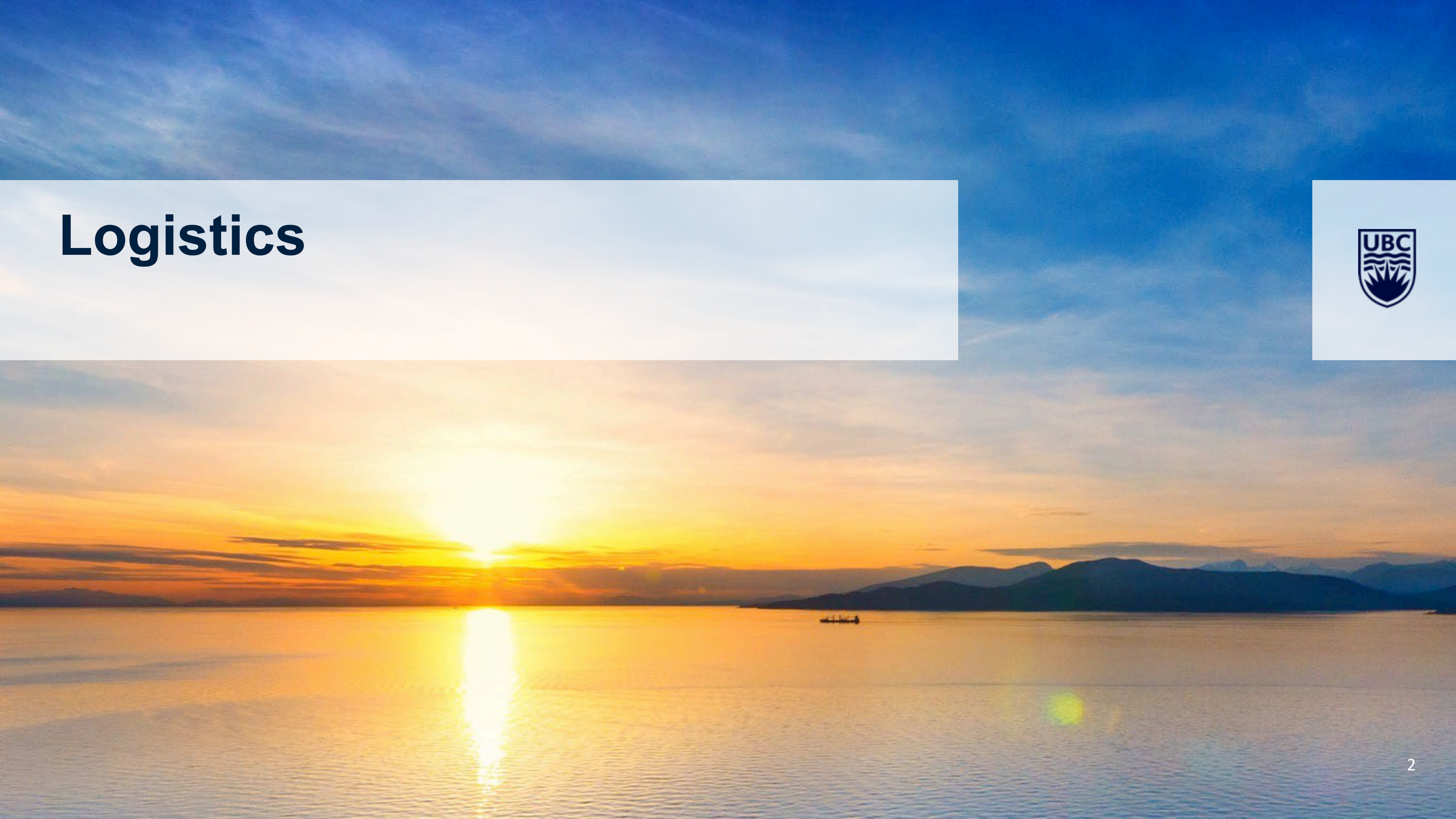
CPSC 416 Distributed Systems

Winter 2022 Term 2 (April 13, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer



Logistics

Deadlines

Project 4 Released. Late Due: April 13, 2023. Grades for on-time have been returned.

Project 5 Released Due: April 13, 2023. **No extensions**.

All project work is due **today** April 13, 2023. Late projects are scaled to 75% of the on-time max.

