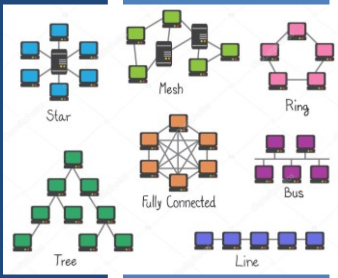
TCSS 558: APPLIED DISTRIBUTED COMPUTING

Introduction to Distributed Systems - II

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



1

OBJECTIVES – 1/9

- Questions from 1/4
- 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

2

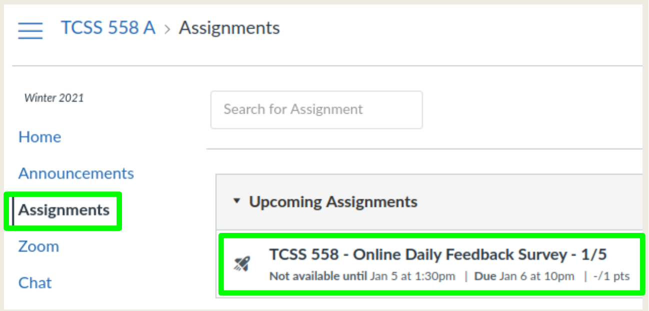
TCSS 558 OFFICE HOURS – WINTER 2024

- 23 of 32 survey responses so far
- Please complete the demographics survey to help set Winter 2024 office hours:
- LINK:
- https://forms.gle/7Vctn8QLrY4kLZvz5

3

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 A > Assignments

Winter 2021

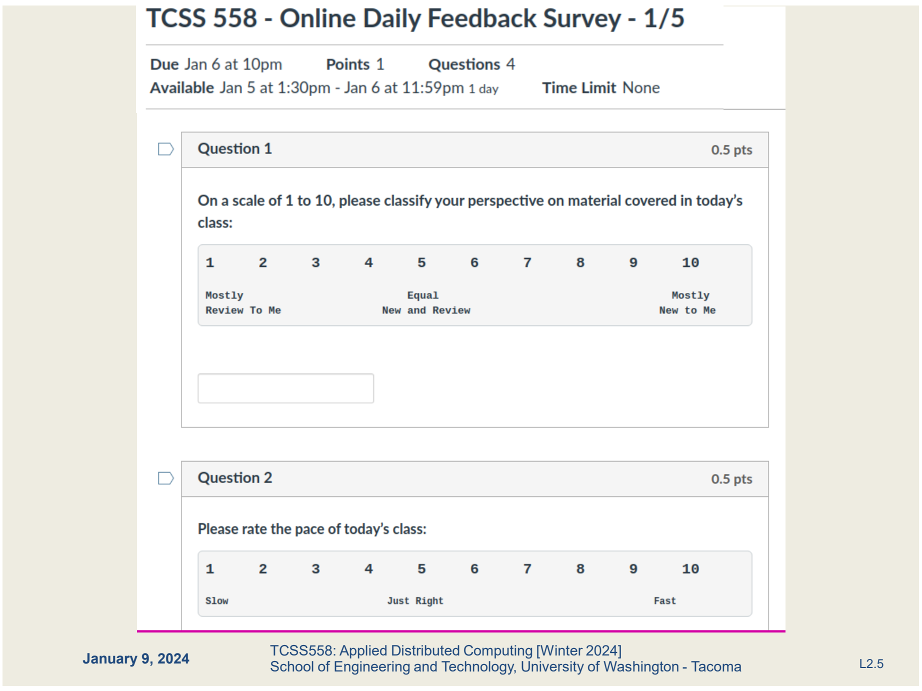
Home
Announcements
Assignments
Zoom
Chat

