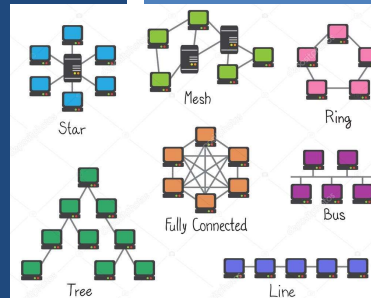


TCSS 558: APPLIED DISTRIBUTED COMPUTING

Chapter 6 – Coordination - III

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



OBJECTIVES – 3/9

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

March 9, 2021

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

L17.2

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 Search for Assignment

Home

Announcements

Assignments

Zoom

Chat

▼ 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

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.3
---------------	---	-------

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

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.4
---------------	---	-------

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (19 respondents):
 - 1-mostly review, 5-equal new/review, 10-mostly new
 - **Average - 6.21** (↓ - *previous 6.87*)
- Please rate the pace of today's class:
 - 1-slow, 5-just right, 10-fast
 - **Average - 5.68** (↓ - *previous 5.73*)

March 9, 2021

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

L17.5

FEEDBACK FROM 3/4

- **Can you visualize how docker swarm works in assignment 2?**
- Use of Docker Swarm is optional for assignment 2
- Docker Swarm combined with an overlay network allows docker containers created across multiple docker hosts using "docker-machine" to be assigned IP addresses on a common interconnected network.
- Containers created across the swarm using the overlay can communicate using TCP
 - *UDP multicast does not work however*
- For assignment 2, you can deploy your multi-node KV store on ec2 across multiple VMs using a swarm + overlay
 - This exercise is optional for assignment #2

March 9, 2021

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

L17.6

DOCKER SWARM + OVERLAY NETWORK

■ Configuration Steps:

1. Launch docker swarm manager node
2. Start swarm manually
3. Copy the token provided. It will be used to add each docker-machine to the swarm.
4. Launch docker-machines as worker nodes in the swarm
5. Create the overlay network
6. Launch individual containers using the overlay network
7. Switch context as needed to the proper docker-machine to place (load-balance) containers
 - `eval $(docker-machine env aw-swarmID)`,
where "aw-swarmID" is a docker-machine node name

See:

<http://faculty.washington.edu/wlloyd/courses/tcss558/assignments/a2/DockerSwarmOverlay-howto.txt>

March 9, 2021

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

L17.7

FEEDBACK - 2

- *I am not clear about total ordered multicasting. Could you please explain in more detail.*
- We will show an example in class today...

March 9, 2021

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

L17.8

OBJECTIVES – 3/9

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

March 9, 2021

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

L17.9

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 9, 2021

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

L17.10

ASSIGNMENT 2

- **Sunday March 14th**
- **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 9, 2021

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

L17.11

OBJECTIVES – 3/9

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

March 9, 2021

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

L17.12

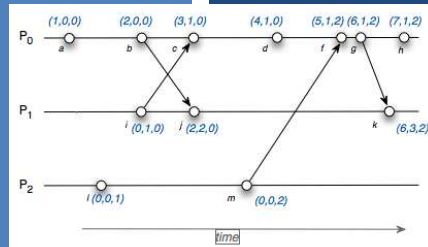
CHAPTER 6 - COORDINATION

- 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*)

March 9, 2021

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

L17.13

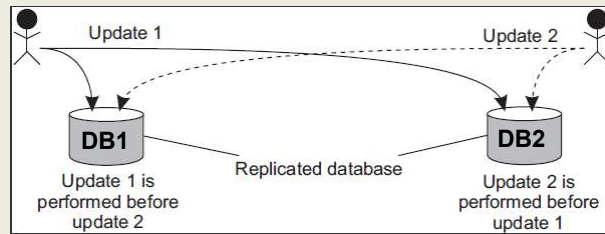


CH. 6.2: LOGICAL CLOCKS

L17.14

TOTAL-ORDERED MULTICASTING

- Consider concurrent updates to a replicated database
- Communication latency between DB1 and DB2 is 250ms