Final Exam: April 20, 2023, DMP 310, 08:30-11:00.

Deadlines

Alternate Path 1 & 2: Review in progress

- Piazza private threads need TLC
    - **Weekly updates due each Monday @ 23:59 PT**
- Final reports due no later than Thursday April 13, 2023 @ 23:59 PT

Instructor Office Hours:

- Zoom Office Hours (Tuesday) @ 13:00-14:00
- Discord (Casual) Office Hours (Thursday) @ 14:00-15:00

TA Office Hours:

- Eric: Friday 9-11 am (in-person and Zoom)
- Japraj: Wednesday 3-5 pm (Zoom)
- Yennis: Thursday 2-4 (Zoom), Friday 2-4 (in-person)

Final Examination

Final Examination Format

Final Examination will consist of **50** questions

There will be 10 **True/False** questions

There will be 40 **Multiple choice** questions

Each question will be worth 2 points. There is no penalty for making an incorrect choice.

The exam will have a maximum time of 2 hours.

You may have a single sheet of notes. **No electronic devices permitted**.

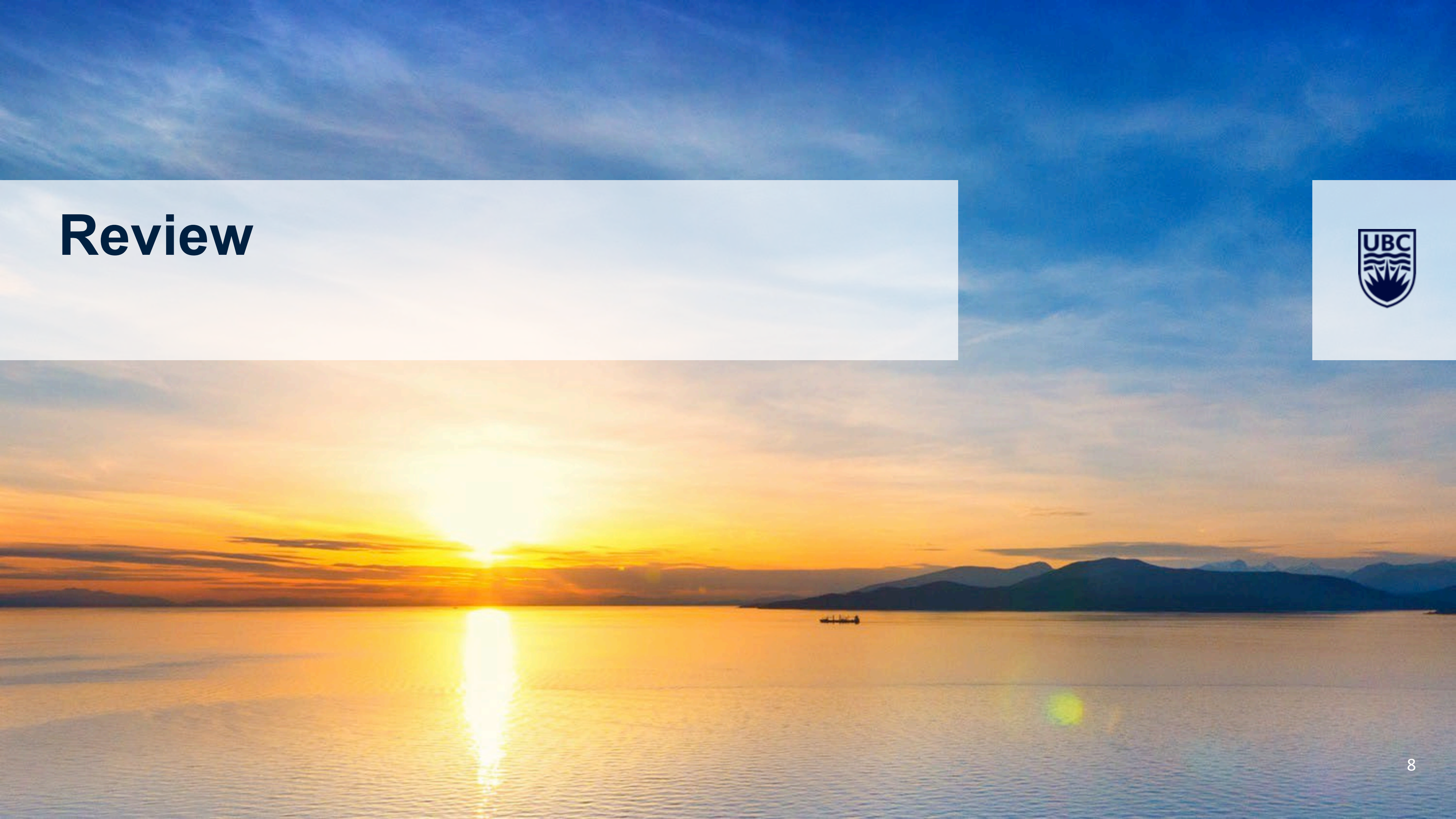
Resources

CPSC 416 2021W2 Practice Questions

Past exam papers: Concurrent and Distributed Systems

Piazza Post: Crowdsourcing the Final Exam

Review



Networking

TCP = reliable transmission over internet connection, unbounded size

UDP = unreliable transmission over internet connection, bounded size

IP = unreliable transmission over internet connection, bounded size

Remote Procedure Call

- Data marshalled into a common format
- Sent via a transport protocol (TCP or UDP)

Examples of Distributed Systems

Bitcoin

Ethereum

Hadoop File System

Cyber-physical systems

Folding@Home

Kafka

Domain Name Service (DNS)

Cloud Services offered by AWS, Azure, GCP, others

Distributed Databases

Motivation for Distributed Systems

Eliminate fate sharing

Improved availability

Enhanced performance and scaling

Characteristics

Use networking to exchange messages

Asynchronous in nature

Minimal central management

Resistant to failure

Key-Value Store

A **very simple database**

They form the root of RDBMS, NoSQL, and File systems

Databases construct resiliency against failures by:

- Relying upon a **persistent journal** (also known as a *database log*, or just log)
- Utilize transactions (ACID)
  - Atomic
  - Consistent
  - Isolated
  - Durable

Consistency

Consistency:

- Distributed systems: across multiple nodes
- Single view (at some point in time)

Strong Consistency:

- Updates are immediately available at all nodes

Eventual Consistency:

- Updates may be visible on some nodes but not others
