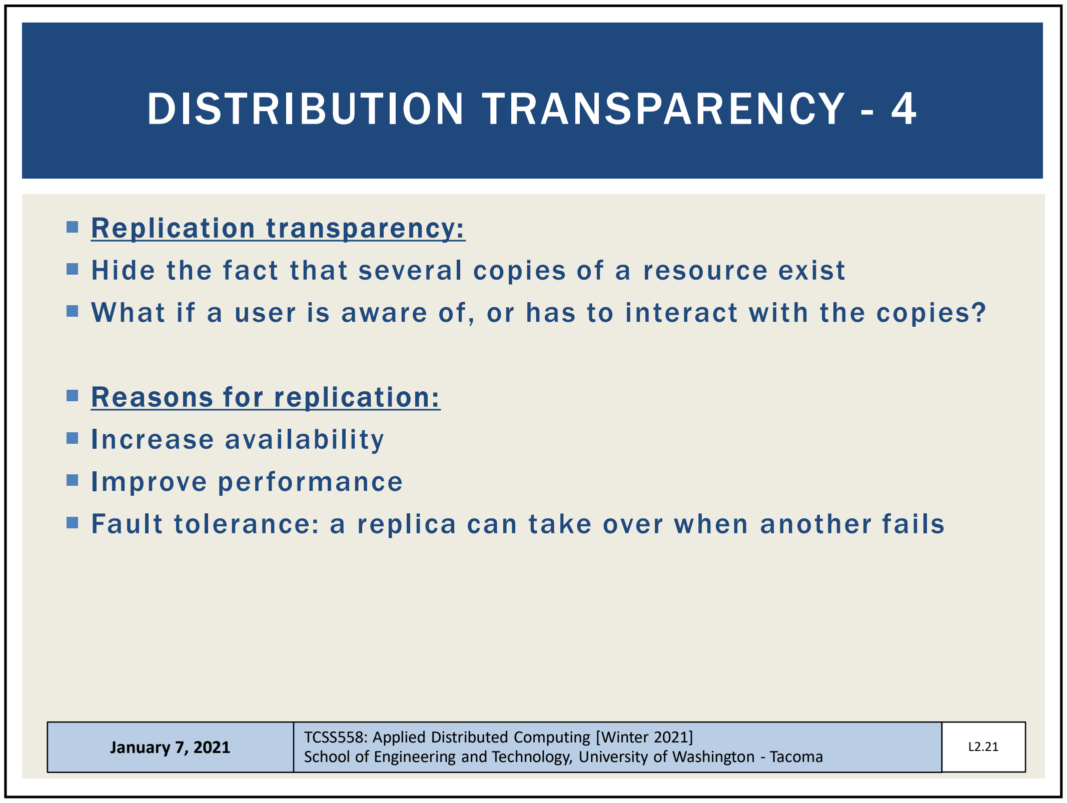
DISTRIBUTION TRANSPARENCY - 3

- **Location transparency:**
- Provided with Uniform resource locator (URLs) …
- Location is abstract: no client reconfiguration needed for relocation
- Users can’t tell where an object physically is
- **Example:** during covid-19 students have location transparency from instructor enabled by Zoom

- **Relocation transparency:**
- Resource(s) can migrate from one server to another
- Initiated by the distributed system, possibly for maintenance
- Should a resource move while in use, users are unable to notice
- **Example:** Student changes Zoom client from laptop to cell phone - instructor does not notice

- **Migration transparency:**
- Feature offered by distributed systems
- Users are unaware if a resource possesses the ability to move to a different location

January 7, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L2.20

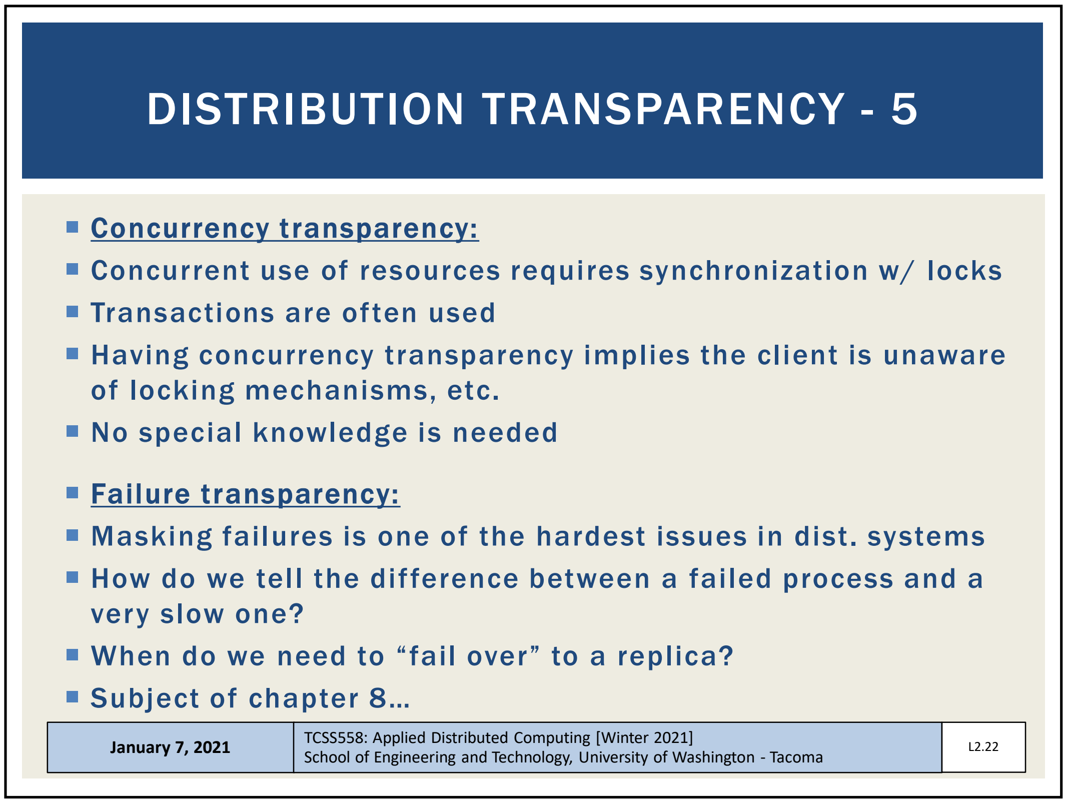


DISTRIBUTION TRANSPARENCY - 4

- **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

January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.21

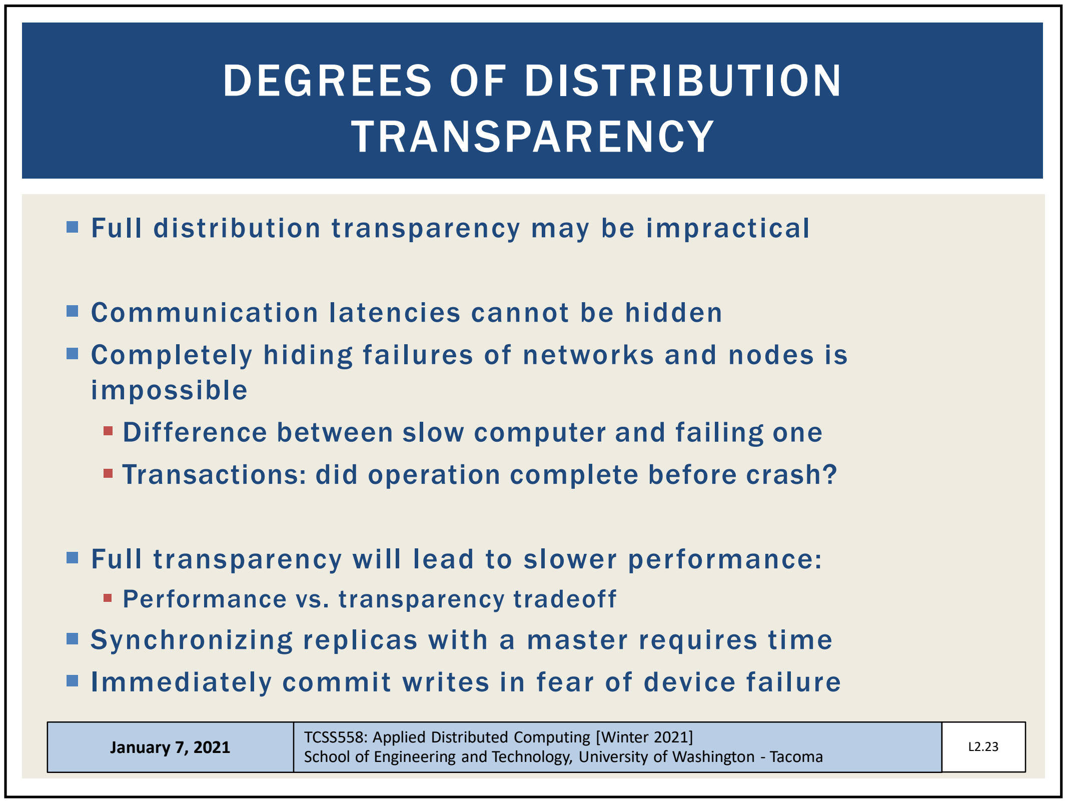


DISTRIBUTION TRANSPARENCY - 5

- **Concurrency transparency:**
- Concurrent use of resources requires synchronization w/ locks
- Transactions are often used
- Having concurrency transparency implies the client is unaware of locking mechanisms, etc.
- No special knowledge is needed

