Lecture Roadmap

• Fault tolerance
• Consensus
• Recall: Paxos
• Case Study: BFT
Failure Handling in Distributed Systems

- **Post-Failure Troubleshooting**
  1. Diagnosis tools
  2. Accountability
  3. Provenance
  4. ... ...

- **Failure Prevention**
- **Changing network paths**
- **Upgrading software components**
- **Outage**

**Service Runtime**

- **Service initialization**
- **Failure Prevention**
- **Failure Tolerance**
Fault Tolerance

- Distributed systems replicate data across multiple servers

![Diagram showing server and replica connections]
Fault Tolerance

- Distributed systems replicate data across multiple servers
  - Replication provides fault-tolerance if servers fail
Fault Tolerance

- Distributed systems replicate data across multiple servers
  - Replication provides fault-tolerance if servers fail

Can we replicate the data across 100,000 nodes?
Distributed systems replicate data across multiple servers:
- Replication provides fault-tolerance if servers fail.
- Allowing clients to access different servers potentially increasing scalability (max throughput).
Fault Tolerance

Client (Beijing)

Server1

Server2

Server3

Client (DC)
Fault Tolerance

Client (Beijing) $\xrightarrow{W(X,1)}$ Server1 $X=0$  
   
   Server2 $X=0$  
   
   Server3 $X=0$  

Client (DC)
Fault Tolerance

Client (Beijing) \( W(X,1) \) Server1 \( X=1 \) Server2 \( X=? \) Server3 \( X=? \)  

R(X) = 1 or 0?  

Client (DC)
Fault Tolerance

Client (Beijing) → Server1 (X=1) → Server2 (X=1) → Server3 (X=1) → Client (DC)

W(X,1) → R(X)=1
Fault Tolerance

Client (Beijing) \( W(X,1) \) Server1 \( X=1 \) Server2 \( X=1 \) Server3 \( X=1 \)

Client (DC) \( R(X)=1 \)

Client (Beijing) \( W(X,1) \) Server1 \( X=1 \) Server2 \( X=0 \) Server3 \( X=0 \)

Client (DC) \( R(X)=0 \)
Fault Tolerance

If I made a 100,000-way replication, my system can achieve availability but not consistency.
We want consistency
Two-Phase Commit Protocol

- Phase 1: Voting phase
  - Get commit agreement from every participant
Two-Phase Commit Protocol

- Phase 1: Voting phase
  - Get commit agreement from every participant
Two-Phase Commit Protocol

• Phase 1: Voting phase
  - Get commit agreement from every participant
  - A single “no” response means that we will have to abort
Two-Phase Commit Protocol

- Phase 2: Commit phase
  - Send the results of the vote to every participant
  - Send abort if any participant voted “no” in Phase 1
Two-Phase Commit Protocol

- Phase 2: Commit phase
  - Get “committed” acknowledgements from every participant
Two-Phase Commit Protocol

- If some nodes fail?
  - Lose availability
  - We need to wait for a long time

Using two-phase can guarantee the consistency, but not fault tolerance
CAP Theorem

Consistency

Availability

Partition Tolerance

Consistency

Availability

Partition tolerance

Paxos

Gossip

2PC
Very Tricky Fault

**ZooKeeper** (synchronization service)
**Issue #335.**

1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader

---

PERMANENT INCONSISTENT REPLICA
**Very Tricky Fault**

**ZooKeeper** (synchronization service)  
**Issue #335.**

1. Nodes A, B, C start (w/ latex txid: 10)  
2. B becomes leader  
3. B crashes  
4. C becomes leader  
5. C commits new txid-value pair (11, X)  
6. A crashes, before committing the new txid 11  
7. C loses quorum and crashes  
8. A and B are back online after C crashes  
9. A becomes leader  
10. A's commits new txid-value pair (11, Y)  
11. C is back online after A's new tx commit  
12. C announces to B (11, X)  
13. B replies diff starting with tx 12  
14. Inconsistency: A has (11, Y), C has (11, X)

**PERMANENT INCONSISTENT REPLICA**
**Very Tricky Fault**

**ZooKeeper** (synchronization service)  
**Issue #335.**

1. Nodes A, B, C start (w/ latex txid: 10)  
2. B becomes leader  
3. B crashes  
4. C becomes leader  
5. C commits new txid-value pair (11, X)  

---

**PERMANENT INCONSISTENT REPLICA**
Very Tricky Fault

ZooKeeper (synchronization service)
Issue #335.