Search for Assignment

Upcoming Assignments

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

4



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
```

January 9, 2024 TCSS558: Applied Distributed Computing [Winter 2024]
School of Engineering and Technology, University of Washington - Tacoma L2.5

5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (26 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.85** (2023 Lecture 1, 6.65)

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.50** (2023 Lecture 1, 5.91)

January 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.6

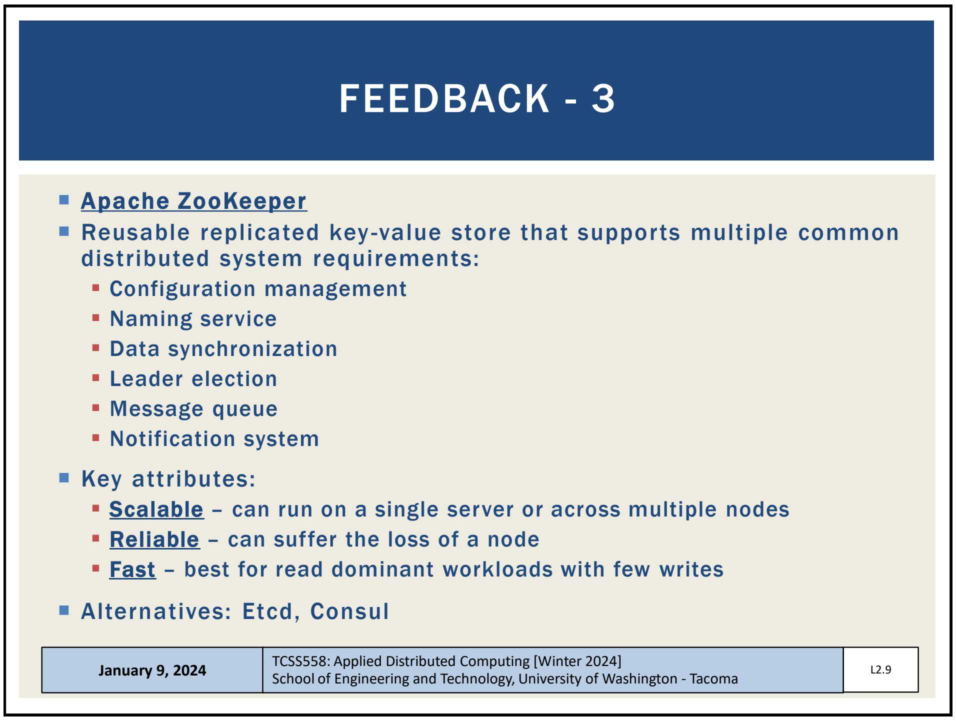
6

FEEDBACK FROM 1/4

- ***What is openness as a design characteristic of a distributed system?***
  - We will cover this today…
- ***Is there an example of something that is like a distributed system, but does not meet the requirements / characterizations ?***
  - A single central server with a hot/warm replica server
  - System provides higher availability than a single server
  - System consists of multiple servers, but only one operates at a time
  - ISSUE: system lacks scalability beyond a single server

FEEDBACK - 2

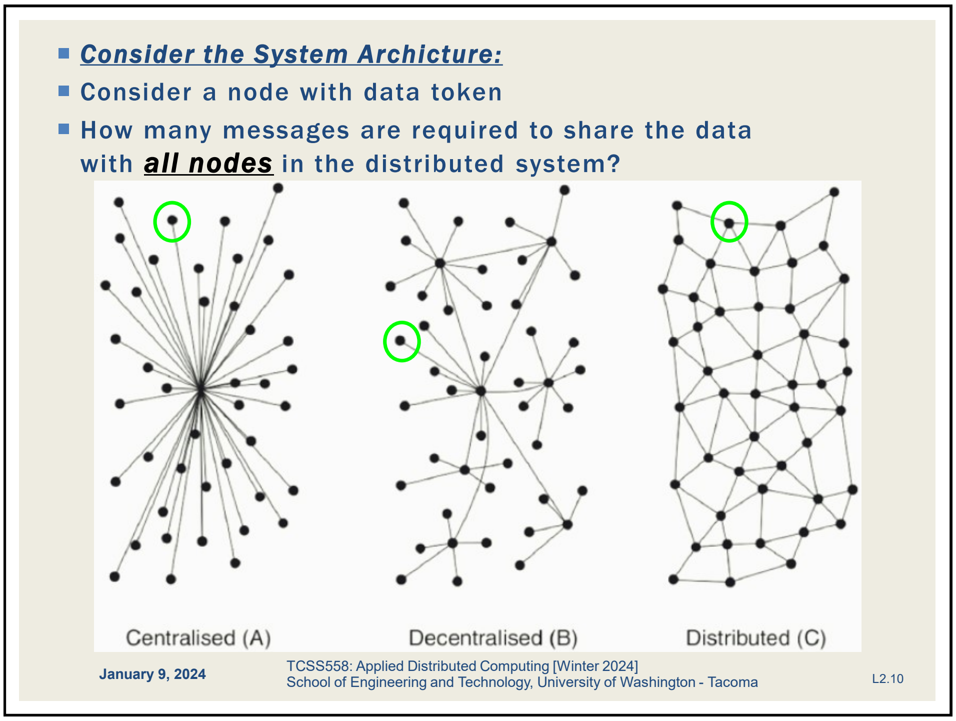
- ***I would like to learn more aboue state/membership tracking.***
- For a distributed system consisting of multiple nodes often there is a need to have the capability to track: ***global system state***
- **EXAMPLE: Apache ZooKeeper:**
  - ZooKeeper provides a reusable implementation for key features required by distributed applications/systems
  - Saves developers from having to 'reinvent the wheel' each time, by offering a common reusable implementation for commonly required distributed systems functions



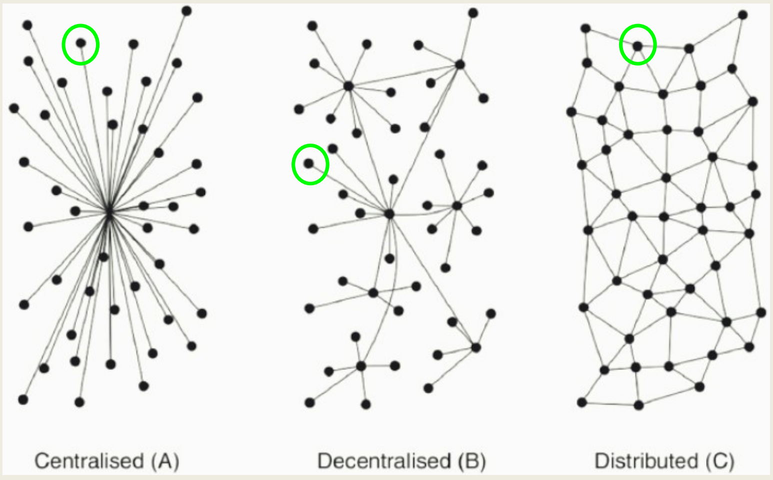
FEEDBACK - 3

- **Apache ZooKeeper**
- Reusable replicated key-value store that supports multiple common distributed system requirements:
  - Configuration management
  - Naming service
  - Data synchronization
  - Leader election
  - Message queue
  - Notification system
- Key attributes:
  - **Scalable** – can run on a single server or across multiple nodes
  - **Reliable** – can suffer the loss of a node
  - **Fast** – best for read dominant workloads with few writes
