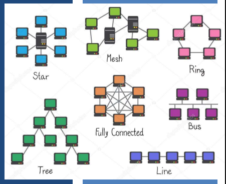
TCSS 558: APPLIED DISTRIBUTED COMPUTING

Chapter 6 – Coordination - IV

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

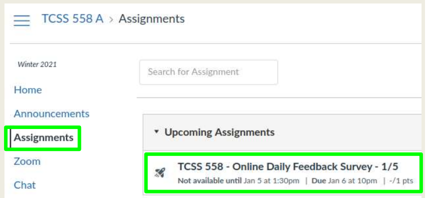


OBJECTIVES – 3/11

- Questions from 3/9
- Assignment 2: Replicated Key Value Store
- Review: Activity 4 – Total Ordered Multicasting
- Chapter 6: Coordination
  - Chapter 6.2: Vector Clocks
- Review: Activity 5 – Causality and Vector Clocks
- Chapter 6: Coordination
  - Chapter 6.3: Distributed Mutual Exclusion
- Practice Final Exam Questions

ONLINE DAILY FEEDBACK SURVEY

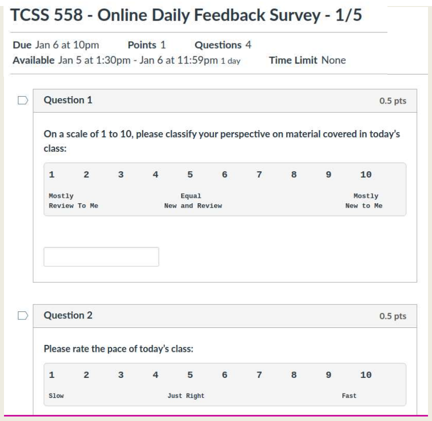
- Daily Feedback Quiz in Canvas – Available After Each Class
- Extra credit available for completing surveys ***ON TIME***
- Tuesday surveys: due by ~ Wed @ 10p
- Thursday surveys: due ~ Mon @ 10p


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


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 Review To Me — Equal New and Review — Mostly 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

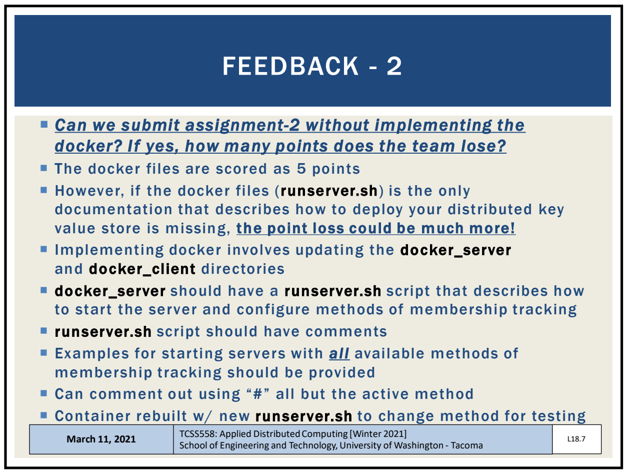
MATERIAL / PACE

- Please classify your perspective on material covered in today's class (17 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- <u>**Average – 6.53** (↑ - *previous 6.21*)</u>

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- <u>**Average – 5.88** (↑ - *previous 5.68*)</u>

FEEDBACK FROM 3/9

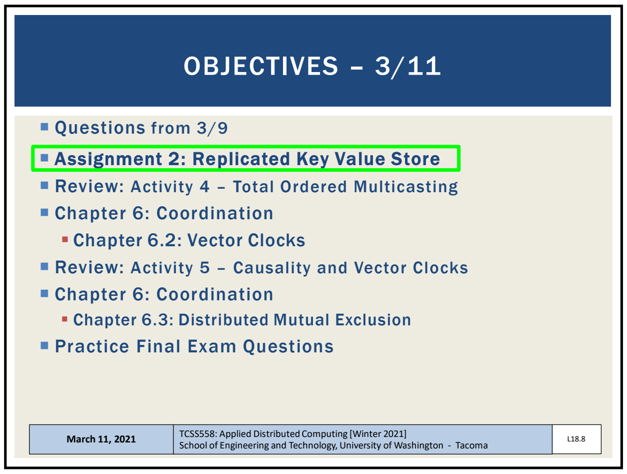
- ***"I understand how to implement a static file membership (static and dynamic list) tracking system when the servers are run on the local machine. But I don't understand how to implement it on docker.***
- ***How will a docker container have access to a txt file on the host system?"***
- No changes are required
- The server inside a docker container scans the local filesystem to check for updates to the membership file (`/tmp/nodes.txt`)
- Using "`sudo docker exec –it <container-id> bash`" user accesses `/tmp/nodes.txt` to make updates
- Need to install a text editor such as "vi":
  - `apt update ; apt upgrade ; apt install vim`
- Server will periodically scan and incorporate updates



FEEDBACK - 2

- ***Can we submit assignment-2 without implementing the docker? If yes, how many points does the team lose?***
- The docker files are scored as 5 points
- However, if the docker files (**runserver.sh**) is the only documentation that describes how to deploy your distributed key value store is missing, **the point loss could be much more!**
- Implementing docker involves updating the **docker_server** and **docker_client** directories
- **docker_server** should have a **runserver.sh** script that describes how to start the server and configure methods of membership tracking
- **runserver.sh** script should have comments
- Examples for starting servers with ***all*** available methods of membership tracking should be provided
- Can comment out using "#" all but the active method
- Container rebuilt w/ new **runserver.sh** to change method for testing

