# Consensus and Consistency: Why Should I Care?

Neha Narula

@neha

Berlin Buzzwords 2014

@neha

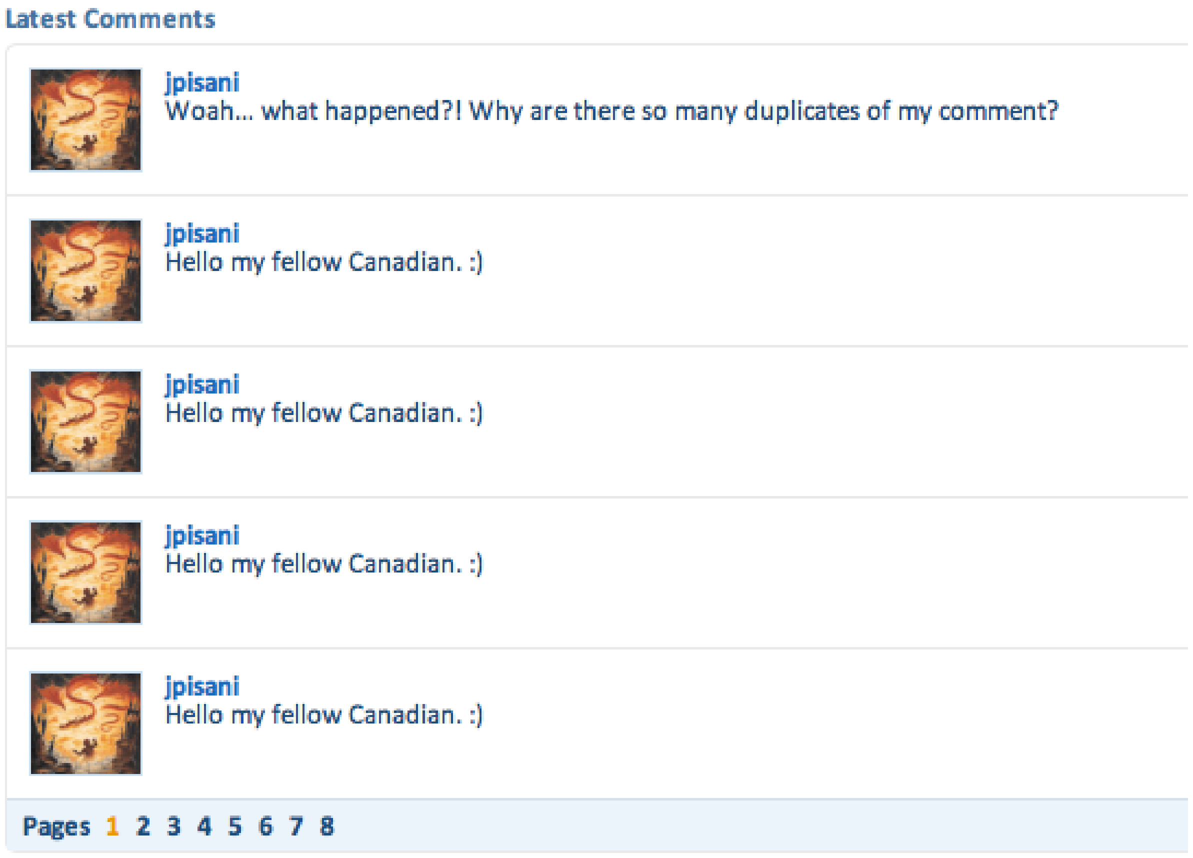
2003-2010 Google™

2008-2014 CSAIL

## This Talk

- Why do we need it?
- Types of consistency
- Consensus
- CAP theorem
- What to do with this?

# How Messed Up Can Things Get?


Latest Comments

jpisani
Woah... what happened?! Why are there so many duplicates of my comment?

jpisani
Hello my fellow Canadian. :)

jpisani
Hello my fellow Canadian. :)

jpisani
Hello my fellow Canadian. :)

jpisani
Hello my fellow Canadian. :)

Pages 1 2 3 4 5 6 7 8

*“The hacker discovered that multiple simultaneous withdrawals are processed essentially at the same time and that the system's software doesn't check quickly enough for a negative balance”*

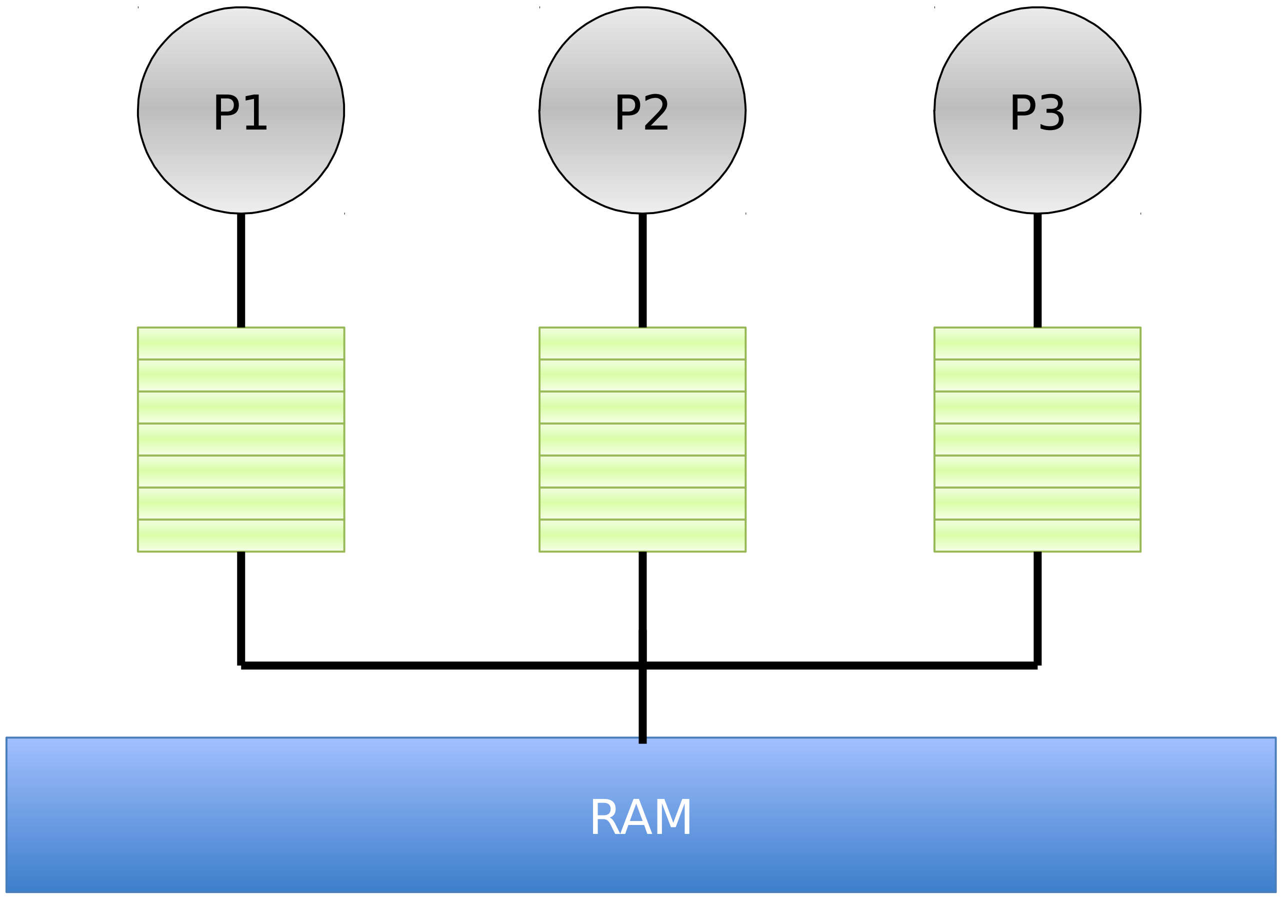
http://arstechnica.com/security/2014/03/yet-another-exchange-hacked-poloniex-loses-around-50000-in-bitcoin/