- **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, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.22

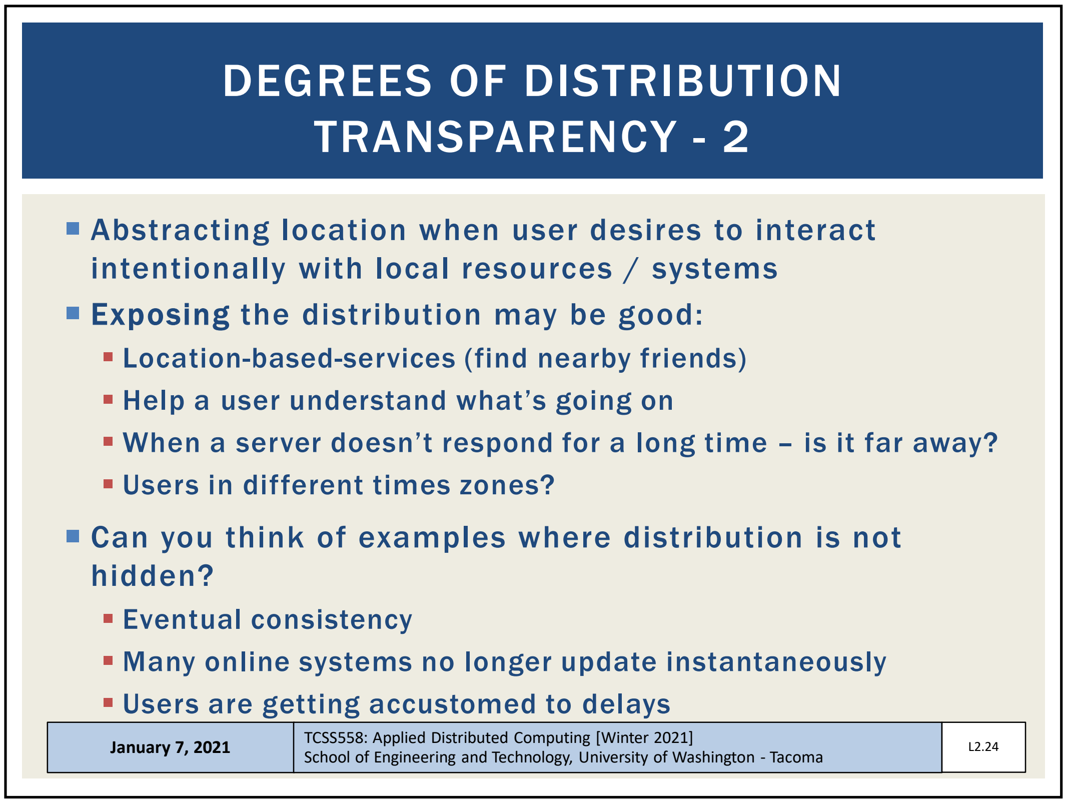


DEGREES OF DISTRIBUTION TRANSPARENCY

- Full distribution transparency may be impractical

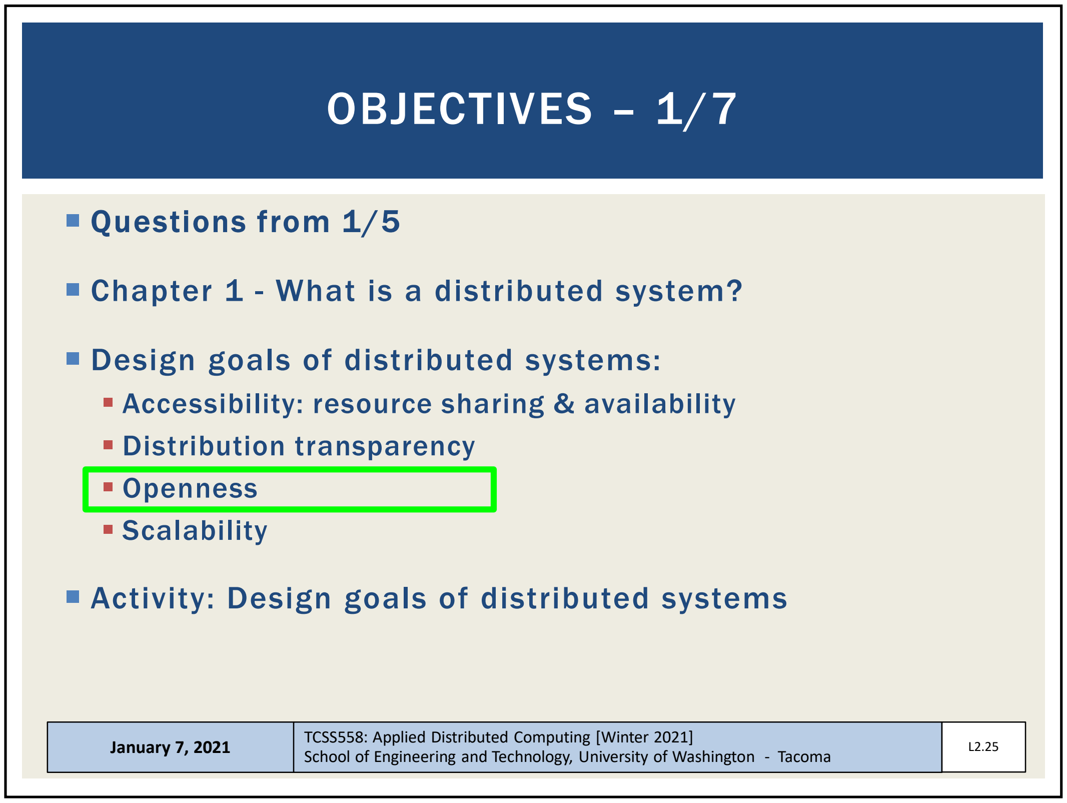
- Communication latencies cannot be hidden
- Completely hiding failures of networks and nodes is impossible
  - Difference between slow computer and failing one
  - Transactions: did operation complete before crash?

- 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

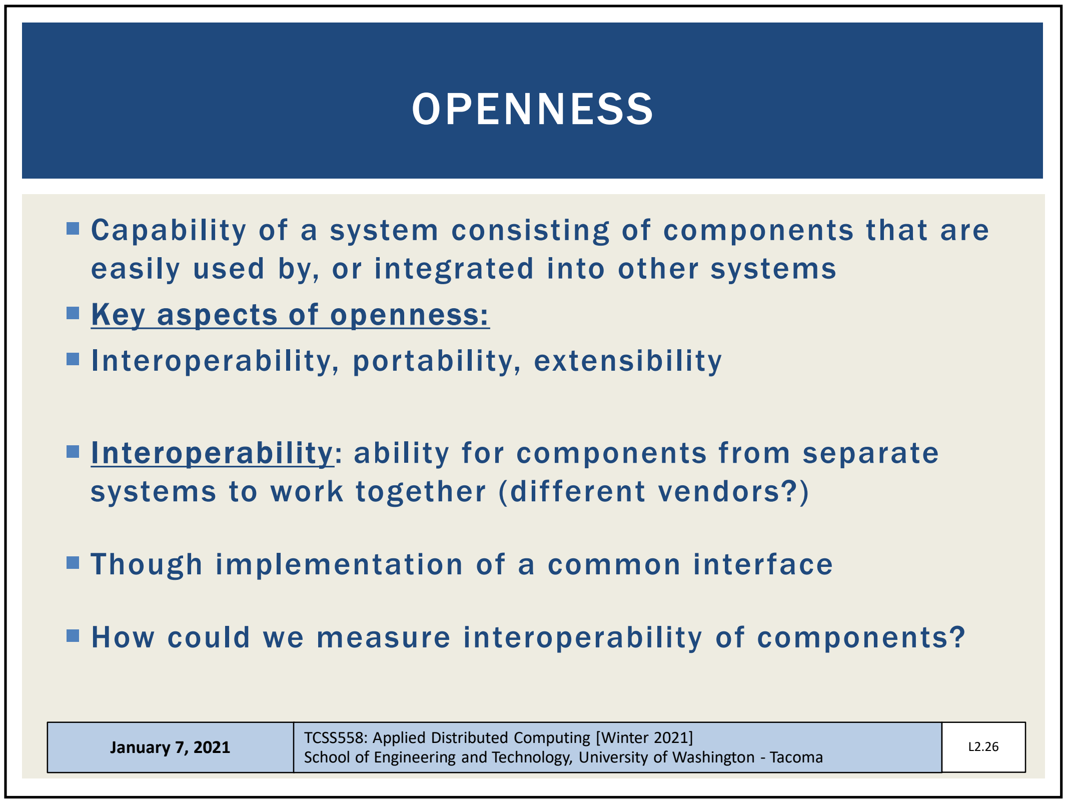
- Abstracting location when user desires to interact intentionally with local resources / systems
- Exposing the distribution may be good:
  - Location-based-services (find nearby friends)
  - Help a user understand what's going on
  - When a server doesn't respond for a long time – is it far away?
  - 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