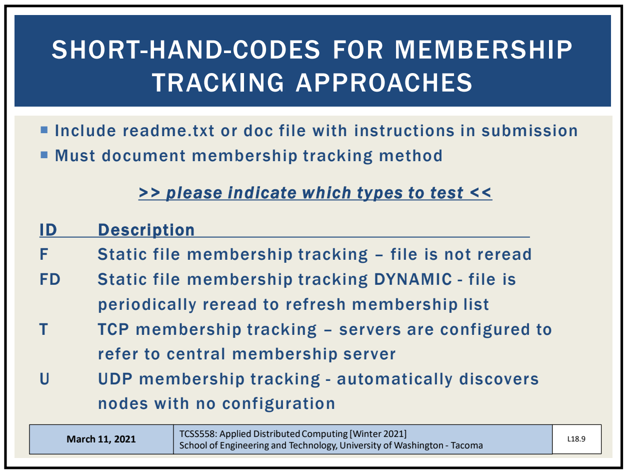
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.7



OBJECTIVES – 3/11

- Questions from 3/9
- **Assignment 2: Replicated Key Value Store**
- Review: Activity 4 – Total Ordered Multicasting
- Chapter 6: Coordination
  - Chapter 6.2: Vector Clocks
- Review: Activity 5 – Causality and Vector Clocks
- Chapter 6: Coordination
  - Chapter 6.3: Distributed Mutual Exclusion
- Practice Final Exam Questions

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.8



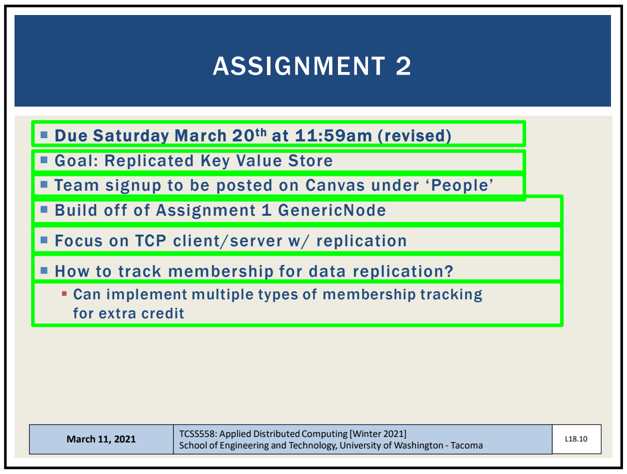
SHORT-HAND-CODES FOR MEMBERSHIP TRACKING APPROACHES

- Include readme.txt or doc file with instructions in submission
- Must document membership tracking method

>> please indicate which types to test <<

ID	Description
F	Static file membership tracking – file is not reread
FD	Static file membership tracking DYNAMIC - file is periodically reread to refresh membership list
T	TCP membership tracking – servers are configured to refer to central membership server
U	UDP membership tracking - automatically discovers nodes with no configuration

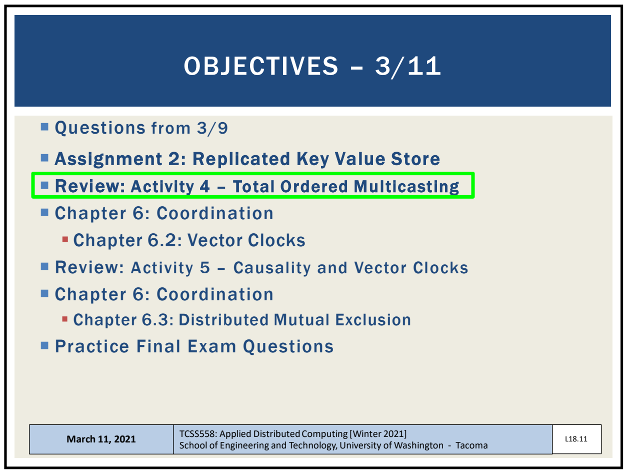
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.9



ASSIGNMENT 2

- **Due Saturday March 20th at 11:59am (revised)**
- Goal: Replicated Key Value Store
- Team signup to be posted on Canvas under 'People'
- Build off of Assignment 1 GenericNode
- Focus on TCP client/server w/ replication
- How to track membership for data replication?
  - Can implement multiple types of membership tracking for extra credit

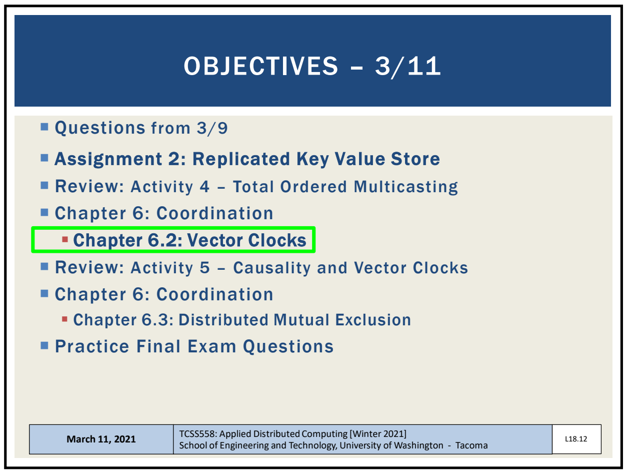
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.10



OBJECTIVES – 3/11

- Questions from 3/9
- **Assignment 2: Replicated Key Value Store**
- **Review: Activity 4 – Total Ordered Multicasting**
- Chapter 6: Coordination
  - Chapter 6.2: Vector Clocks
- Review: Activity 5 – Causality and Vector Clocks
- Chapter 6: Coordination
  - Chapter 6.3: Distributed Mutual Exclusion
- Practice Final Exam Questions

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.11



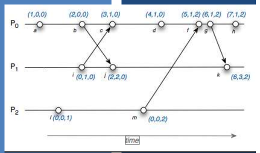
OBJECTIVES – 3/11

- Questions from 3/9
- **Assignment 2: Replicated Key Value Store**
- Review: Activity 4 – Total Ordered Multicasting
- Chapter 6: Coordination
  - **Chapter 6.2: Vector Clocks**
- Review: Activity 5 – Causality and Vector Clocks
- Chapter 6: Coordination
  - Chapter 6.3: Distributed Mutual Exclusion
- Practice Final Exam Questions