Consistency guarantees help us reason about our code and avoid subtle bugs

# Consistency

- Most misused word in distributed systems
- C as in ACID
- C as in CAP
- C as in sequential, causal, eventual, strict consistency

# Cache Coherence

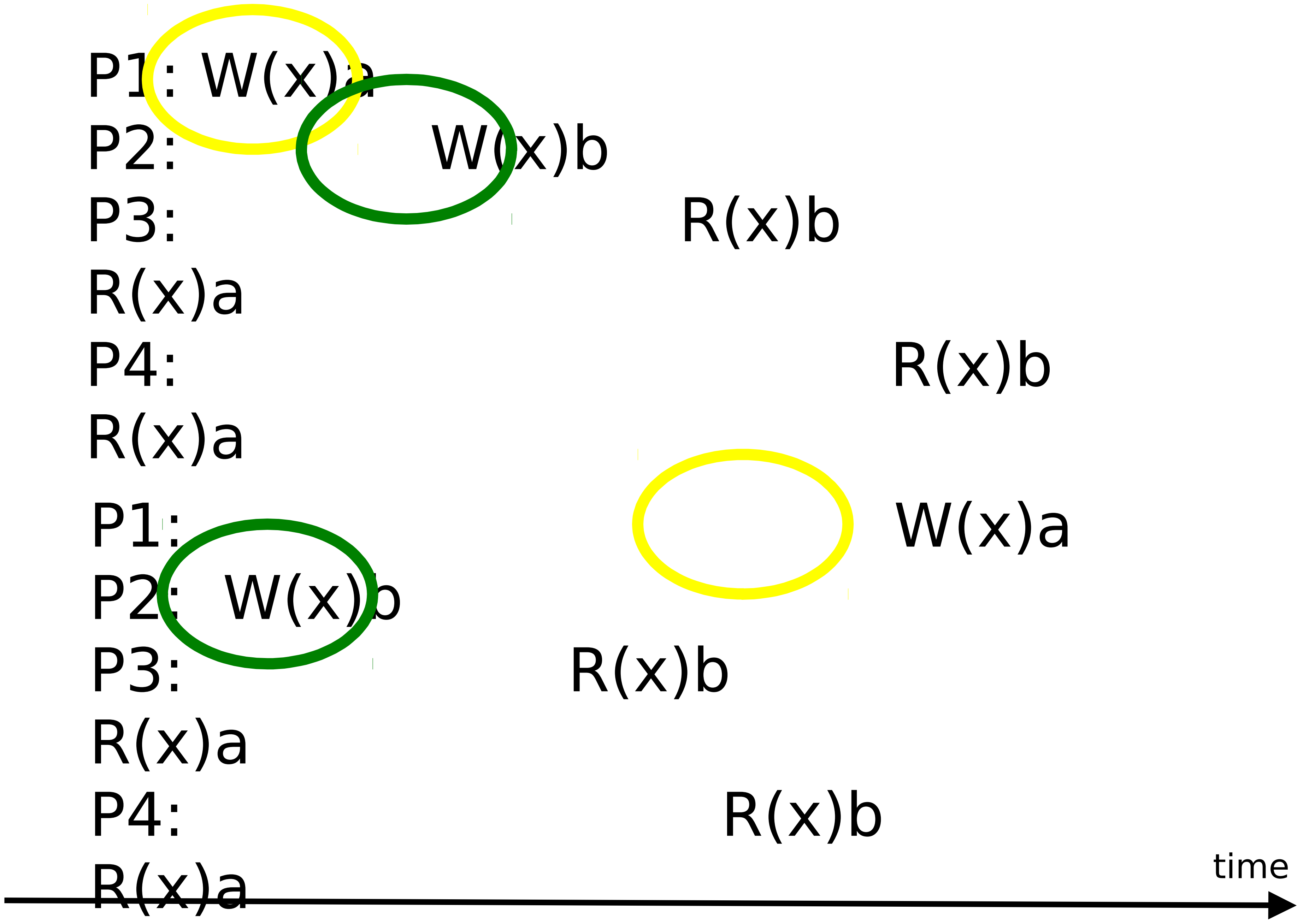


## Sequential Consistency

The result of any execution is as if the reads and writes were executed in *some* order

Order doesn’t have to match time!

Order does have to match what each process sees


P1: W(x)a
P2: W(x)b
P3: R(x)b R(x)a
P4: R(x)b R(x)a
P1: W(x)a
P2: W(x)b
P3: R(x)b R(x)a
P4: R(x)b R(x)a
time

# External Consistency

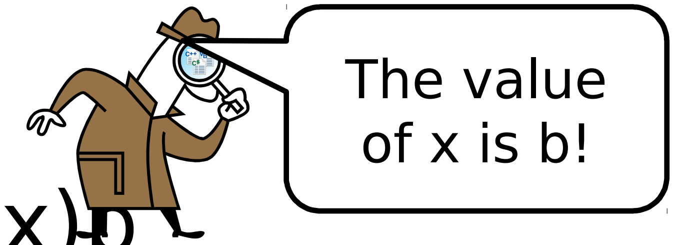
Everything that sequential consistency has

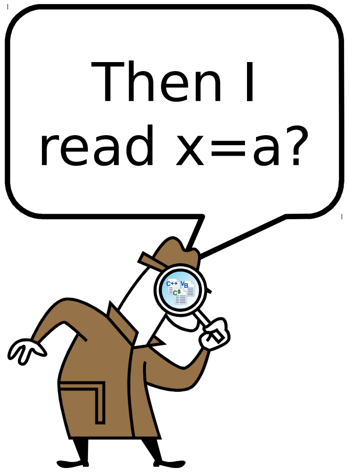
Except results actually match *time.*

An external observer

# Not Externally Consistent

P1: W(x)a
P2: W(x)b
P3: R(x)b
R(x)a
P4: R(x)b
R(x)a





time

## Distributed System

- Communication, not shared memory
- Replication without cache coherence
- *Time* becomes a fuzzy concept

## Eventual Consistency

If no new updates are made to the object, eventually all accesses will return the last updated value.

# Eventual Consistency

If no new updates are made to the object, eventually all accesses will return ~~the last updated value~~ the same value.

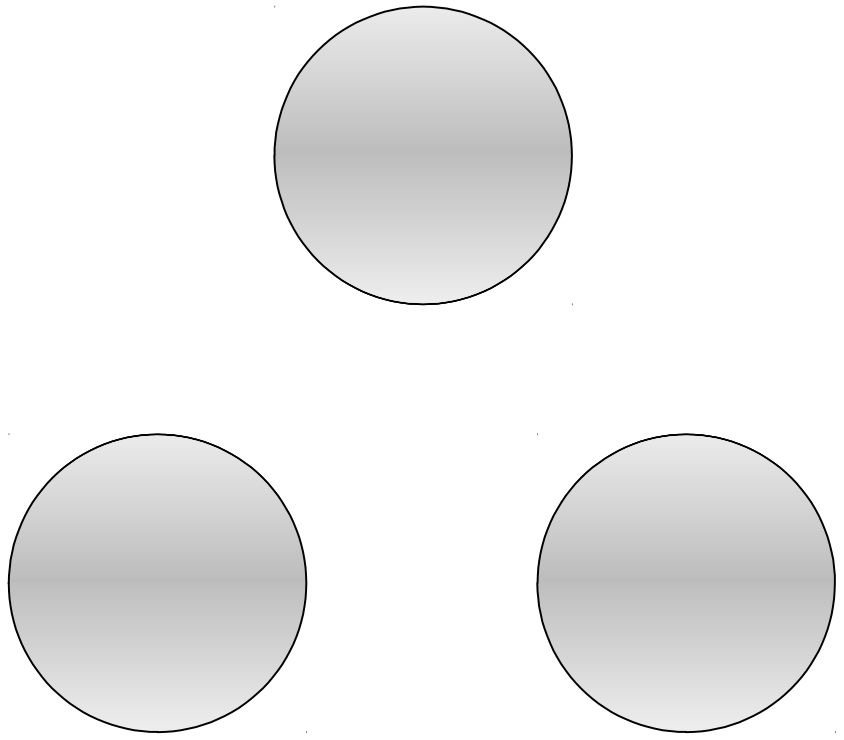
(What is last, really?)