OBJECTIVES – 1/7

- Questions from 1/5
- Chapter 1 - What is a distributed system?
- Design goals of distributed systems:
  - Accessibility: resource sharing & availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems

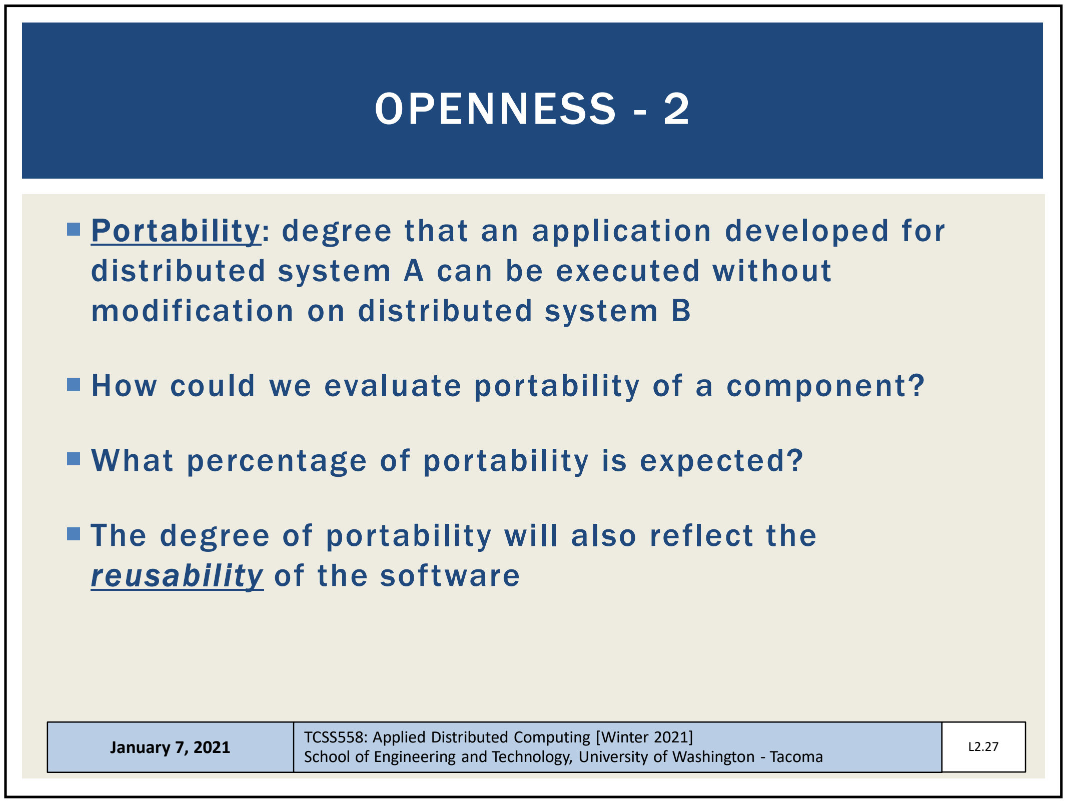
January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.25



OPENNESS

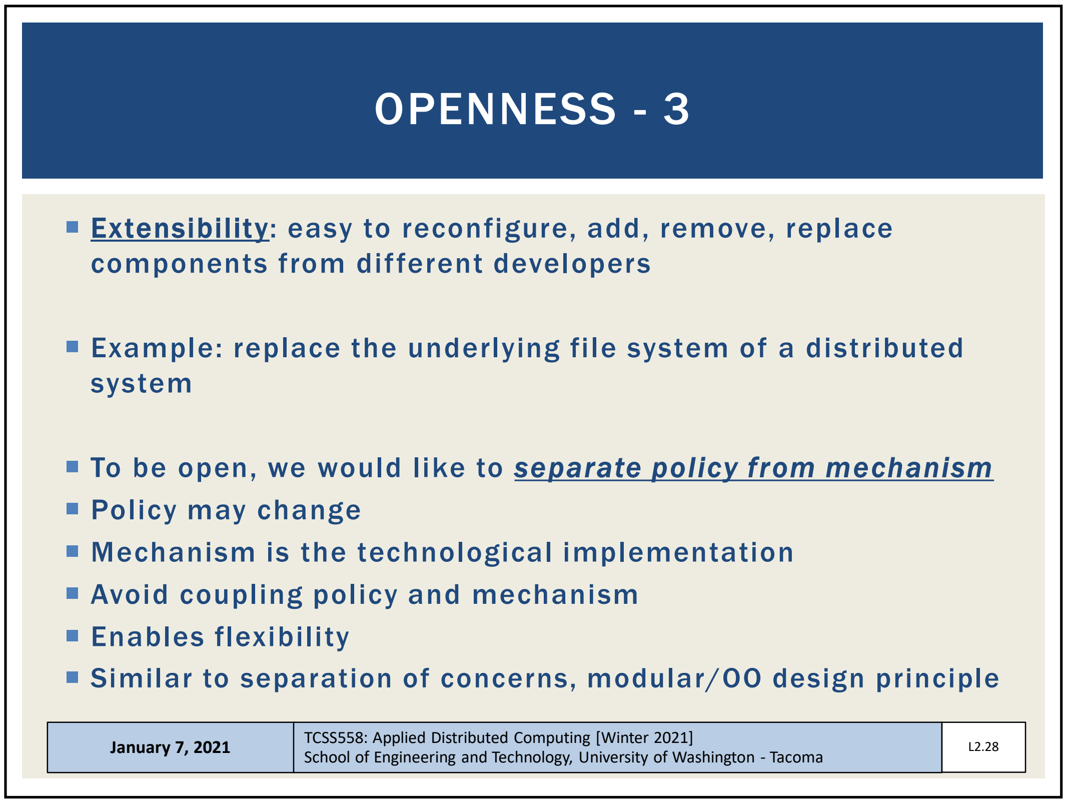
- Capability of a system consisting of components that are easily used by, or integrated into other systems
- **Key aspects of openness:**
- Interoperability, portability, extensibility
- **Interoperability**: ability for components from separate systems to work together (different vendors?)
- Though implementation of a common interface
- How could we measure interoperability of components?

January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.26



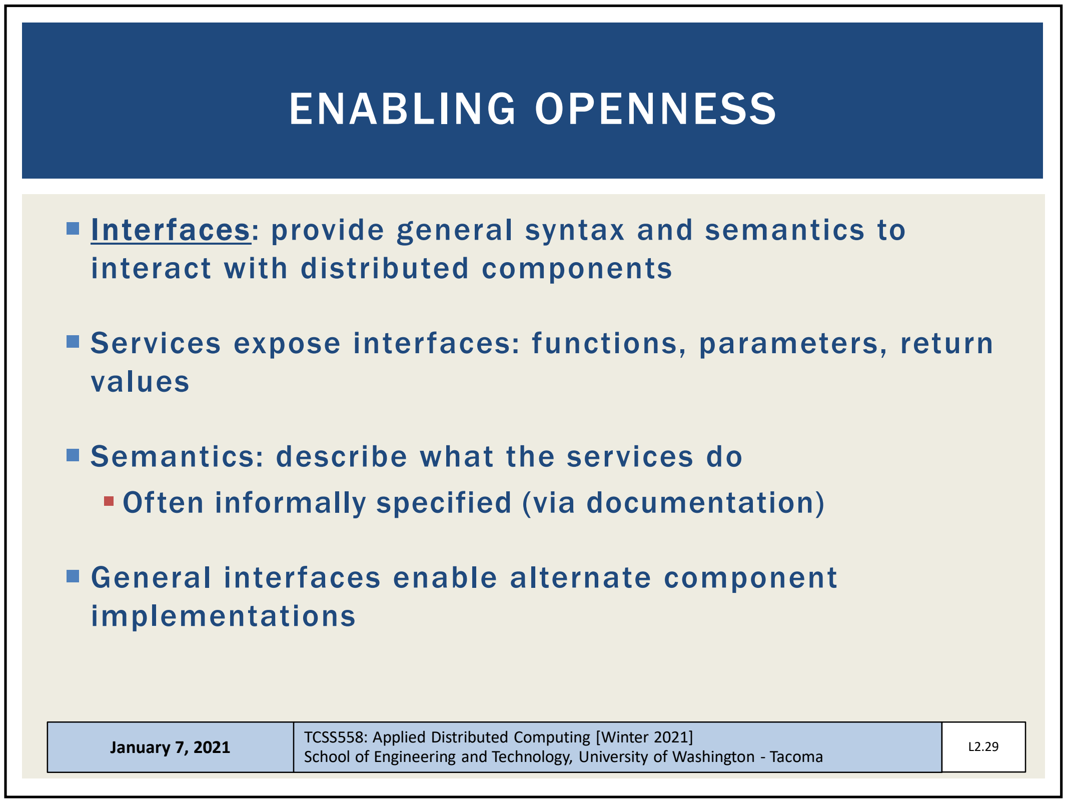
OPENNESS - 2

- **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?
- The degree of portability will also reflect the ***reusability*** of the software