- Alternatives: Etcd, Consul

9



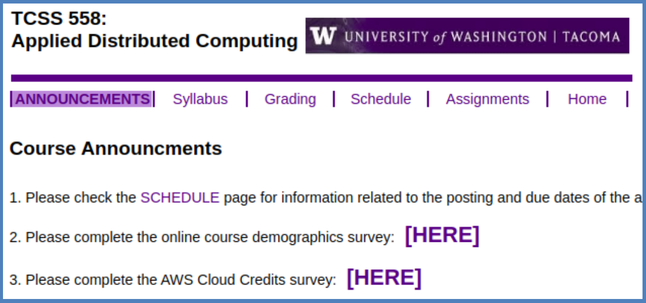
- ***Consider the System Archicture:***
- Consider a node with data token
- How many messages are required to share the data with ***all nodes*** in the distributed system?



10

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]**

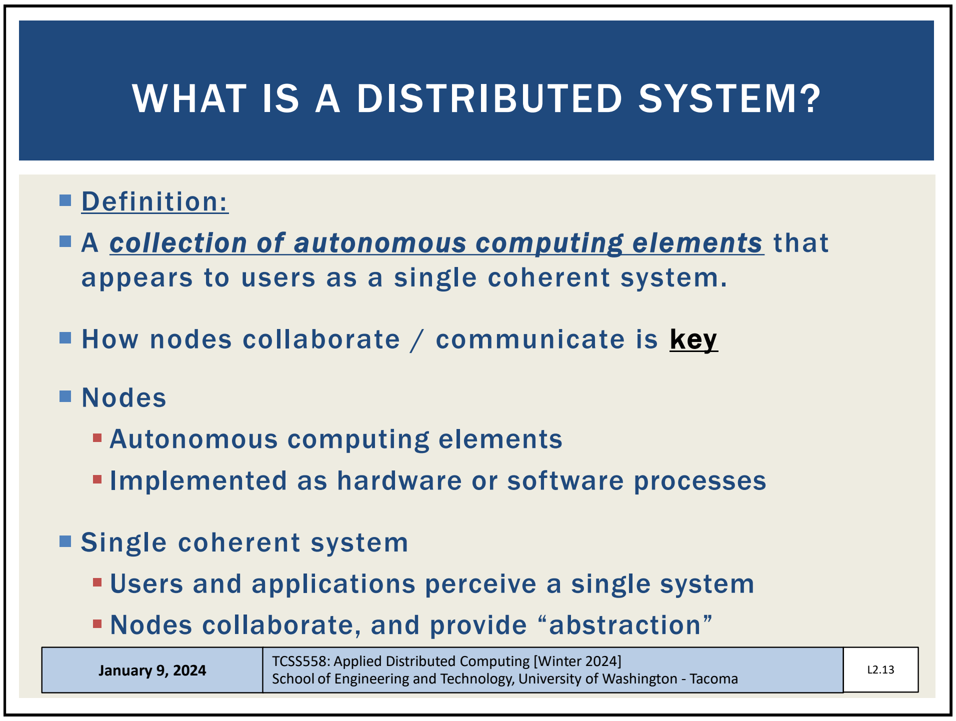
3. Please complete the AWS Cloud Credits survey: **[HERE]**

11

OBJECTIVES – 1/9

- Questions from 1/4
- **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

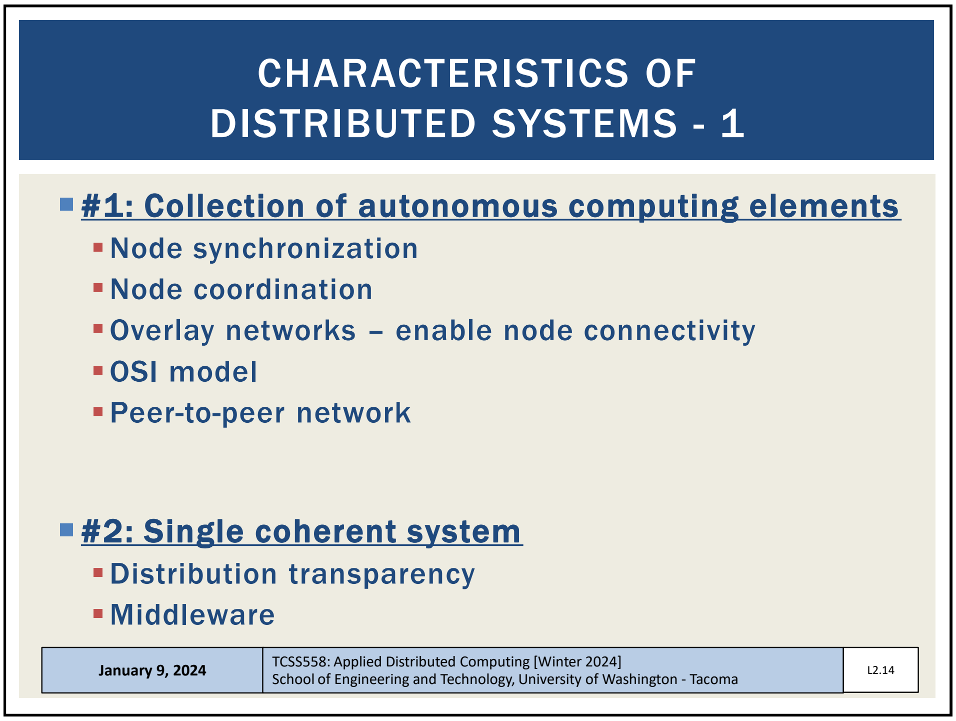
12



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"

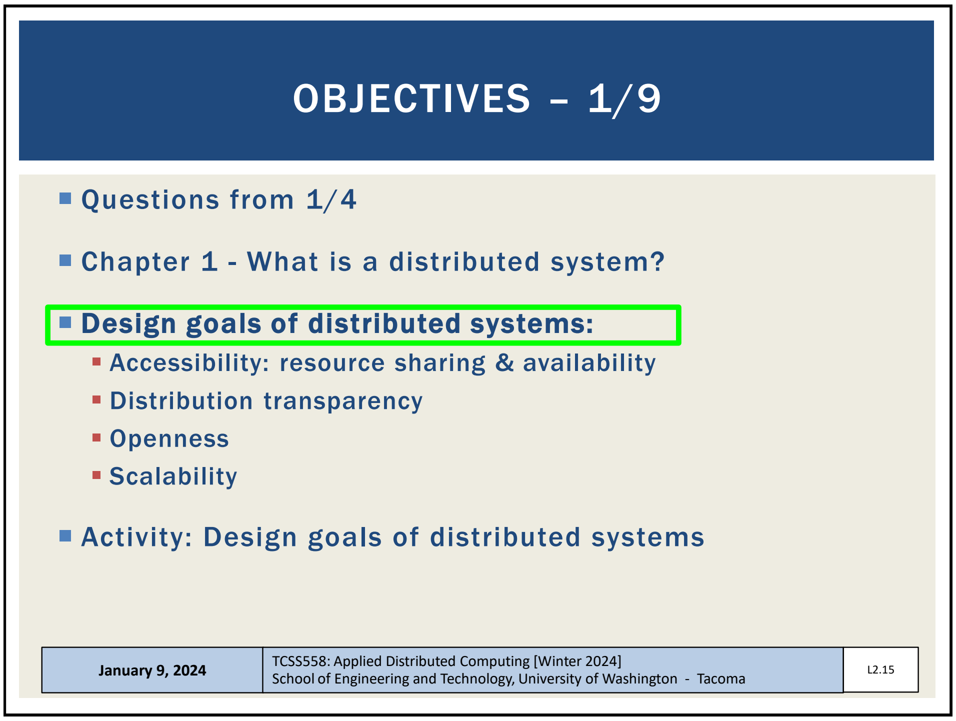
13



CHARACTERISTICS OF DISTRIBUTED SYSTEMS - 1

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

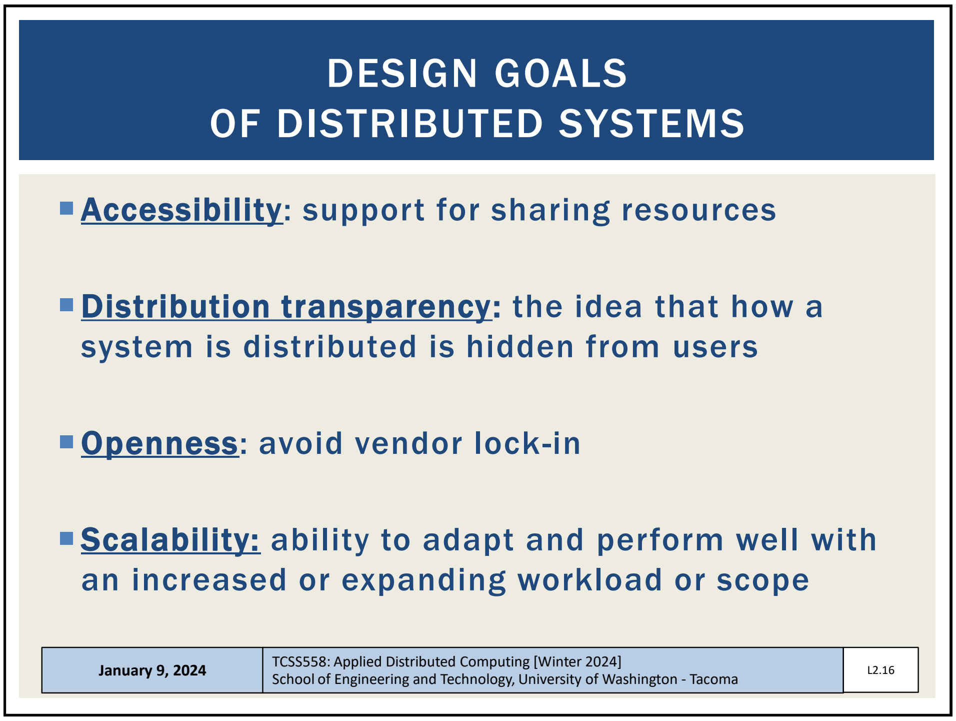
14



OBJECTIVES – 1/9

- Questions from 1/4
- 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

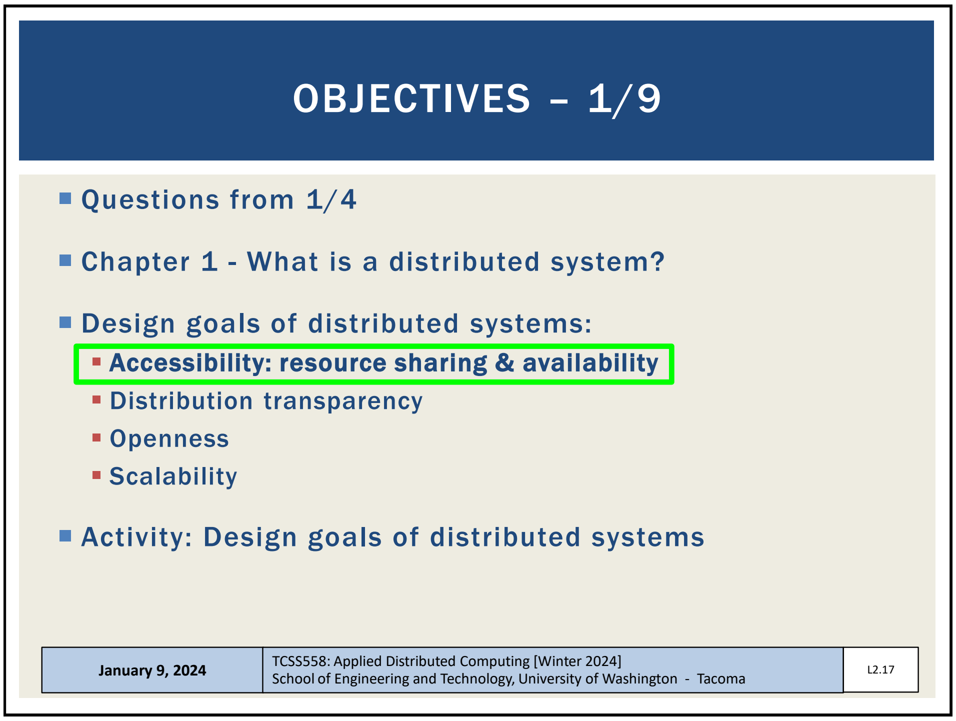
15



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

16

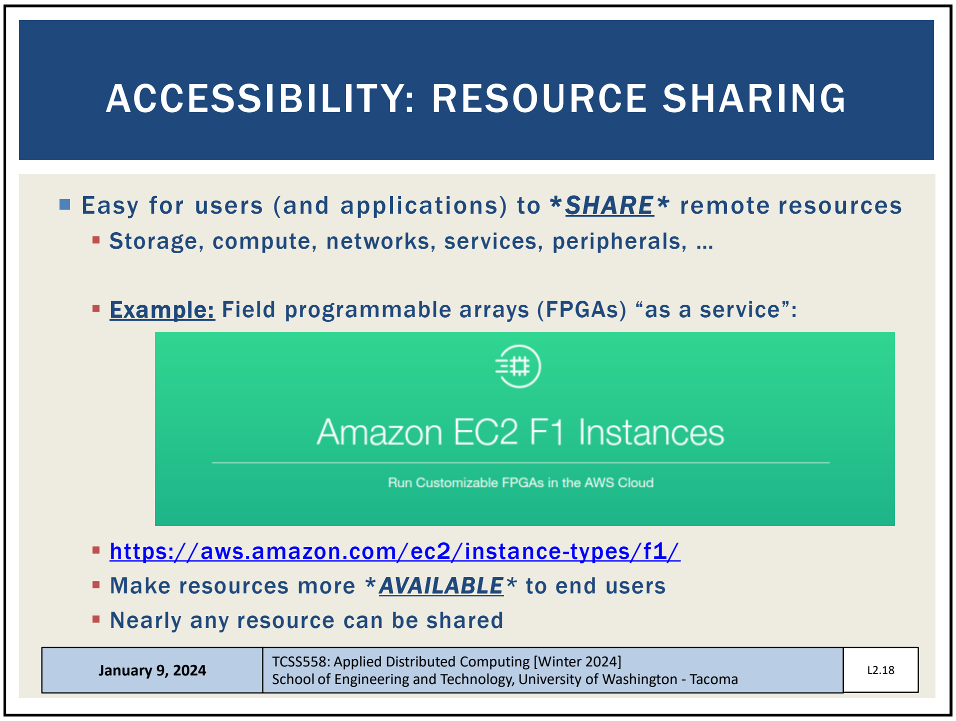


OBJECTIVES – 1/9

- Questions from 1/4
- 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 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.17

17



ACCESSIBILITY: RESOURCE SHARING

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

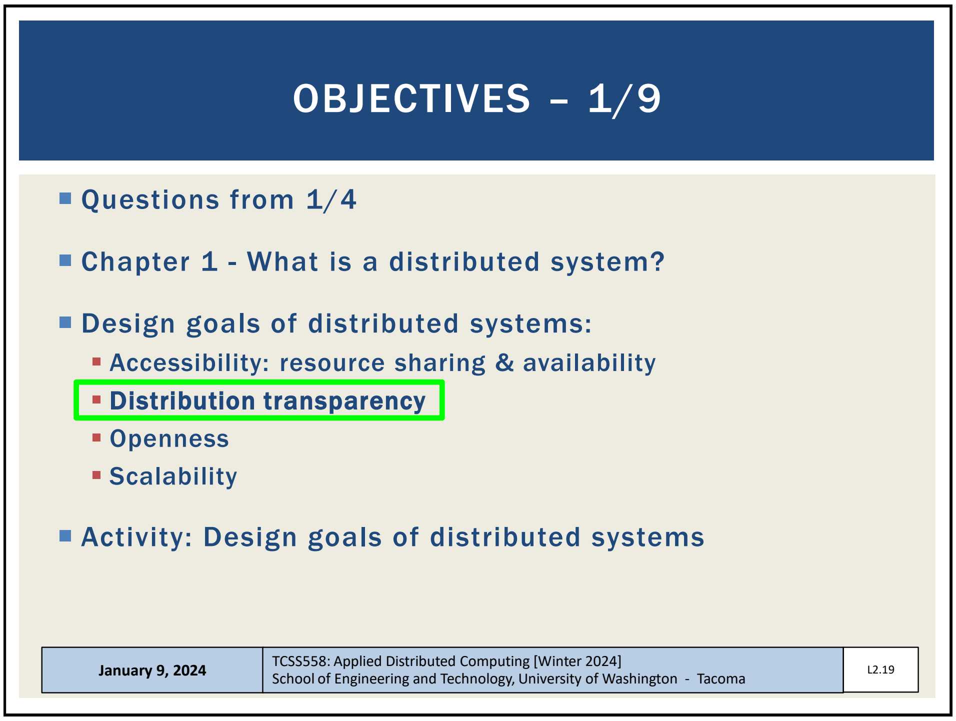
Amazon EC2 F1 Instances

Run Customizable FPGAs in the AWS Cloud

  - https://aws.amazon.com/ec2/instance-types/f1/
  - Make resources more ***AVAILABLE*** to end users
  - Nearly any resource can be shared

January 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.18

18

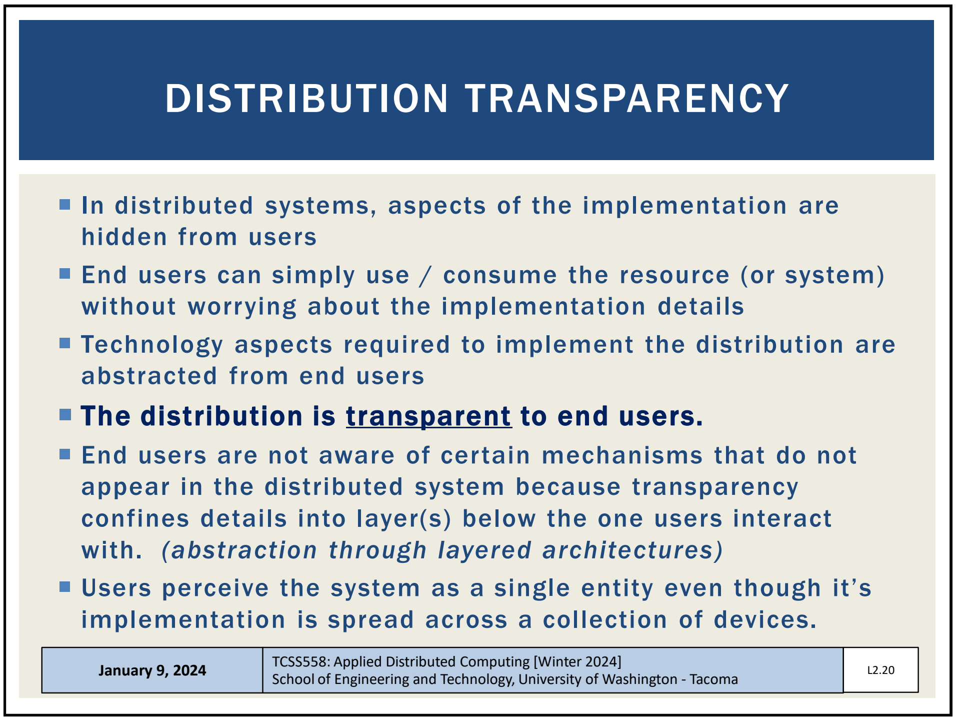


OBJECTIVES – 1/9

- Questions from 1/4
- 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 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.19

19

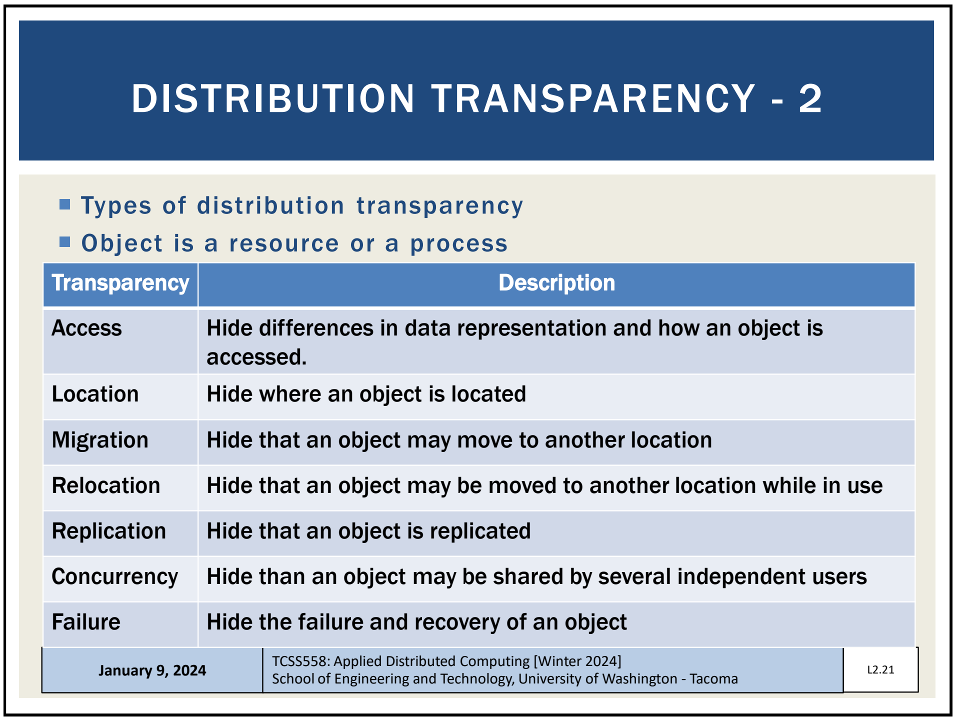


DISTRIBUTION TRANSPARENCY

- 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.

January 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.20

20

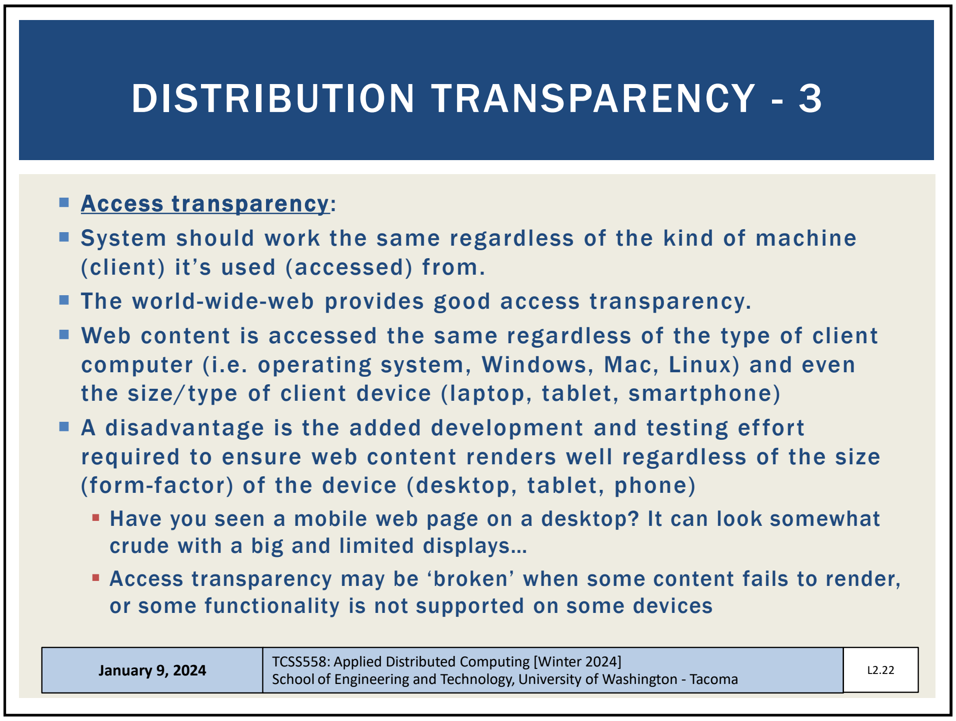


DISTRIBUTION TRANSPARENCY - 2

- 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
Migration	Hide that an object may move to another location