(And when do we stop writing?)

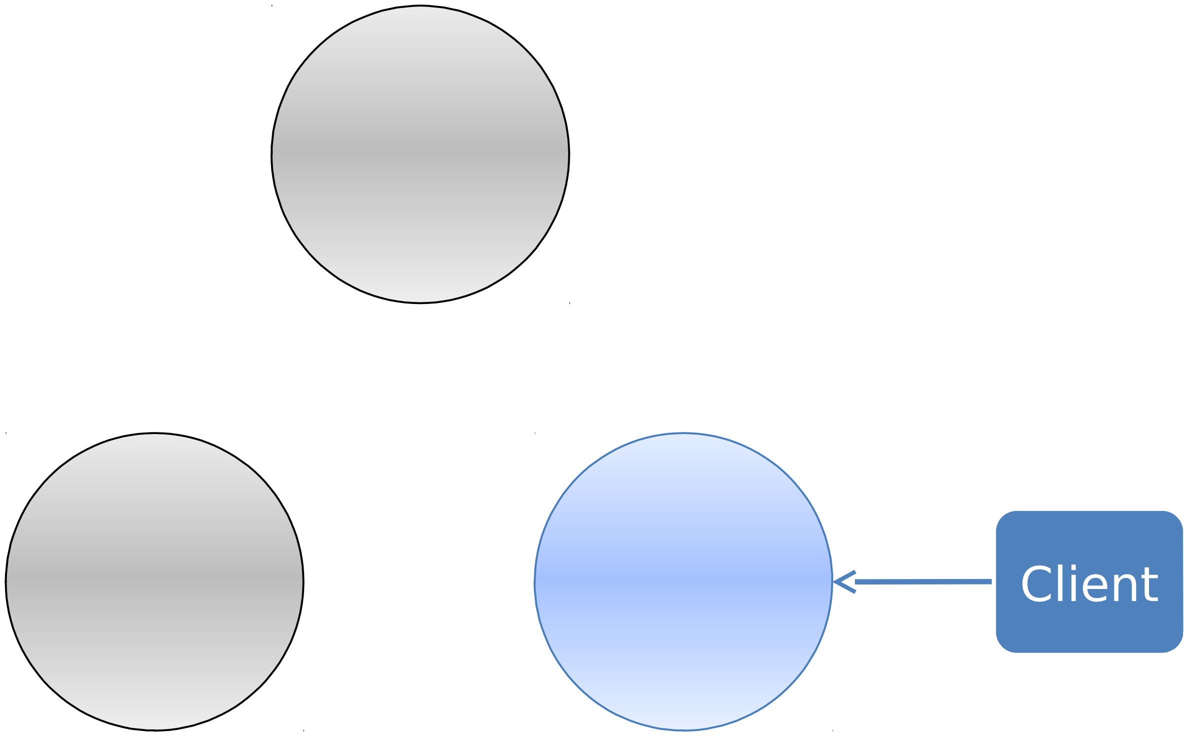
# CONSENSUS

THIS WOULD WORK A LOT BETTER IF YOU'D JUST AGREE WITH ME.