OPENNESS - 3

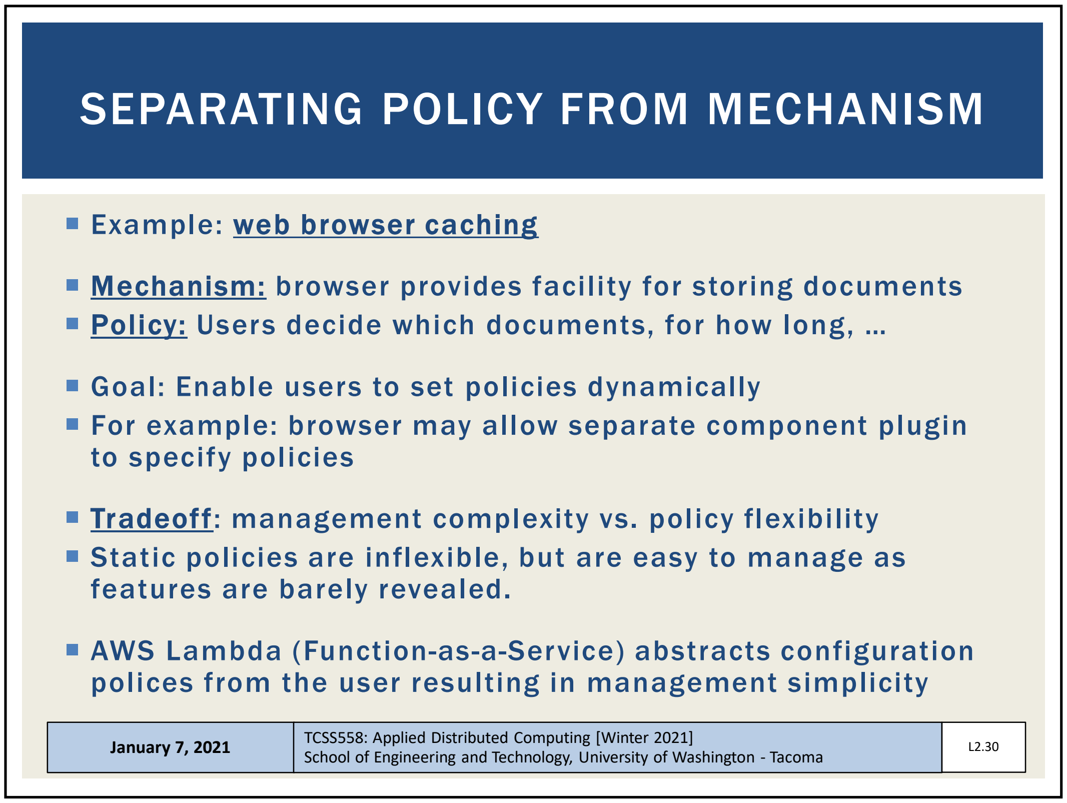
- **Extensibility**: easy to reconfigure, add, remove, replace components from different developers
- Example: replace the underlying file system of a distributed system
- To be open, we would like to ***separate policy from mechanism***
- Policy may change
- Mechanism is the technological implementation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



ENABLING OPENNESS

- **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

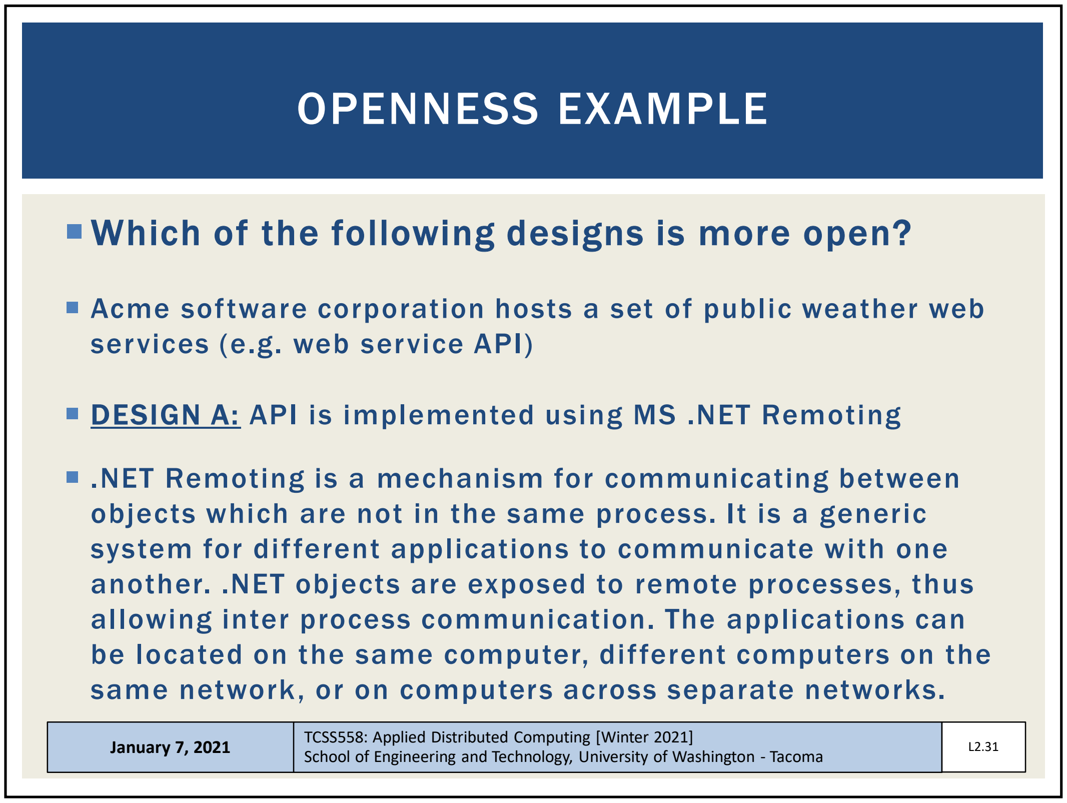
January 7, 2021 | TCSS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma | L2.29



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

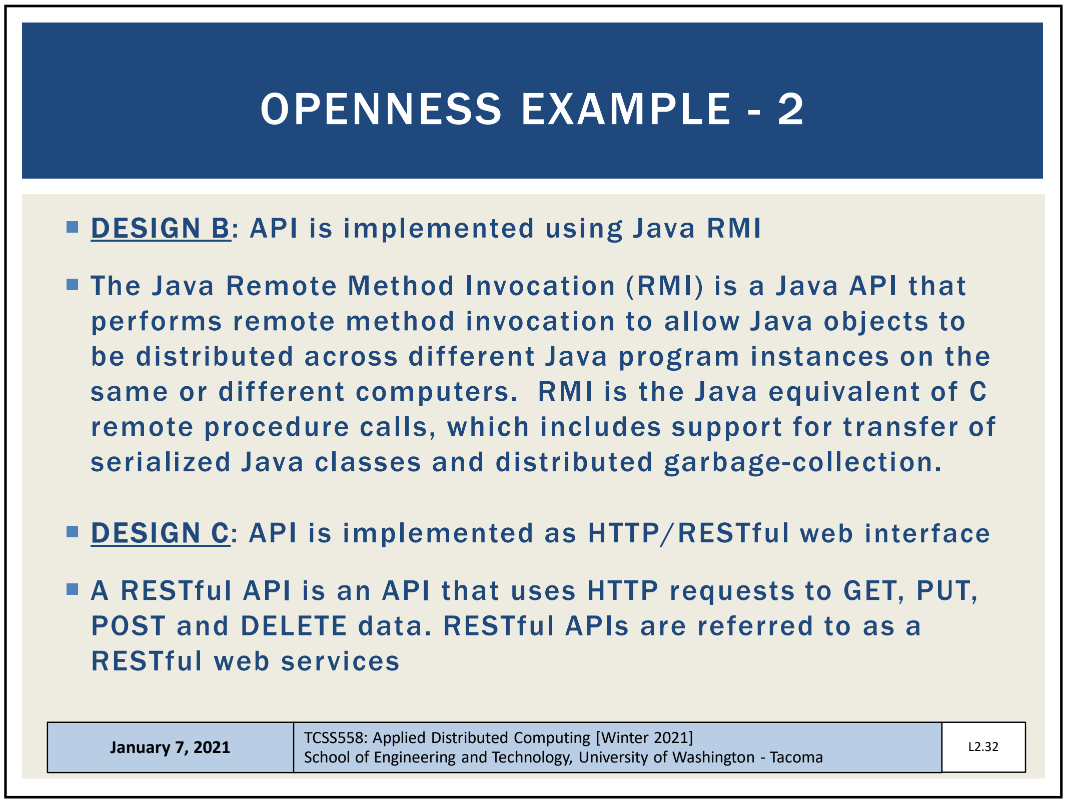
January 7, 2021 | TCSS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma | L2.30



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.