- **Initial Account balance: \$1,000**
- **Update #1: Deposit \$100**
- **Update #2: Add 1% Interest**
- **Total Ordered Multicasting needed**

March 9, 2021

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

L17.15

TOTAL-ORDERED MULTICASTING EXAMPLE

- Two messages (m_1, m_2) must be distributed, to two processes (p_1, p_2)
- We assume messages have correct lamport clock timestamps
- $m_1(10, p_1, \text{add } \$100)$
- $m_2(12, p_2, \text{add } 1\% \text{ interest})$
- Each process maintains a queue of messages
- Arriving messages are placed into queues ordered by the Lamport clock timestamp
- In each queue, each message must be acknowledged by every process in the system before operations can be applied to the local database

March 9, 2021

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

L17.16

TOTAL-ORDERED MULTICASTING EXAMPLE

- Two messages (m_1, m_2) must be distributed, to two processes (p_1, p_2)
- We assume messages have correct lamport clock timestamps
- $m_1(10, p_1, \text{add } \$100)$

Key point:

Multicast messages are also received by the sender (*itself*)

Arriving messages are placed into queues ordered by the Lamport clock timestamp

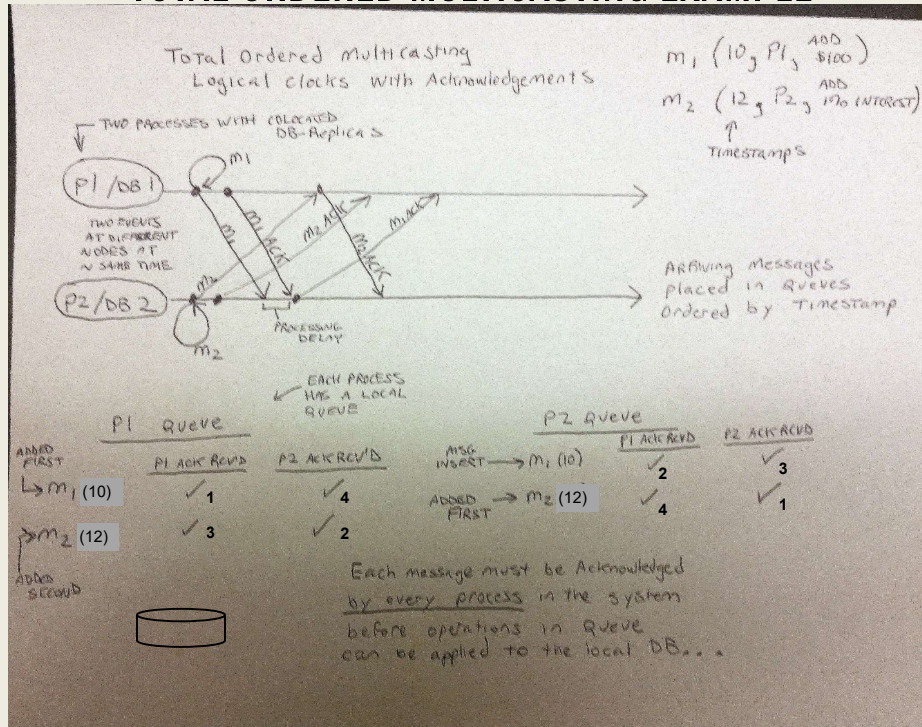
- In each queue, each message must be acknowledged by every process in the system before operations can be applied to the local database