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.12

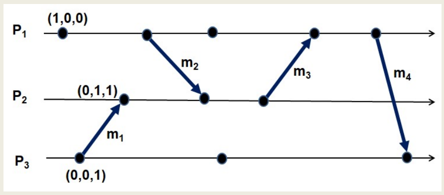
CHAPTER 6 - COORDINATION

- 6.1 Clock Synchronization
  - Physical clocks
  - Clock synchronization algorithms
- 6.2 Logical clocks
  - Lamport clocks
  - Vector clocks
- 6.3 Mutual exclusion
- 6.4 Election algorithms
- 6.6 Distributed event matching *(light)*
- 6.7 Gossip-based coordination *(light)*



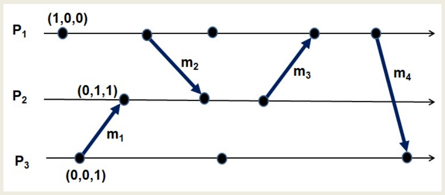
CH. 6.2: LOGICAL CLOCKS

VECTOR CLOCKS EXAMPLE - 3



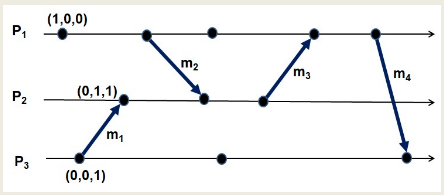
- Provide a vector clock label for unlabeled events

VECTOR CLOCKS EXAMPLE - 4



- TRUE/FALSE:
- The sending of message m_3 is causally dependent on the sending of message m_1.
- The sending of message m_2 is causally dependent on the sending of message m_1.

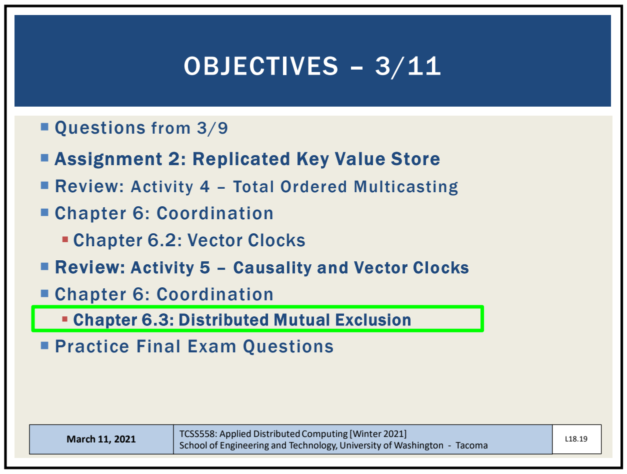
VECTOR CLOCKS EXAMPLE - 5



- TRUE/FALSE:
- P_1 (1,0,0) and P_3 (0,0,1) may be concurrent events.
- P_2 (0,1,1) and P_3 (0,0,1) may be concurrent events.
- P_1 (1,0,0) and P_2 (0,1,1) may be concurrent events.

OBJECTIVES – 3/11

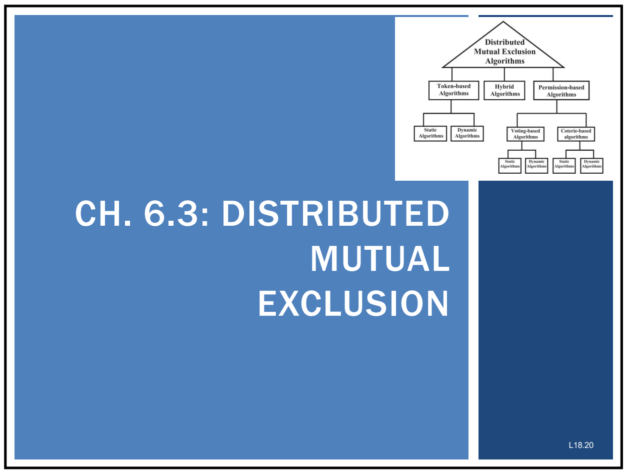
- Questions from 3/9
- **Assignment 2: Replicated Key Value Store**
- Review: Activity 4 – Total Ordered Multicasting
- Chapter 6: Coordination
  - Chapter 6.2: Vector Clocks
- **Review: Activity 5 – Causality and Vector Clocks**
- Chapter 6: Coordination
  - Chapter 6.3: Distributed Mutual Exclusion
- Practice Final Exam Questions



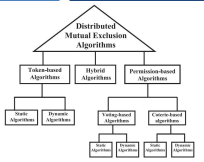
OBJECTIVES – 3/11

- Questions from 3/9
- **Assignment 2: Replicated Key Value Store**
- Review: Activity 4 – Total Ordered Multicasting
- Chapter 6: Coordination
  - Chapter 6.2: Vector Clocks
- **Review: Activity 5 – Causality and Vector Clocks**
- Chapter 6: Coordination
  - **Chapter 6.3: Distributed Mutual Exclusion**
- Practice Final Exam Questions

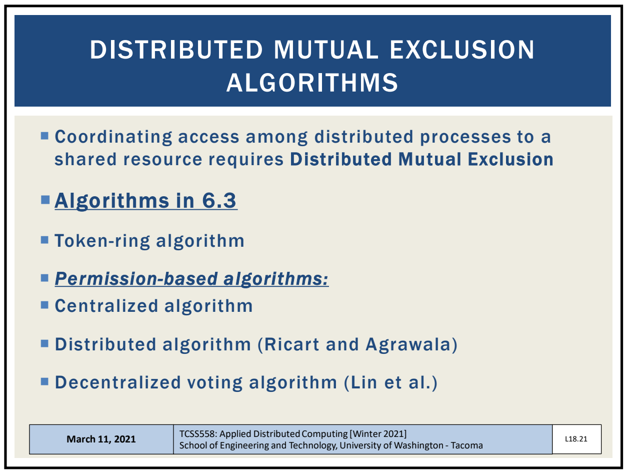
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.19