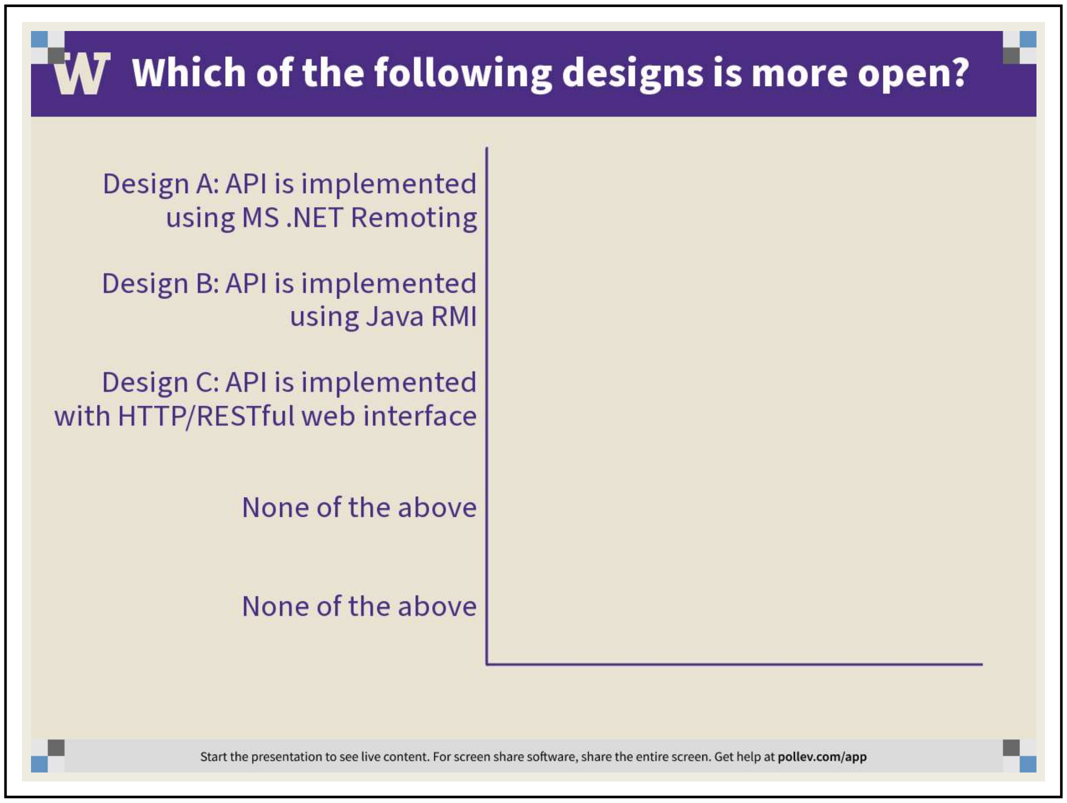
January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.31



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, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.32



Which of the following designs is more open?

Design A: API is implemented using MS .NET Remoting

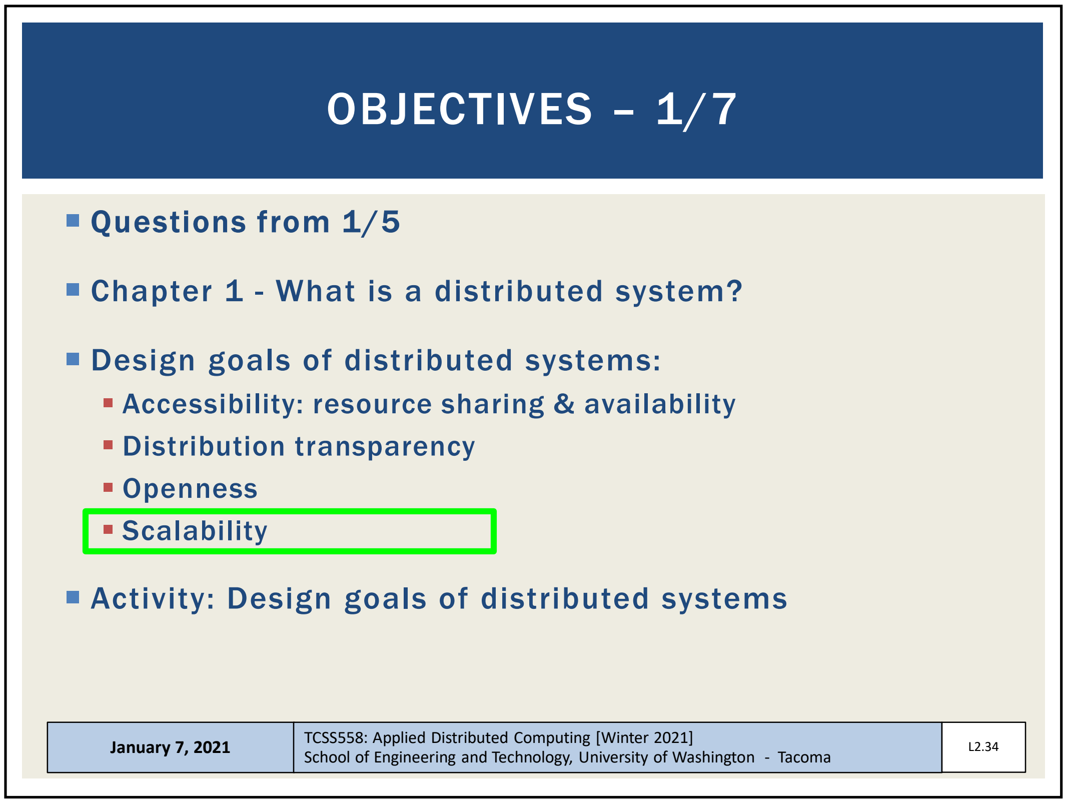
Design B: API is implemented using Java RMI

Design C: API is implemented with HTTP/RESTful web interface

None of the above

None of the above

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**



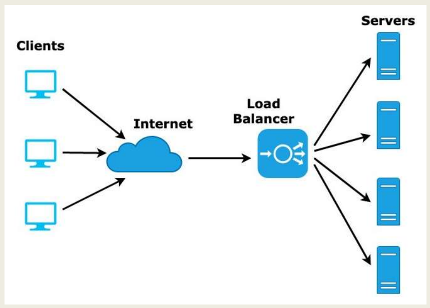
OBJECTIVES – 1/7

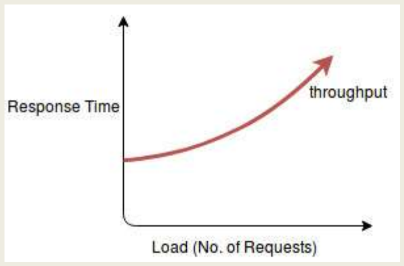
- Questions from 1/5
- Chapter 1 - What is a distributed system?
- Design goals of distributed systems:
  - Accessibility: resource sharing & availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems

January 7, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L2.34

SCALABILITY

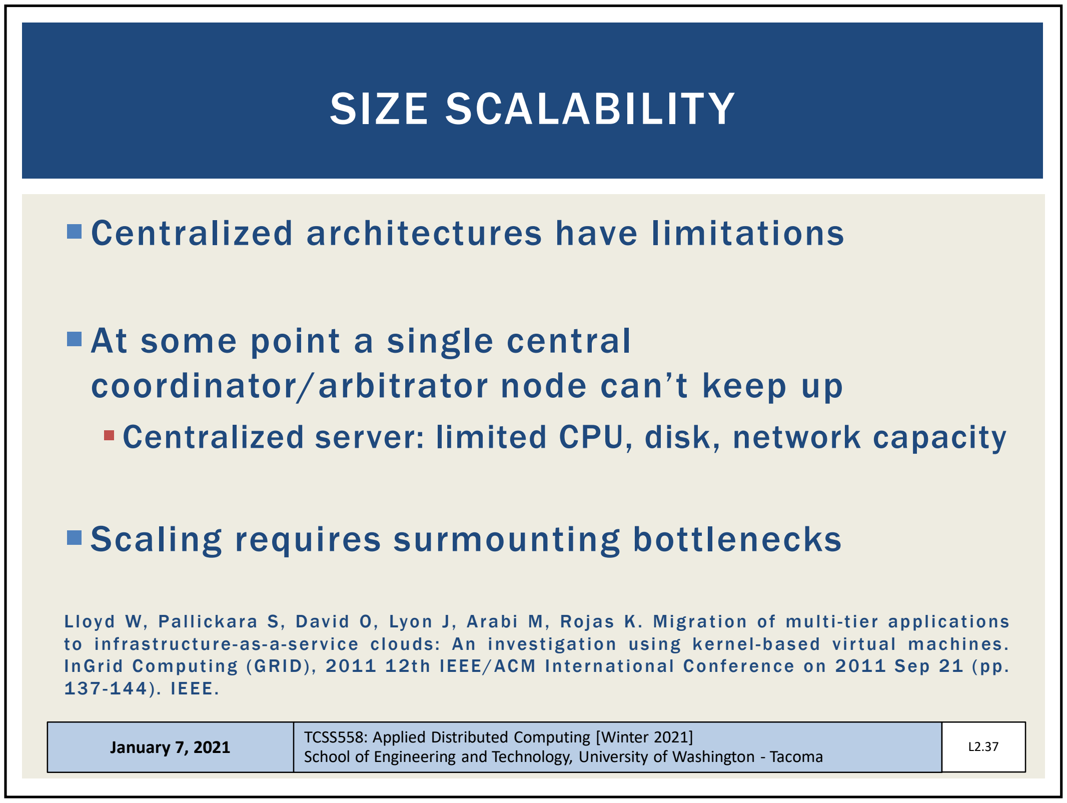
- The capability of a system to handle a growing amount of work by adding resources to the system
- Scalability is measured over multiple dimensions
- Two types: ***horizontal*** (scale out by adding more nodes) and ***vertical*** (scale up by adding resource to a single node)





