Distributed Consensus/Consistency- Part 1: Context (Two-General Problem and Applications)

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)

Example: Changing Course Requirement

Someone proposes a change

Consensus: Those present in class vote on it based on notion of consistency

Consistency: Those involved learn about the vote

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

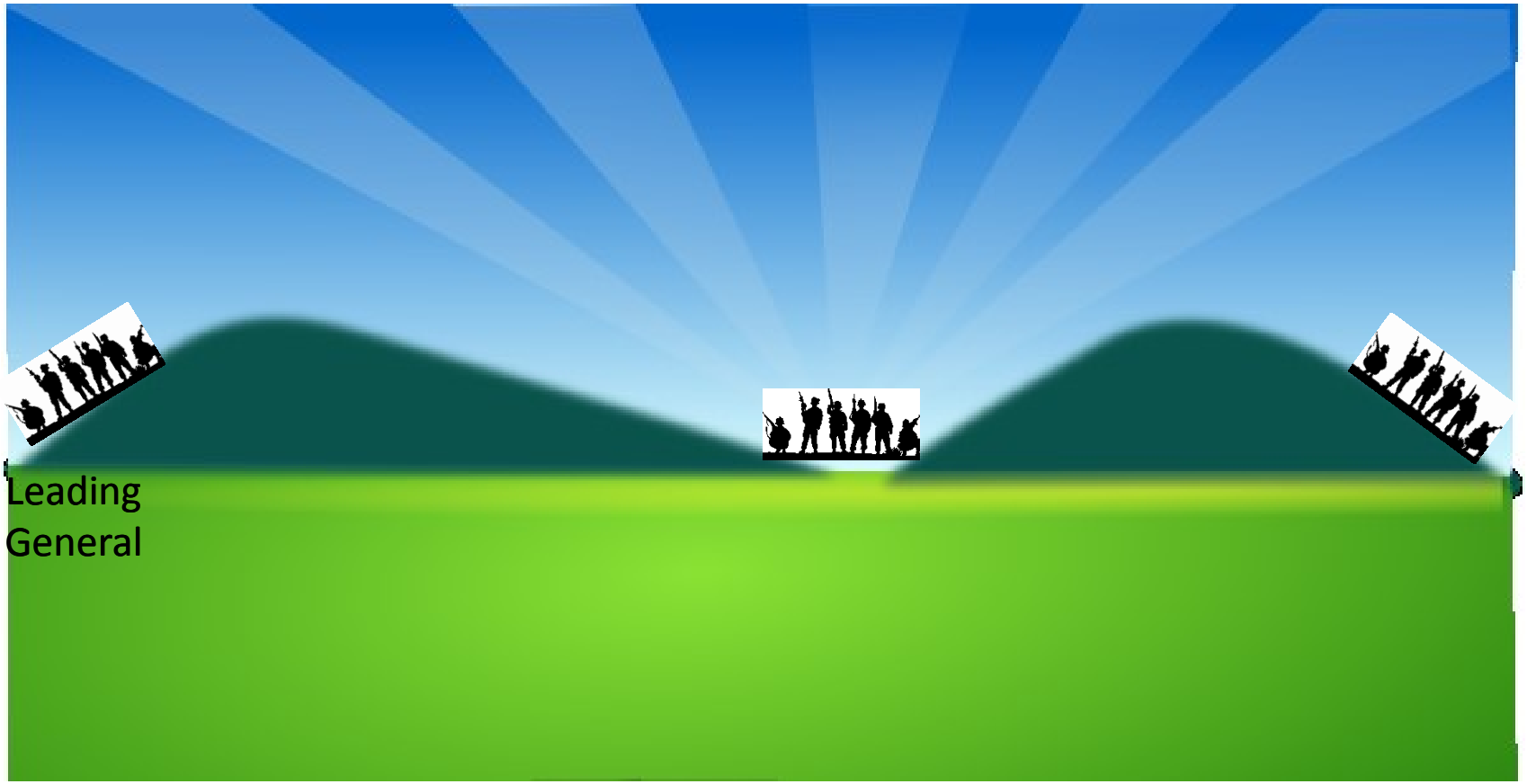
Intuition and relationship among well-known, practical algorithms

Design space with intermediate algorithms

Extendible abstractions covering the design space

Breadth rather than depth!