CH. 6.3: DISTRIBUTED MUTUAL EXCLUSION



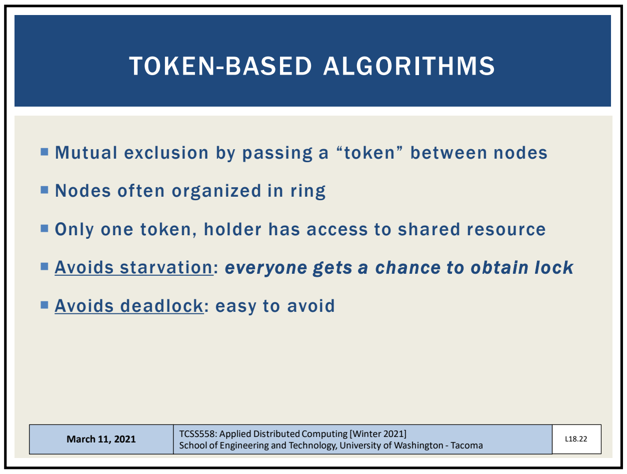
L18.20



DISTRIBUTED MUTUAL EXCLUSION ALGORITHMS

- Coordinating access among distributed processes to a shared resource requires **Distributed Mutual Exclusion**
- **Algorithms in 6.3**
- Token-ring algorithm
- ***Permission-based algorithms:***
- Centralized algorithm
- Distributed algorithm (Ricart and Agrawala)
- Decentralized voting algorithm (Lin et al.)

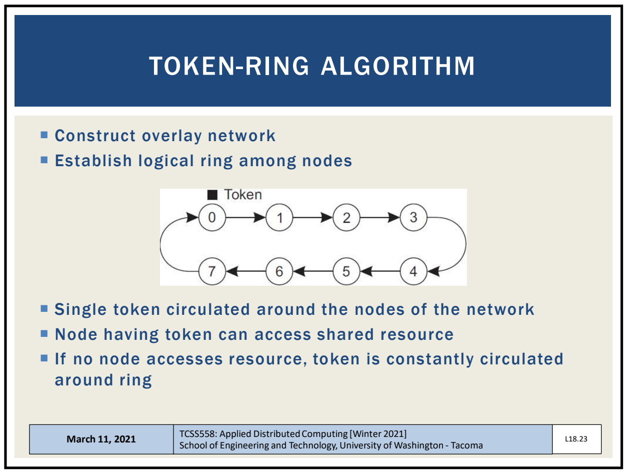
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.21



TOKEN-BASED ALGORITHMS

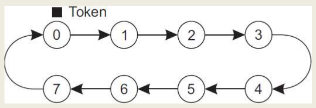
- Mutual exclusion by passing a "token" between nodes
- Nodes often organized in ring
- Only one token, holder has access to shared resource
- Avoids starvation: ***everyone gets a chance to obtain lock***
- Avoids deadlock: easy to avoid

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.22



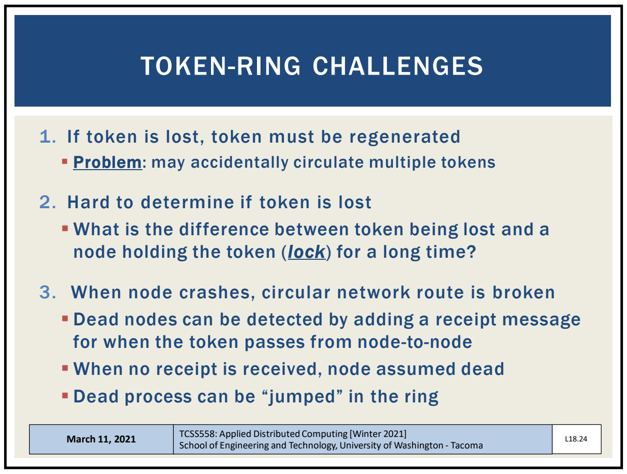
TOKEN-RING ALGORITHM

- Construct overlay network
- Establish logical ring among nodes



- Single token circulated around the nodes of the network
- Node having token can access shared resource
- If no node accesses resource, token is constantly circulated around ring

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.23



TOKEN-RING CHALLENGES

1. If token is lost, token must be regenerated
   - **Problem**: may accidentally circulate multiple tokens
2. Hard to determine if token is lost
   - What is the difference between token being lost and a node holding the token (***lock***) for a long time?
3. When node crashes, circular network route is broken
   - Dead nodes can be detected by adding a receipt message for when the token passes from node-to-node
   - When no receipt is received, node assumed dead
   - Dead process can be "jumped" in the ring

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.24

DISTRIBUTED MUTUAL EXCLUSION ALGORITHMS - 3

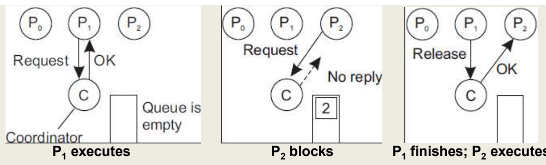
- **Permission-based algorithms**
- Processes must require permission from other processes before first acquiring access to the resource
  - CONTRAST: Token-ring did not ask nodes for permission

- **Centralized algorithm**
- Elect a single leader node to coordinate access to shared resource(s)
- Manage mutual exclusion on a distributed system similar to how it mutual exclusion is managed for a single system
- Nodes must all interact with leader to obtain ***"the lock"***

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.25

CENTRALIZED MUTUAL EXCLUSION

Permission granted from coordinator \/ No response from coordinator



- When resource not available, coordinator can block the requesting process, or respond with a reject message
- P2 must ***poll*** the coordinator if it responds with reject otherwise can wait if simply blocked
- Requests granted permission fairly using FIFO queue
- Just three messages: (request, grant (OK), release)

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.26

CENTRALIZED MUTUAL EXCLUSION - 2

- **Issues**
- Coordinator is a single point of failure
- Processes can't distinguish dead coordinator from ***"blocking"*** when resource is unavailable
  - No difference between CRASH and Block (*for a long time*)
- Large systems, coordinator becomes performance bottleneck
  - Scalability: Performance does not scale