Relocation	Hide that an object may be moved to another location while in use
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

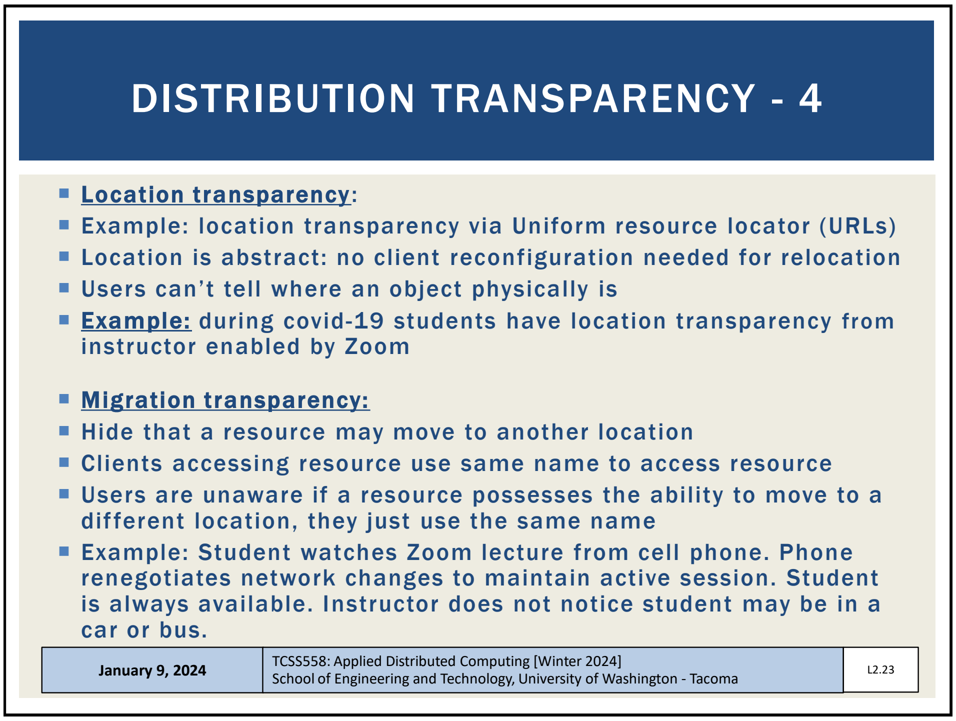
21



DISTRIBUTION TRANSPARENCY - 3

- **Access transparency**:
- System should work the same regardless of the kind of machine (client) it's used (accessed) from.
- The world-wide-web provides good access transparency.
- Web content is accessed the same regardless of the type of client computer (i.e. operating system, Windows, Mac, Linux) and even the size/type of client device (laptop, tablet, smartphone)
- A disadvantage is the added development and testing effort required to ensure web content renders well regardless of the size (form-factor) of the device (desktop, tablet, phone)
  - Have you seen a mobile web page on a desktop? It can look somewhat crude with a big and limited displays…
  - Access transparency may be 'broken' when some content fails to render, or some functionality is not supported on some devices

22



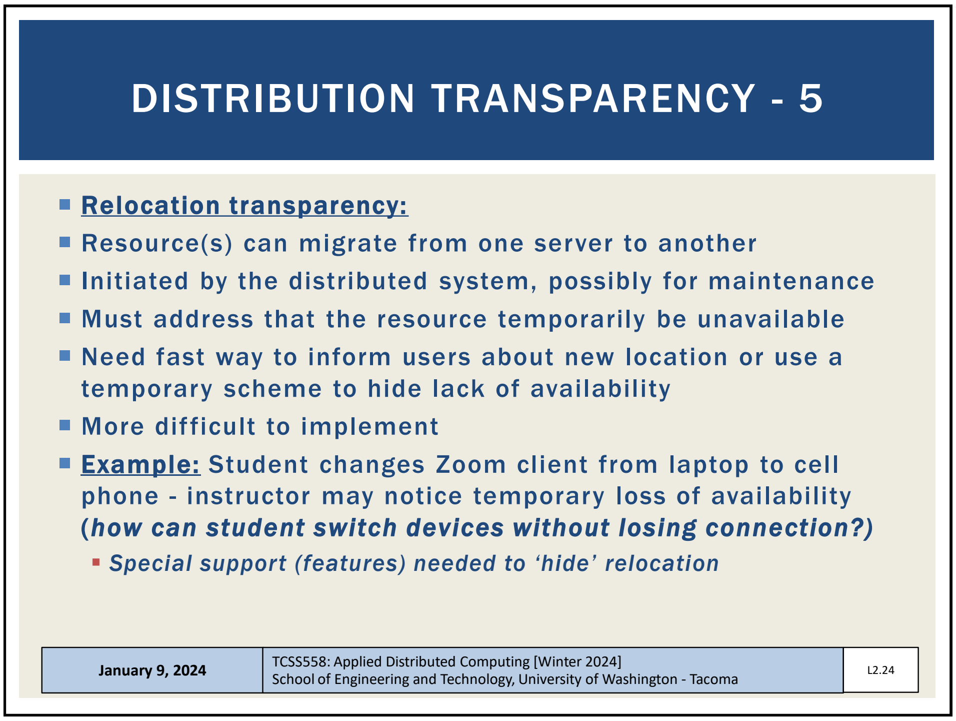
DISTRIBUTION TRANSPARENCY - 4

- **Location transparency**:
- Example: location transparency via 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

- **Migration transparency:**
- Hide that a resource may move to another location
- Clients accessing resource use same name to access resource
- Users are unaware if a resource possesses the ability to move to a different location, they just use the same name
- Example: Student watches Zoom lecture from cell phone. Phone renegotiates network changes to maintain active session. Student is always available. Instructor does not notice student may be in a car or bus.

January 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.23

23

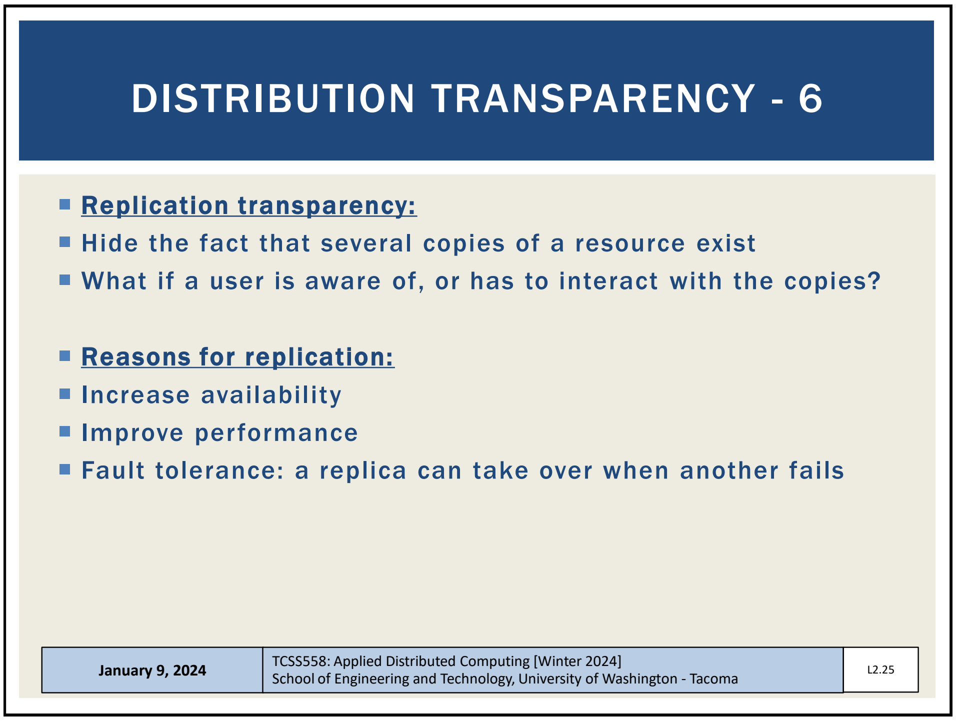


DISTRIBUTION TRANSPARENCY - 5

- **Relocation transparency:**
- Resource(s) can migrate from one server to another
- Initiated by the distributed system, possibly for maintenance
- Must address that the resource temporarily be unavailable
- Need fast way to inform users about new location or use a temporary scheme to hide lack of availability
- More difficult to implement