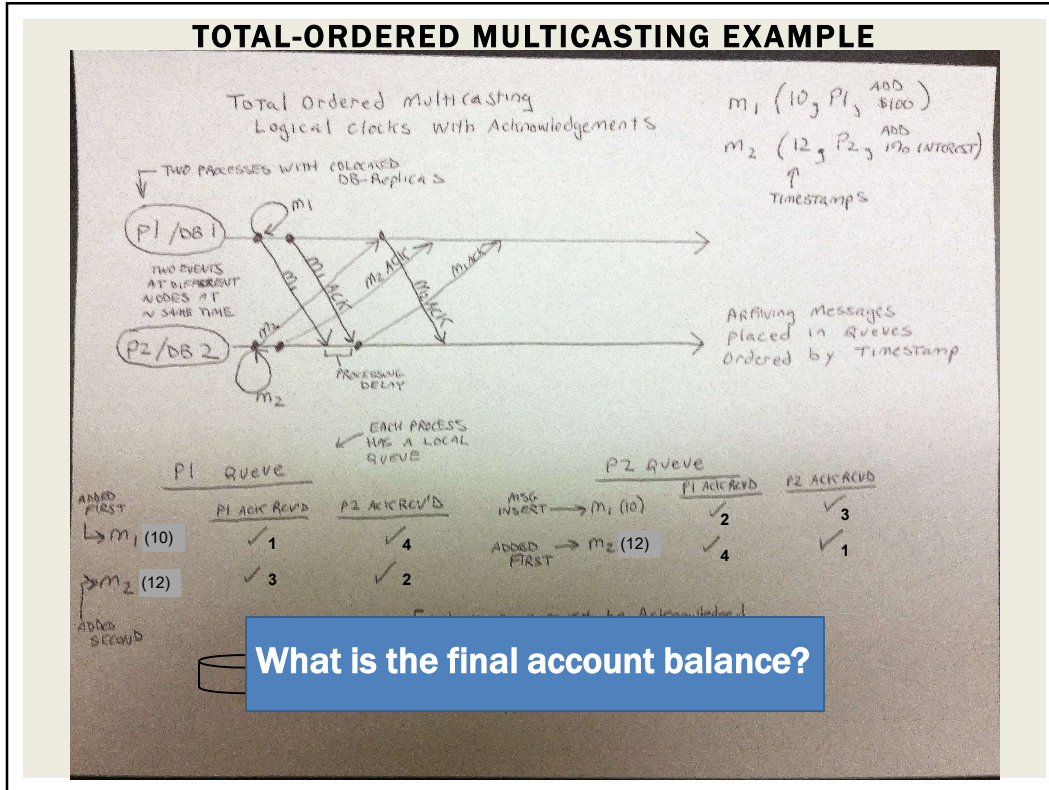
March 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
 School of Engineering and Technology, University of Washington - Tacoma

L17.17

TOTAL-ORDERED MULTICASTING EXAMPLE



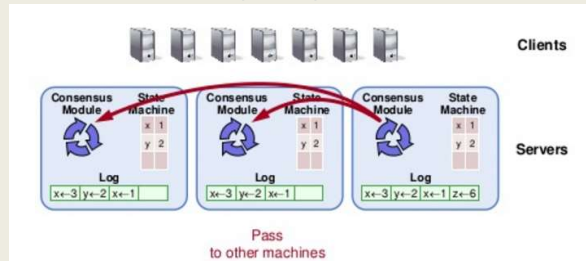


TOTAL-ORDERED MULTICASTING - 2

- Each message timestamped with local logical clock of sender
- **Multicast messages are also received by the sender (*itself*)**
- Assumptions:
 - Messages from same sender received in order they were sent
 - No messages are lost
- When messages arrive they are placed in local queue ordered by timestamp
- Receiver **multicasts** acknowledgement of message receipt to other processes
 - Time stamp of message receipt is lower the acknowledgement
- This process **replicates** queues across sites
- Messages delivered to application (database) only when message at the head of the queue has been acknowledged by **every process in the system**

TOTAL-ORDERED MULTICASTING - 3

- Can be used to implement replicated state machines (RSMs)
- Concept is to replicate event queues at each node
- (1) **Using logical clocks** and (2) **exchanging acknowledgement messages**, allows for events to be “**totally**” ordered in replicated event queues
- Events can be applied “**in order**” to each (distributed) replicated state machine (RSM)



March 9, 2021

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

L17.21

OBJECTIVES – 3/9

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

March 9, 2021

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

L17.22

VECTOR CLOCKS

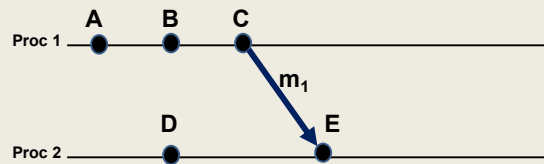
- Lamport clocks don't help to determine causal ordering of messages
- Vector clocks capture causal histories and can be used as an alternative
- But what is causality? ...