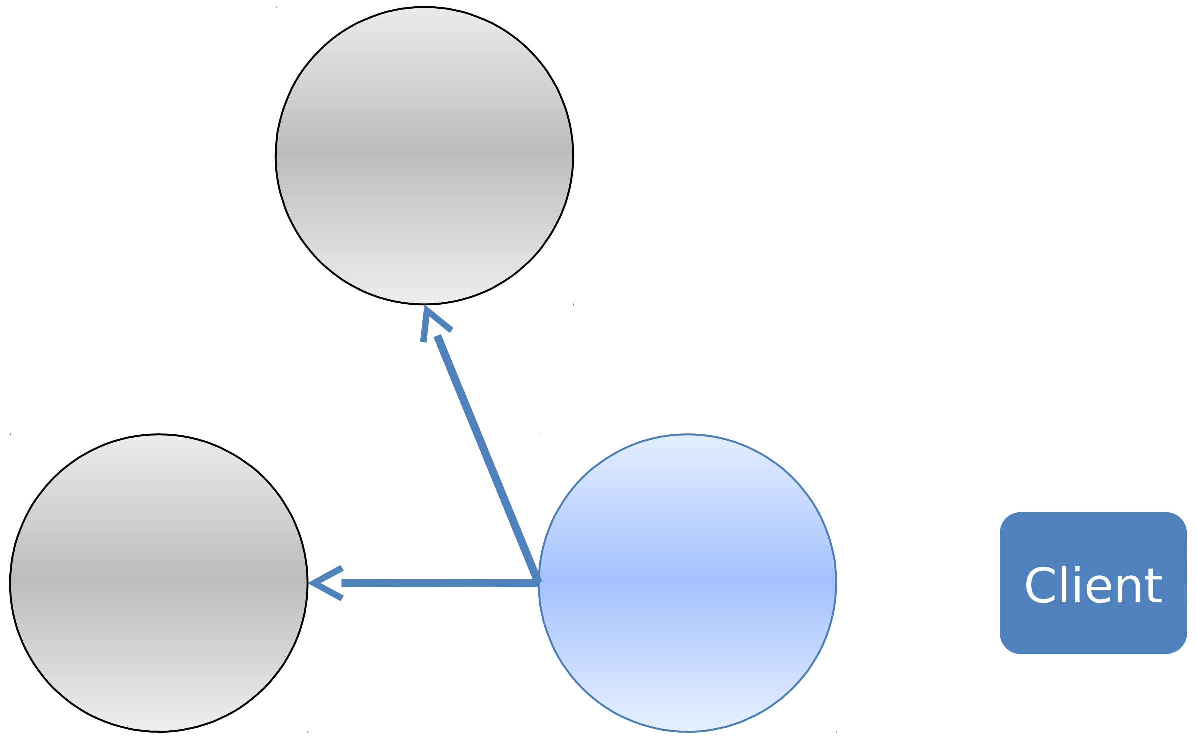
## Paxos



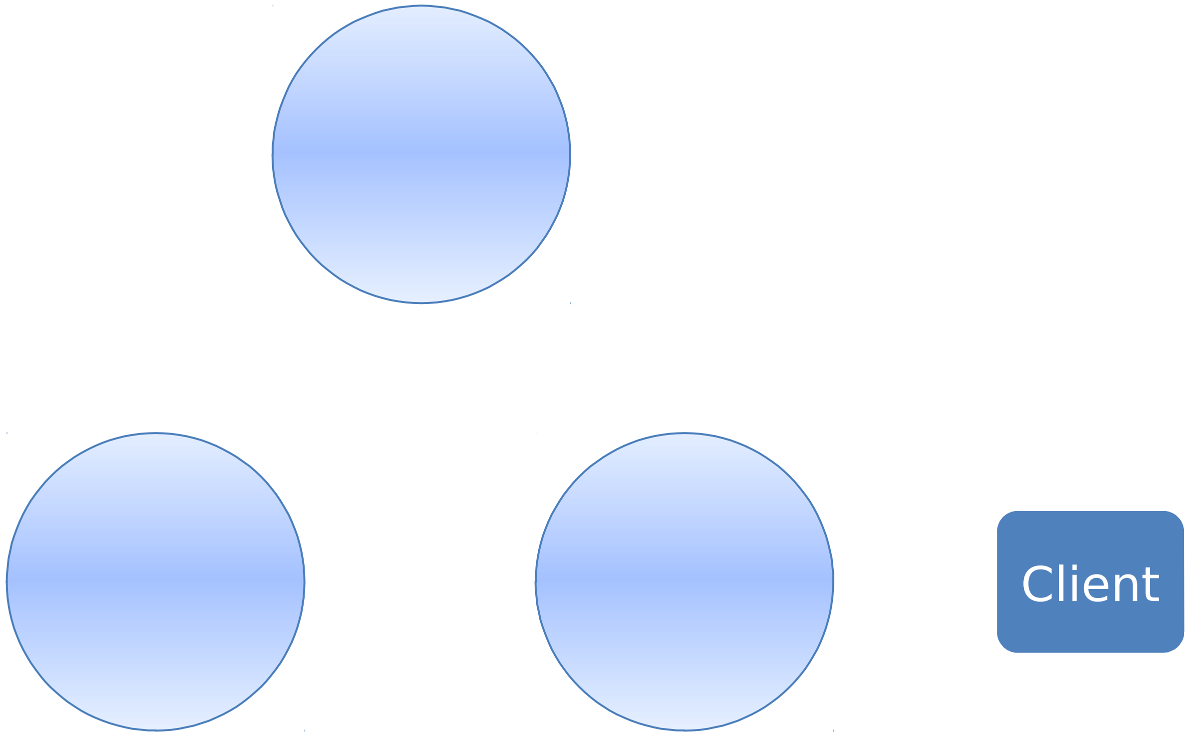
# Propose blue



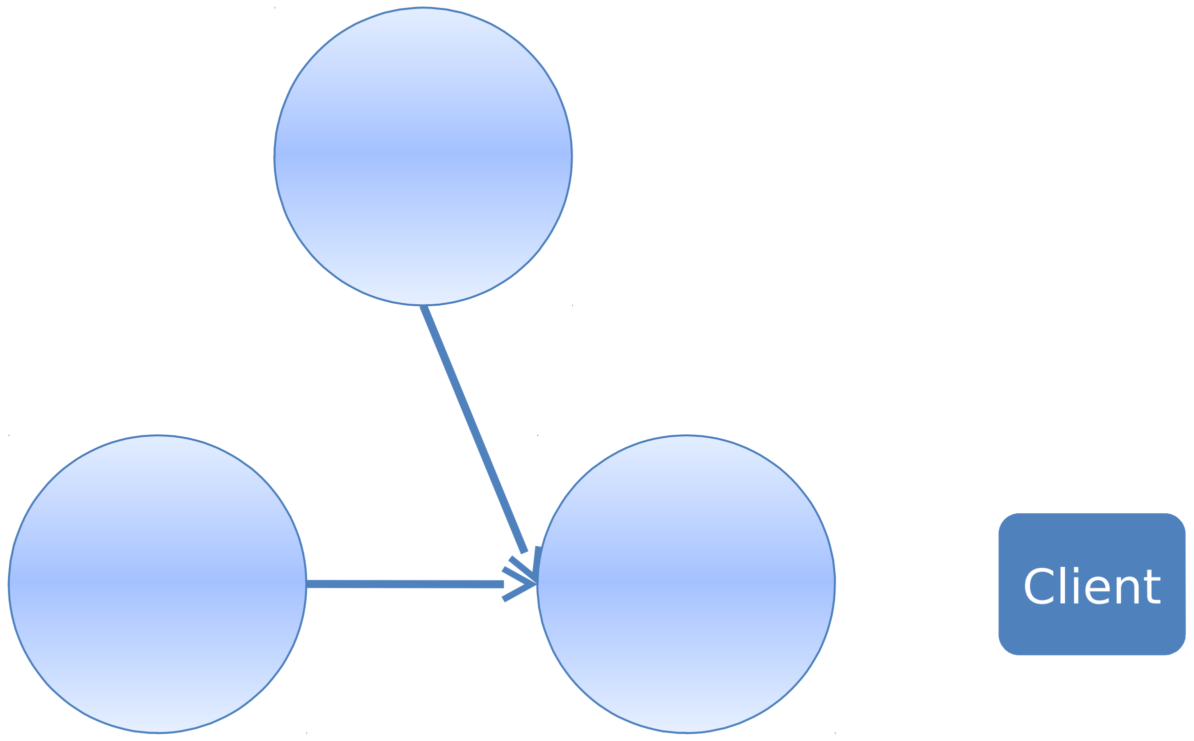
# Prepare (n, blue)



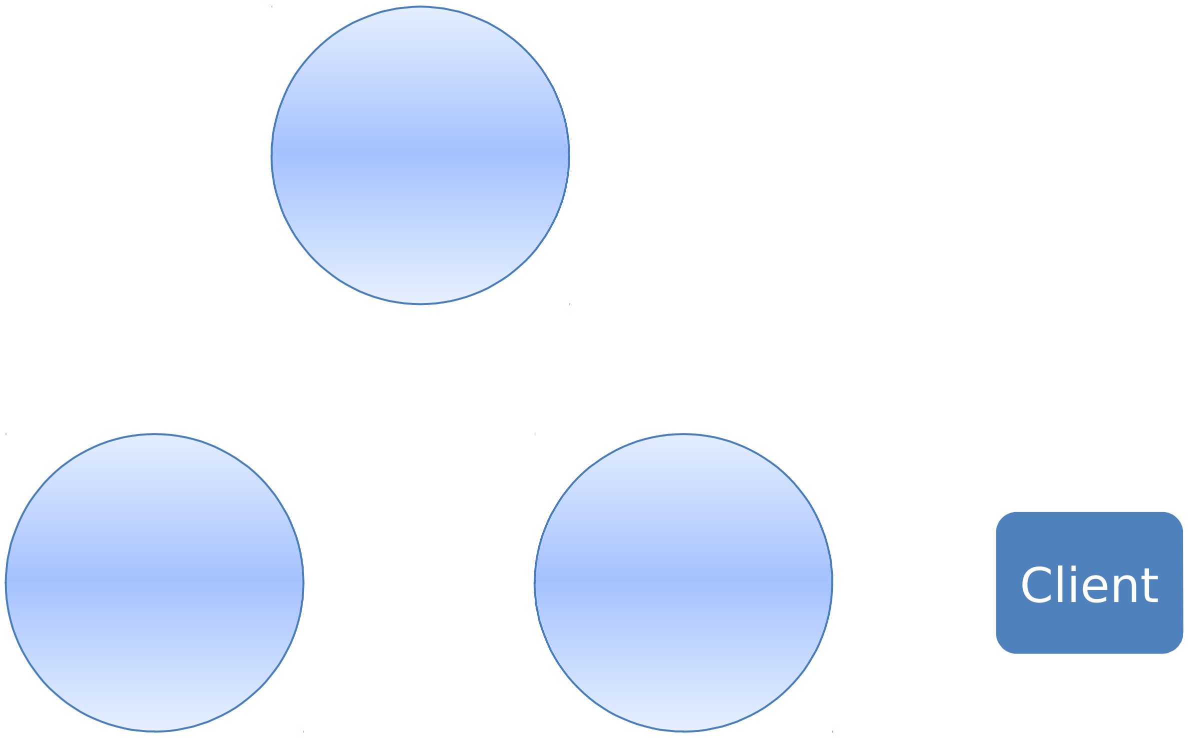
## Prepare (n, blue)