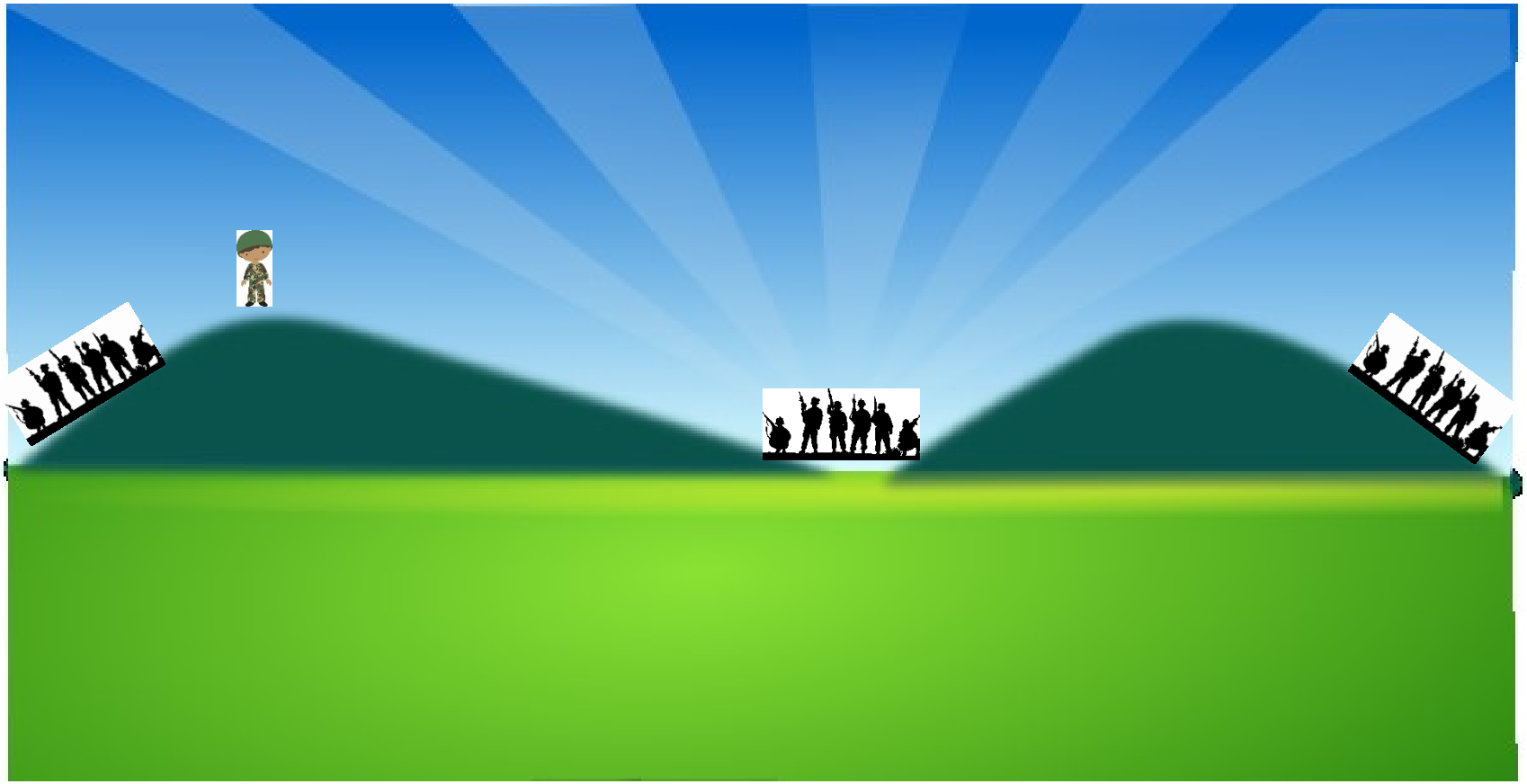
Two-General Problem



Attack time: At Dusk

Attack time: At Dusk

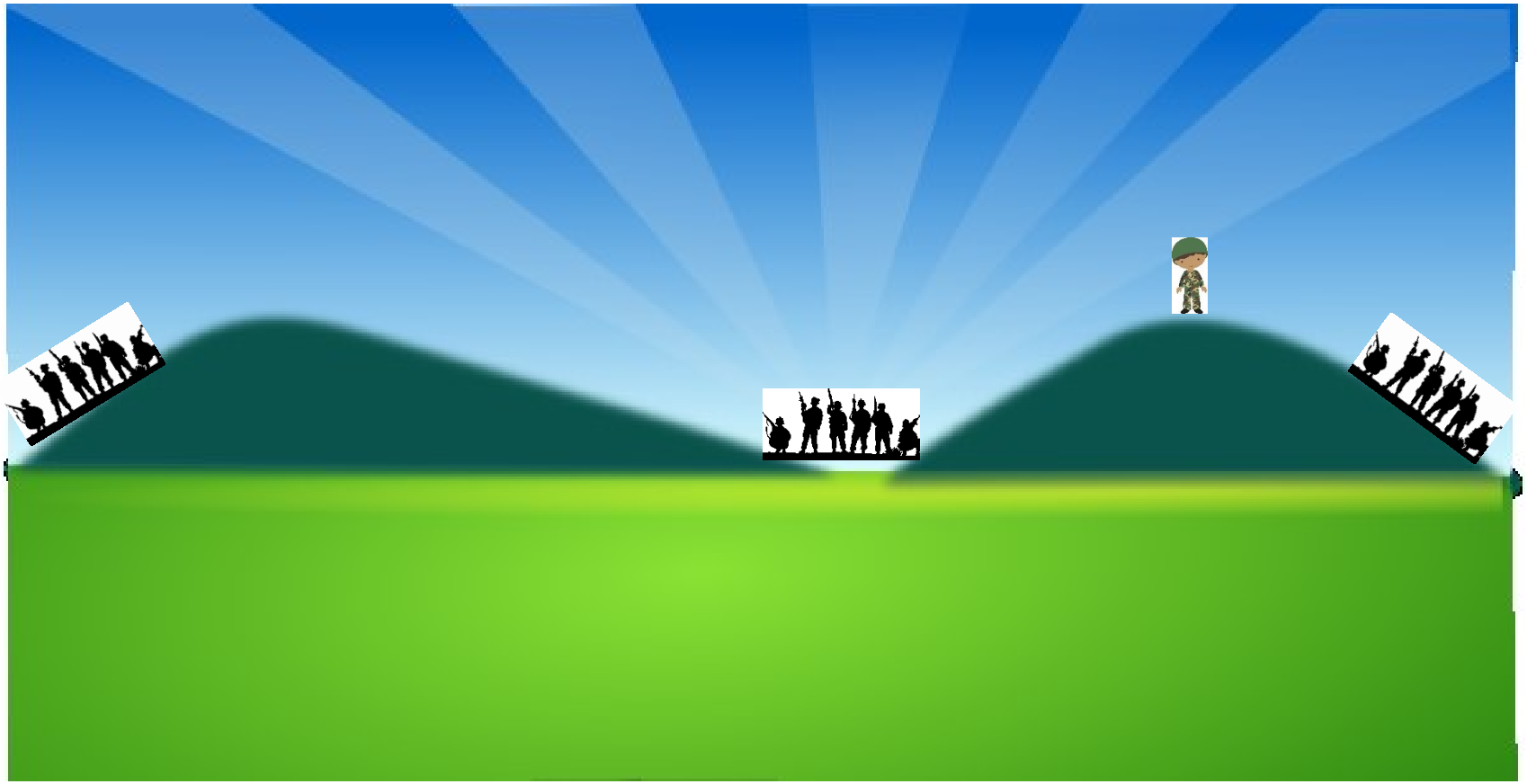
Two-General Problem: Send Messenger



Attack time: At Dawn

Attack time: At Dusk

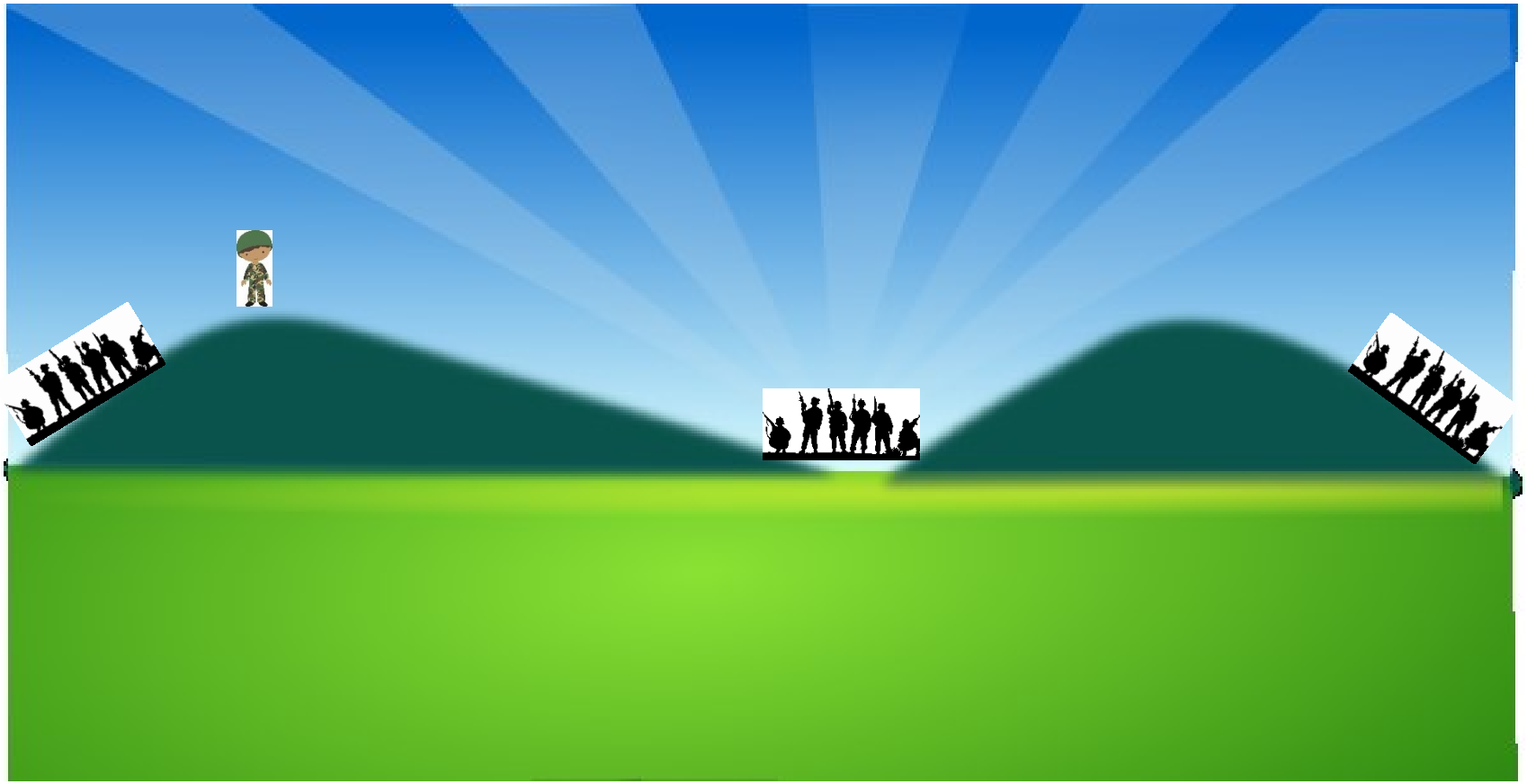
Two-General Problem: Send Acker



Attack time: At Dawn

Attack time: At Dawn

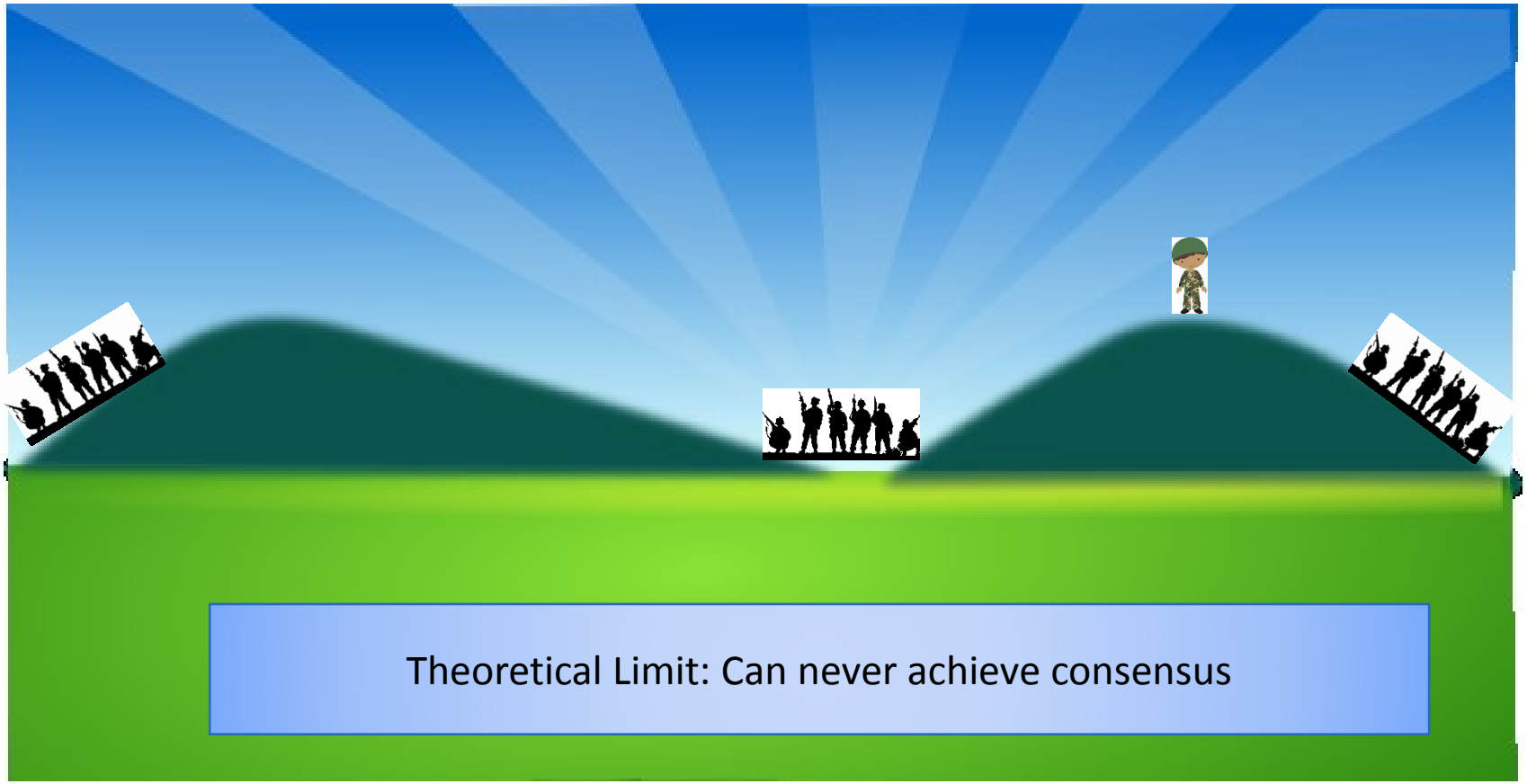
Two-General Problem: Send Ack Acker



Attack time: At Dawn

Attack time: At Dusk

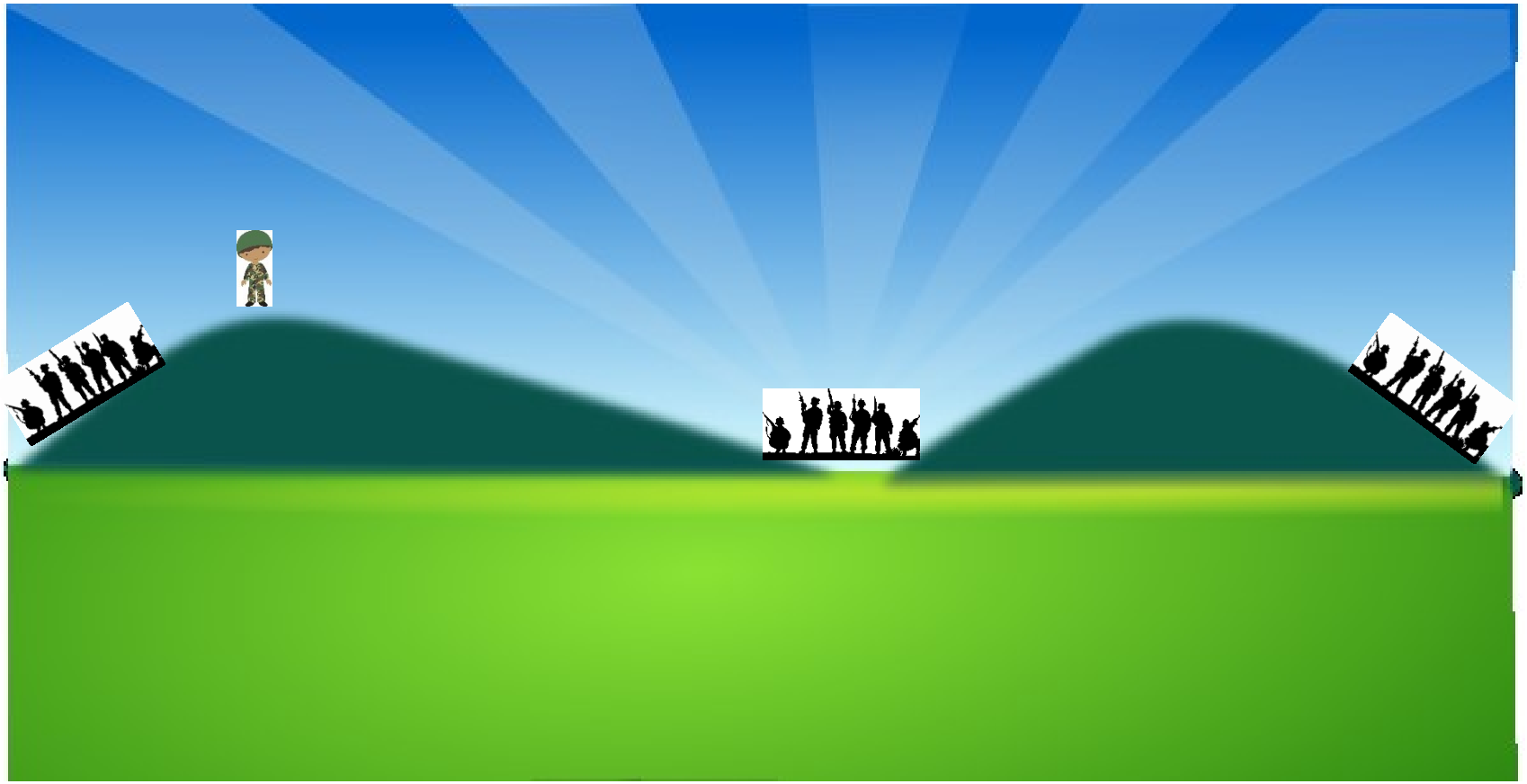
Two-General Problem: Ack*



Attack time: At Dawn

Attack time: At Dawn