March 9, 2021

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

L17.23

WHAT IS CAUSALITY?



- Having a causal relationship between two events (A and E) indicates that event E results from the occurrence of event A.
- When one event results from another, there is a causal relationship between the two events.
- This is also referred to as cause and effect.

March 9, 2021

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

L17.24

CAUSALITY - 2

- **Disclaimer:**
- Without knowing actual information contained in messages, it is not possible to state with certainty that there is a causal relationship or perhaps a conflict

- Lamport/Vector clocks can help us suggest possible causality
- But we never know for sure...

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.25
---------------	---	--------

CAUSALITY - 3

- Consider the messages:

P ₁	P ₂	P ₃
0	0	0
6	8	10
12	16	20
18	24	30
24	32	40
30	40	50
36	48	60
42	61	70
48	69	80
70	77	90
76	85	100

- P2 receives m1, and subsequently sends m3
- **Causality:** Sending m3 *may* depend on what's contained in m1
- P2 receives m2, receiving m2 is **not** related to receiving m1
- **Is sending m3 causally dependent on receiving m2?**

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.26
---------------	---	--------

VECTOR CLOCKS

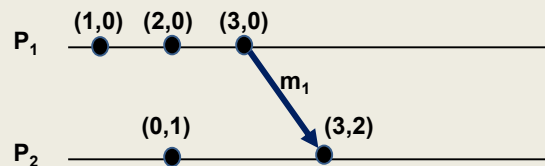
- Vector clocks help keep track of **causal history**
- If two local events happened at process P, then the causal history $H(p_2)$ of event p_2 is $\{p_1, p_2\}$
- P sends messages to Q (event p_3)
- Q previously performed event q_1
- Q records arrival of message as q_2
- Causal histories merged at Q $H(q_2) = \{p_1, p_2, p_3, q_1, q_2\}$
- Fortunately, can simply store history of last event, as a vector clock $\rightarrow H(q_2) = (3, 2)$
- Each entry corresponds to the last event at the process

March 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.27

VECTOR CLOCKS - 2



- Each process maintains a vector clock which
 - Captures number of events at the local process (e.g. logical clock)
 - Captures number of events at all other processes
- Causality is captured by:
 - For each event at P_i , the vector clock (VC_i) is incremented
 - The msg is timestamped with VC_i ; and sending the msg is recorded as a new event at P_i
 - P_j adjusts its VC_j choosing the **max** of: the message timestamp - or- the local vector clock (VC_j)

March 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.28

VECTOR CLOCKS - 3

- P_j knows the # of events at P_i based on the timestamps of the received message
- P_j learns how many events have occurred at other processes based on timestamps in the vector
- These events *“may be causally dependent”*
- **In other words:** they may have been necessary for the message(s) to be sent...

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.29
---------------	---	--------

VECTOR CLOCKS EXAMPLE

▪ Local clock is underlined

CAUSALITY

m_2	m_4	$m_2 < m_4$	$m_2 > m_4$	Conclusion
(2,1,0)	(4,3,0)	Yes	No	m2 may causally precede m4

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.30
---------------	---	--------

VECTOR CLOCKS EXAMPLE - 2

m_2	m_4	$m_2 < m_4$	$m_2 > m_4$	Conclusion
(4,1,0)	(2,3,0)	No	No	m_2 and m_4 may conflict

- P3 can't determine if m_4 may be causally dependent on m_2
- Is m_4 causally dependent on m_3 ?

March 9, 2021

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

L17.31

VECTOR CLOCKS EXAMPLE - 3

- Provide a vector clock label for unlabeled events

March 9, 2021

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

L17.32

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 .

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.33
---------------	---	--------

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.