SCALABILITY DIMENSIONS

- **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

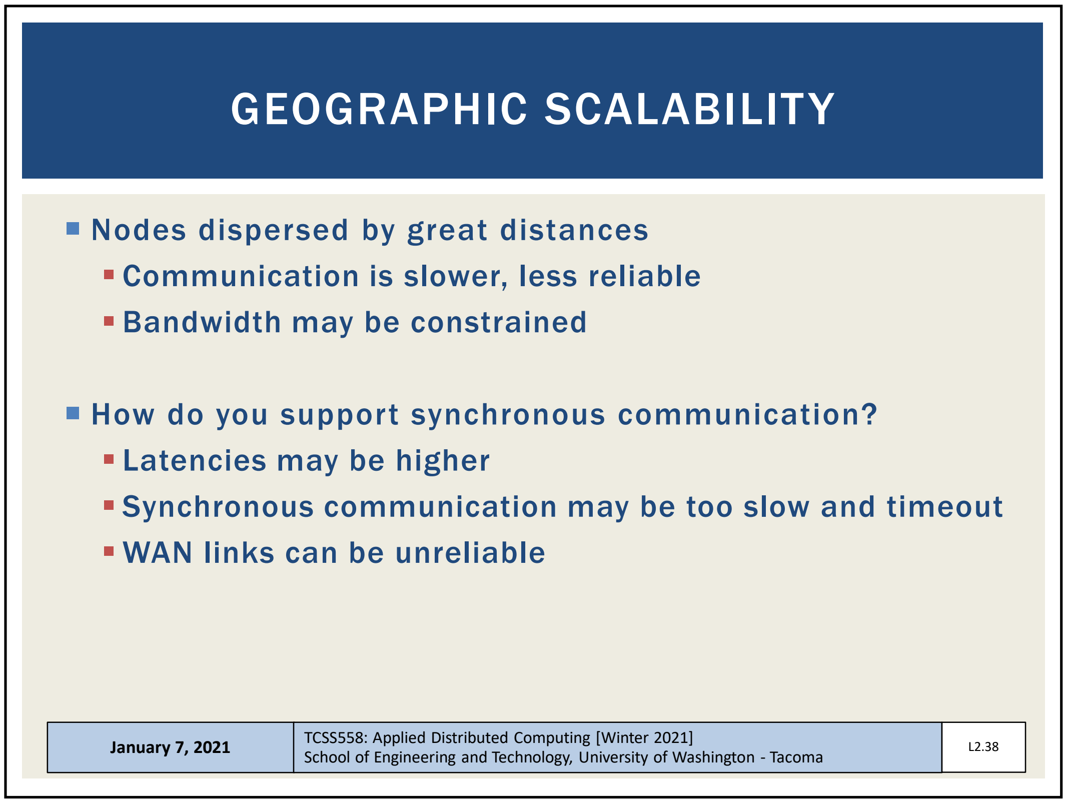


SIZE SCALABILITY

- 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.

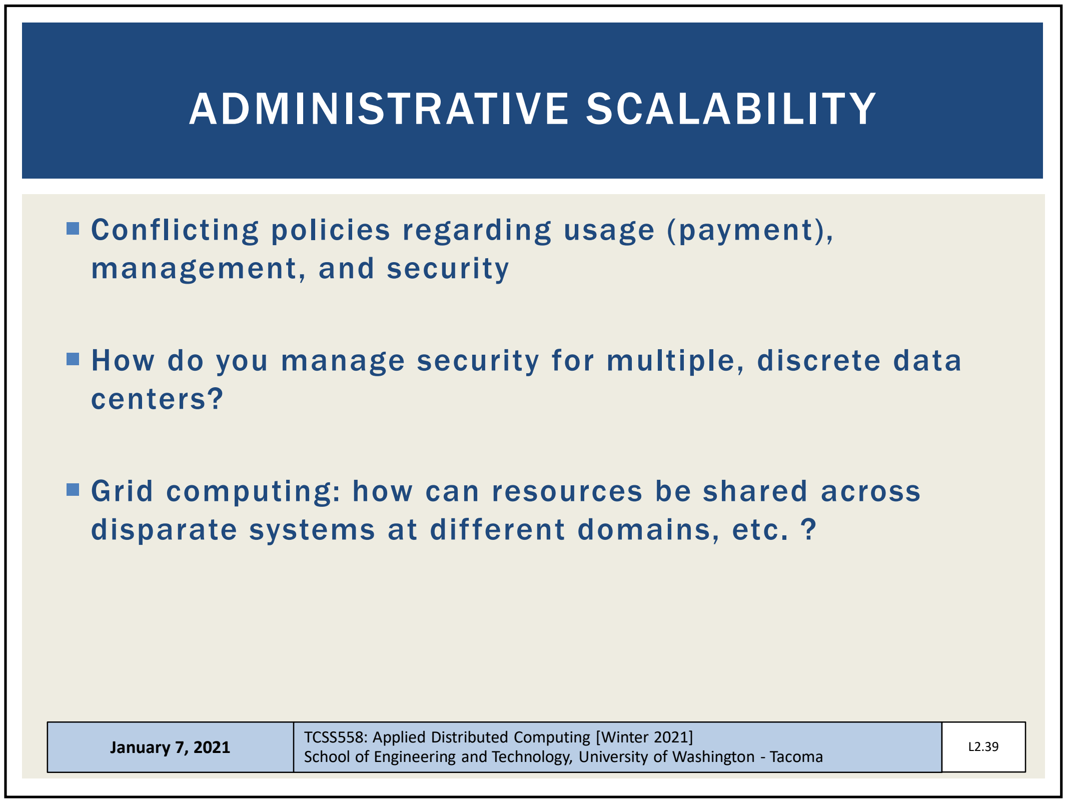
January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.37



GEOGRAPHIC SCALABILITY

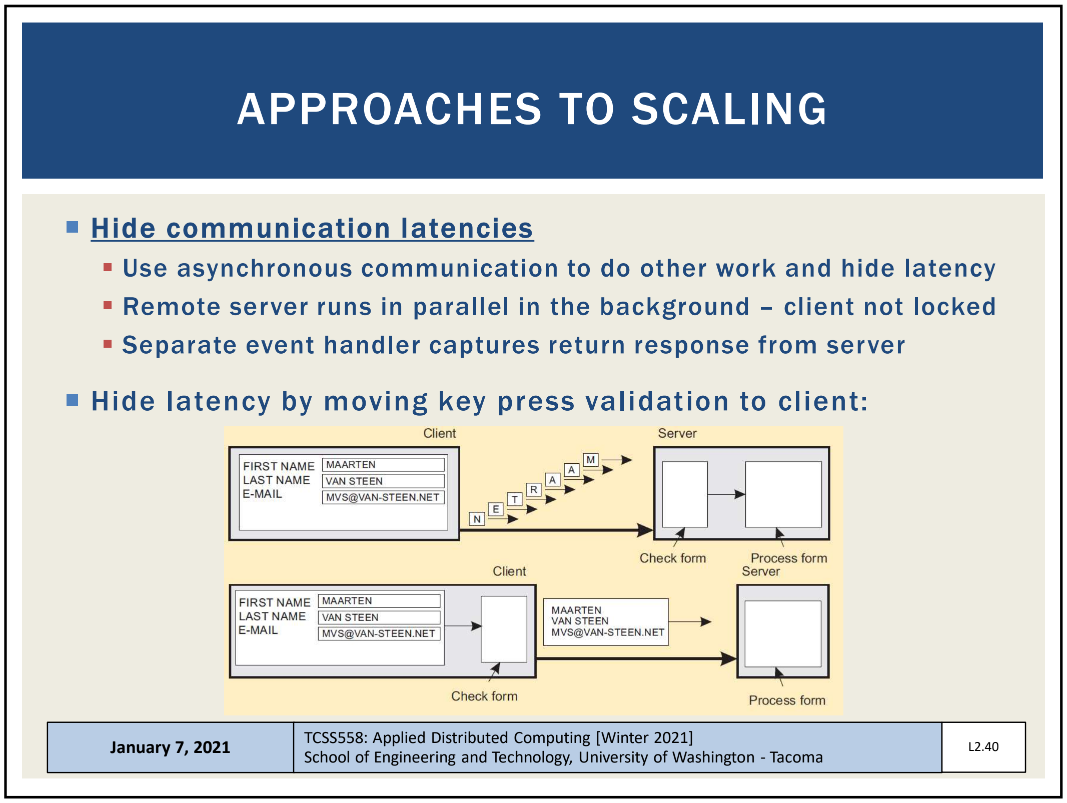
- Nodes dispersed by great distances
  - Communication is slower, less reliable
  - Bandwidth may be constrained
- How do you support synchronous communication?
  - Latencies may be higher
  - Synchronous communication may be too slow and timeout
  - WAN links can be unreliable