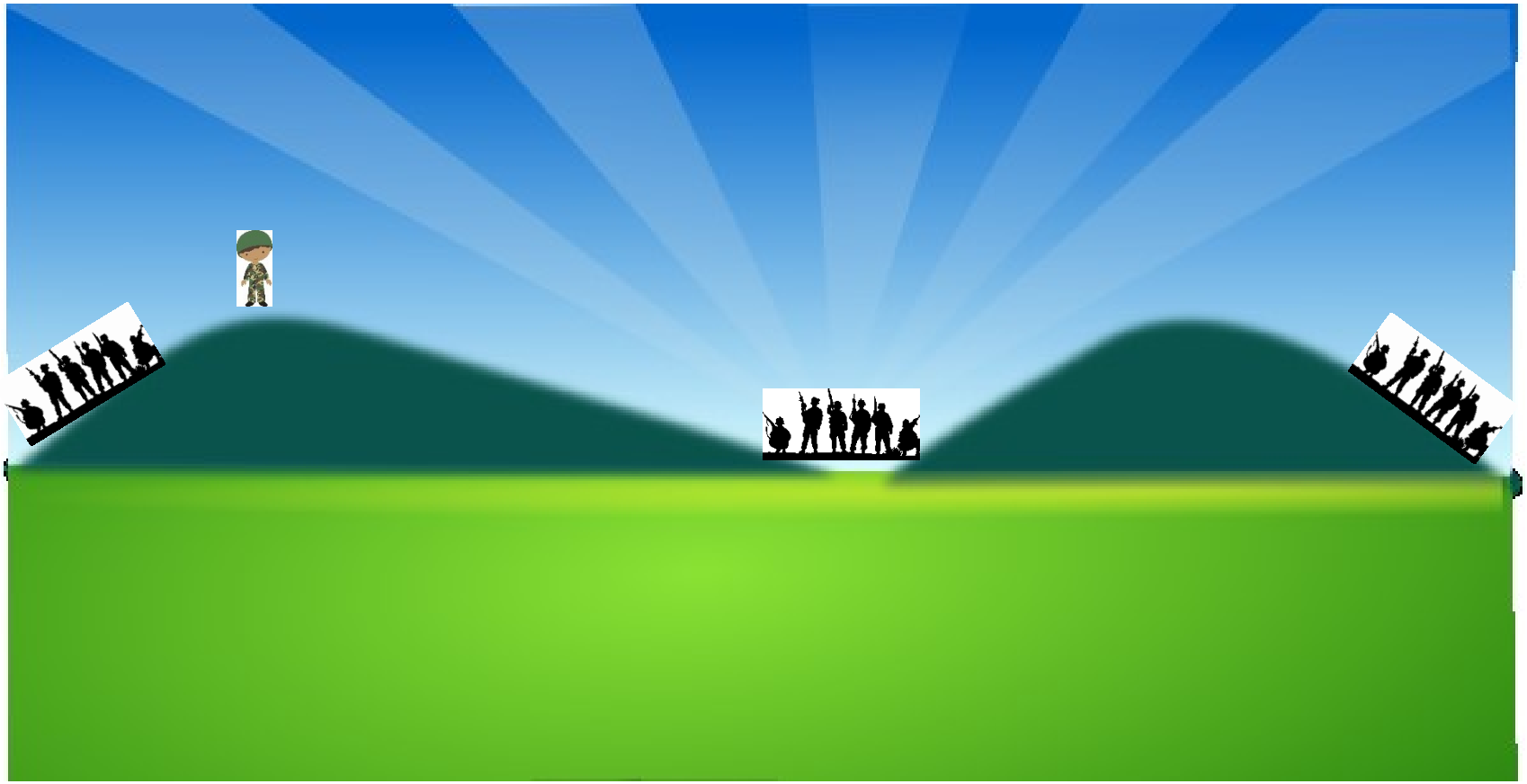
Two-General Problem: Send Messenger



Attack time: At Dawn

Attack time: At Dusk

Two-General Problem: Send Duplicate



Attack time: At Dawn

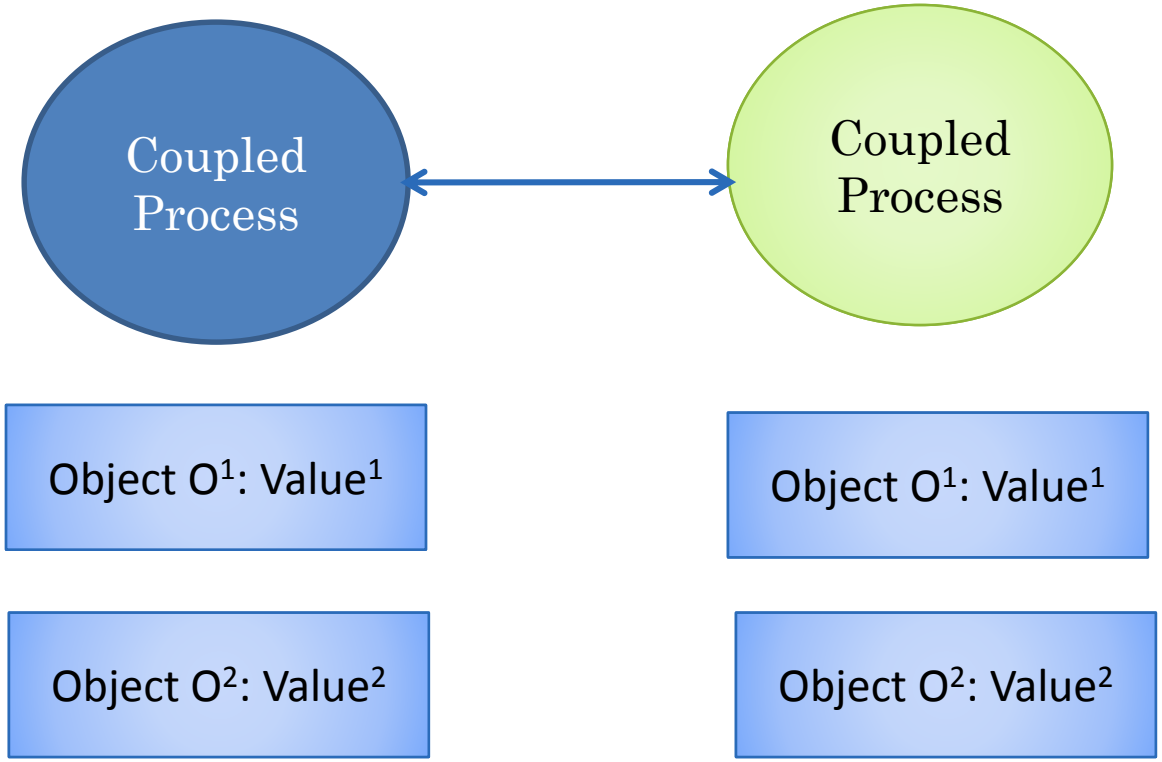
Attack time: At Dawn

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?



Client-Server State Consistency

Question 27 of 27

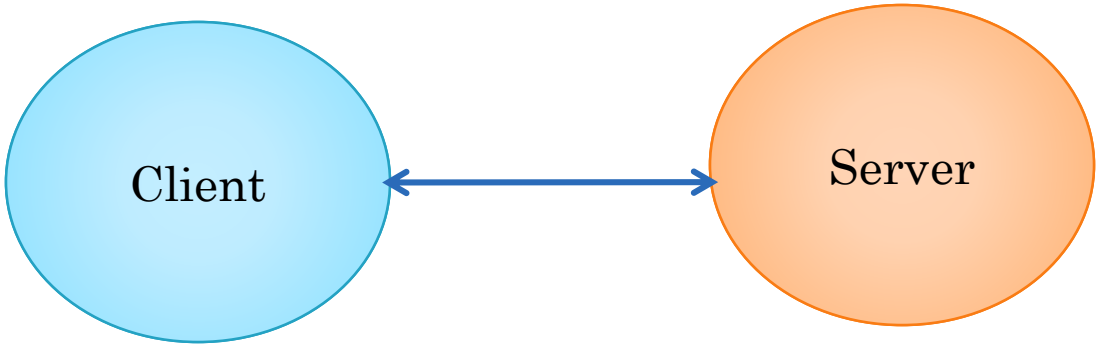
5.0 Points. Point(s) deducted for incorrect answer: 5.0

In GIPC, on completion of a remote function call, the called site sends back to the callings site the return value (or an exception) along with an ID identifying the call.

