CS 4530 & CS 5500 Software Engineering

Lecture 9.2: Strategies for Engineering Distributed Software

Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences

Learning Objectives for this Lesson

By the end of this lesson, you should be able to…

- Characterize the benefits of replication and partitioning in distributed systems
- Evaluate the tradeoffs between consistency and availability in distributed systems

Why expand to distributed systems?

- Scalability
- Performance
- Latency
- Availability
- Fault Tolerance

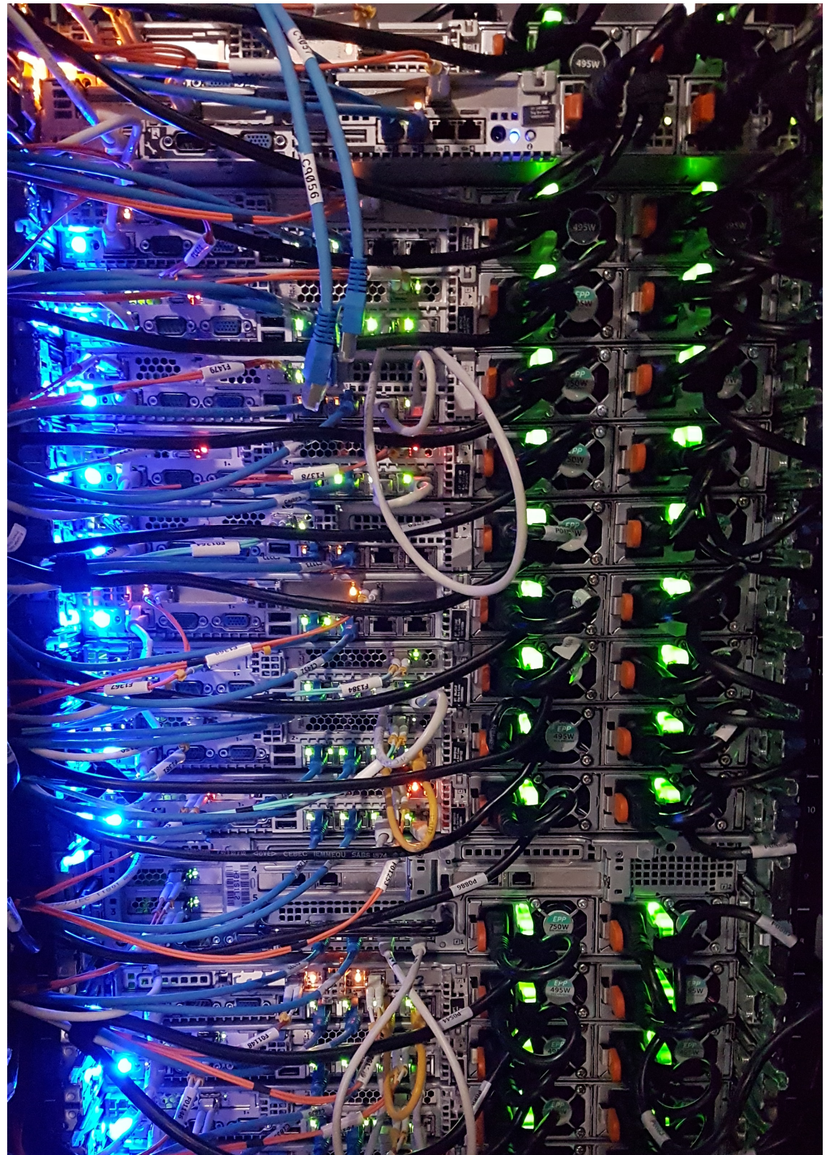
“Distributed Systems for Fun and Profit”, Takada