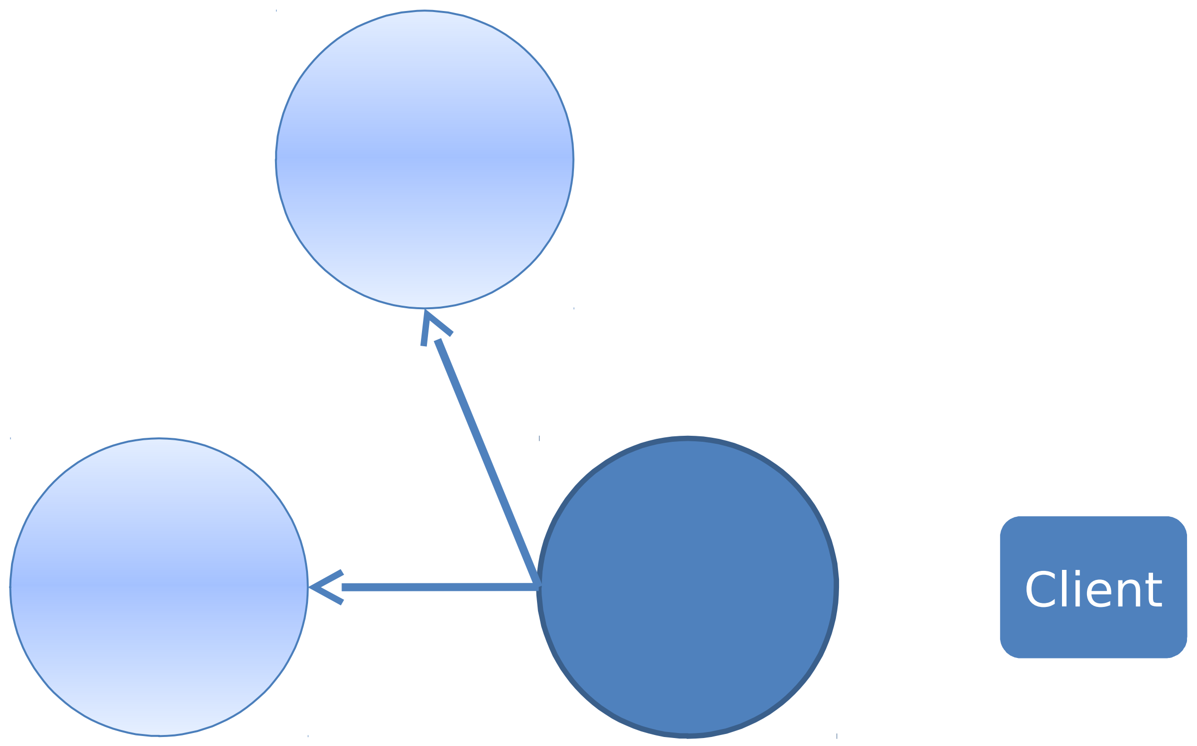
# Prepare OK



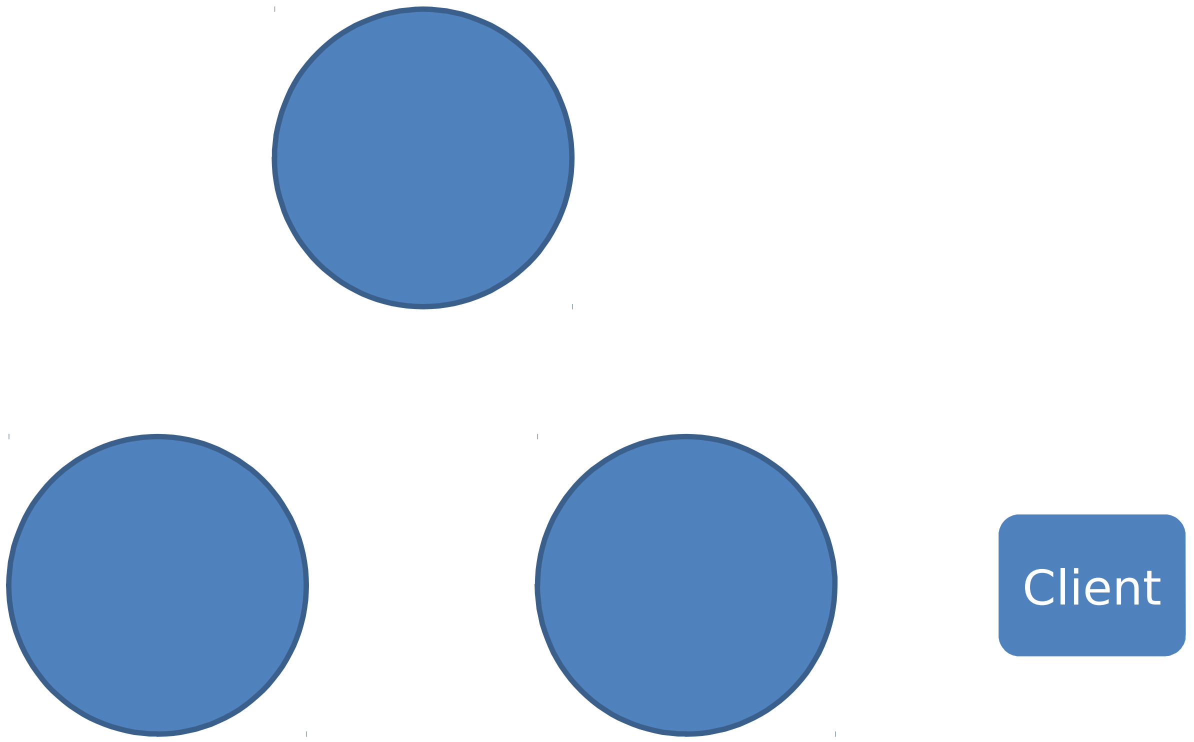
## Prepare OK