- **Benefits**
- Simplicity:
  Easy to implement compared to distributed alternatives

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.27

DISTRIBUTED ALGORITHM

- Ricart and Agrawala [1981], use total ordering of all events
  - Leverages Lamport logical clocks

- Package up resource request message (AKA Lock Request)
- Send to all nodes
- Include:
  - Name of resource
  - Process number
  - Current (logical) time

- Assume messages are sent reliably
  - No messages are lost

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.28

DISTRIBUTED ALGORITHM - 2

- When each node receives a request message they will:

1. Say OK (***if the node doesn't need the resource***)
2. Make **no reply**, queue request (***node is using the resource)***
3. ***If node is also waiting to access the resource:*** perform a timestamp comparison -
   1. Send OK if requester has lower logical clock value
   2. Make **no reply** if requester has higher logical clock value

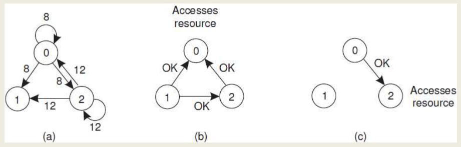
- Nodes sit back and wait for all nodes to grant permission

- Requirement: every node must know the entire membership list of the distributed system

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.29

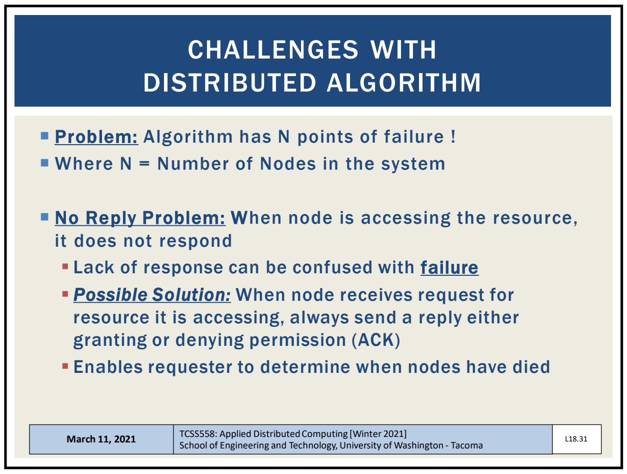
DISTRIBUTED ALGORITHM - 3

- Node 0 and Node 2 simultaneously request access to **resource**
- Node 0's time stamp is lower (8) than Node 2 (12)
- Node 1 and Node 2 grant Node 0 access
- Node 1 is not interested in the resource, it OKs both requests



- **In case of conflict, lowest timestamp wins!**
  - Node 2 rejects its own request (1@) in favor of node 0 (8)

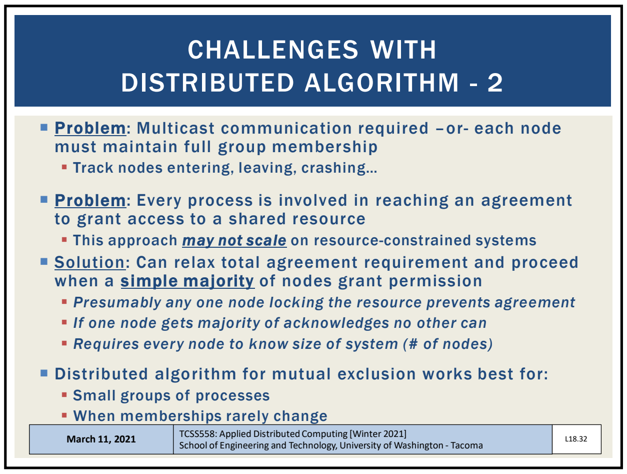
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.30



CHALLENGES WITH DISTRIBUTED ALGORITHM

- **Problem:** Algorithm has N points of failure !
- Where N = Number of Nodes in the system
- **No Reply Problem:** When node is accessing the resource, it does not respond
  - Lack of response can be confused with **failure**
  - ***Possible Solution:*** When node receives request for resource it is accessing, always send a reply either granting or denying permission (ACK)
  - Enables requester to determine when nodes have died

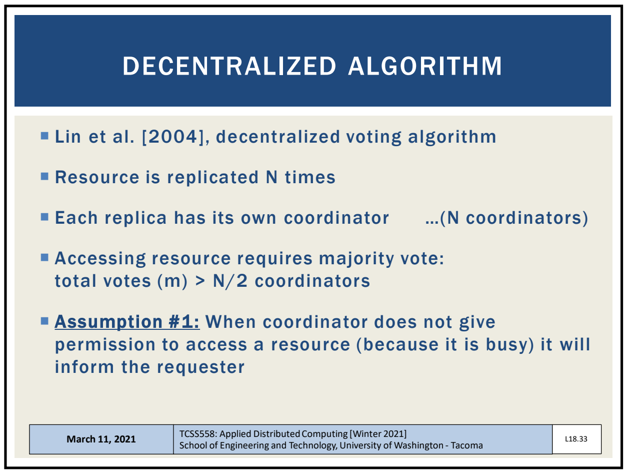
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.31



CHALLENGES WITH DISTRIBUTED ALGORITHM - 2

- **Problem**: Multicast communication required –or- each node must maintain full group membership
  - Track nodes entering, leaving, crashing…
- **Problem**: Every process is involved in reaching an agreement to grant access to a shared resource
  - This approach ***may not scale*** on resource-constrained systems
- **Solution**: Can relax total agreement requirement and proceed when a **simple majority** of nodes grant permission
  - *Presumably any one node locking the resource prevents agreement*
  - *If one node gets majority of acknowledges no other can*
  - *Requires every node to know size of system (# of nodes)*
- Distributed algorithm for mutual exclusion works best for:
  - Small groups of processes
  - When memberships rarely change