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.34
---------------	---	--------

**WE WILL RETURN AT
3:01 PM**



OBJECTIVES – 3/9

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

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.36
---------------	---	--------

OBJECTIVES – 3/9

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

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.37
---------------	---	--------

OBJECTIVES – 3/9

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

March 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.38
---------------	---	--------

CH. 6.3: DISTRIBUTED MUTUAL EXCLUSION

L17.39

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 9, 2021	TCCS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.40
---------------	---	--------

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 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.41
---------------	---	--------

TOKEN-RING ALGORITHM

- Construct overlay network
- Establish logical ring among nodes

■ Token

```
graph LR; 0((0)) --> 1((1)); 1 --> 2((2)); 2 --> 3((3)); 3 --> 4((4)); 4 --> 5((5)); 5 --> 6((6)); 6 --> 7((7)); 7 --> 0
```

- 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 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.42
---------------	---	--------

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 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.43

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**”

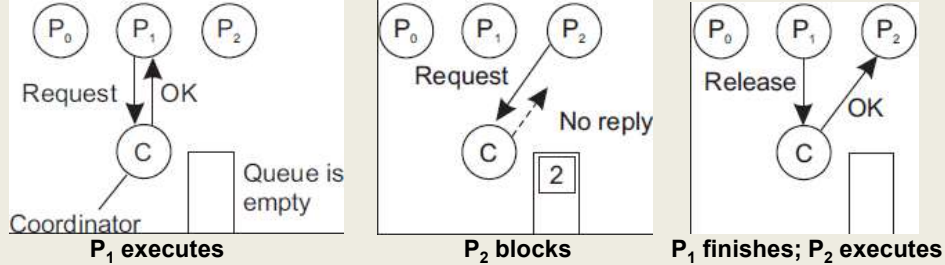
March 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.44

CENTRALIZED MUTUAL EXCLUSION

Permission granted from coordinator \vee 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 9, 2021

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

L17.45

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 9, 2021

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

L17.46

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 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.47

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

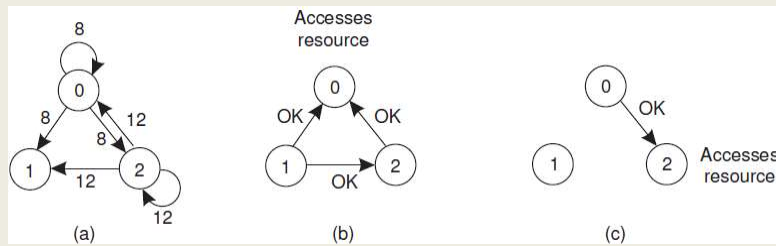
March 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.48

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 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.49

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 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.50

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 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.51

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 9, 2021

TCCS558: Applied Distributed Computing [Winter 2021]
School of Engineering and Technology, University of Washington - Tacoma

L17.52

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 9, 2021

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

L17.53

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	N	m	p	Violation
8	5	3 sec/hour	$< 10^{-15}$	8	5	30 sec/hour	$< 10^{-10}$
8	6	3 sec/hour	$< 10^{-18}$	8	6	30 sec/hour	$< 10^{-11}$
16	9	3 sec/hour	$< 10^{-27}$	16	9	30 sec/hour	$< 10^{-18}$
16	12	3 sec/hour	$< 10^{-36}$	16	12	30 sec/hour	$< 10^{-24}$
32	17	3 sec/hour	$< 10^{-52}$	32	17	30 sec/hour	$< 10^{-35}$
32	24	3 sec/hour	$< 10^{-73}$	32	24	30 sec/hour	$< 10^{-49}$

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

March 9, 2021

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

L17.54

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 9, 2021

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

L17.55

When poll is active, respond at [PollEv.com/wesleylloyd641](https://poll-ev.com/wesleylloyd641)

Text **WESLEYLLOYD641** to **22333** once to join

W 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 poll-ev.com/app

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

W 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

W 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](https://poll-ev.com/wesleylloyd641)
Text **WESLEYLLOYD641** to **22333** once to join

W 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 poll-ev.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 9, 2021	TCSS558: Applied Distributed Computing [Winter 2021] School of Engineering and Technology, University of Washington - Tacoma	L17.60
---------------	---	--------

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

March 9, 2021

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

L17.61

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

March 9, 2021

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

L17.62

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

March 9, 2021

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

L17.63

QUESTIONS



March 9, 2021

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

L17.64