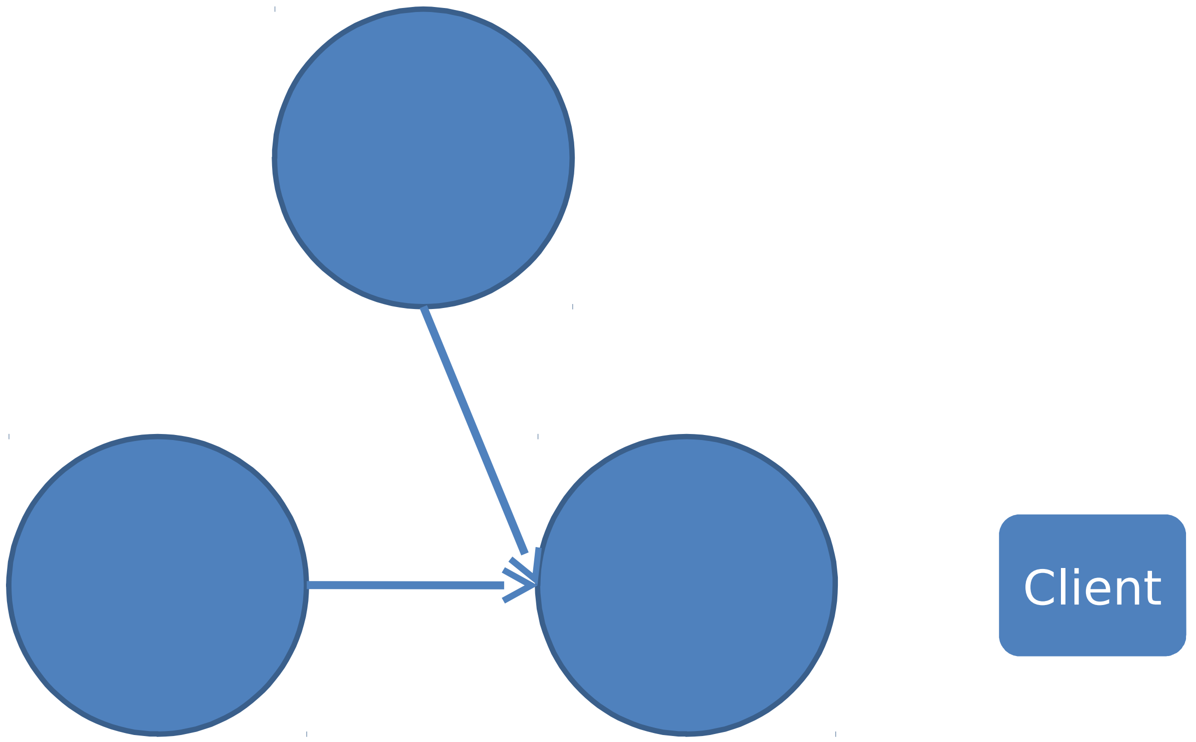
# Accept (n, blue)



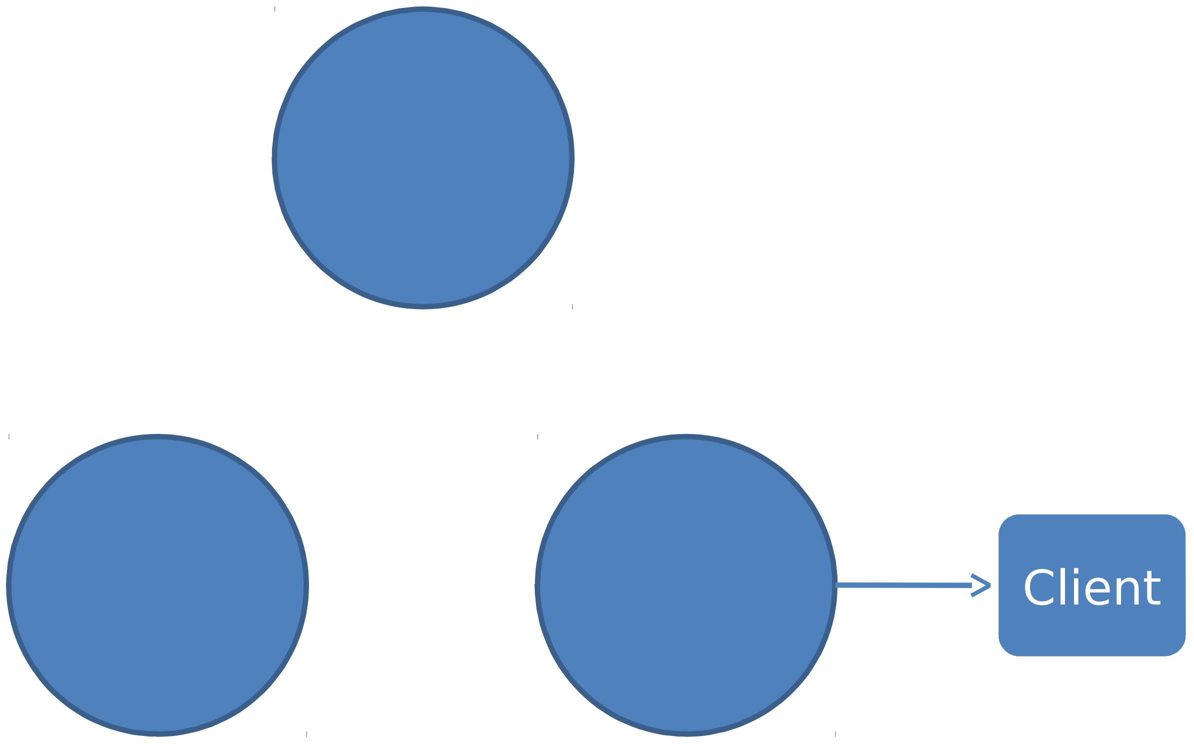
## Accept (n, blue)