1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader
3. B crashes
4. C becomes leader
5. C commits new txid-value pair (11, X)
6. A crashes, before committing the new txid 11

PERMANENT INCONSISTENT REPLICA
Very Tricky Fault

**ZooKeeper** (synchronization service)

**Issue #335.**

1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader
3. B crashes
4. C becomes leader
5. C commits new txid-value pair (11, X)
6. A crashes, before committing the new txid 11
7. C loses quorum and C crashes

---

**PERMANENT INCONSISTENT REPLICA**
Very Tricky Fault

**ZooKeeper** (synchronization service)
Issue #335.

1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader
3. B crashes
4. C becomes leader
5. C commits new txid-value pair (11, X)
6. A crashes, before committing the new txid 11
7. C loses quorum and C crashes
8. A and B are back online after C crashes
9. A becomes leader
10. A's commits new txid-value pair (11, Y)
11. C is back online after A's new tx commit
12. C announce to B (11, X)
13. B replies diff starting with tx12
14. Inconsistency: A has (11, Y), C has (11, X)

PERMANENT INCONSISTENT REPLICA
**Very Tricky Fault**

**ZooKeeper** (synchronization service)

**Issue #335.**

1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader
3. B crashes
4. C becomes leader
5. C commits new txid-value pair (11, X)
6. A crashes, before committing the new txid 11
7. C loses quorum and C crashes
8. A and B are back online after C crashes
9. A becomes leader
10. A's commits new txid-value pair (11, Y)

**PERMANENT INCONSISTENT REPLICA**
**Very Tricky Fault**

**ZooKeeper** (synchronization service)

**Issue #335.**

1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader
3. B crashes
4. C becomes leader
5. C commits new txid-value pair (11, X)
6. A crashes, before committing the new txid 11
7. C loses quorum and C crashes
8. A and B are back online after C crashes
9. A becomes leader
10. A’s commits new txid-value pair (11, Y)
11. C is back online after A’s new tx commit
12. C announces to B (11, X)
13. B replies diff starting with tx 12
14. Inconsistency: A has (11, Y), C has (11, X)

**PERMANENT INCONSISTENT REPLICA**
Lecture Roadmap

- Fault tolerance
- **Consensus**
- Recall: Paxos
- Case Study: BFT
Consensus

• Definition:
  - A general agreement about something
  - An idea or opinion that is shared by all the people in a group

• Given a set of processors, each with an initial value:
  - **Termination**: All non-faulty processes eventually decide on a value
  - **Agreement**: All processes that decide do so on the same value
  - **Validity**: The value that has been decided must have proposed by some process
Consensus / Agreement Problem

• Goal: N processes want to agree on a value

• Correctness:
  - All N nodes agree on the same value
  - The agreed value has been proposed by some node
Consensus / Agreement Problem

• Goal: \( N \) processes want to agree on a value

• Correctness:
  - All \( N \) nodes agree on the same value
  - The agreed value has been proposed by some node

• Fault-tolerance:
  - If \( \leq F \) faults in a window, consensus reached eventually
  - Liveness not guaranteed: If \( > F \) faults, no consensus
  - Given goal of \( F \), what is \( N \)? Depends on fault model ("Crash fault" need \( 2F+1 \); Byzantine fault needs \( 3F+1 \))
**Consensus / Agreement Problem**

