

# Distributed Computing III

*Murphy was an optimist*

CSSE6400

Richard Thomas

April 11, 2022

Question

What communication faults may occur?

Question

What communication faults may occur?

Answer

- Message not delivered

### Question

What communication faults may occur?

### Answer

- Message not delivered
- Message delayed

### Question

What communication faults may occur?

### Answer

- Message not delivered
- Message delayed
- Receiver failed

### Question

What communication faults may occur?

### Answer

- Message not delivered
- Message delayed
- Receiver failed
- Receiver busy

### Question

What communication faults may occur?

### Answer

- Message not delivered
- Message delayed
- Receiver failed
- Receiver busy
- Reply not received

## Question

What communication faults may occur?

## Answer

- Message not delivered
  - Message delayed
  - Receiver failed
  - Receiver busy
  - Reply not received
  - Reply delayed
- Lost in transit
  - Network delay or receiver overloaded, but message will be processed later
  - Receiver software has crashed or node has died
  - Receiver temporarily not replying (e.g. garbage collection has frozen other processes)
  - Request was processed but reply lost in transit
  - Reply will be received later



Question

How do we detect faults?

### Question

How do we detect faults?

### Answer

- No listener on port – RST or FIN packet

### Question

How do we detect faults?

### Answer

- No listener on port – RST or FIN packet
- Process crashes – Monitor report failure

### Question

How do we detect faults?

### Answer

- No listener on port – RST or FIN packet
- Process crashes – Monitor report failure
- IP address not reachable – unreachable packet

### Question

How do we detect faults?

### Answer

- No listener on port – RST or FIN packet
- Process crashes – Monitor report failure
- IP address not reachable – unreachable packet
- Query switches

## Question

# How do we detect faults?

## Answer

- No listener on port – RST or FIN packet
- Process crashes – Monitor report failure
- IP address not reachable – unreachable packet
- Query switches
- Timeout

- Assumes node is running & reachable. OS should close or refuse connection. Error packet may be lost in transit.
- Assumes node is running & reachable. Most reliable.
- Router has to determine address is not reachable, which is no easier than for your application.
- Need permissions to do this. Will only have this in your own data centre.
- UDP reduces network transmission time guarantee – does not perform retransmission

Question

What to do if fault is detected?

### Question

What to do if fault is detected?

### Answer

- Retry
- Restart

- How many retries? How often?
- Exponential backoff with jitter
- How long to wait to restart?
- Too long reduces responsiveness.
- Unacknowledged messages need to be sent to other nodes – reducing performance.
- Too short may prematurely declare nodes dead.
- May lead to contention – two nodes processing the same request.
- May lead to cascading failure – load is sent to other nodes, slowing them down so they are then declared dead ....



#### Definition 1. Idempotency

Repeating an operation does not change receiver's state.

- Idempotent consumer pattern
- Tag messages with an ID, so repeated messages can be ignored
- Or, redo messages that do not change state (e.g. queries)

## Byzantine Generals Problem



- $n$  generals need to agree on plan
  - Can only communicate via messenger
  - Messenger may be delayed or lost
  - Some generals are traitors
    - Send dishonest messages
    - Pretend to have not received message
- Link analogy to Byzantine faults
  - Mention idea of Byzantine fault-tolerant systems
  - e.g. safety critical systems, blockchain, ...

Question

How to order asynchronous messages?

### Question

## How to order asynchronous messages?

### Answer

- Timestamps?
  - Can't keep clocks in sync
  - Limited clock precision
- Trying to sync with NTP is unreliable
- Network delays during sync
- Clock drift between syncs
- Finite precision – two events may end up with the same timestamp, if they occur in quick succession

Consistency

Eventual Consistency weak guarantee

Linearisability strong guarantee

Causal Ordering strong guarantee

## Linearisability

- Once value is written, all reads see same value
  - Regardless of replica read from

#### Linearisability

- Once value is written, all reads see same value
  - Regardless of replica read from
- Single-leader replication
  - Read from leader
  - Read from synchronous follower

## Linearisability

- Once value is written, all reads see same value
  - Regardless of replica read from
- Single-leader replication
  - Read from leader
  - Read from synchronous follower
- Multi-leader replication can't be linearised



## Linearisability

- Once value is written, all reads see same value
    - Regardless of replica read from
  - Single-leader replication
    - Read from leader
    - Read from synchronous follower
  - Multi-leader replication can't be linearised
  - Leaderless replication
    - Lock value on quorum before writing
- Abstraction over replicated database
  - Used when uniqueness needs to be guaranteed
  - e.g. Multiple withdrawals from an account
  - SLR – defeats most performance benefits
  - Leaderless – similar performance cost to SLR