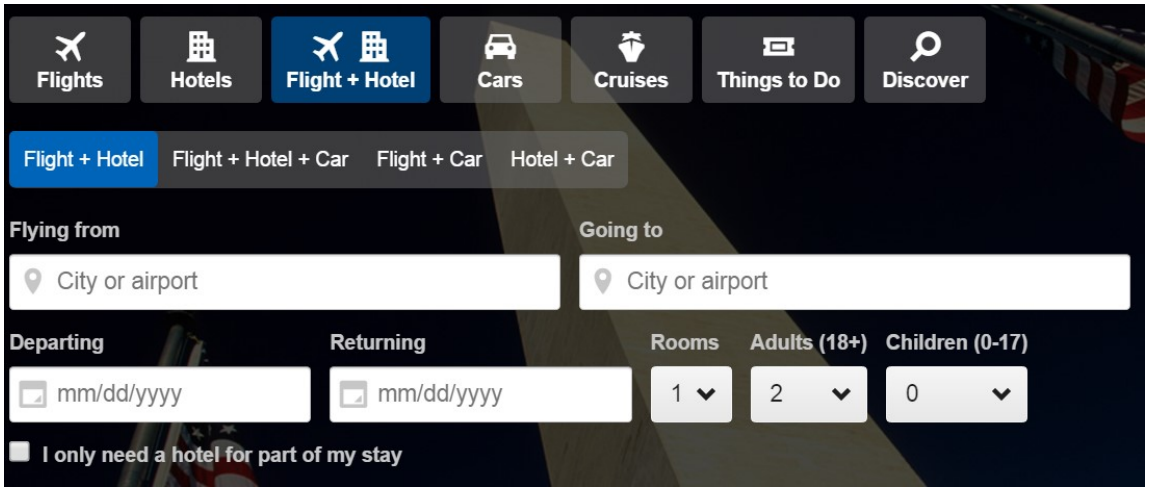
- ○ **True**
- ◉ **False**

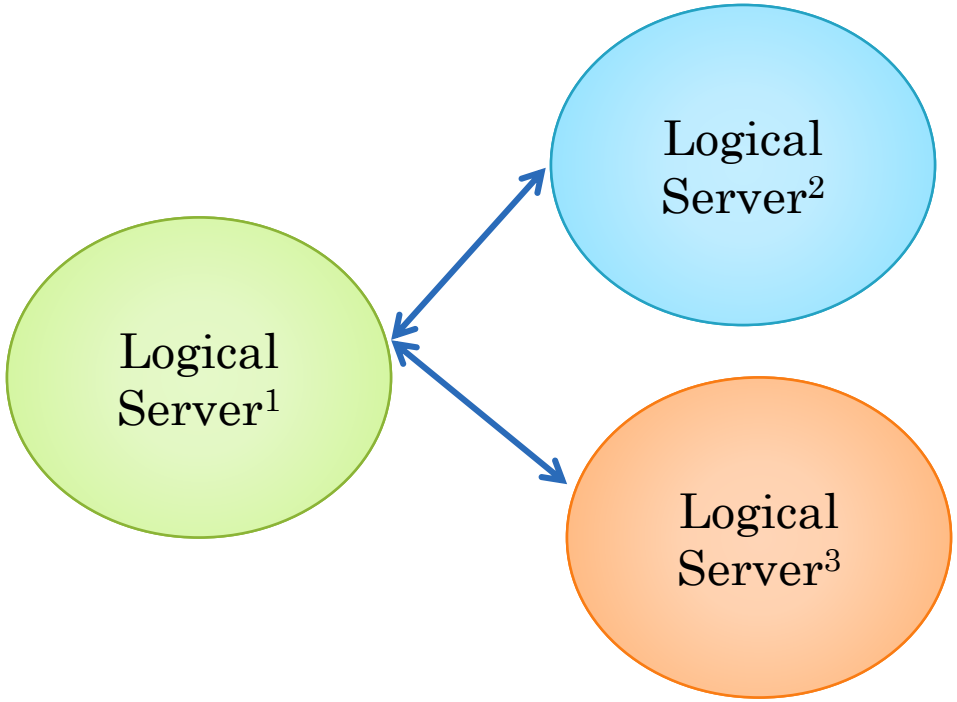
Reset Selection

Previous Next Save Exit

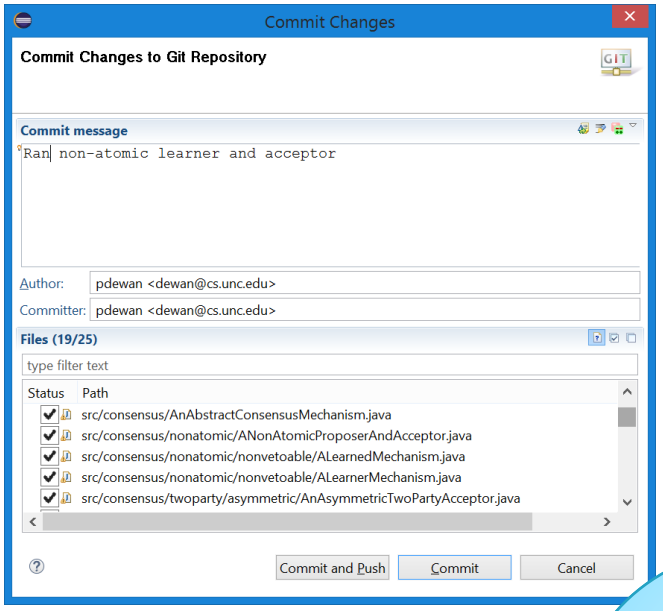


Consistency of Nested Transaction Commits





Physical Replica Diff Consistency



Reliability

Sharing with Performance

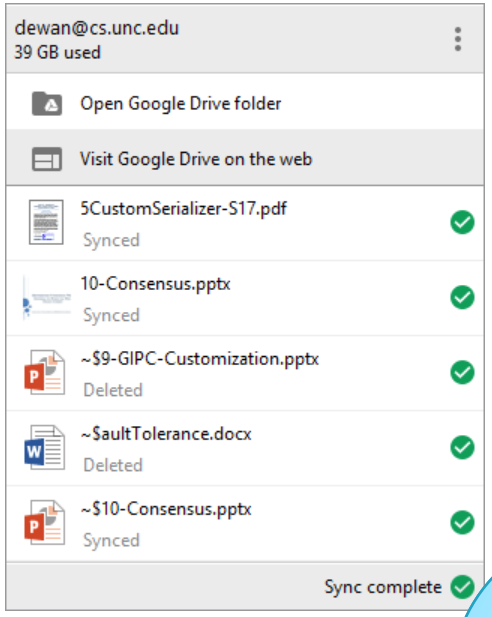
State shared: Diffs

Writeable Client Physical Replica[2]

Writeable Server Physical Replica[1]

Writeable Client Physical Replica[3]