8 Fallacies of Distributed Computing

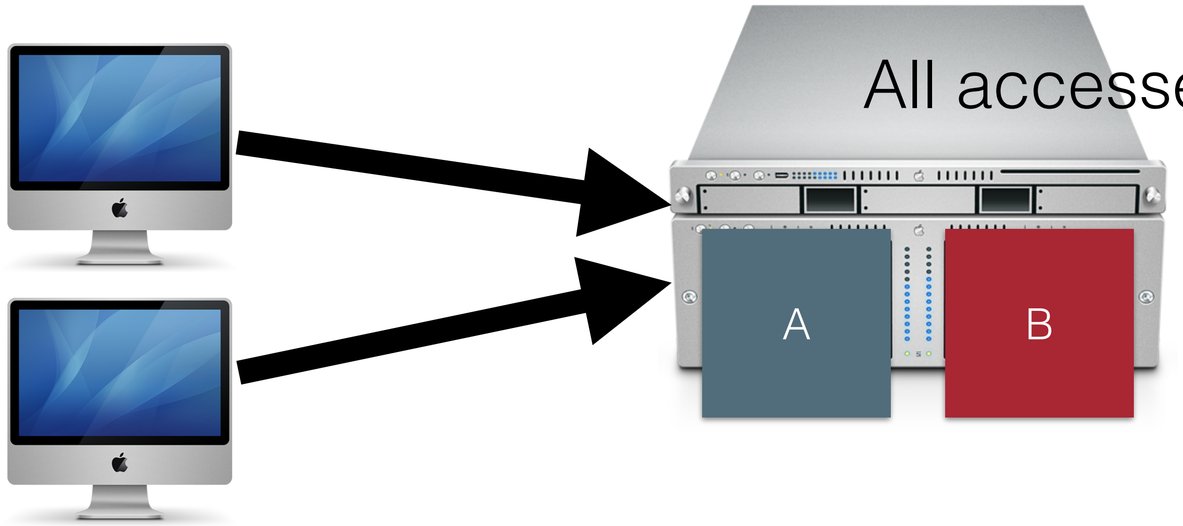
Sun Microsystems, early 1990's

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. Transport cost is zero
7. The network is homogeneous
8. There is one administrator