- **Example:** Student changes Zoom client from laptop to cell phone - instructor may notice temporary loss of availability (***how can student switch devices without losing connection?)***
  - *Special support (features) needed to 'hide' relocation*

January 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.24

24

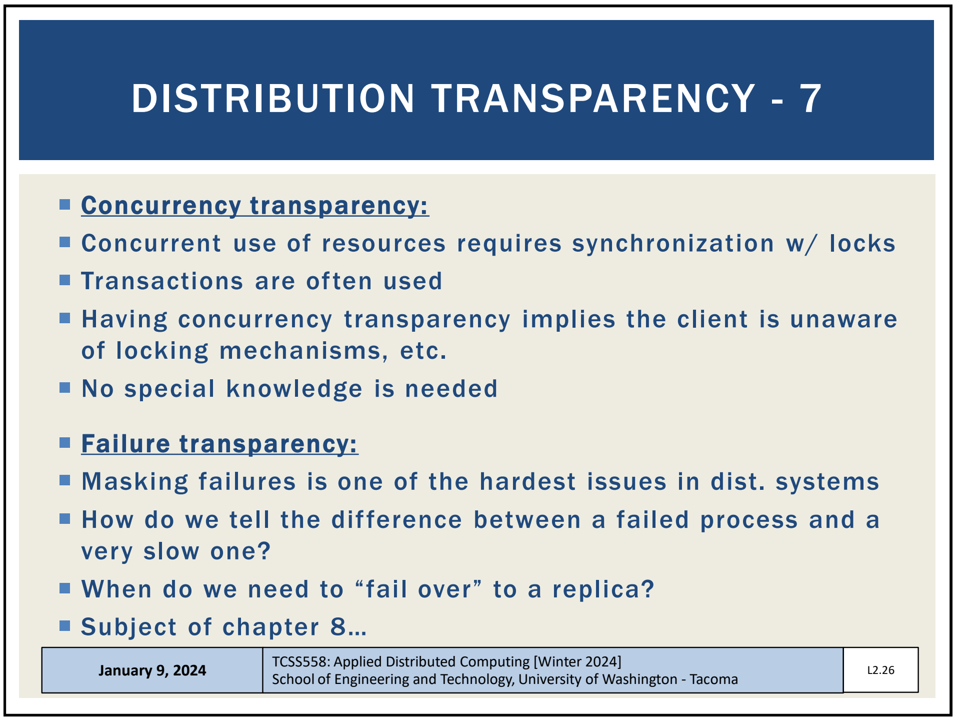


DISTRIBUTION TRANSPARENCY - 6

- **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

25

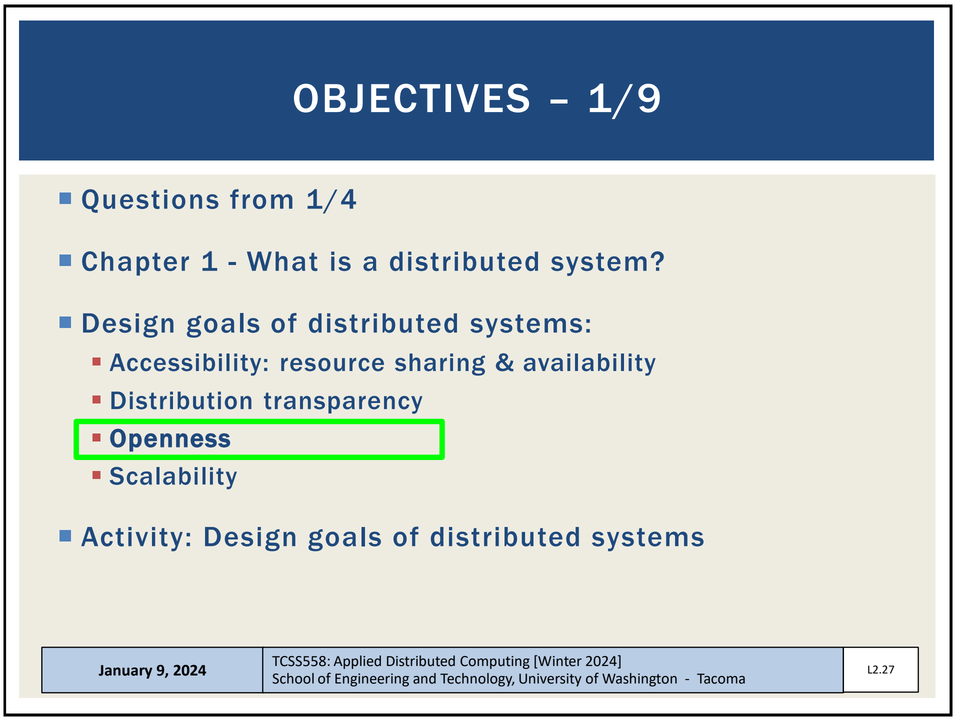


DISTRIBUTION TRANSPARENCY - 7

- **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…

26

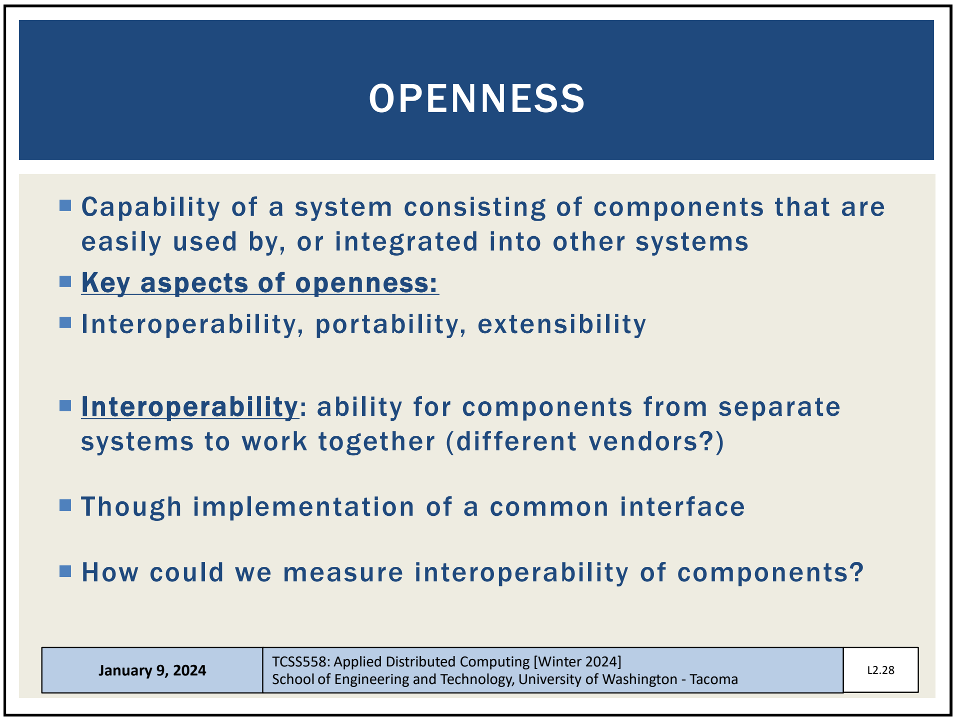


OBJECTIVES – 1/9

- Questions from 1/4
- 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 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.27

27

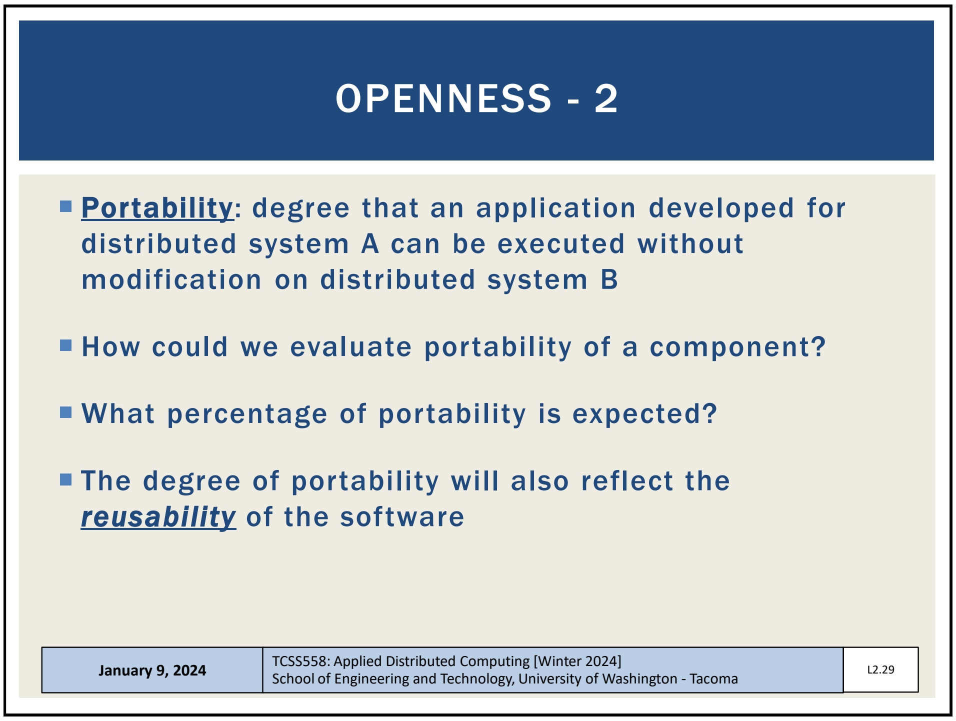


OPENNESS

- 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 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.28

28

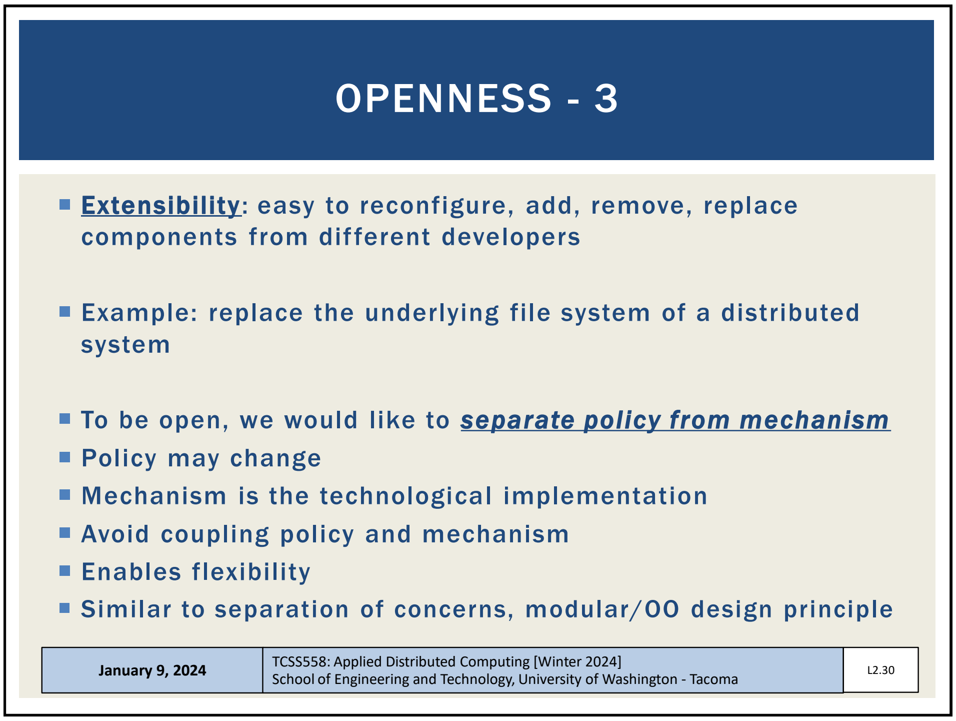


OPENNESS - 2

- **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

January 9, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L2.29

29

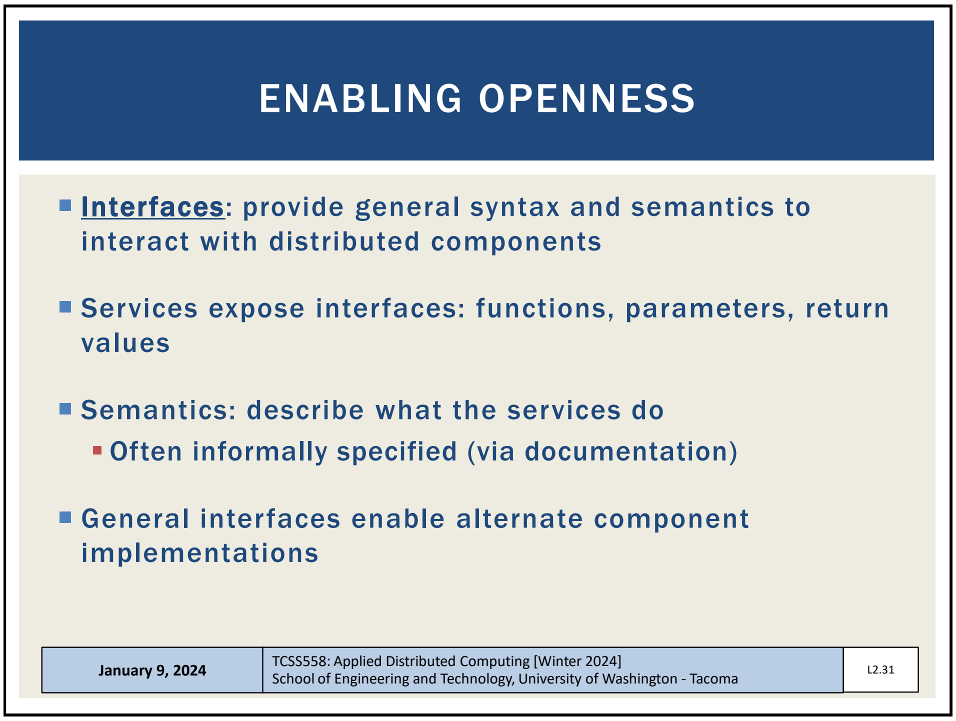


OPENNESS - 3

- **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

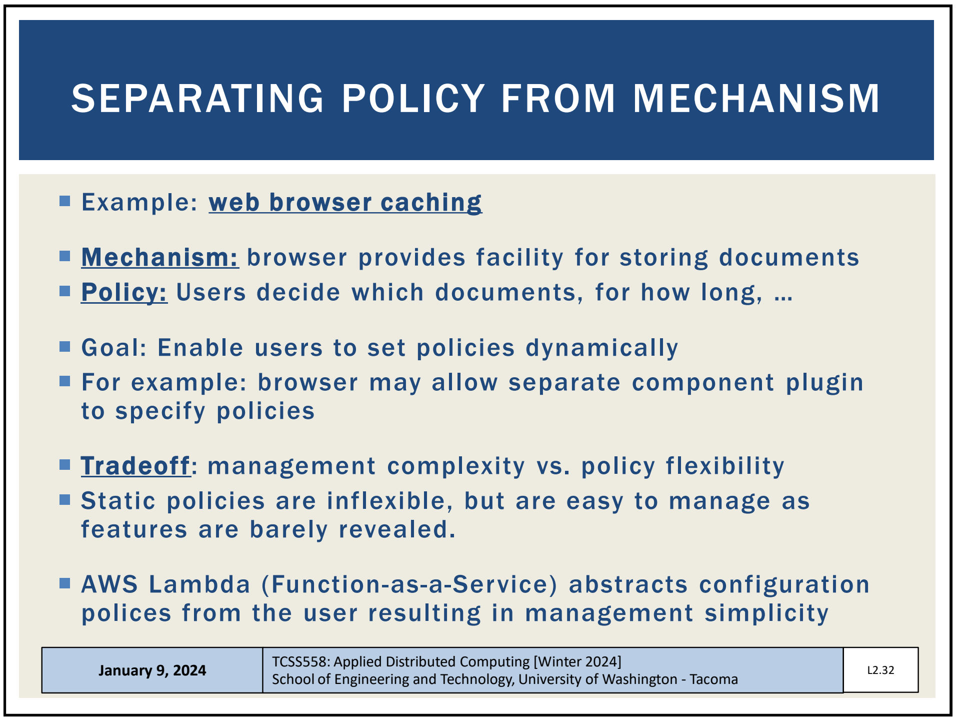
January 9, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L2.30

30



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



SEPARATING POLICY FROM MECHANISM

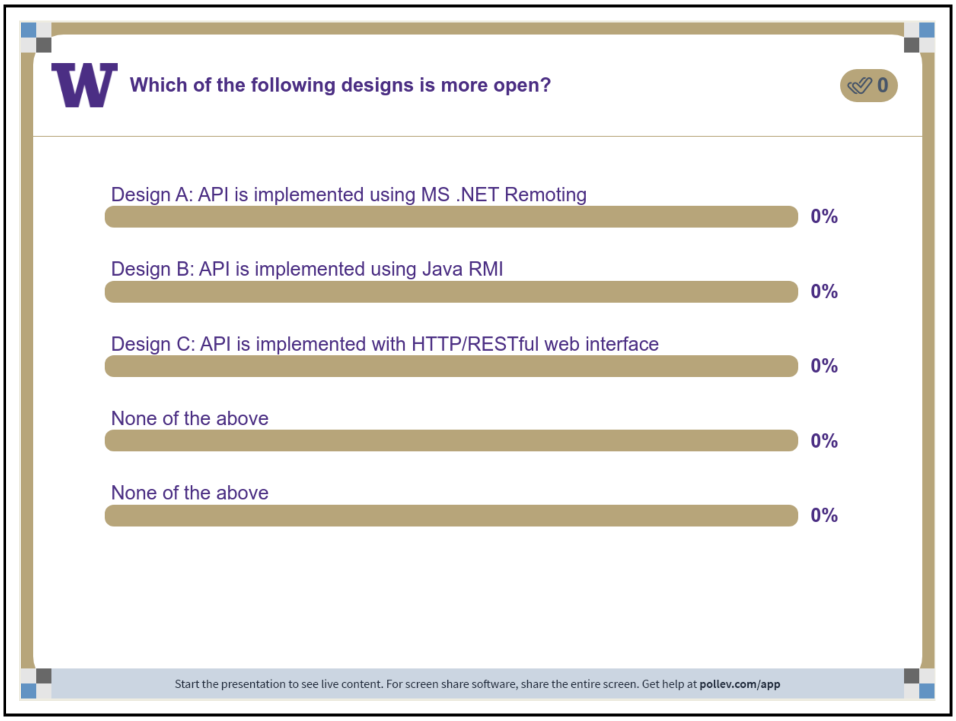
- Example: **web browser caching**
- **Mechanism:** browser provides facility for storing documents
- **Policy:** Users decide which documents, for how long, …
- Goal: Enable users to set policies dynamically
- For example: browser may allow separate component plugin to specify policies
- **Tradeoff**: 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

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… weatherbit ?)
- **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.

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



Which of the following designs is more open?

0

Design A: API is implemented using MS .NET Remoting 0%

Design B: API is implemented using Java RMI 0%

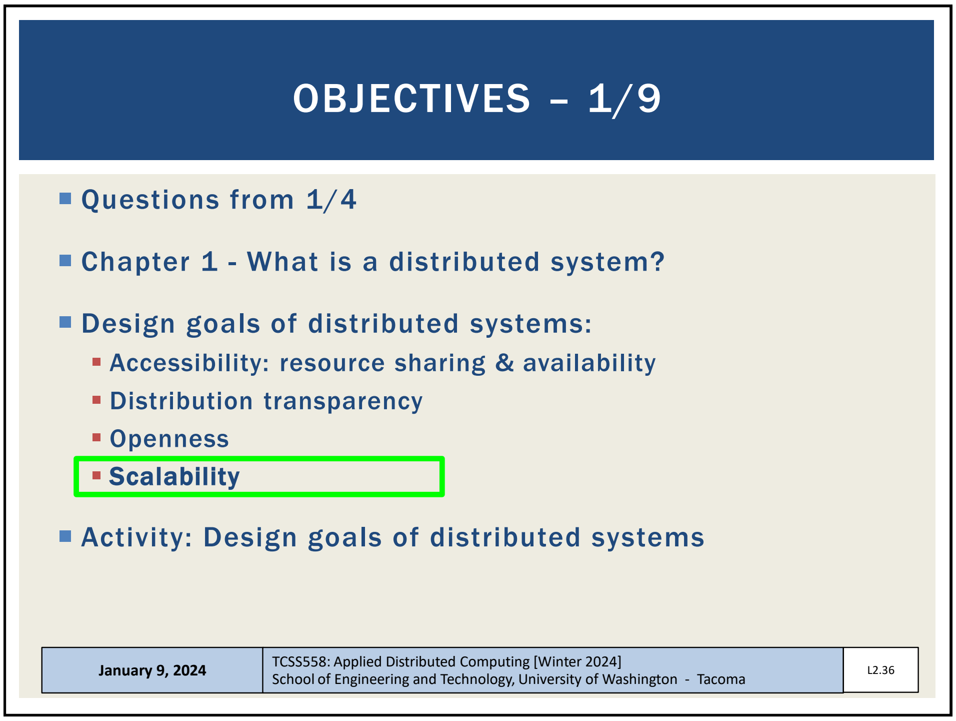