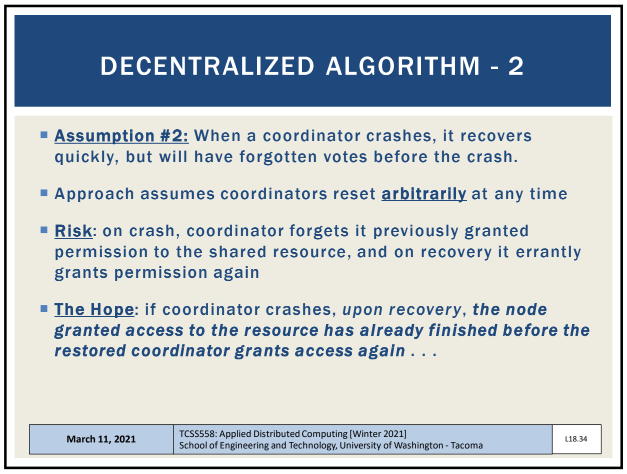
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.32



DECENTRALIZED ALGORITHM

- Lin et al. [2004], decentralized voting algorithm
- Resource is replicated N times
- Each replica has its own coordinator …(N coordinators)
- Accessing resource requires majority vote: total votes (m) > N/2 coordinators
- **Assumption #1:** When coordinator does not give permission to access a resource (because it is busy) it will inform the requester

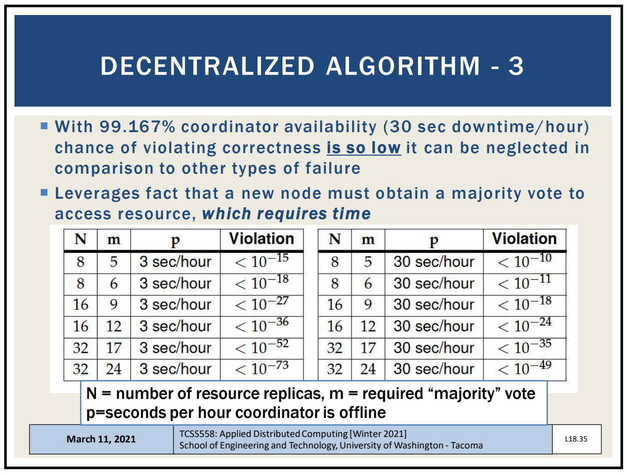
March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.33



DECENTRALIZED ALGORITHM - 2

- **Assumption #2:** When a coordinator crashes, it recovers quickly, but will have forgotten votes before the crash.
- Approach assumes coordinators reset **arbitrarily** at any time
- **Risk**: on crash, coordinator forgets it previously granted permission to the shared resource, and on recovery it errantly grants permission again
- **The Hope**: if coordinator crashes, *upon recovery*, ***the node granted access to the resource has already finished before the restored coordinator grants access again . . .***

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.34



DECENTRALIZED ALGORITHM - 3

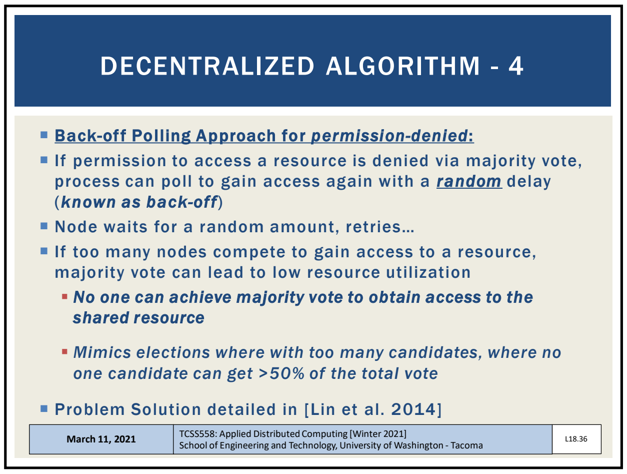
- With 99.167% coordinator availability (30 sec downtime/hour) chance of violating correctness **is so low** it can be neglected in comparison to other types of failure
- Leverages fact that a new node must obtain a majority vote to access resource, ***which requires time***

N	m	p	Violation
8	5	3 sec/hour	$< 10^{-15}$
8	6	3 sec/hour	$< 10^{-18}$
16	9	3 sec/hour	$< 10^{-27}$
16	12	3 sec/hour	$< 10^{-36}$
32	17	3 sec/hour	$< 10^{-52}$
32	24	3 sec/hour	$< 10^{-73}$

N	m	p	Violation
8	5	30 sec/hour	$< 10^{-10}$
8	6	30 sec/hour	$< 10^{-11}$
16	9	30 sec/hour	$< 10^{-18}$
16	12	30 sec/hour	$< 10^{-24}$
32	17	30 sec/hour	$< 10^{-35}$
32	24	30 sec/hour	$< 10^{-49}$

N = number of resource replicas, m = required "majority" vote
p=seconds per hour coordinator is offline

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.35

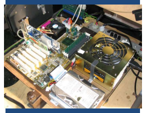


DECENTRALIZED ALGORITHM - 4

- **Back-off Polling Approach for *permission-denied:***
- If permission to access a resource is denied via majority vote, process can poll to gain access again with a ***random*** delay (***known as back-off***)
- Node waits for a random amount, retries…
- If too many nodes compete to gain access to a resource, majority vote can lead to low resource utilization
  - ***No one can achieve majority vote to obtain access to the shared resource***
  - ***Mimics elections where with too many candidates, where no one candidate can get >50% of the total vote***
- Problem Solution detailed in [Lin et al. 2014]

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.36

WE WILL RETURN AT 3:01 PM



When poll is active, respond at **PollEv.com/wesleylloyd641**
Text **WESLEYLLOYD641** to **22333** once to join

Which algorithm offers the best scalability to support distributed mutual exclusion in a large distributed system?