#### Causal Order

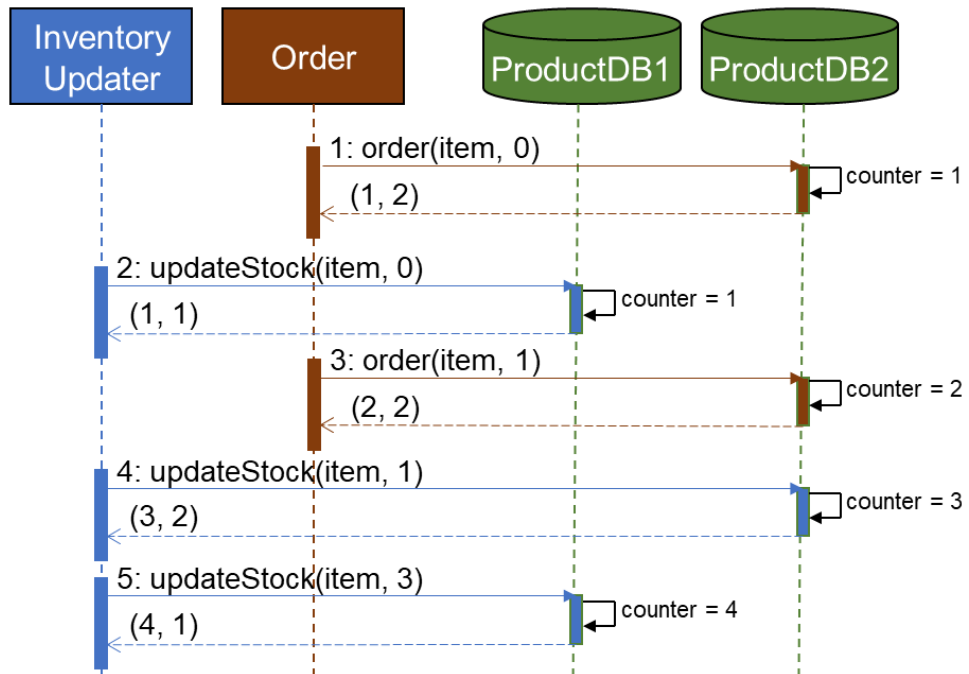
- Order is based on causality
  - What event needs to happen before another
  - Allows concurrent events

#### Causal Order

- Order is based on causality
  - What event needs to happen before another
  - Allows concurrent events
- Single-leader replication
  - Record sequence number of writes in log
  - Followers read log to execute writes

## Causal Order

- Order is based on causality
    - What event needs to happen before another
    - Allows concurrent events
  - Single-leader replication
    - Record sequence number of writes in log
    - Followers read log to execute writes
  - Lamport timestamps
- Linearisation defines a total order
  - Causal ordering defines a partial order
  - e.g. Git repo history with branching as causal order
  - Not as strict as linearisability, so less performance cost



### Definition 2. Consensus

A set of nodes in the system agree on some aspect of the system's state.

Abstraction to make it easier to reason about system state.

## Consensus Properties

**Uniform Agreement** All nodes must agree on the decision

**Integrity** Nodes can only vote once

**Validity** Result must have been proposed by a node

**Termination** Every node that doesn't crash must decide

- Uniform agreement and integrity are key
- Validity avoids nonsensical solutions (e.g. always agreeing to a null decision)
- Termination enforces fault tolerance, it requires that progress is made towards a solution

### Definition 3. Atomic Commit

All nodes participating in a distributed transaction need to form consensus to complete the transaction.

Based on transaction atomicity from ACID.

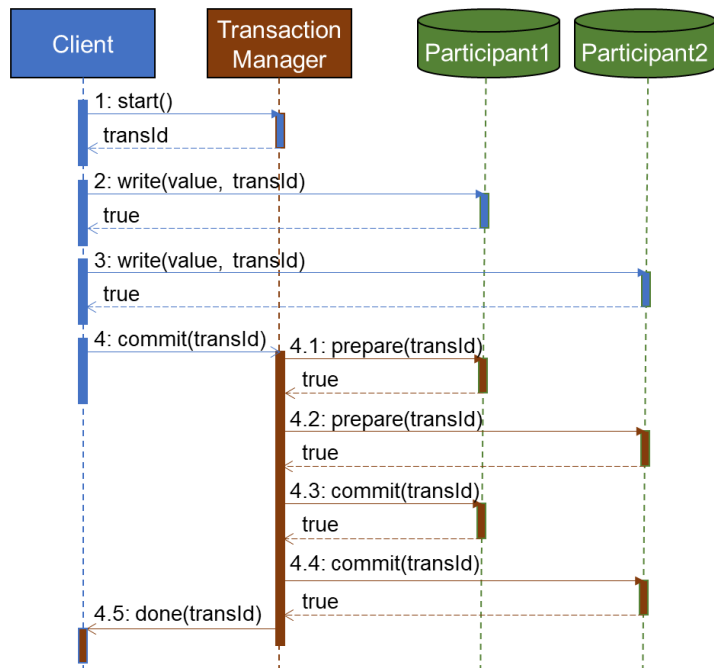


## Two-Phase Commit

**Prepare** Confirm nodes can commit transaction

**Commit** Finalise commit once consensus is reached

- Abort if consensus can't be reached



- Transaction ID used to track writes
- Prepare does all steps of a commit, aside from confirming it – It cannot be revoked by participant
- Commit intent is recorded in log before sending to participants
- Even if a participant fails, commit can proceed when it recovers
- Comment on performance costs