- Eventually all nodes will be consistent

Causal Consistency:

- Updated to *causally related operations* is ordered
- Non-related operations, no guarantees

Failure Models

Fail-stop:

- Failed node **ceases operation**

Fail-slow:

- Failing node may respond, but increasingly slowly

Byzantine:

- Failing node provides incorrect/false information

Clock/Time

Absolute Time

Logical Time (Logical Clock)

Clock-based ordering

- Scalar
- Vector
- Matrix

Time Synchronization

Physical clocks: do not necessarily agree, drift, vary

NTP: synchronize clocks, may roll backwards

Global timestamps: solve problem, but have bound limitations

Logical time: captures causality in a distributed system

Database

Provide storage of information

- Persistent
- Structured

Guarantees:

- Consistency
- Concurrent access control
- Failure resiliency

Database journal/log

A *database journal*

- Key tool to ensure data consistency
- Used during database *recovery*
- Forms the basis of implementing ACID properties (even with failure)

Key concepts:

- Recovery **to a consistent state**
- Provides a simple abstraction for *replicating databases*

Key issues:

- Journal may be written *ahead* of the database
- Causally linked access to database *must* be ordered

Global State

Processes: active agents inside a distributed system

Channels: messages transiting the system

State Transitions: event driven

- Processing a *received message* permutes the process state.

Run: **any valid sequence of events**

Observation: different runs may have equivalent outcomes. These are *isomorphic*.

- Causally unlinked
- Causally linked and commutative operations

Distributed System State

Cut: snapshot

Consistent cut: cut that **obeys causality**

Inconsistent cut: cut that **cannot guarantee** causality

Global snapshot

- Simplest to *reason* about
- Not realistic (why?)

Recall: Chandy-Lamport Algorithm. We get *a* possible (consistent) state.

Consensus

Agreement between distributed processes on shared state

- ***Value***
- ***Action***
- ***Timestamp***
- ***Transaction outcome***

Consensus allows a system to be correct

FLP

Model: asynchronous, single fault, fail-stop

Basic point:

- It is possible that the “deciding message” can be delayed arbitrarily long
- If the “deciding message” should have been sent by the failed node, the system may not ever reach consensus (e.g., the deciding message is the one the failed node should have sent.)