Token-ring algorithm
Centralized algorithm
Distributed algorithm
Decentralized voting algorithm
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**

When poll is active, respond at **PollEv.com/wesleylloyd641**
Text **WESLEYLLOYD641** to **22333** once to join

Which algorithm(s) involve blocking (no reply) when a resource is not available? (check all that apply)

Token-ring algorithm
Centralized Algorithm
Distributed algorithm
Decentralized voting algorithm
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**

When poll is active, respond at **PollEv.com/wesleylloyd641**
Text **WESLEYLLOYD641** to **22333** once to join

Which algorithm(s) involve arriving at a consensus (majority opinion) to determine whether a node should be granted access to a resource? (check all that apply)

Token-ring algorithm
Centralized algorithm
Distributed algorithm
Decentralized voting algorithm
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**

When poll is active, respond at **PollEv.com/wesleylloyd641**
Text **WESLEYLLOYD641** to **22333** once to join

Which algorithm(s) have N points of failure, where N = Number of Nodes in the system? (check all that apply)

Token-ring algorithm
Centralized algorithm
Distributed algorithm
Decentralized voting algorithm
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**

DISTRIBUTED MUTUAL EXCLUSION ALGORITHMS REVIEW

- Which algorithm offers the best scalability to support distributed mutual exclusion in a large distributed system?
- (A) Token-ring algorithm
- (B) Centralized algorithm
- (C) Distributed algorithm
- (D) Decentralized voting algorithm

March 11, 2021 | TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma | L18.42

DISTRIBUTED MUTUAL EXCLUSION ALGORITHMS REVIEW - 2

- Which algorithm(s) involve blocking (no reply) when a resource is not available?
- (A) Token-ring algorithm
- (B) Centralized algorithm
- (C) Distributed algorithm
- (D) Decentralized voting algorithm

DISTRIBUTED MUTUAL EXCLUSION ALGORITHMS REVIEW - 3

- Which algorithm(s) involve arriving at a consensus (majority opinion) to determine whether a node should be granted access to a resource?
- (A) Token-ring algorithm
- (B) Centralized algorithm
- (C) Distributed algorithm
- (D) Decentralized voting algorithm

DISTRIBUTED MUTUAL EXCLUSION ALGORITHMS REVIEW - 4

- Which algorithm(s) have N points of failure, where N = Number of Nodes in the system?
- (A) Token-ring algorithm
- (B) Centralized algorithm
- (C) Distributed algorithm
- (D) Decentralized voting algorithm

OBJECTIVES – 3/11

- Questions from 3/9
- **Assignment 2: Replicated Key Value Store**
- Review: Activity 4 – Total Ordered Multicasting
- Chapter 6: Coordination
  - Chapter 6.2: Vector Clocks
- **Review: Activity 5 – Causality and Vector Clocks**
- Chapter 6: Coordination
  - Chapter 6.3: Distributed Mutual Exclusion
- **Practice Final Exam Questions**

PRACTICE QUESTIONS

PRACTICE QUESTIONS

PRACTICE EXAM QUESTIONS

- We will take a break until ~4:30pm
- At ~4:30pm we will spend approximately 1 hour reviewing solutions to the practice TCSS 558 exam questions
- Solutions will be **recorded** as a separate Zoom recording, and shared using Canvas via an announcement
- We will meet at ~4:30pm **using the same Zoom link**
- Attendance is optional

QUESTION 1: MULTI-TIERED ARCHITECTURE

- For a multi-tiered architecture describe the differences between a vertical distribution and a horizontal distribution of components (Lecture 6)?
- >>Address specifically implications of these distributions for ***scalability*** of distributed systems.

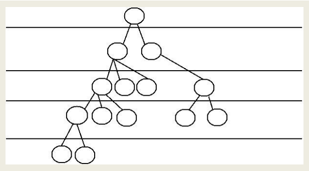
QUESTION 2: CENTRALIZED SERVER ARCHITECTURE

- Consider a traditional centralized server architecture where many client nodes communicate with a single server node.
- Consider the four design goals of distributed systems from Chapter 1: Resource sharing, Distribution Transparency, Openness, and Scalability.
- Describe challenges with ensuring these design goals when adopting a centralized server architecture.
- >> Consider citing an example if helpful.

QUESTION 3: ARCHITECTURE DIFFERENCES

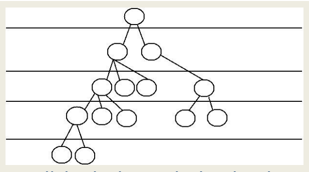
- Describe two communication differences between a traditional connection oriented client/server architecture, and a publish/subscribe architecture where clients and servers communicate by interacting with tuples in a shared data space.

QUESTION 4: UNSTRUCTURED PEER-TO-PEER NETWORK



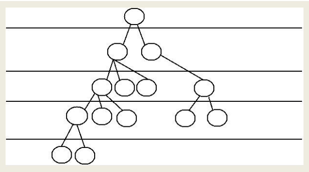
- Fourteen nodes communicate using an unstructured peer-to-peer network using random walks. The head node pictured at the top of the graph for this network receives a client request to retrieve a data element. Starting at the head node using message flooding without a specified time-to-live (TTL), how many messages are sent to locate the data item?

QUESTION 4 (2): UNSTRUCTURED PEER-TO-PEER NETWORK



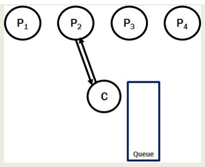
- Using a random walk beginning at the head node at the top of the graph where only one walk per level is performed without a specified time-to-live (TTL), how many nodes will be visited?
- Given this number of node visitations, and considering that the data element is not replicated in the network as it exists at only one node, what is the probability (in %) that the data element will be found?

QUESTION 4 (3): UNSTRUCTURED PEER-TO-PEER NETWORK



- If we perform two parallel walks without a TTL, what is the worst-case probability (in %) of finding the data element?
- For this scenario, what is the best-case probability (in %) of finding the data element?