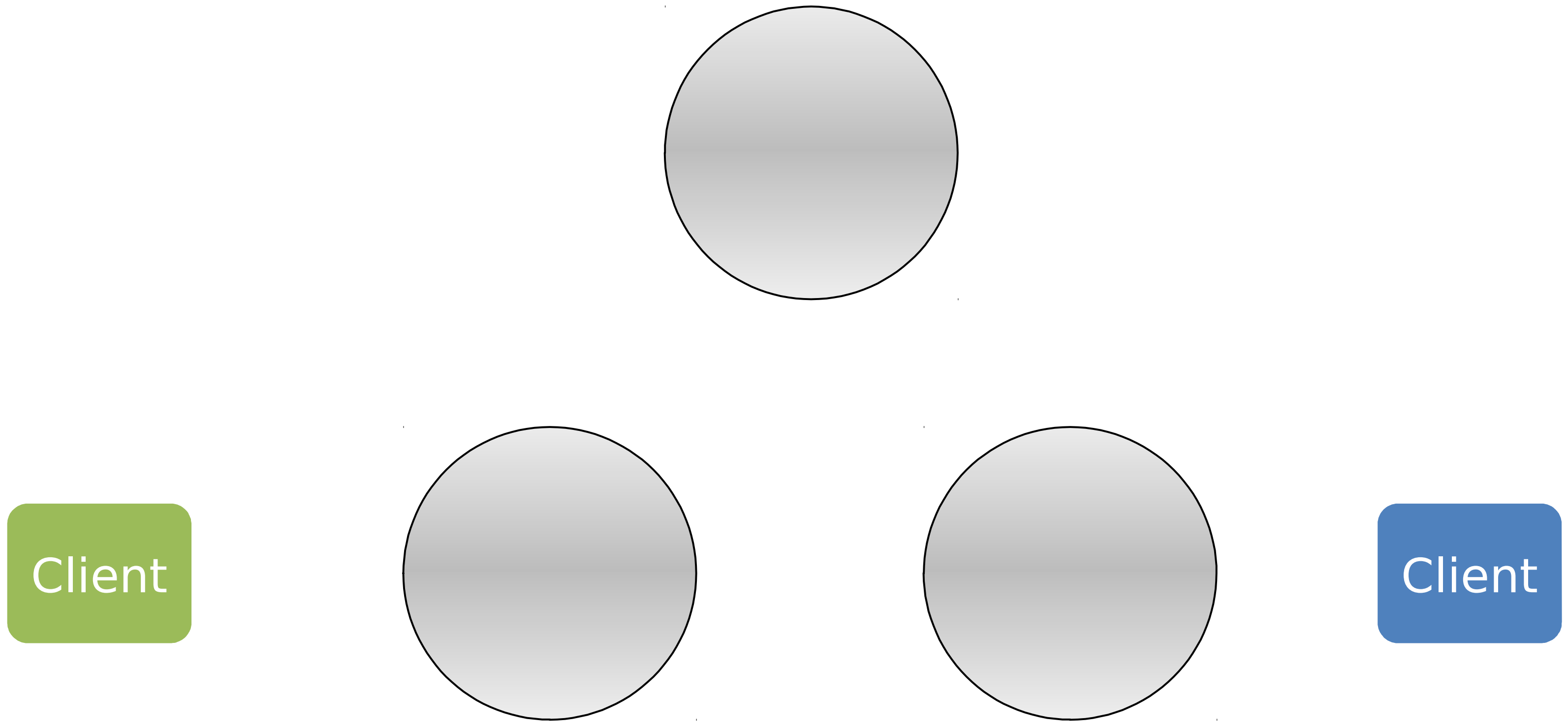
# Accept OK



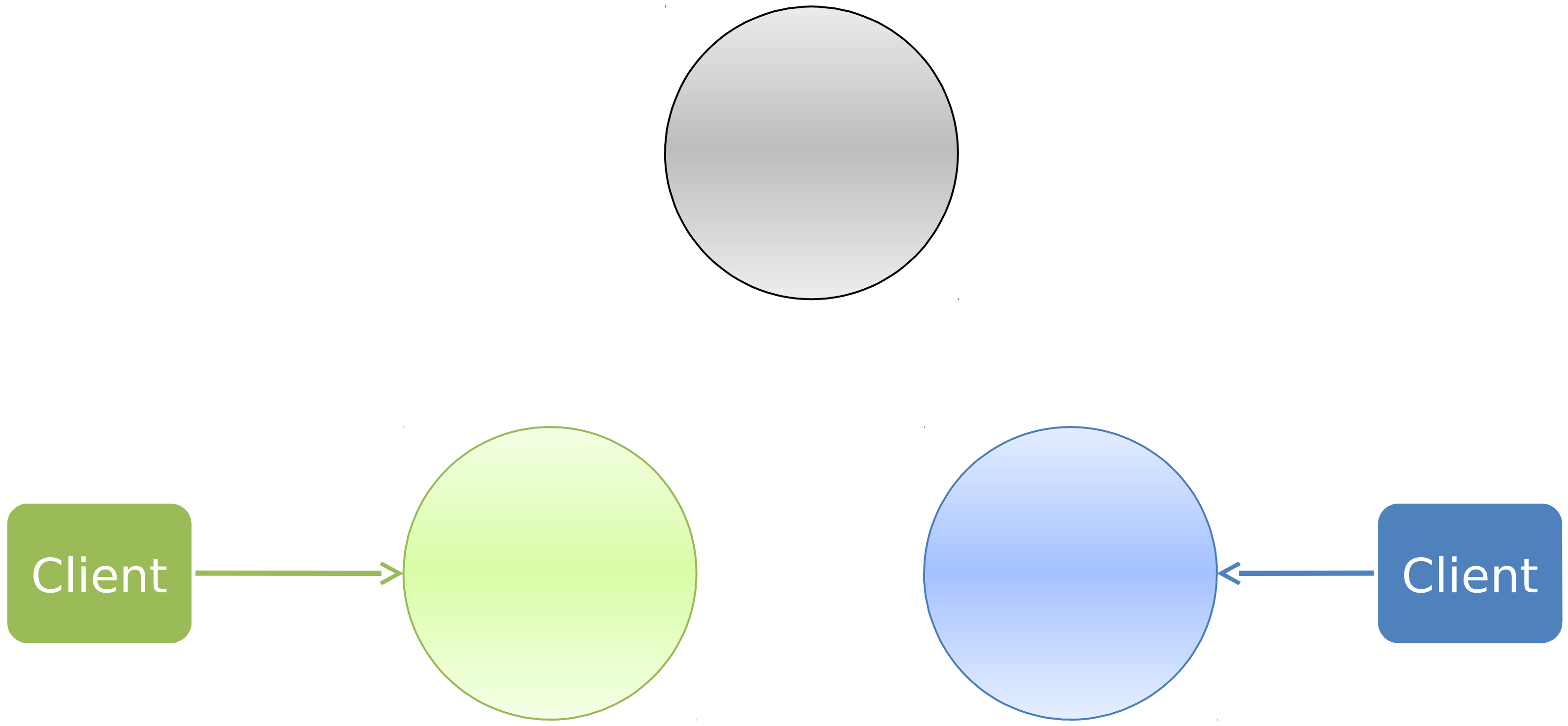
# Decided blue



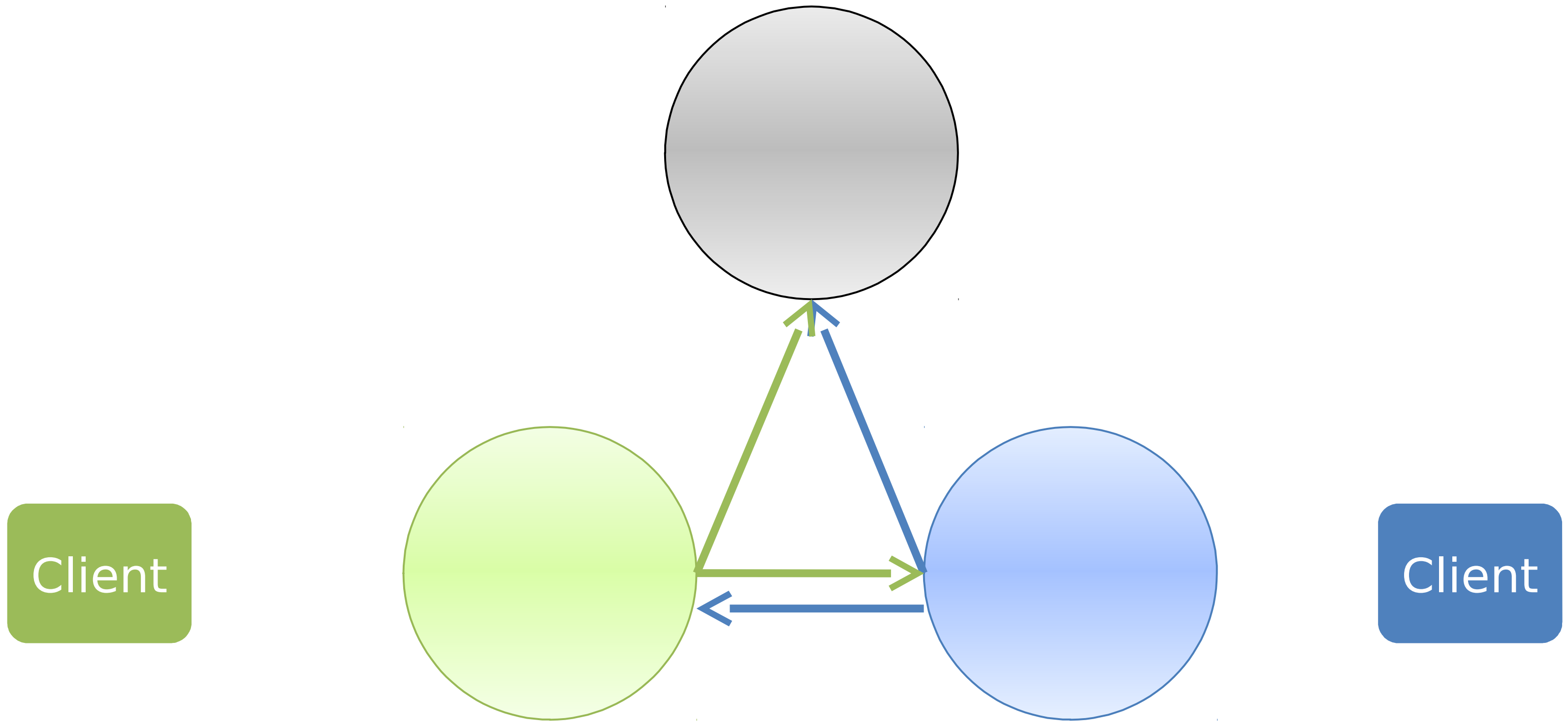
# Paxos



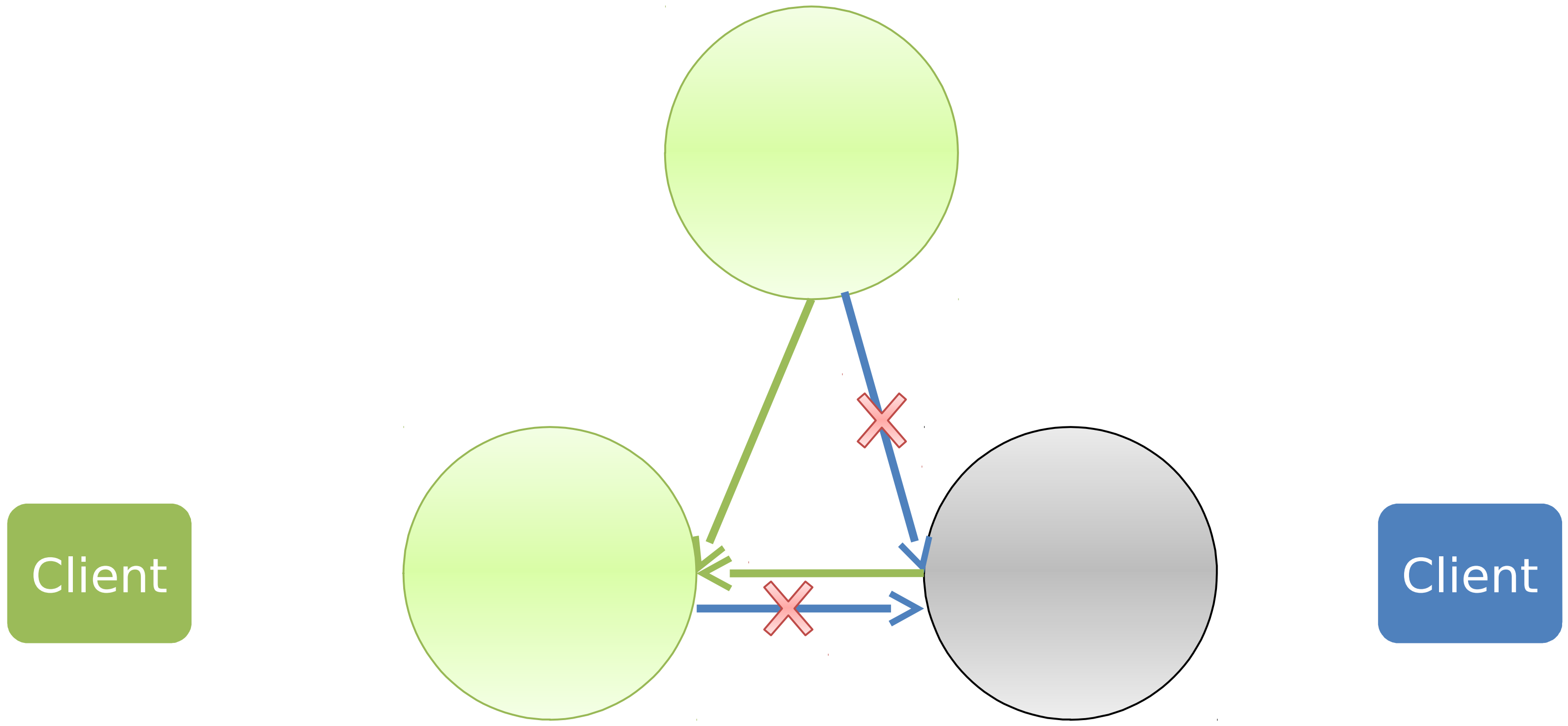
# Paxos



# Conflicting Prepares



## Highest Proposal # Wins



```
proposer(v):
 while not decided:
  choose n, unique and higher than any n seen so far
  send prepare(n) to all servers including self
  if prepare_ok(n_a, v_a) from majority:
   v' = v_a with highest n_a; choose own v otherwise
   send accept(n, v') to all
   if accept_ok(n) from majority:
    send decided(v') to all

acceptor's state:
 n_p (highest prepare seen)
 n_a, v_a (highest accept seen)

acceptor prepare_handler(n):
 if n > n_p
  n_p = n
  reply prepare_ok(n_a, v_a)
 else
  reply prepare_reject

acceptor accept_handler(n, v):
 if n >= n_p
  n_p = n
  n_a = n
  v_a = v
  reply accept_ok(n)
 else
  reply accept_reject
```

## Global Log Primitive

- Each operation (read or write) as an entry in the log
- Everyone agrees on the log
- Everyone applies operations in log order
- Externally consistent

ZAB (Zookeeper), Viewstamped Replication

Paxos lets us guarantee correctness with a functioning majority

Paxos does not guarantee *liveness*

# CAP Theorem

- Brewer’s PODC talk: “Consistency, Availability, Partition-tolerance: choose two” in 2000
  - Partition-tolerance is a failure model
  - Choice: can you process reads and writes during a partition or not?

- FLP result – “Impossibility of Distributed Consensus with One Faulty Process” in 1985