Replication

Primary-Backup

Chain Replication

CRAQ

Fault Tolerance

Fault: software/hardware exhibit incorrect or unexpected behaviour

Error: observe a fault

Question: why are unobserved faults dangerous in distributed systems?

Recovery:

- We use persistent information to return to consistency
- Checkpoints, Journals, Replication

A Survey of Rollback-Recovery Protocols in Message-Passing Systems

Corfu

Transactions

2 Phase Commit

2 Phase Locking

TrueTime

3 Phase Commit (asynchronous 2PC)

Consensus

Quorum Consensus:

- Intersection of readers and writers
- Weighted voting
- Crumbling walls (matrix consensus, with a minority of nodes)

CAP Theorem

Consistency

- Strongest consistency model is **sequential** consistency
- Weaker models exist (e.g., linear consistency)

Availability

- Can we continue servicing clients

Partition tolerance

- Networks are garbage
- Many failures are *transient*
- How to handle *without* sacrificing **C** or **A**?

Consensus Algorithms

Three common elements:

- Let’s pick a leader
- Let’s propose some mutations
- Let’s record our decisions

Why?

- Leader driven approaches fit best with 2PC/3PC systems
- We need to be able to *reliably* update our database
- Journal/log allows us to recover nodes

Challenges: ensuring the nodes that did *not* decide don’t give the wrong answer

Algorithms

Viewstamped Replication

- Pick the leader by network address
- View defines the current epoch (leader)
- Uses quorum consensus to replicate data

Paxos

- Leaderless consensus protocol
  - Anyone can propose
  - Reaches a consensus decision
- Can be “chained together”
  - Multi-Paxos: pick a leader, then leader makes proposals.
  - Combine *ballot* with *proposal*. De-facto *view* model

Algorithms

PMMC

- Implements multi-Paxos
- Focuses on *log value selection*
- Includes **many optimizations**

Raft

- Implements Viewstamped Replication (leader selected by network addres)
- Focuses on *log value selection*
- Includes **many optimizations**

Note: PMMC and Raft are *different* because the way they organize their logs is different.

Formal Verification

Model construction

TLA+

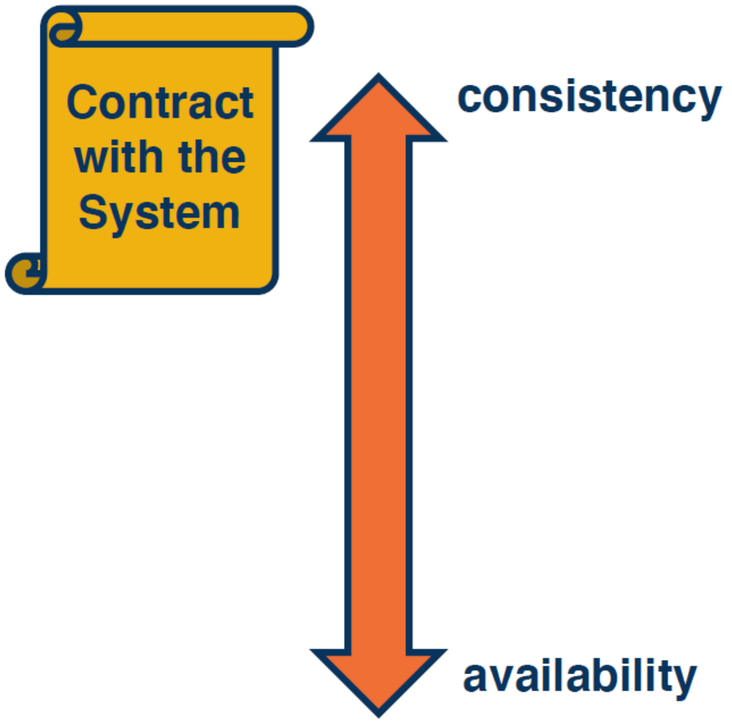
Consistency Models

Strong consistency, linearizability

Sequential consistency: guaranteed single ord

Causal consistency: enforce “happens before”