January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.38



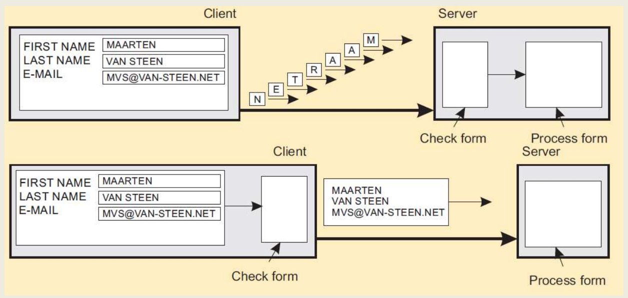
ADMINISTRATIVE SCALABILITY

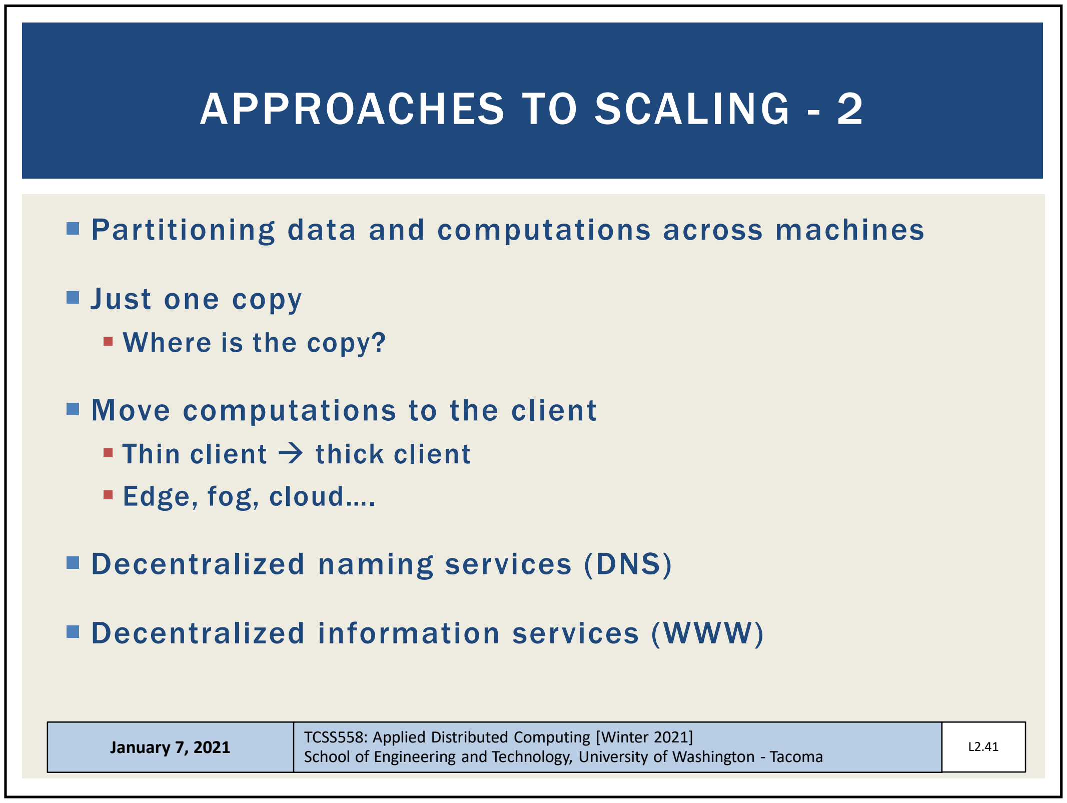
- Conflicting policies regarding usage (payment), management, and security
- How do you manage security for multiple, discrete data centers?
- Grid computing: how can resources be shared across disparate systems at different domains, etc. ?



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:

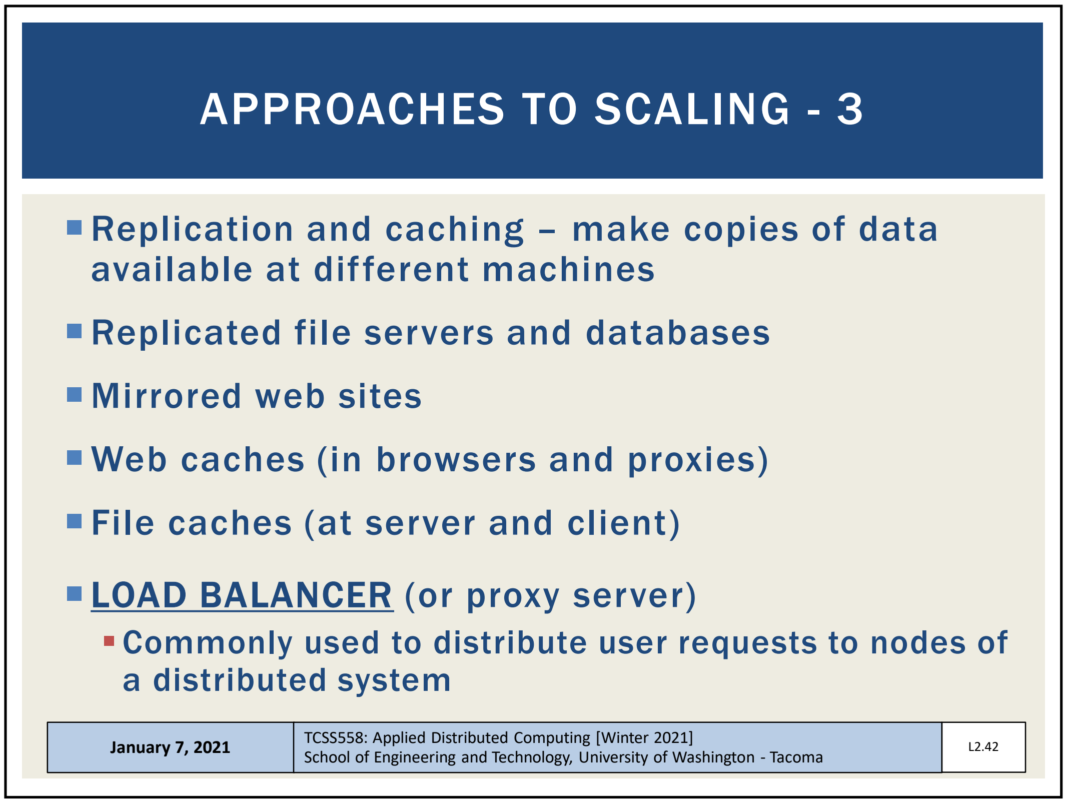




APPROACHES TO SCALING - 2

- 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)

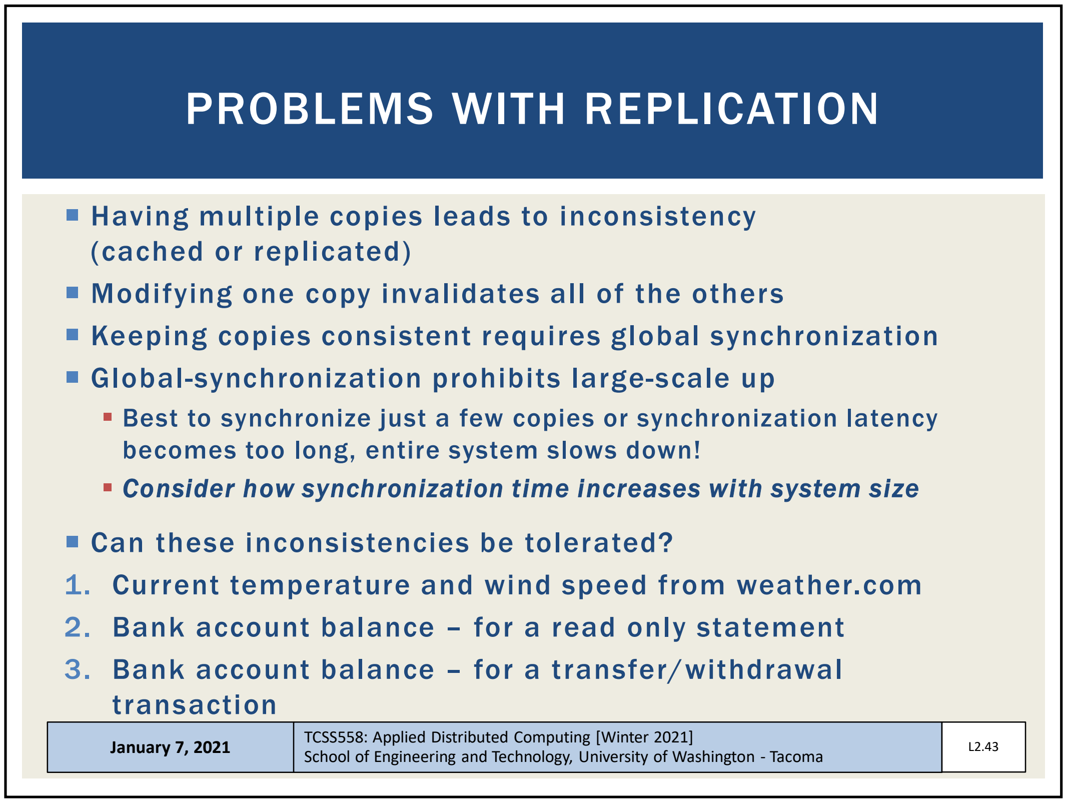
January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.41



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)
- <u>**LOAD BALANCER**</u> (or proxy server)
  - Commonly used to distribute user requests to nodes of a distributed system