  - Asynchronous model; cannot tell the difference between message delay and failure

## What does this mean?

It’s impossible to decide anything on the internet?

## NP-hard

# What does CAP mean?

It’s impossible to 100% of the time decide everything on the internet if we can’t rely on synchronous messaging

We can 100% of the time decide everything if partitions heal (we know the upper bound on message delays)

We can still play Candy Crush

## ~~CAP~~
## Consistency vs. Performance

Paxos is many rounds of messages.

How do we reduce # messages while:

- Producing a correct ordering of reads and writes and
- Handling failures and making progress?

## Real World Systems

Google’s distributed database Spanner:

*“We believe it is better to have application programmers deal with performance problems due to overuse of transactions as bottlenecks arise, rather than always coding around the lack of transactions.”*

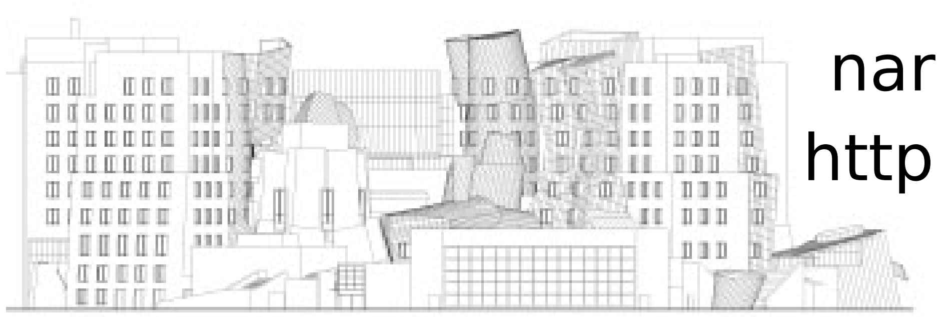
## Summary

- Consistency makes our lives a lot easier and programming with guarantees is HARD.
- We should be focusing on how to improve the performance of consistent systems instead of worrying about impossibility results.

# Further Reading

- Fischer, Lynch, Paterson: Impossiblity of Consensus with One Faulty Process. Journal of the ACM, 1985
- Henry Robinson: http://the-paper-trail.org/blog/a-brief-tou
- Eric Brewer: http://www.infoq.com/articles/cap-twelve

# Thanks!



narula@mit.edu

http://nehanaru.l
a

@neha

CSAIL

The Stata Center via emax: http://hip.cat/emax/