Physical Properties of networks



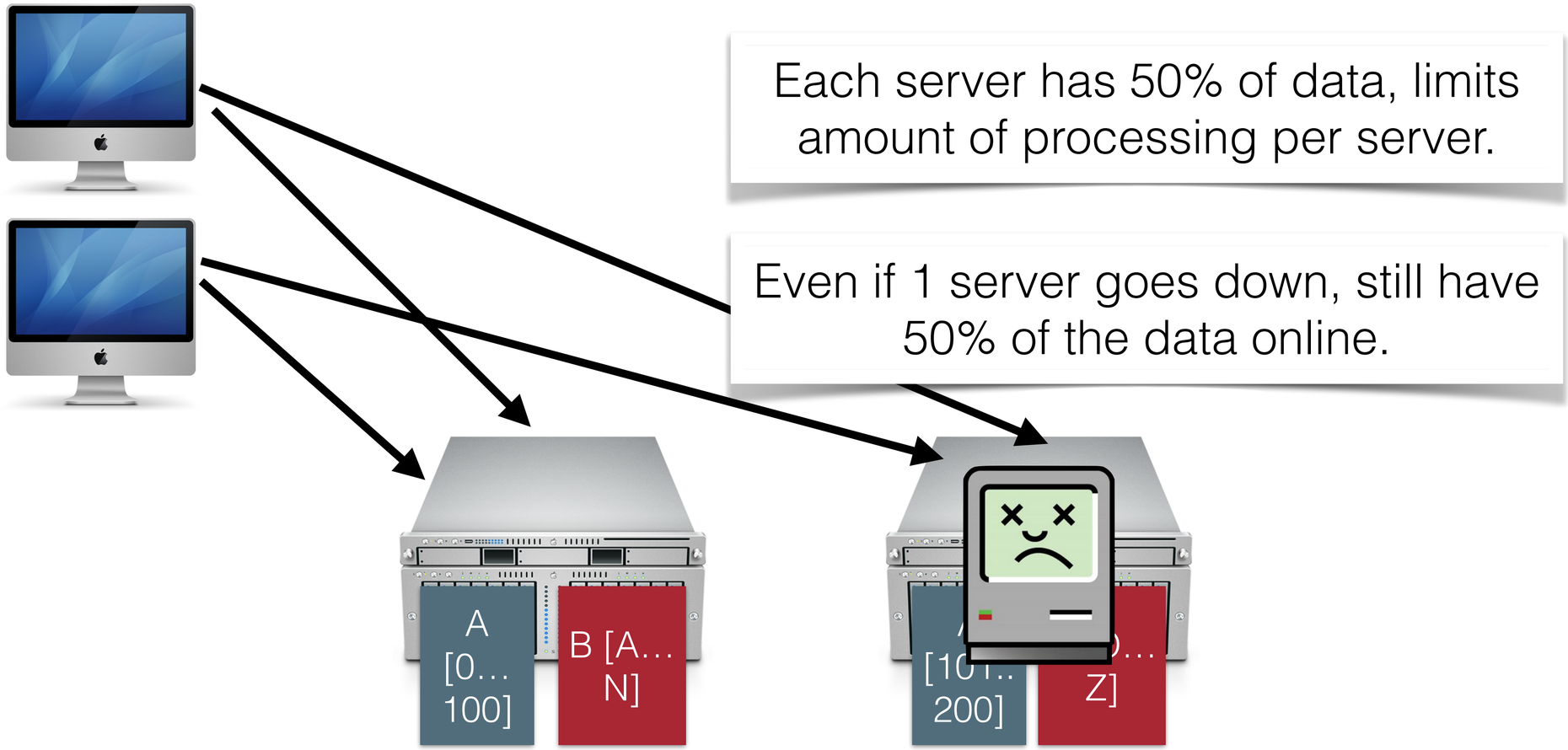
Recurring Solution #1: Partitioning



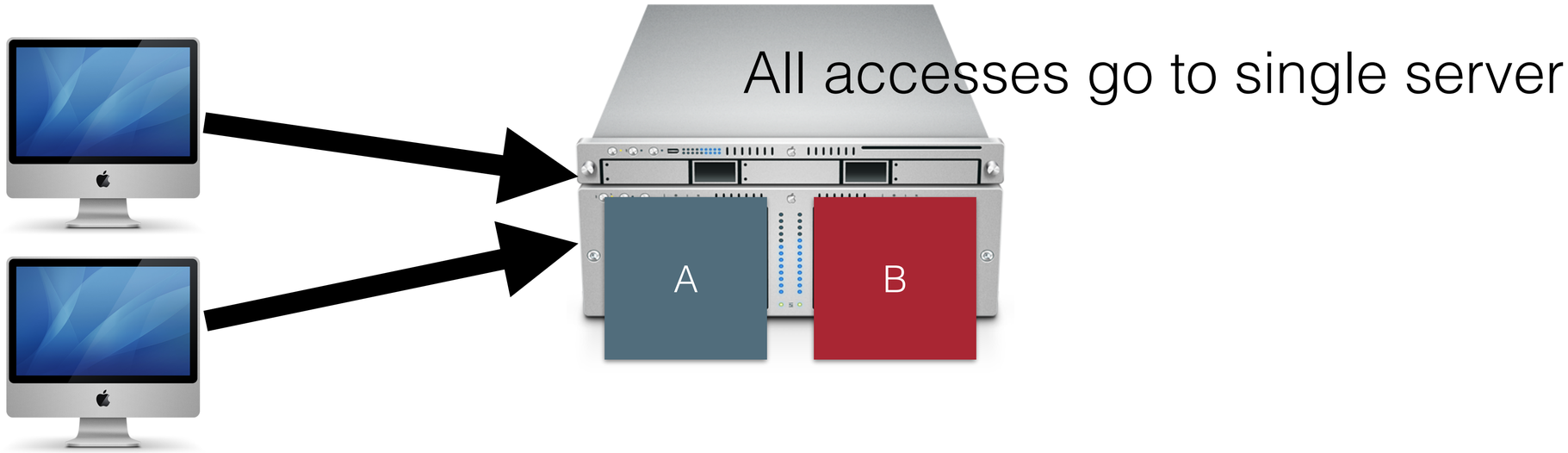
All accesses go to single server