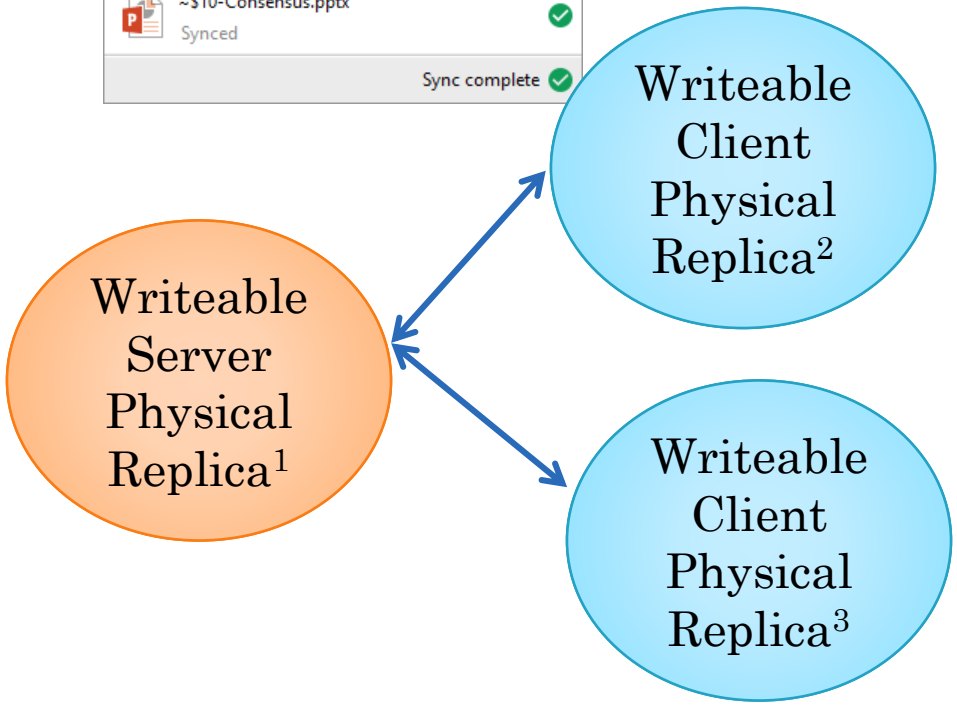
Physical Replica Snapshot Consistency



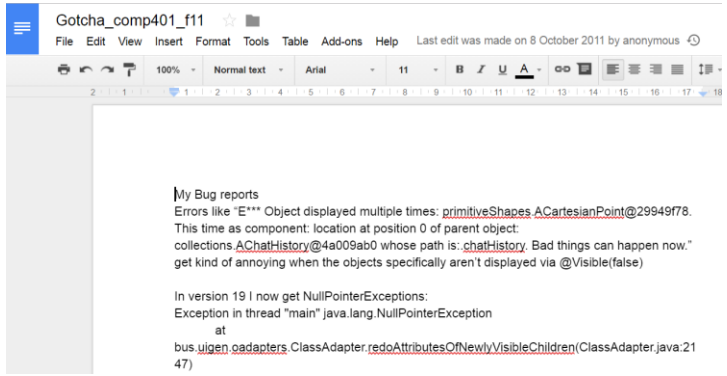
Reliability

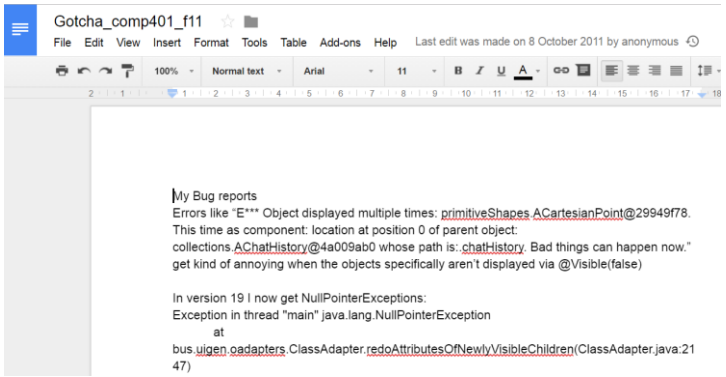
Sharing with Performance

State shared: Snapshot



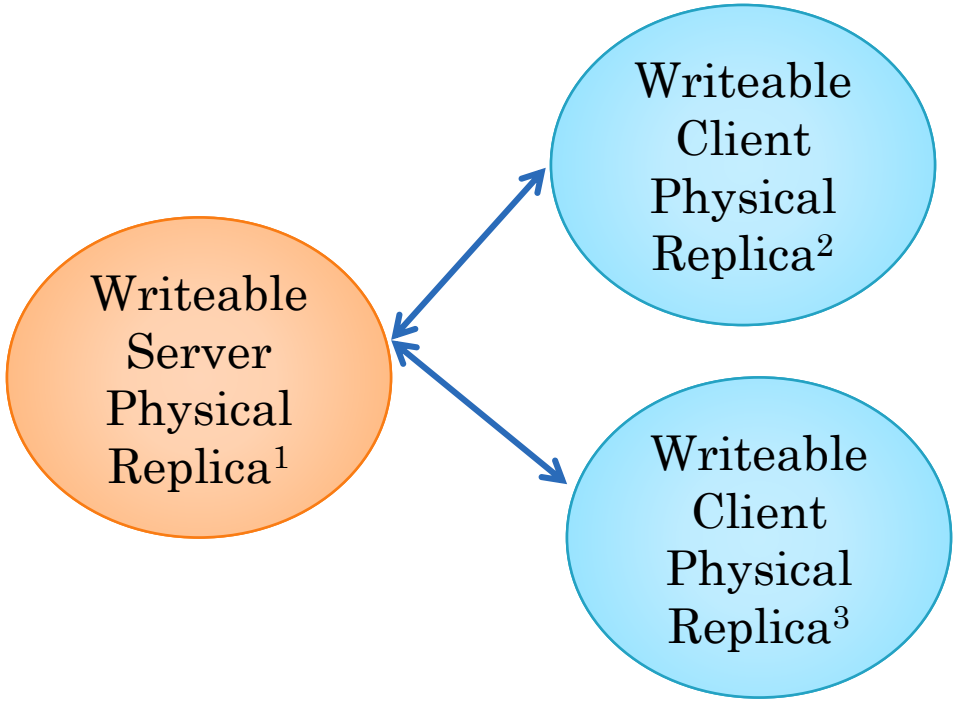
Physical Replica Command Consistency



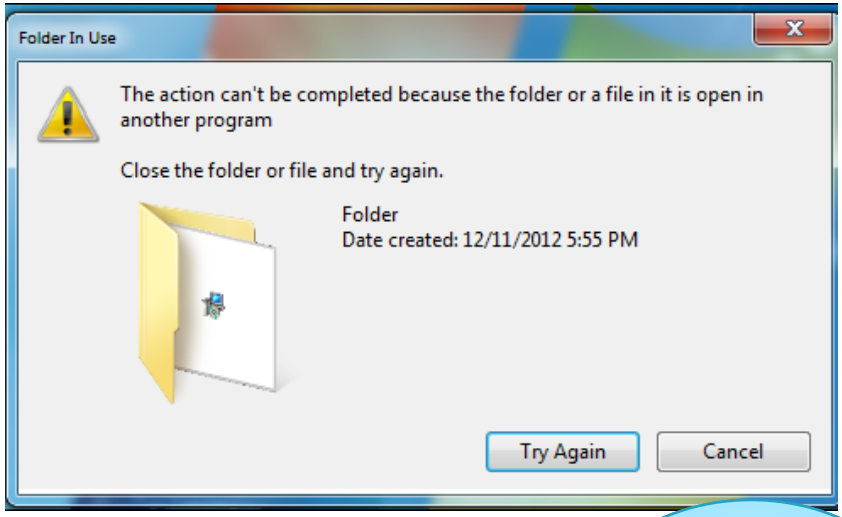


Sharing with Performance

State shared: Next command



Meta (Consistency-State) Consistency



Sharing with Performance

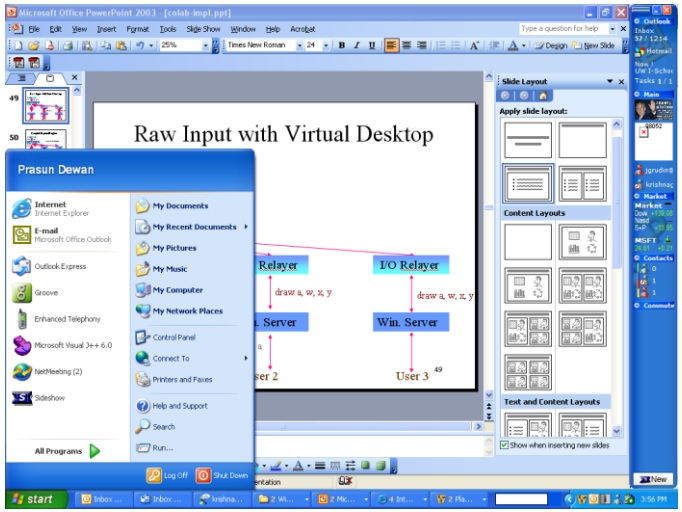
State shared: Locks, Consistency, Communication Protocols

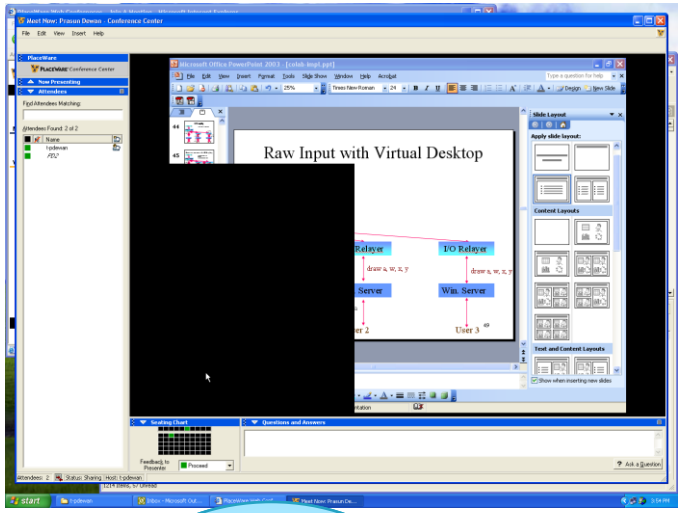
Writeable Server Physical Replica[1]

Writeable Client Physical Replica[2]

Writeable Client Physical Replica[3]

Lock Consistency





Sharing with Performance

State shared: Locks

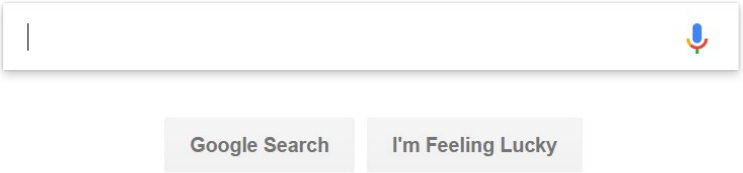
Writeable Server Physical Replica[1]

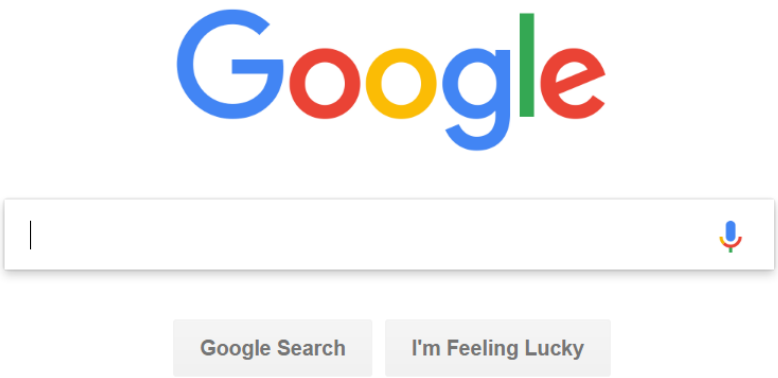
Writeable Client Physical Replica[2]

Writeable Client Physical Replica[3]

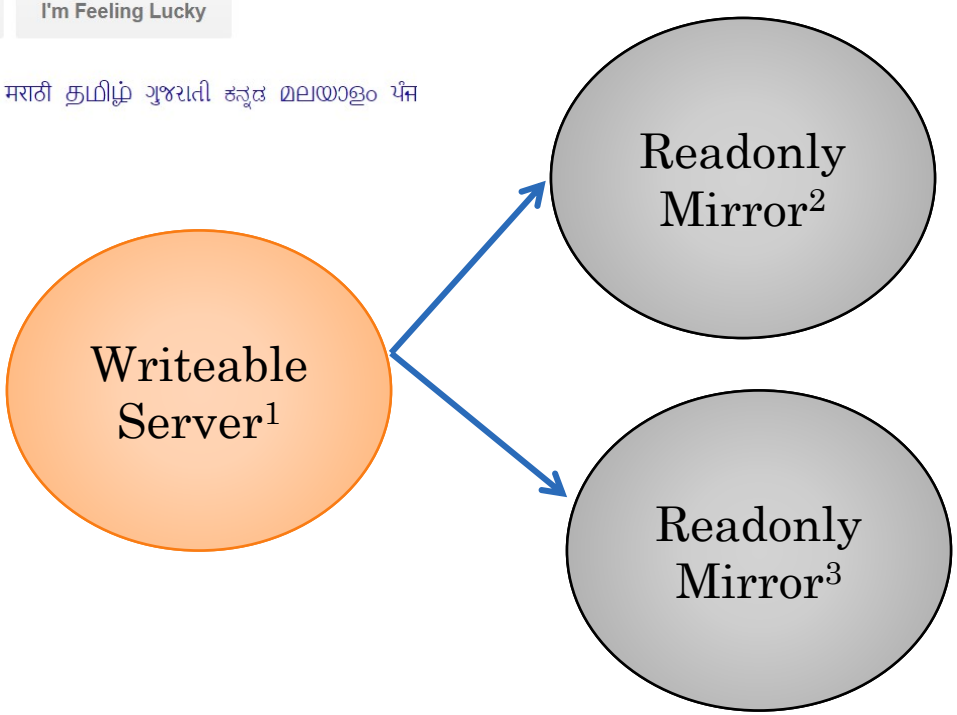
One directional arrows?