Eventual consistency: recover from transient e

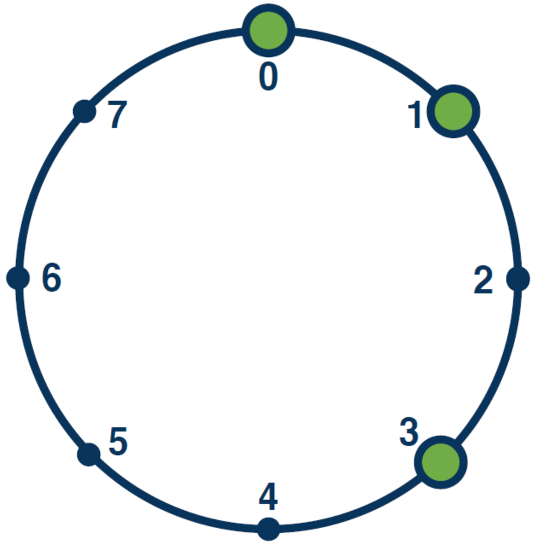


Chord Distributed Hash Table Ring

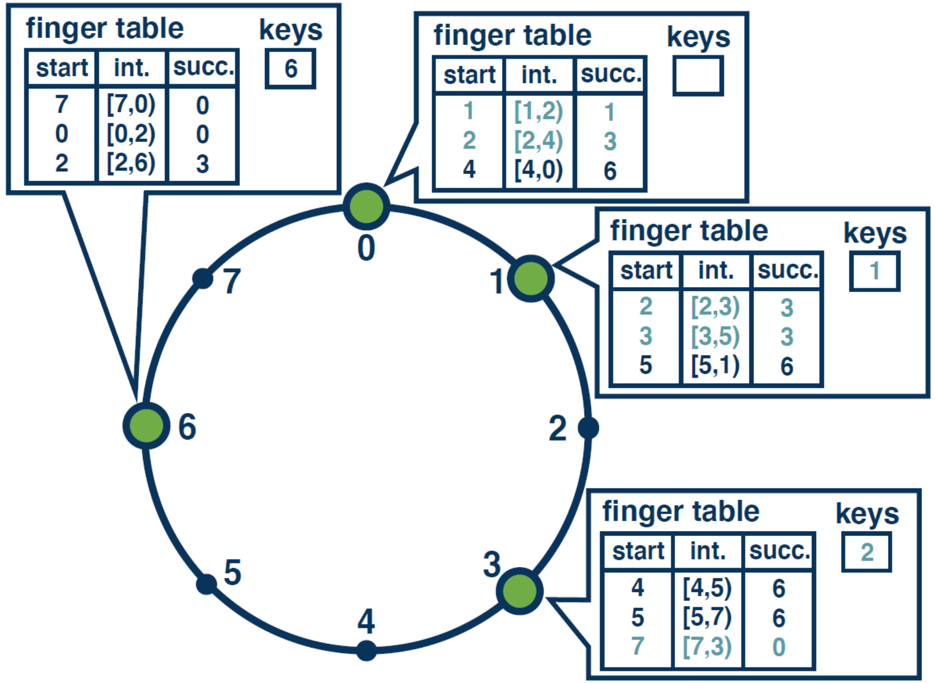
Use cryptographic secure hash algorithm (SHA)

- Maps keys to a fixed length numeric value
- Maps IP addresses to a fixed length numeric value

Ring is N nodes {0,…N-1}



Chord: Managing the Ring



finger table

start	int.	succ.
7	[7,0)	0
0	[0,2)	0
2	[2,6)	3

keys

6

finger table

start	int.	succ.
1	[1,2)	1
2	[2,4)	3
4	[4,0)	6

keys

finger table

start	int.	succ.
2	[2,3)	3
3	[3,5)	3
5	[5,1)	6

keys

1

finger table

start	int.	succ.
4	[4,5)	6
5	[5,7)	6
7	[7,3)	0

keys

2

Nodes joining and departing

Redistributed data

Update finger tables

Improve performance with additional metadata

Probabilistic system performance guarantees

Lesson Review

Final Examination Format

Final Examination will consist of **50** questions

There will be 10 **True/False** questions

There will be 40 **Multiple choice** questions

Each question will be worth 2 points. There is no penalty for making an incorrect choice.

The exam will have a maximum time of 2 hours.

You may have a single sheet of notes. **No electronic devices permitted**.

Questions?

THE UNIVERSITY OF BRITISH COLUMBIA

THE UNIVERSITY OF BRITISH COLUMBIA