Recurring Solution #1: Partitioning

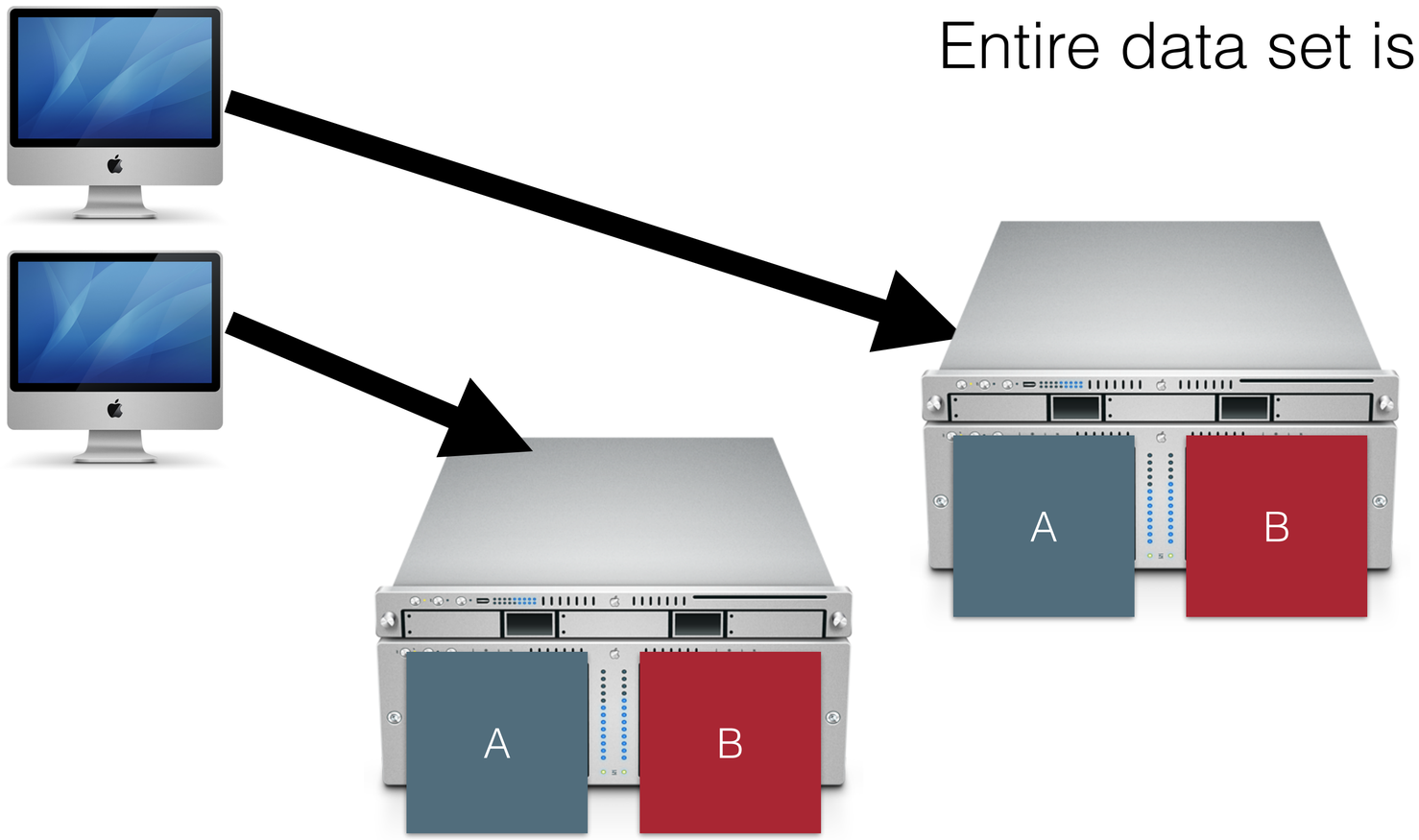
- Divide data up in some (hopefully logical) way
- Makes it easier to process data concurrently (cheaper reads)