Physical Replica: Mirroring

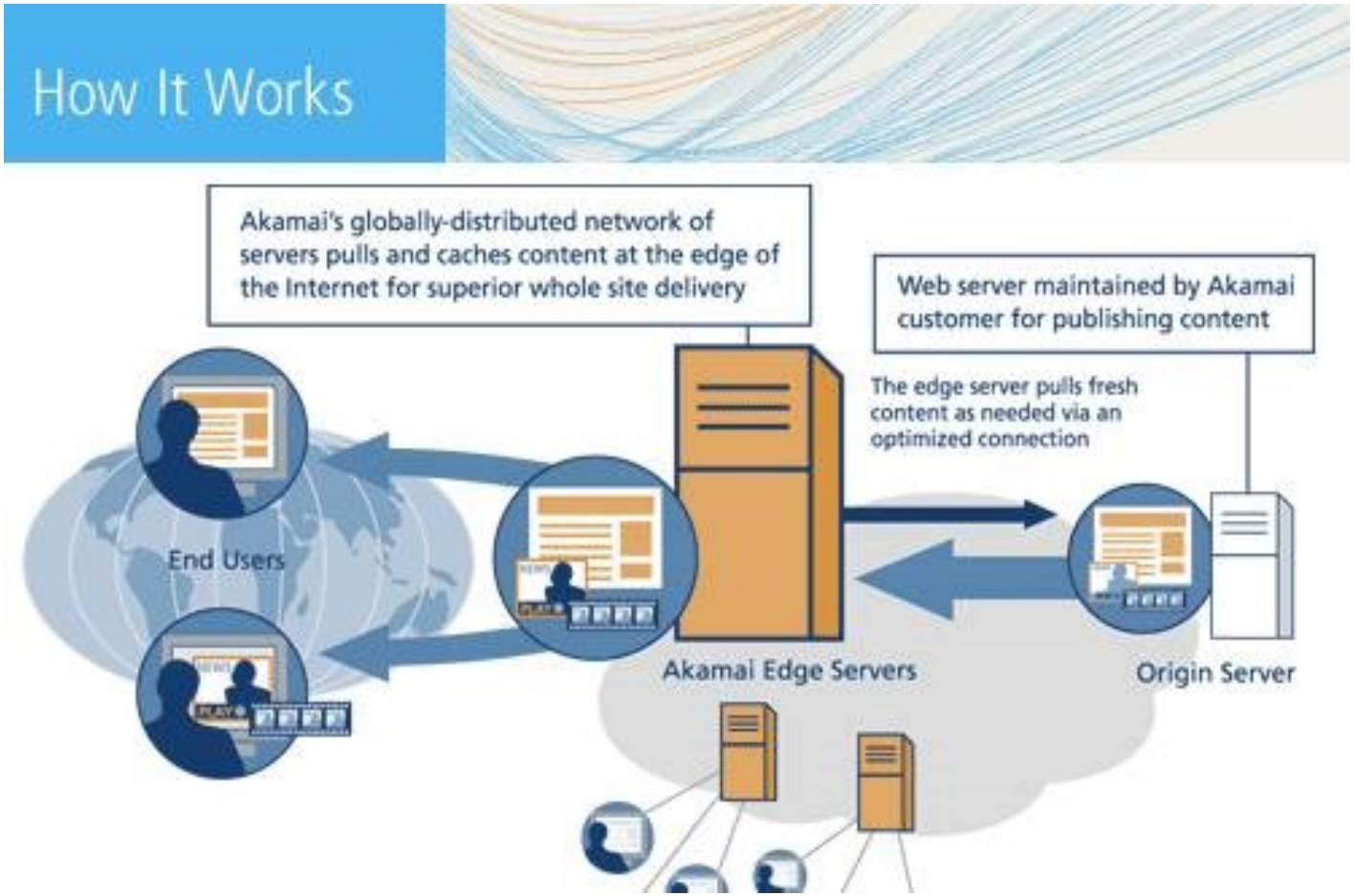




Sharing with Performance



Akamai Mirroring

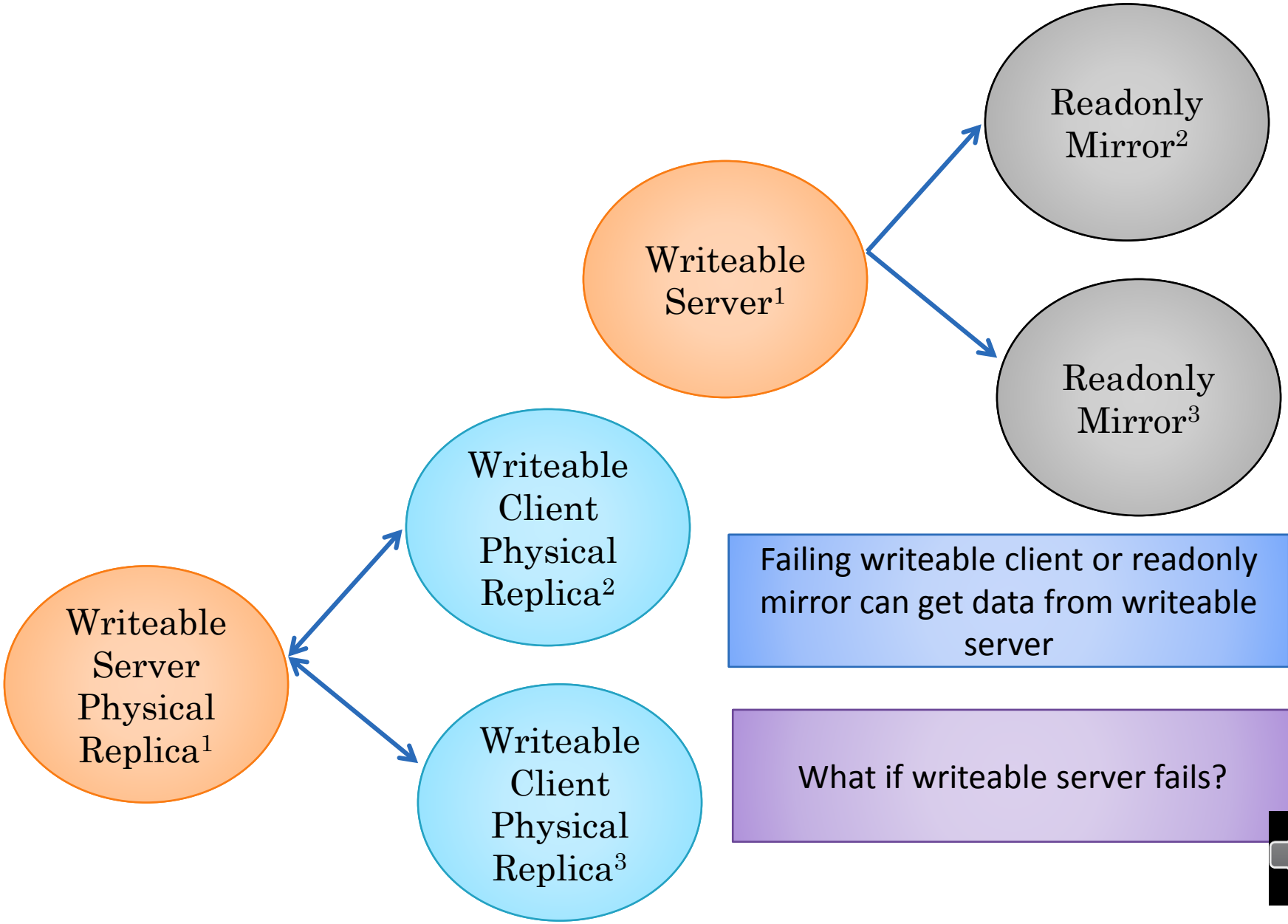


From google images search (Akamai Architecture)

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

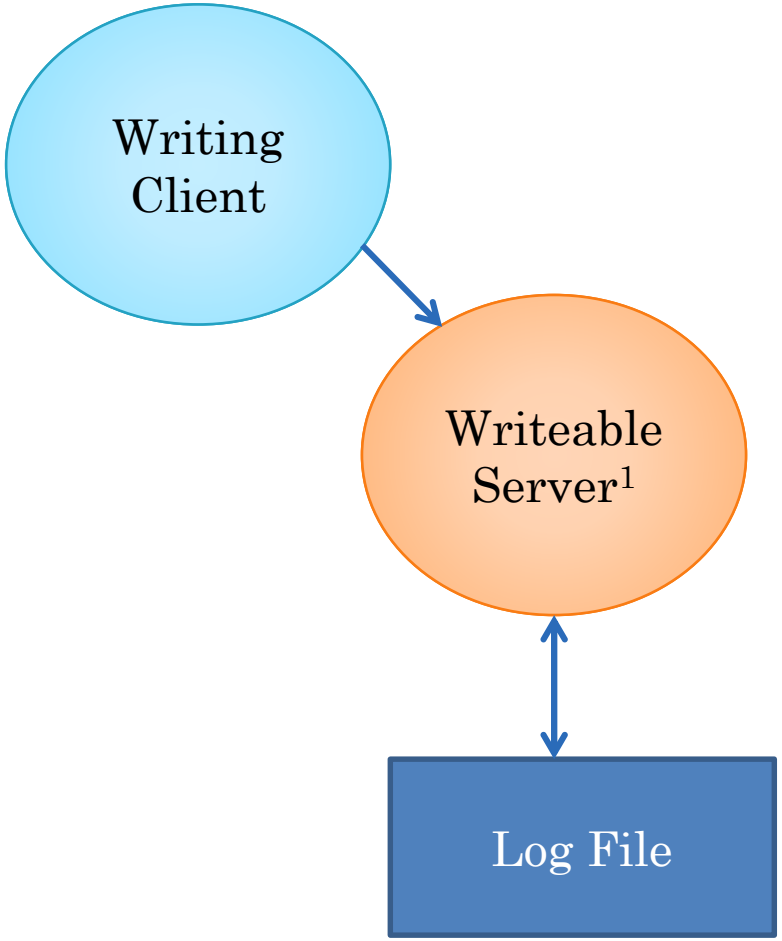
Fault Tolerance



Failing writeable client or readonly mirror can get data from writeable server

What if writeable server fails?