Design C: API is implemented with HTTP/RESTful web interface 0%

None of the above 0%

None of the above 0%

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

35



OBJECTIVES – 1/9

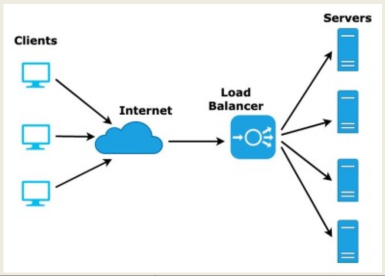
- Questions from 1/4
- Chapter 1 - What is a distributed system?
- Design goals of distributed systems:
  - Accessibility: resource sharing & availability
  - Distribution transparency
  - Openness
  - **Scalability**
- Activity: Design goals of distributed systems

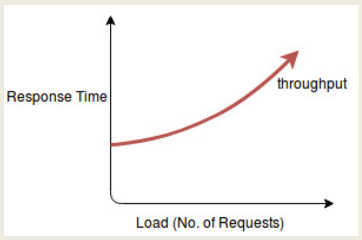
January 9, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L2.36

36

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)



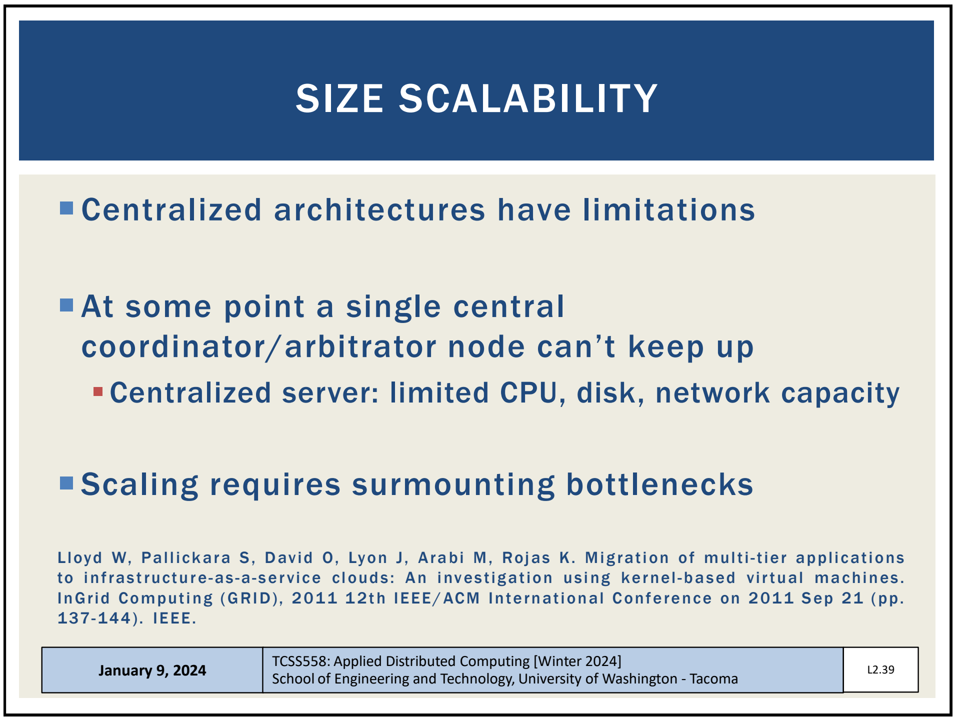


37

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

38



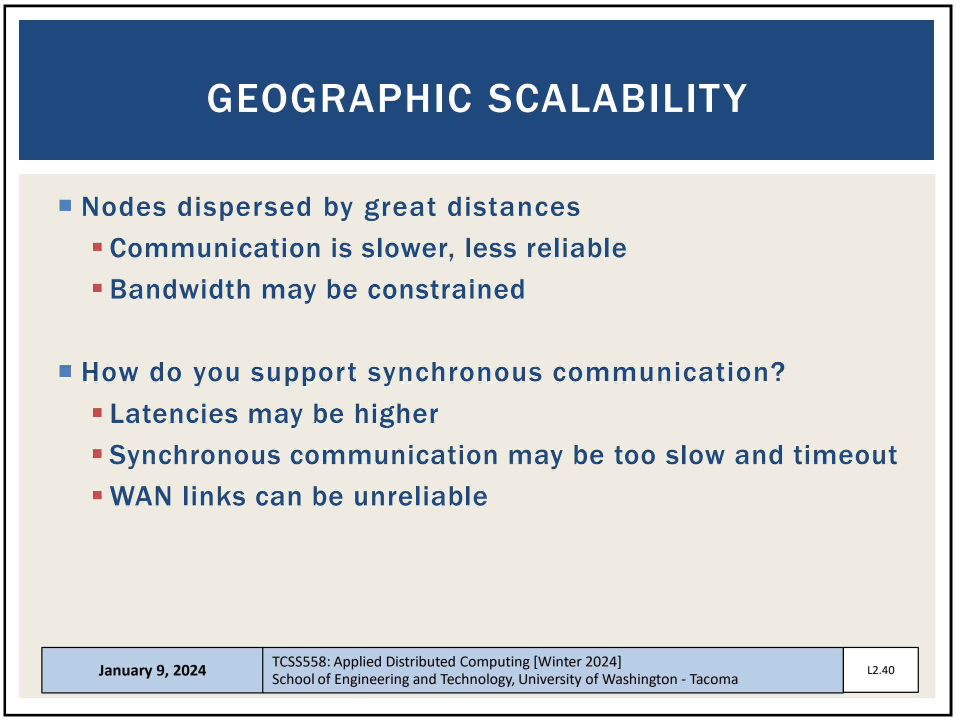
SIZE SCALABILITY

- 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 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.39

39



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 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.40

40

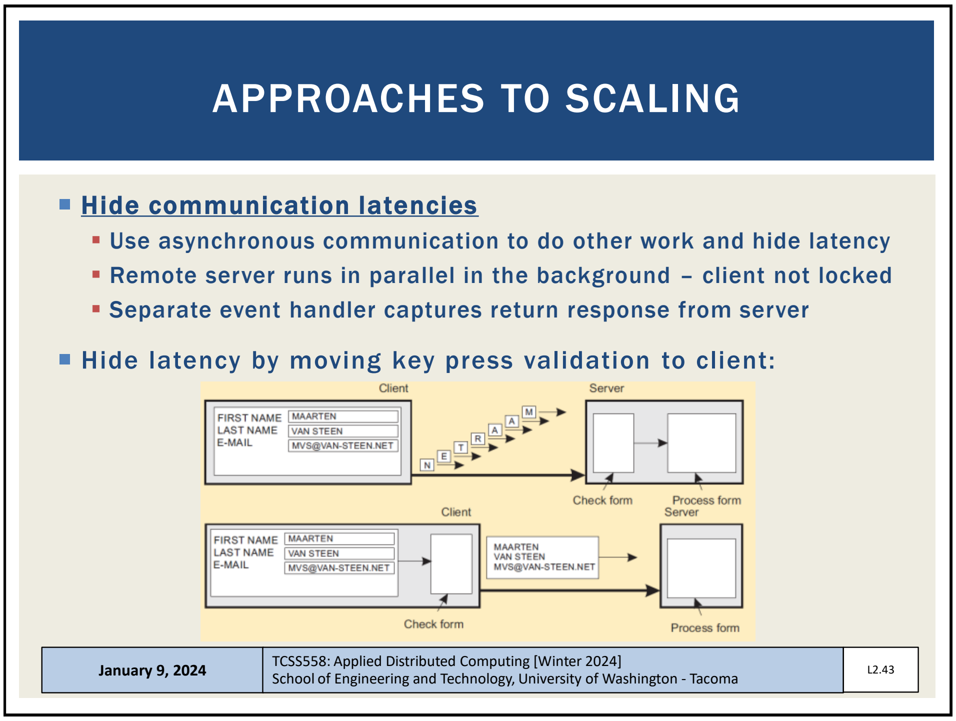
ADMINISTRATIVE SCALABILITY

- 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. ?

41

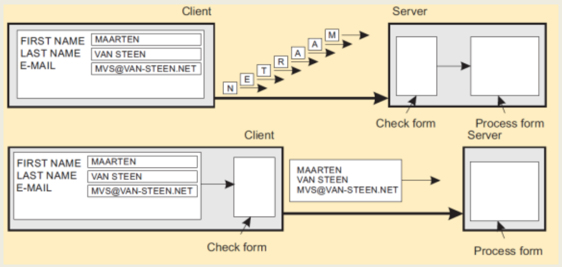
WE WILL RETURN AT 4:50PM

42



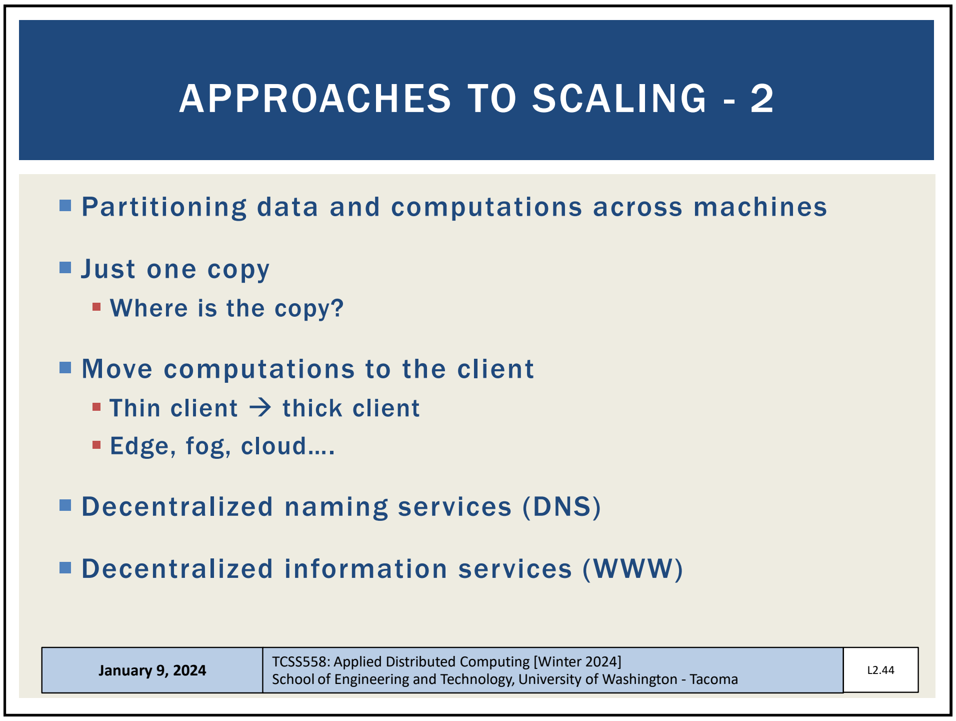
APPROACHES TO SCALING

- **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 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.43

43



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 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.44

44

APPROACHES TO SCALING - 3

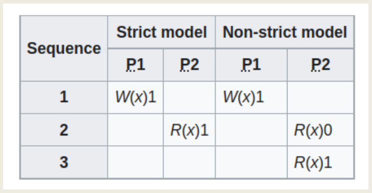
- **Replication** and **caching** – make copies of data available at different machines
- Replicated file servers and databases
- Mirrored web sites
- Web caches (in browsers and proxies)
- File caches (at server and client)
- **LOAD BALANCER** (or proxy server)
  - Commonly used to distribute user requests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

PROBLEMS WITH REPLICATION

- Having multiple copies of data leads to inconsistency (cached or replicated copies)
- 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 and distance between nodes***