January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.42

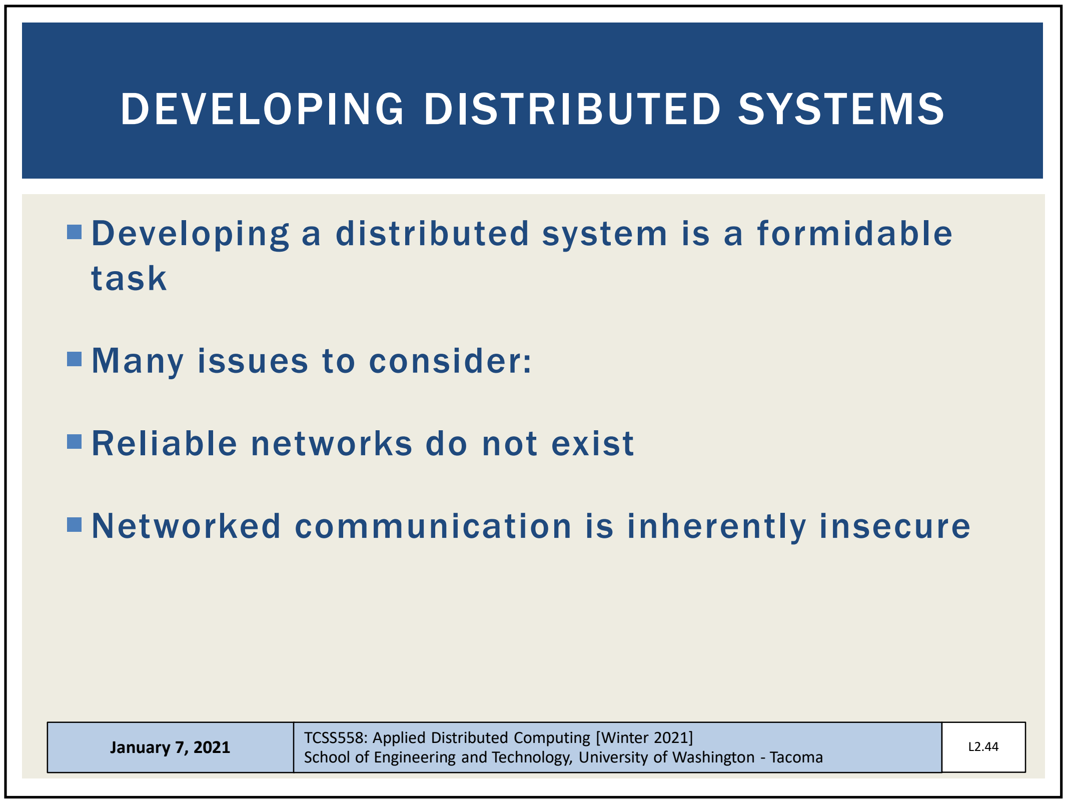


PROBLEMS WITH REPLICATION

- 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

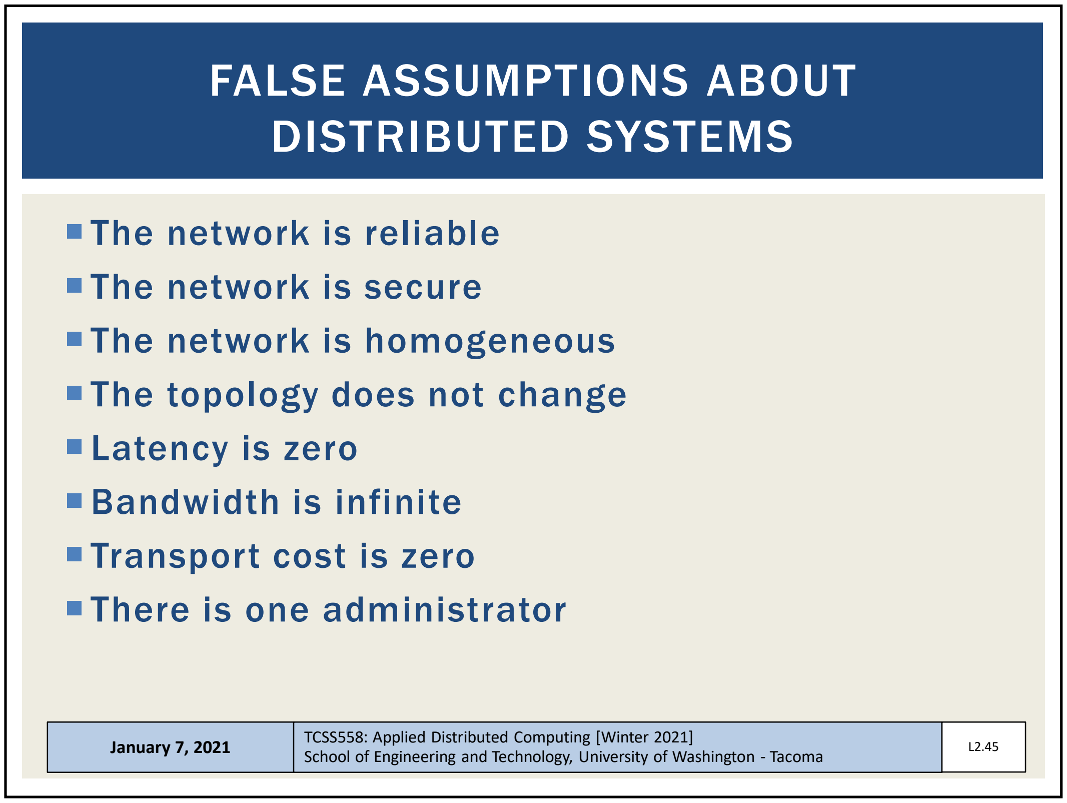
January 7, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.43



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, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L2.44



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

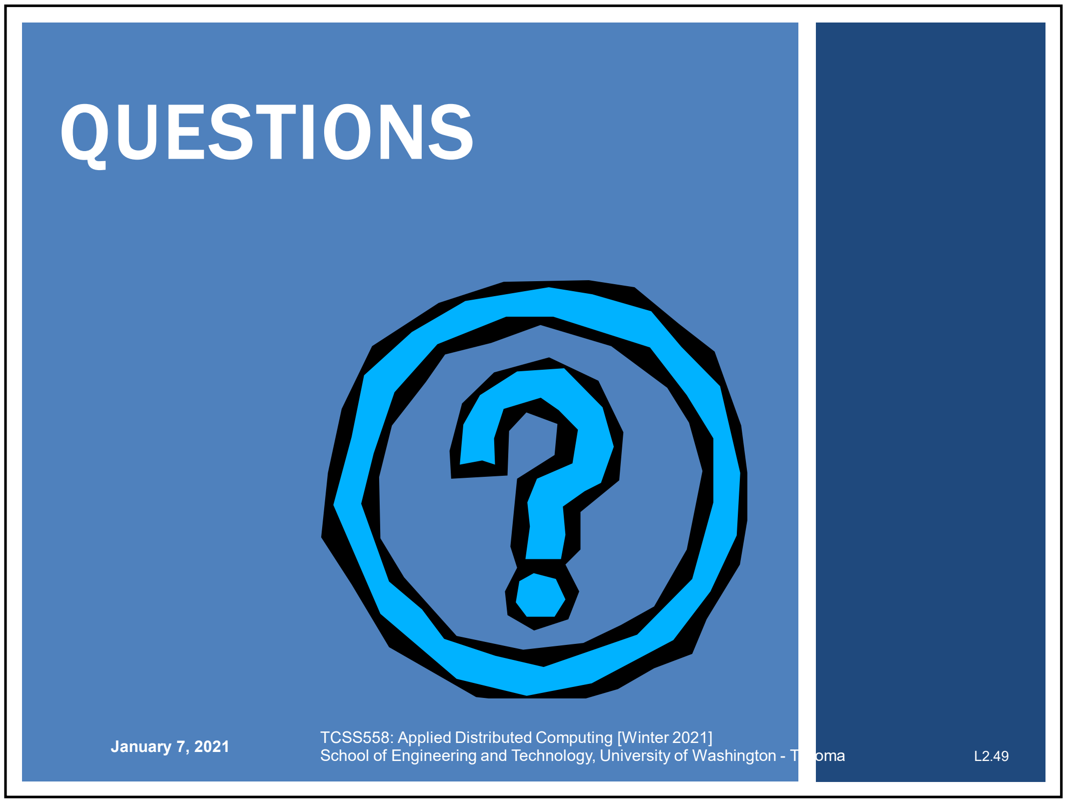
WE WILL RETURN AT 2:35PM

OBJECTIVES – 1/7

- Questions from 1/5
- Chapter 1 - What is a distributed system?
- Design goals of distributed systems:
  - Accessibility: resource sharing & availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems

CLASS ACTIVITY 1

- We will form groups of ~2-3 and enter breakout rooms
- Each group will complete a Google Doc worksheet
- Add names to Google Doc as they appear in Canvas
- Once completed, one person submits a PDF of the Google Doc to Canvas
- Instructor will score all group members based on the uploaded PDF file
- To get started:
  - Log into your UW Google Account
  - Link to shared Google Drive
  - Follow link:
    https://tinyurl.com/y5nmmro9



QUESTIONS

January 7, 2021 TCSS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma L2.49