- **Goal:** N processes want to agree on a value

- **Correctness:**
  - All N nodes agree on the same value
  - The agreed value has been proposed by some node

- **Fault-tolerance:**
  - If $\leq F$ faults in a window, consensus reached eventually
  - Liveness not guaranteed: If $> F$ faults, no consensus
  - Given goal of F, what is N? Depends on fault model ("Crash fault" need 2F+1; Byzantine fault needs 3F+1)

  Odd Number
Lecture Roadmap

• Fault tolerance
• Consensus
• Recall: Paxos
• Case Study: BFT
Paxos

• **Safety (nothing bad happens):**
  - Only a single value is chosen
  - Only a proposed value can be chosen
  - Only chosen values are learned by processes

• **Liveness (some good things happen eventually):**
  - Some proposed value eventually chosen if fewer than half of processes fail
  - If value is chosen, a process eventually learns it
Paxos

- Three conceptual roles:
  - **Proposers**: propose values
  - **Acceptors**: accept values, where chosen if majority accept
  - **Learners**: learn the outcome (the chosen value)
Proposer A

Prepare request

\[ n=2, v=5 \]

Proposer B

Acceptor A

Acceptor B

Acceptor C

Learner
Paxos

Proposer A

Proposer B

Acceptors:
- Acceptor A
  - $[n=2, v=5]$
- Acceptor B
  - $[n=2, v=5]$
- Acceptor C

Learner
Paxos

Proposer A

Proposer B

Acceptor A

Acceptor B

Acceptor C

Learner

Prepare request

Prepare response

Prepare response

[n=2, v=5]

[no previous]

[no previous]
Paxos

Proposer A

Proposer B

Acceptor A

Acceptor B

Acceptor C

Learner

Prepare request

[n=2, v=5]

Prepare request

[n=4, v=8]

Prepare request

[n=2, v=5]

Prepare response

[no previous]

Prepare response

[no previous]
The Paxos algorithm is illustrated in the diagram. It involves a proposer, acceptors, and a learner. The proposer initiates a proposal with a value. Each acceptor processes the proposal and responds. The learner then acknowledges the accepted proposal. The diagram shows the sequence of messages exchanged during the Paxos process, including prepare requests and responses.
Proposer A

Proposer B

Acceptor A

Acceptor B

Acceptor C

Learner

Prepare request
[n=2, v=5]

Prepare response
[no previous]

Prepare response
[no previous]

Prepare request
[n=4, v=8]

Accept request
[n=4, v=8]
Proposer A
Proposer B
Acceptor A
Acceptor B
Acceptor C
Learner

Paxos

Prepare request
\[n=2, v=5\]
Prepare request
\[n=4, v=8\]
Prepare request
\[n=2, v=5\]
Prepare request
\[n=4, v=8\]
Prepare request
\[n=2, v=5\]
Prepare request
\[n=4, v=8\]

Prepare response
\[no previous\]
Prepare response
\[no previous\]
Prepare response
\[no previous\]
Prepare response
\[no previous\]
Prepare response
\[no previous\]
Prepare response
\[no previous\]

Accept request
\[n=4, v=8\]
Accept request
\[n=4, v=8\]
Accept request
\[n=4, v=8\]
Accept request
\[n=4, v=8\]
Accept request
\[n=4, v=8\]
Accept request
\[n=4, v=8\]

Chosen value
\[v=8\]
If we do not wait for the majority

Preparation request

Proposer A

Proposer B

Acceptor A

Acceptor B

Acceptor C

Learner

\[ n=2, v=5 \]
If we do not wait for the majority

Proposer A
[n=2, v=5]

Proposer B

Acceptor A
[n=2, v=5]

Acceptor B
[n=3, v=0]

Acceptor C
[n=4, v=1]

Learner
Lecture Roadmap

• Fault tolerance
• Consensus
• Recall: Paxos
• Case Study: BFT