DISTRIBUTED
CONSSENSUS/CONSISTENCY-
PART 1: CONTEXT (TWO-GENERAL
PROBLEM AND APPLICATIONS)

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)
**Example: Changing Course Requirement**

1. Someone proposes a change
2. Consensus: Those present in class vote on it based on notion of consistency
3. Consistency: Those involved learn about the vote
4. Consensus/Consistency: We reach agreement on who knows what (the result of the vote)
CONSISTENCY VS. CONSENSUS

- Consensus
  - Algorithm for coupled processes reach agreement on whether request made by a process should be executed.
  - Multiple forms of consensus

- Consistency
  - Algorithm for coupled processes to react in some well defined way to an agreed change
  - Multiple kinds of consistency.

- Awareness/consensus
  - Consistency does not occur instantaneously
  - Algorithm to discover what others know
  - Multiple forms of agreement
**Scope**

- Intuition and relationship among well-known, practical algorithms
- Design space with intermediate algorithms
- Extendible abstractions covering the design space
- Breadth rather than depth!
TWO-GENERAL PROBLEM

Leading General

Attack time: At Dusk

Attack time: At Dusk
TWO-GENERAL PROBLEM: SEND MESSENGER

Attack time: At Dawn

Attack time: At Dusk
TWO-GENERAL PROBLEM: SEND ACKER

Attack time: At Dawn

Attack time: At Dawn
TWO-GENERAL PROBLEM: SEND ACK ACKER

Attack time: At Dawn

Attack time: At Dusk
Two-General Problem: Ack*

Theoretical Limit: Can never achieve consensus

Attack time: At Dawn
TWO-GENERAL PROBLEM: SEND MESSENGER

Attack time: At Dawn

Attack time: At Dusk
TWO-GENERAL PROBLEM: SEND DUPLICATE

Attack time: At Dawn

Attack time: At Dawn
TWO-GENERAL PROBLEM SOLUTION

- Sending General Algorithm
  - If # retransmissions < R, retransmit after time $T^1$
  - Consensus achieved iff
    - Ack received after less than R retransmissions

- Obeying General Algorithm
  - Send ack in response to each duplicate message
  - Consensus achieved iff
    - No retransmission received within time $T^2 = f(R, T1)$

- Consensus cannot be achieved instantaneously!
GENERAL PROBLEM

Practical examples of consistency?

Coupled Process

Object $O^1$: Value $^1$

Object $O^2$: Value $^2$

Coupled Process

Object $O^1$: Value $^1$

Object $O^2$: Value $^2$
In GIPC, on completion of a remote function call, the called site sends back to the calling site the return value (or an exception) along with an ID identifying the call.

- True
- False

Reset Selection
CONSISTENCY OF NESTED TRANSACTION COMMITS
PHYSICAL REPLICA DIFF CONSISTENCY

Reliability

Sharing with Performance

State shared: Diffs

Writeable Server Physical Replica

Writeable Client Physical Replica

Writeable Client Physical Replica

Writeable Client Physical Replica

Writeable Client Physical Replica

Writeable Client Physical Replica
Physical Replica Snapshot Consistency

Reliability

Sharing with Performance

State shared: Snapshot

Writeable Server Physical Replica\(^1\)

Writeable Client Physical Replica\(^1\)

Writeable Client Physical Replica\(^2\)

Writeable Client Physical Replica\(^3\)
**Physical Replica Command Consistency**

Sharing with Performance

State shared: Next command

- **Writeable Server Physical Replica**
- **Writeable Client Physical Replica\(^2\)**
- **Writeable Client Physical Replica\(^3\)**
META (CONSISTENCY-STATE) CONSISTENCY

Sharing with Performance

State shared: Locks, Consistency, Communication Protocols

Writeable Server Physical Replica¹

Writeable Client Physical Replica²

Writeable Client Physical Replica³
Lock Consistency

Sharing with Performance

State shared: Locks

Writeable Server Physical Replica

Writeable Client Physical Replica

Writeable Client Physical Replica

One directional arrows?
Physical Replica: Mirroring

- Writeable Server\(^1\)
  - Readonly Mirror\(^2\)
  - Readonly Mirror\(^3\)

Sharing with Performance
AKAMAI MIRRORING

How It Works

Akamai's globally-distributed network of servers pulls and caches content at the edge of the Internet for superior whole site delivery.

Web server maintained by Akamai customer for publishing content.

The edge server pulls fresh content as needed via an optimized connection.

From google images search (Akamai Architecture)
Classifying Consistency Scenarios

- Multiple client UIs commit to single server
  - Browser-Sakai
- Nested transaction involving multiple logical servers
  - Travelocity
- Physical replication with multiple changers
  - Diff-based with divergence (Git)
  - Snapshot-based (Google Drive, OneDrive)
  - Command-based: replicated state machines (Google Docs, LiveMeeting)
- Lock and other meta/configuration state
  - Live Meeting
- Physical mirroring
  - Akamai
What if writeable server fails?
Failing writeable client or readonly mirror can get data from writeable server

What if writeable server fails?

**Log File**

1. **Writing Client**
2. **Writable Server**
3. **Log File**
4. **Log file error?**
5. **Multiple disks (RAID) to create “stable” storage**
6. **Master corruption or stable storage error**
7. **Down time**
**Static Master-Slave Replication**

- **Writing Client**
- **Writable Master**
- **Log File**
- **Readonly Slave**
- **Readonly Slave**
- **Reading Client**
- **Down time**

Diagram showcases the flow of data and replication in a static master-slave configuration.
Do slaves subsume mirrors?
Location of Slaves?

Synchronization time can be high if slaves are on the edge.
Data Centers

Synchronization time low if slaves are in master’s data center

Slaves provide fault tolerance and read load balancing rather than edge performance

Multiple data centers for some service?

Geographically dispersed writes (and reads) and power-failure tolerance
Multiple masters can provide write load balancing in a single or multiple data centers, and power-fault tolerance in multiple centers.

Multiple masters can provide edge performance for geographically dispersed writes.
**Master-Master Replication**

Solution: Increase probability of disjoint writes (Client writing to some account writes to home or nearby data center)

Unsequenced conflicts in overlapping writes (account creation still possible)
CLASSIFYING CONSISTENCY ScENarios

- Multiple client UIs commit to single server
  - Browser-Sakai
- Nested transaction involving multiple logical servers
  - Travelocity
- Physical replication with multiple changers
  - Diff-based with divergence (Git)
  - Snapshot-based (Google Drive, OneDrive)
  - Command-based: replicated state machines (Google Docs, LiveMeeting)
- Lock and other meta/configuration state
  - Live Meeting
- Physical mirroring
  - Akamai
- Master (primary)-slave(backup) replication
- Master-master replication
  - Disjoint unrelated writes to different sequencing masters
  - Overlapping writes