Log File



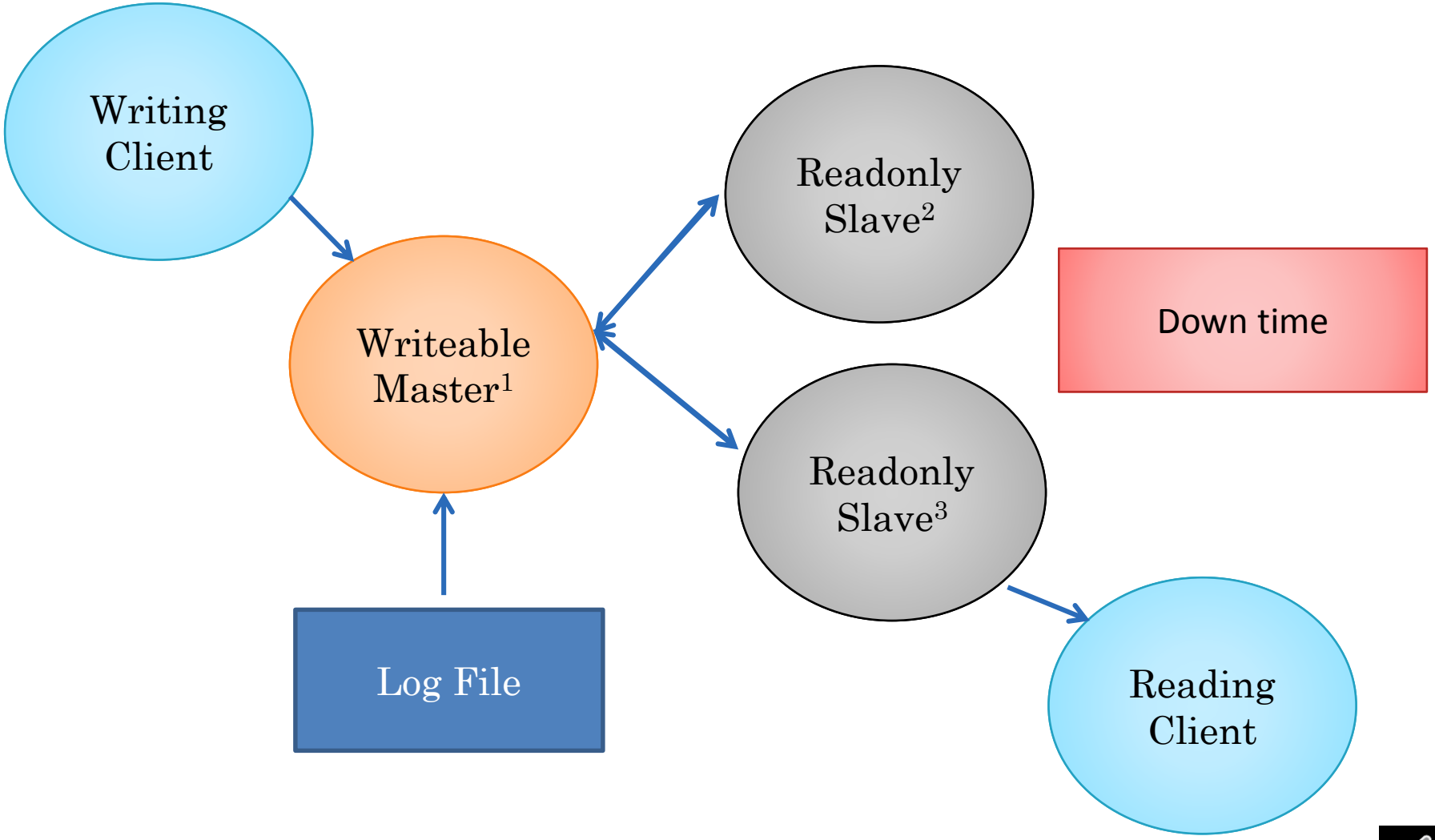
Log file error?

Multiple disks (RAID) to create "stable" storage

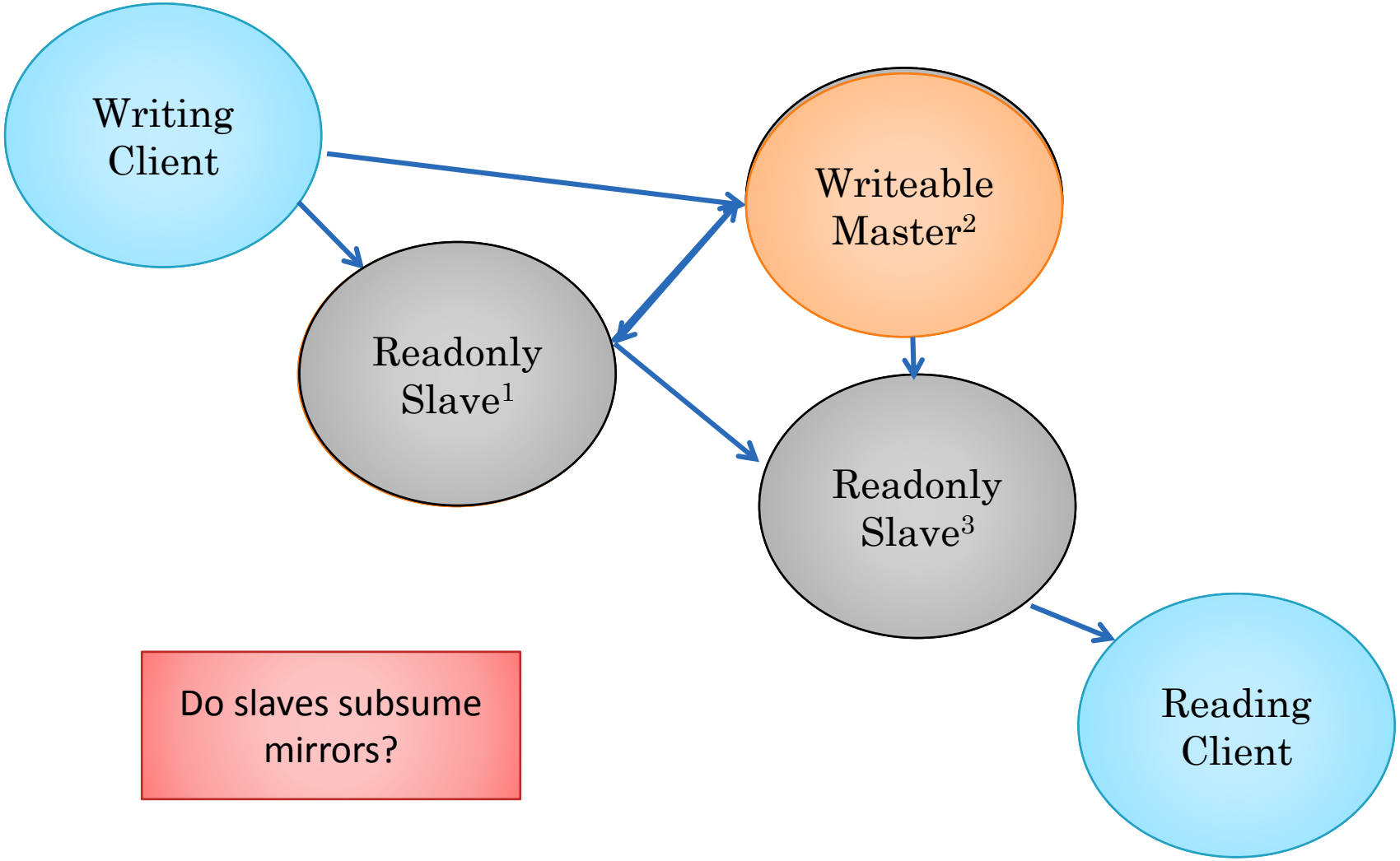
Master corruption or stable storage error

Down time

Static Master-Slave Replication

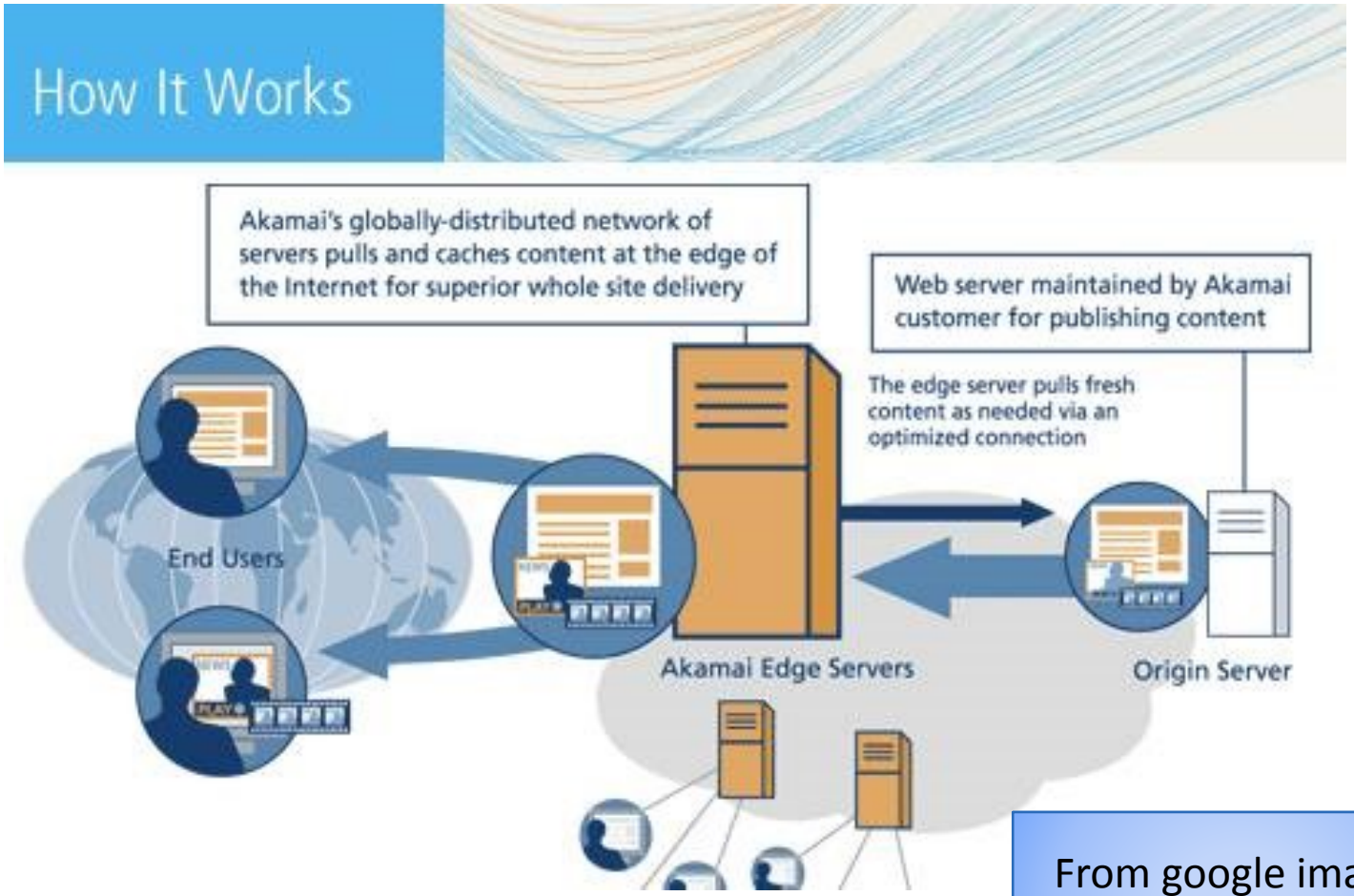


Dynamic Master-Slave Replication



Do slaves subsume mirrors?