- To address synchronization time, distributed systems designers will relax data consistency guarantees…

TYPES OF CONSISTENCY

- **Consistency Models:** Contract between the programmer and a system, where the system guarantees if the programmer follows rules for operations on (distributed/replicated) data, data will be consistent and the results of reading, writing, or updating memory will be predictable
- **Strict Consistency** – a write to a variable by any node needs to be seen instantaneously by all nodes



Sequence	Strict model		Non-strict model	
	P1	P2	P1	P2
1	W(x)1		W(x)1	
2		R(x)1		R(x)0
3				R(x)1

- Non-strict model, for P2: x can be 0 or 1

January 9, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L2.47

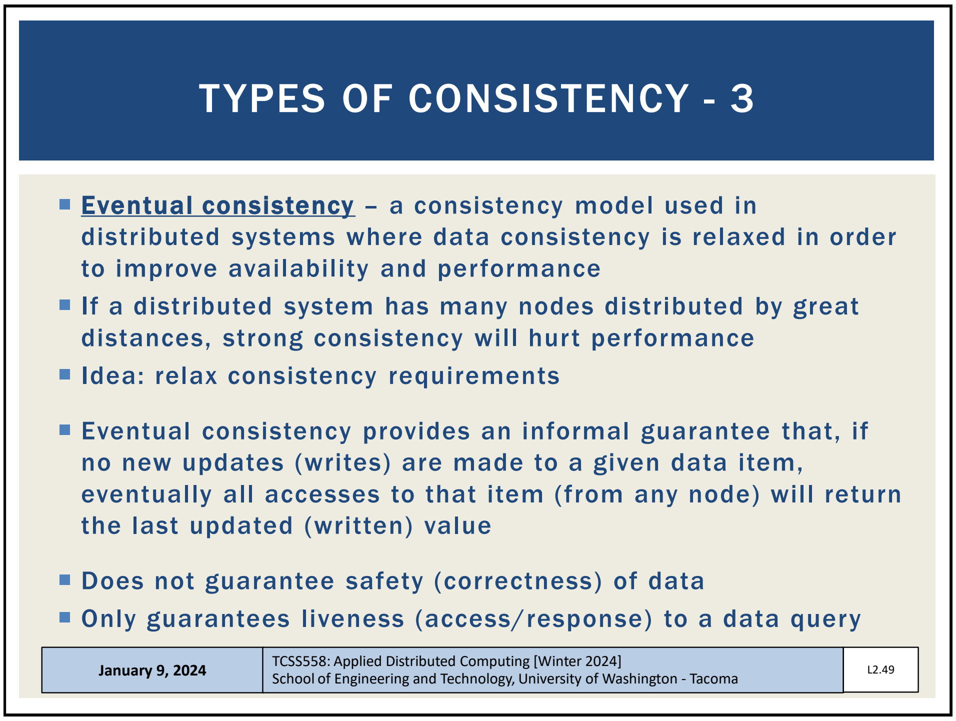
47

TYPES OF CONSISTENCY - 2

- **Strong consistency** – data is locked during writes and replication across nodes. No other nodes can access data for read/write when locked
- Sacrifices availability and performance for consistency
- Amazon Simple Storage Service (S3) now offers strong consistency

January 9, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L2.48

48

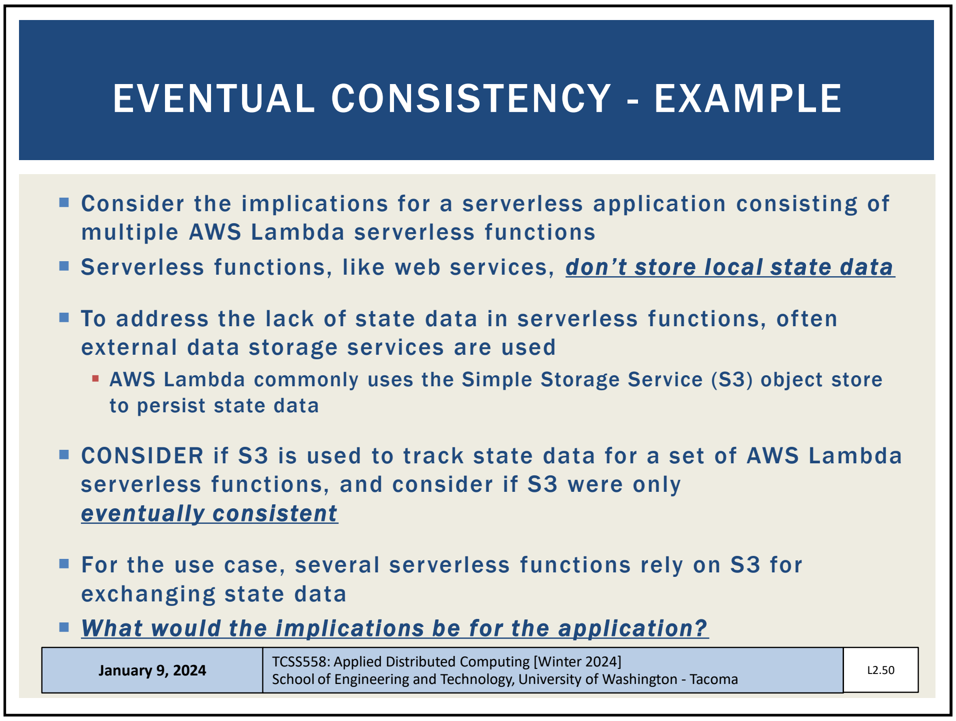


TYPES OF CONSISTENCY - 3

- **Eventual consistency** – a consistency model used in distributed systems where data consistency is relaxed in order to improve availability and performance
- If a distributed system has many nodes distributed by great distances, strong consistency will hurt performance
- Idea: relax consistency requirements
- Eventual consistency provides an informal guarantee that, if no new updates (writes) are made to a given data item, eventually all accesses to that item (from any node) will return the last updated (written) value
- Does not guarantee safety (correctness) of data
- Only guarantees liveness (access/response) to a data query

January 9, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L2.49

49

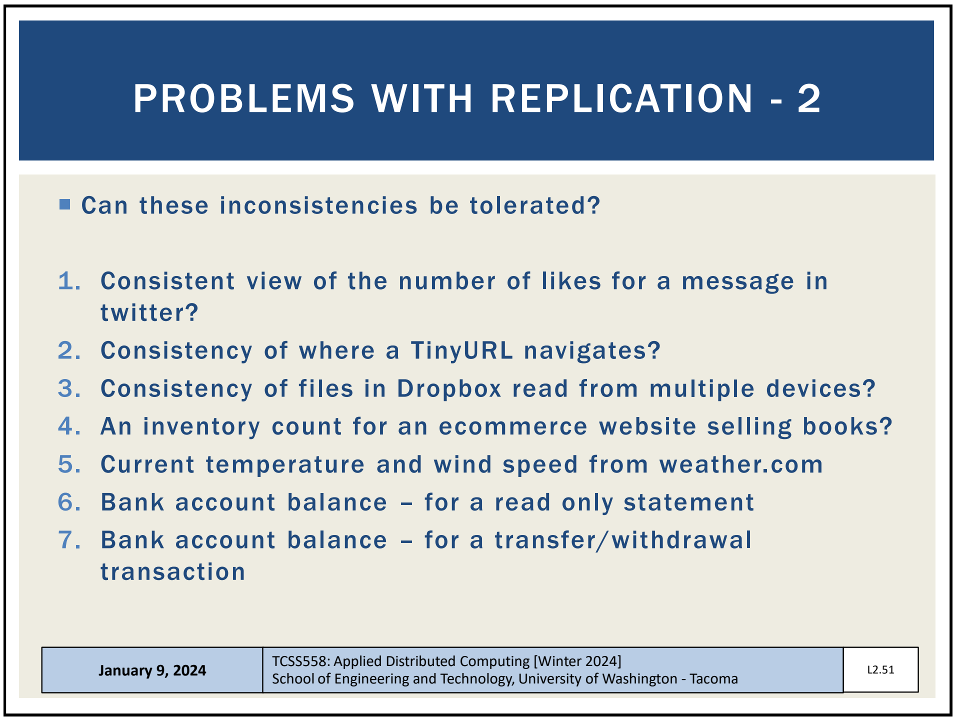


EVENTUAL CONSISTENCY - EXAMPLE

- Consider the implications for a serverless application consisting of multiple AWS Lambda serverless functions
- Serverless functions, like web services, ***don't store local state data***