QUESTION 5: DISTRIBUTED MUTUAL EXCLUSION

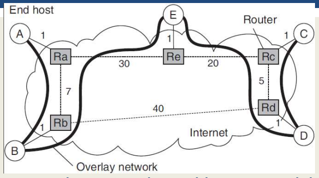


- List one advantage, and one disadvantage for centralized distributed mutual exclusion:
- Advantage: Disadvantage:

QUESTION 6: TIME MANIA

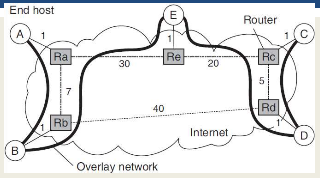
- Approaches to synchronizing time across all of the nodes of a distributed system focus on ensuring either one or both of the following: **accuracy** and/or **precision**
- For each time tracking approach below, identify whether it provides accuracy, precision, or both for coordinating time across the nodes in a distributed system.
- NTP:
- Berkeley:
- Lamport Clocks:
- Vector Clocks:

QUESTION 7: OVERLAY NETWORKS



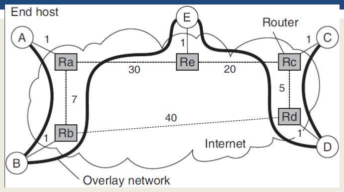
- In the figure, an overlay network provides connectivity among the nodes: A, B, C, D, and E.
- The overlay network is implemented using "underlying" networks. In this case, the underlying network consists of a series of routers: Ra, Rb, Rc, Rd, and Re. Network "Weights" are assigned to each of the links between the routers indicating approximate communication delay. For example, the communication delay between Ra and Rb is 7 units, whereas the communication delay between node A and Ra is just 1 unit.

QUESTION 7: OVERLAY NETWORKS



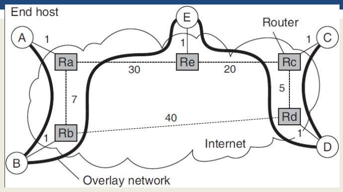
- When nodes communicate using the overlay network, they must route messages via (by way of) the "overlay" links. In the diagram above, there are overlay links between: A → B, B → E, E → D, and D → C.

QUESTION 7: OVERLAY NETWORKS

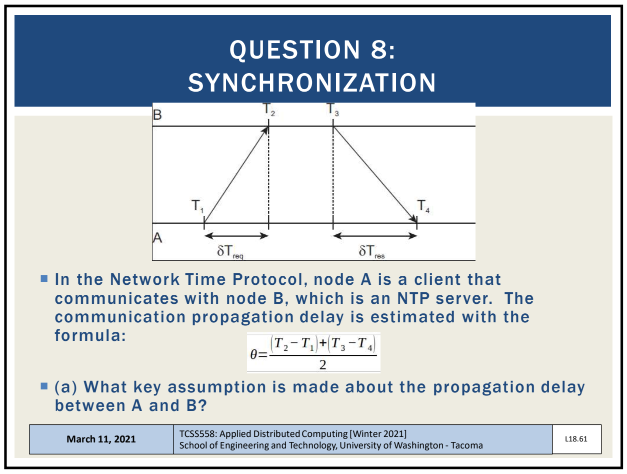


- (A) What is the network delay when routing a message using the overlay network from node D to B? _______________units
- (B) What is the network delay when sending this same message from node D to B via the most efficient path using the underlying network? _______________ units

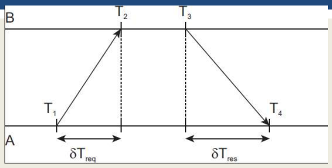
QUESTION 7: OVERLAY NETWORKS



- (C) Network "Stretch" is the ratio of the overlay network delay to the underlying network delay. For this example, what is the network stretch? _______________units



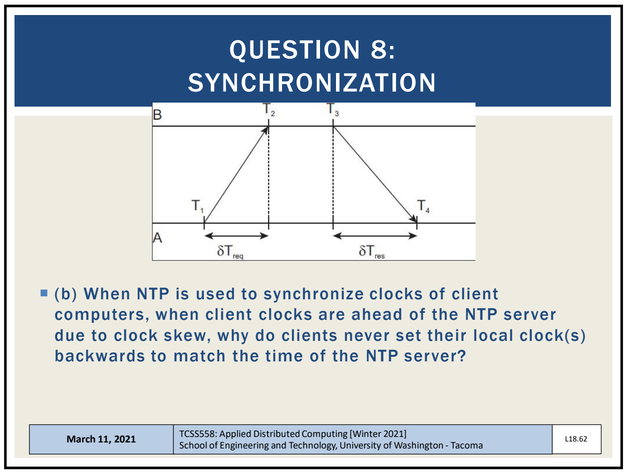
QUESTION 8: SYNCHRONIZATION



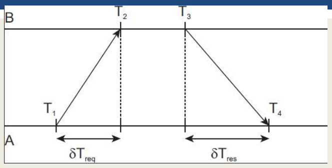
- In the Network Time Protocol, node A is a client that communicates with node B, which is an NTP server. The communication propagation delay is estimated with the formula:

$$\theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

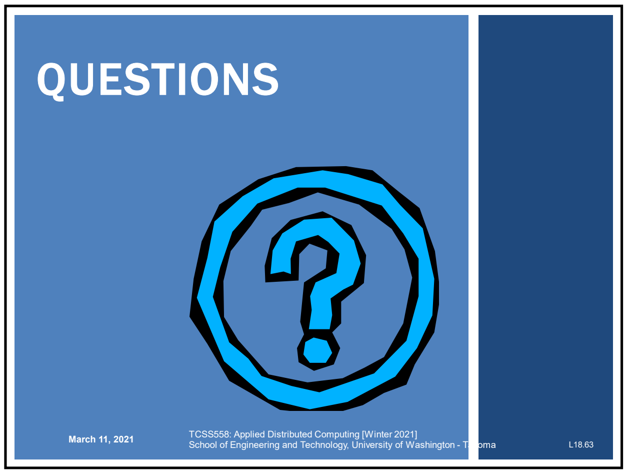
- (a) What key assumption is made about the propagation delay between A and B?



QUESTION 8: SYNCHRONIZATION



- (b) When NTP is used to synchronize clocks of client computers, when client clocks are ahead of the NTP server due to clock skew, why do clients never set their local clock(s) backwards to match the time of the NTP server?



QUESTIONS