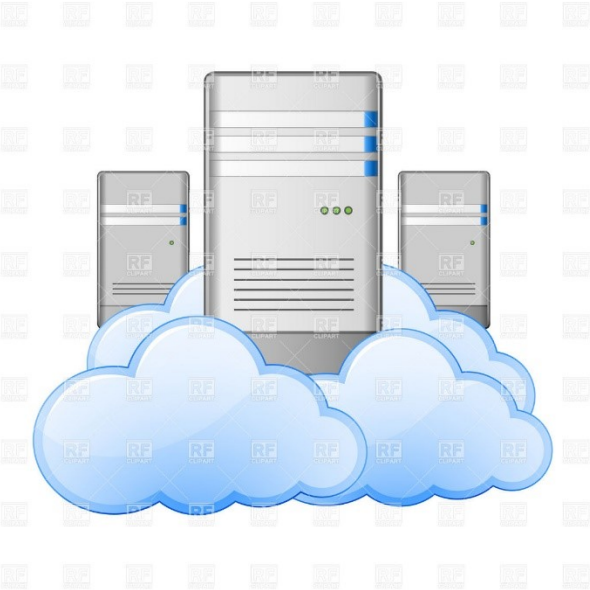
Location of Slaves?

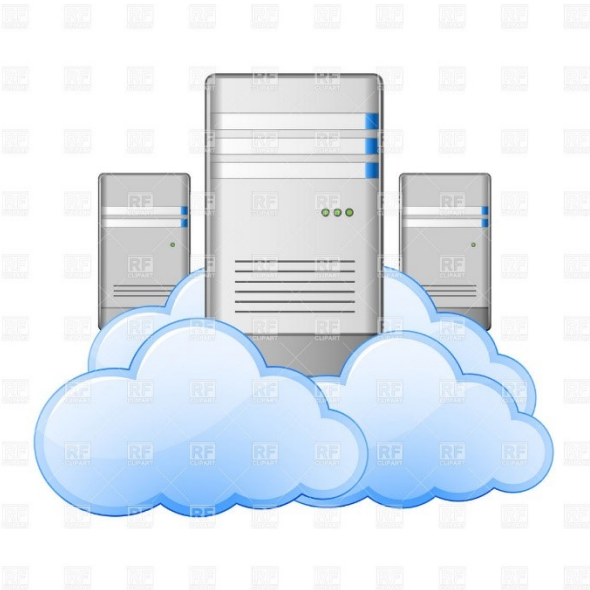


Synchronization time can be high if slaves are on the edge

From google images search (Akamai Architecture)

Data Centers





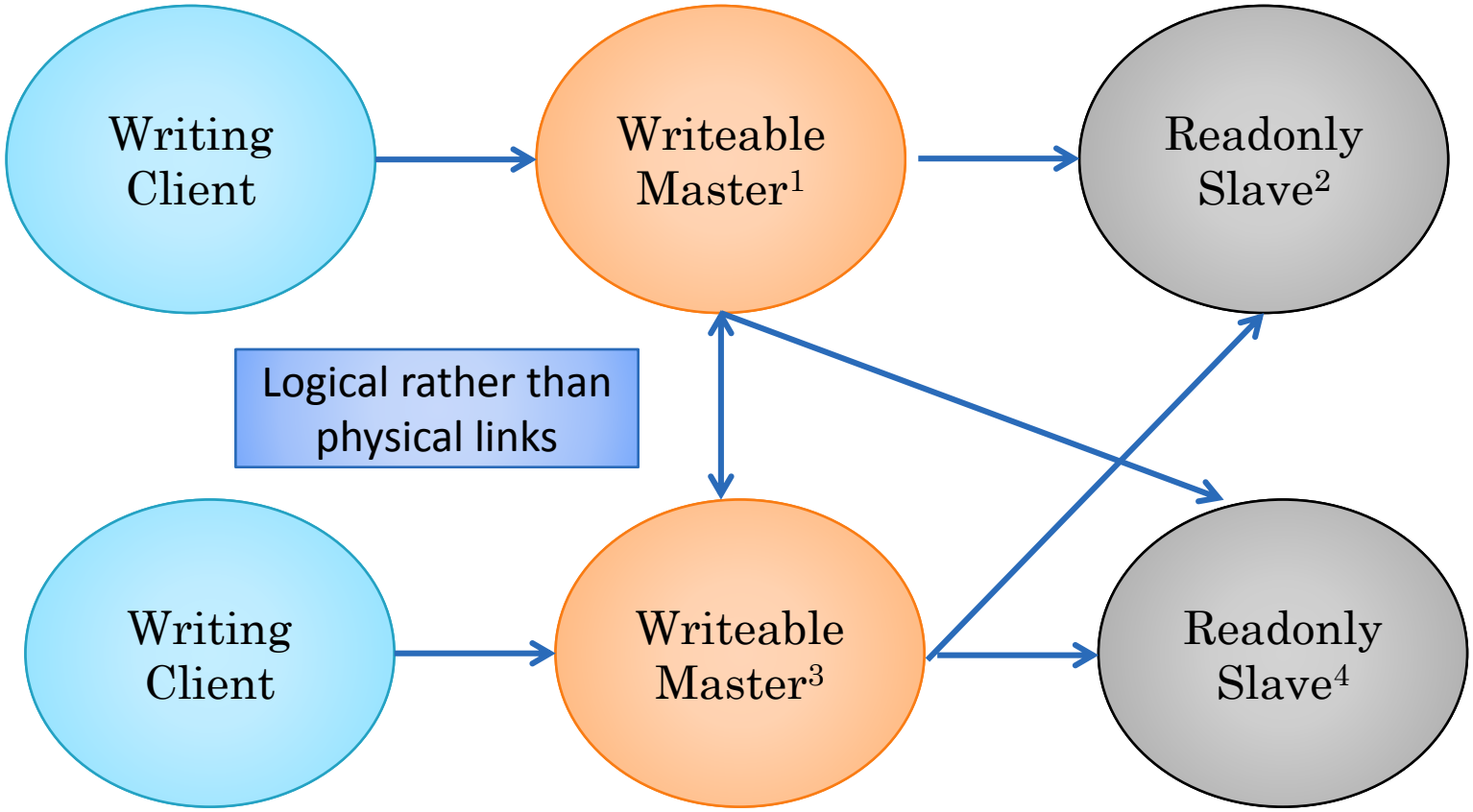
Synchronization time low if slaves are in master's data center

Multiple data centers for some service?

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

Geographically dispersed writes (and reads) and power-failure toleance

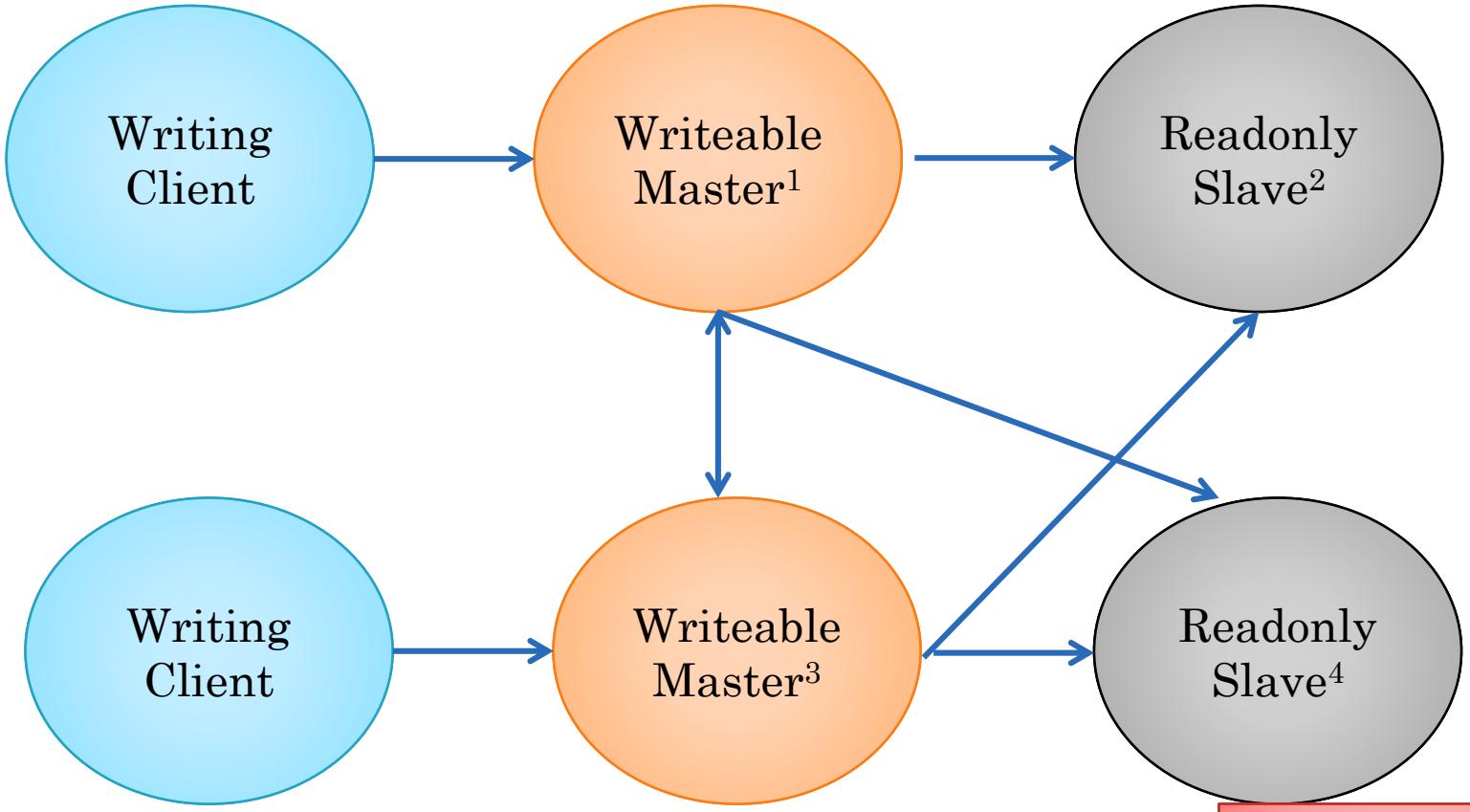
Master-Master Replication



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



Unsequenced (by a single master) write conflicts possible!

Solutionn: 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