- To address the lack of state data in serverless functions, often external data storage services are used
  - AWS Lambda commonly uses the Simple Storage Service (S3) object store to persist state data
- CONSIDER if S3 is used to track state data for a set of AWS Lambda serverless functions, and consider if S3 were only ***eventually consistent***
- For the use case, several serverless functions rely on S3 for exchanging state data
- ***What would the implications be for the application?***

January 9, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L2.50

50



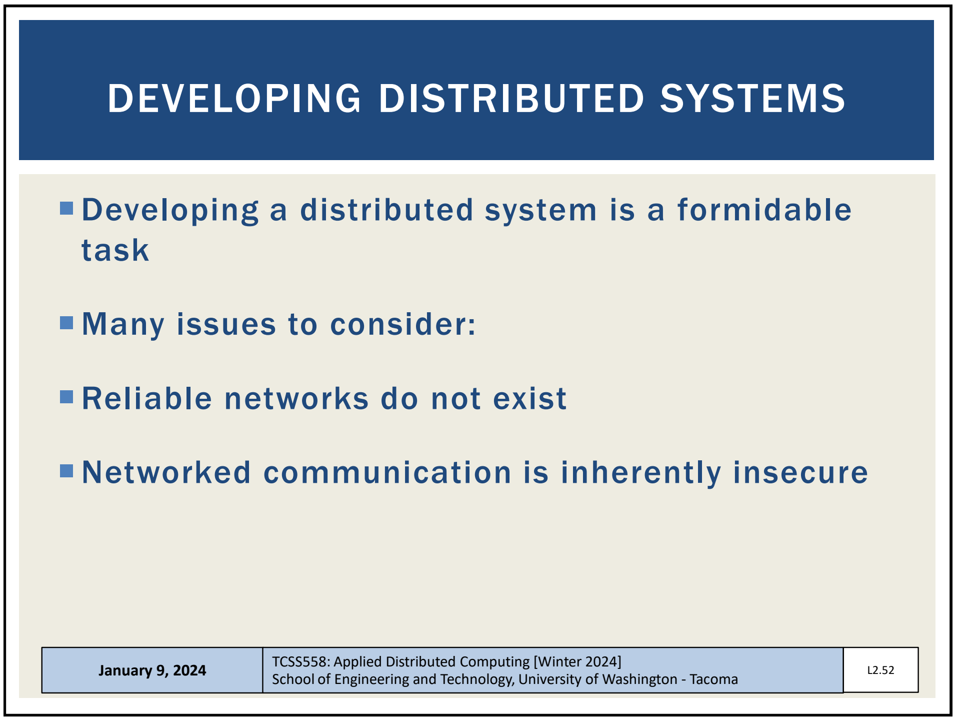
PROBLEMS WITH REPLICATION - 2

- Can these inconsistencies be tolerated?

1. Consistent view of the number of likes for a message in twitter?
2. Consistency of where a TinyURL navigates?
3. Consistency of files in Dropbox read from multiple devices?
4. An inventory count for an ecommerce website selling books?
5. Current temperature and wind speed from weather.com
6. Bank account balance – for a read only statement
7. Bank account balance – for a transfer/withdrawal transaction

January 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.51

51

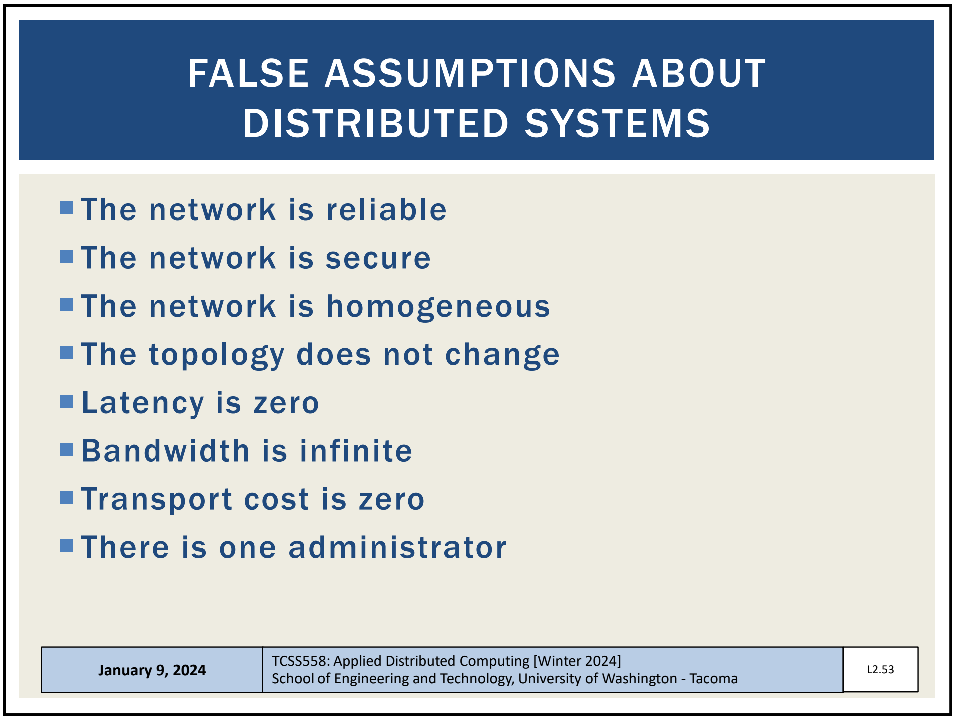


DEVELOPING DISTRIBUTED SYSTEMS

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

January 9, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L2.52

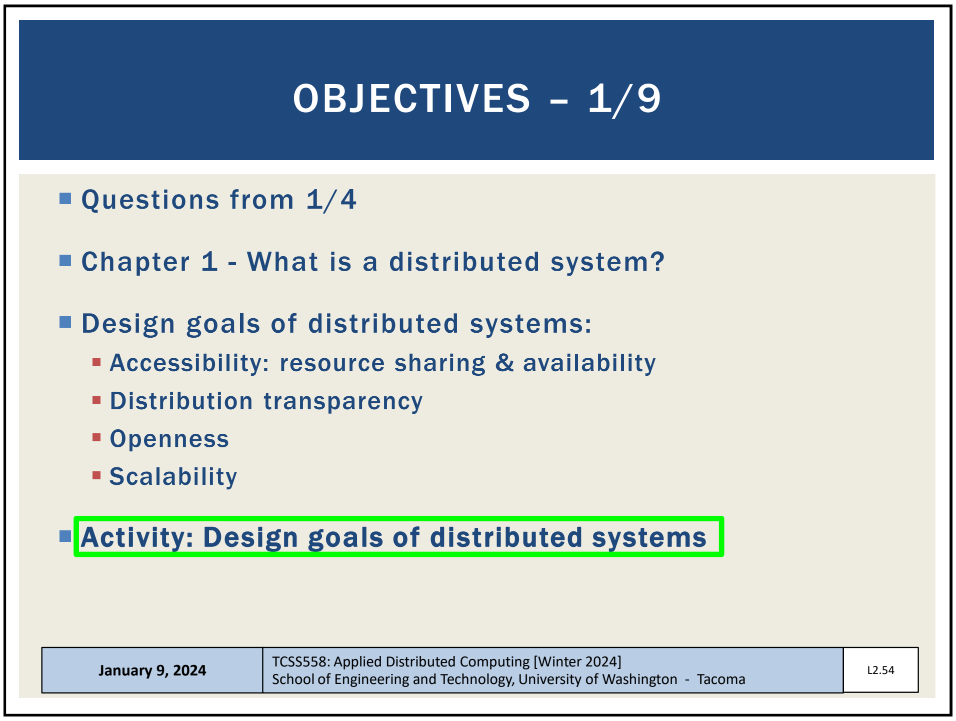
52



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

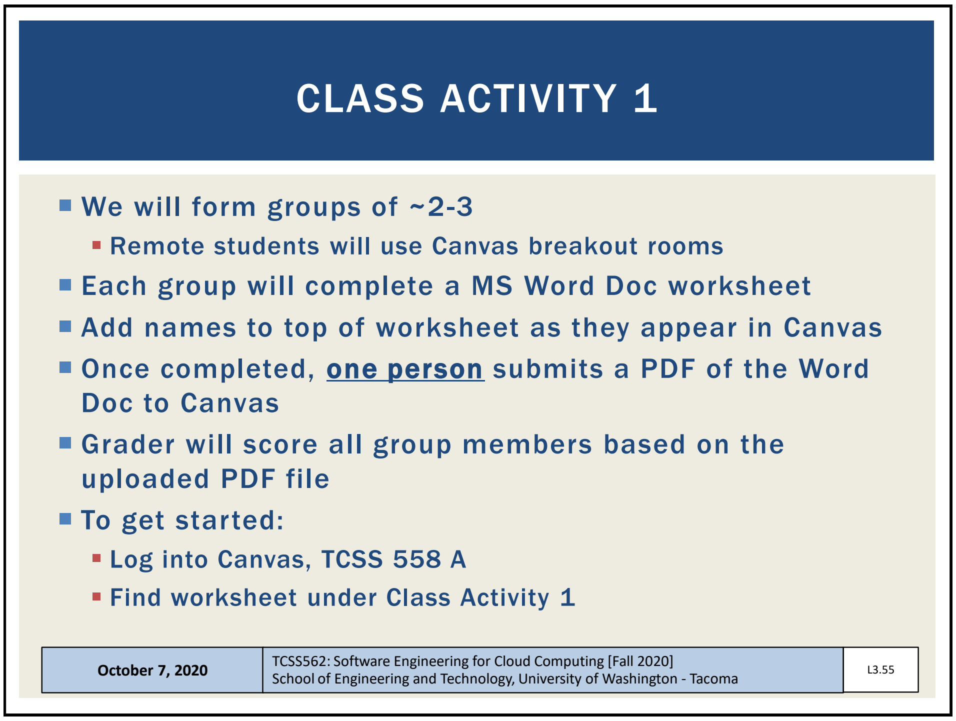
53



OBJECTIVES – 1/9

- Questions from 1/4
- 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**

54

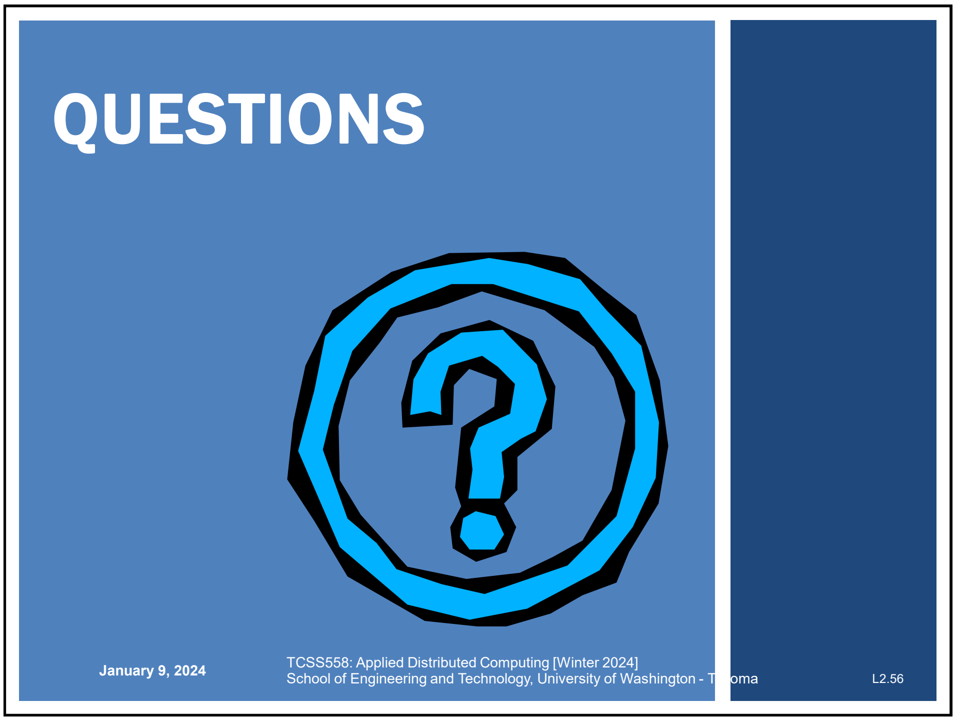


CLASS ACTIVITY 1

- We will form groups of ~2-3
  - Remote students will use Canvas breakout rooms
- Each group will complete a MS Word Doc worksheet
- Add names to top of worksheet as they appear in Canvas
- Once completed, **one person** submits a PDF of the Word Doc to Canvas
- Grader will score all group members based on the uploaded PDF file
- To get started:
  - Log into Canvas, TCSS 558 A
  - Find worksheet under Class Activity 1

October 7, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L3.55

55



QUESTIONS

January 9, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L2.56

56