Recurring Solution #2: Replication

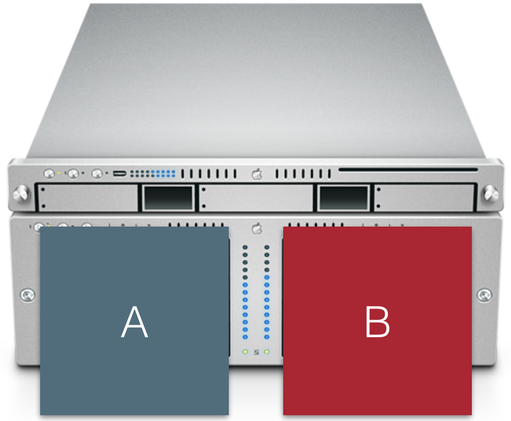


Recurring Solution #2: Replication

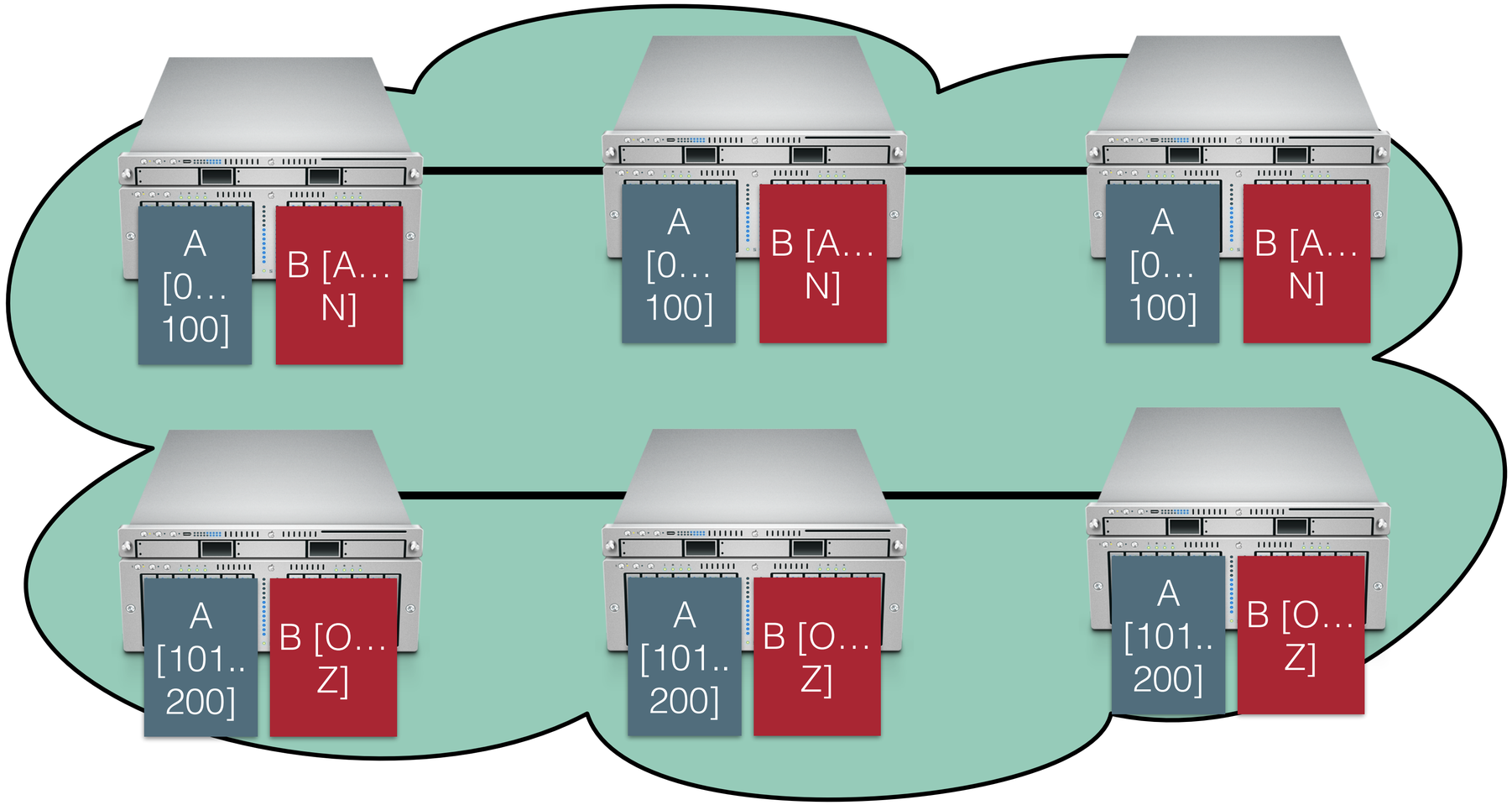


Entire data set is copied

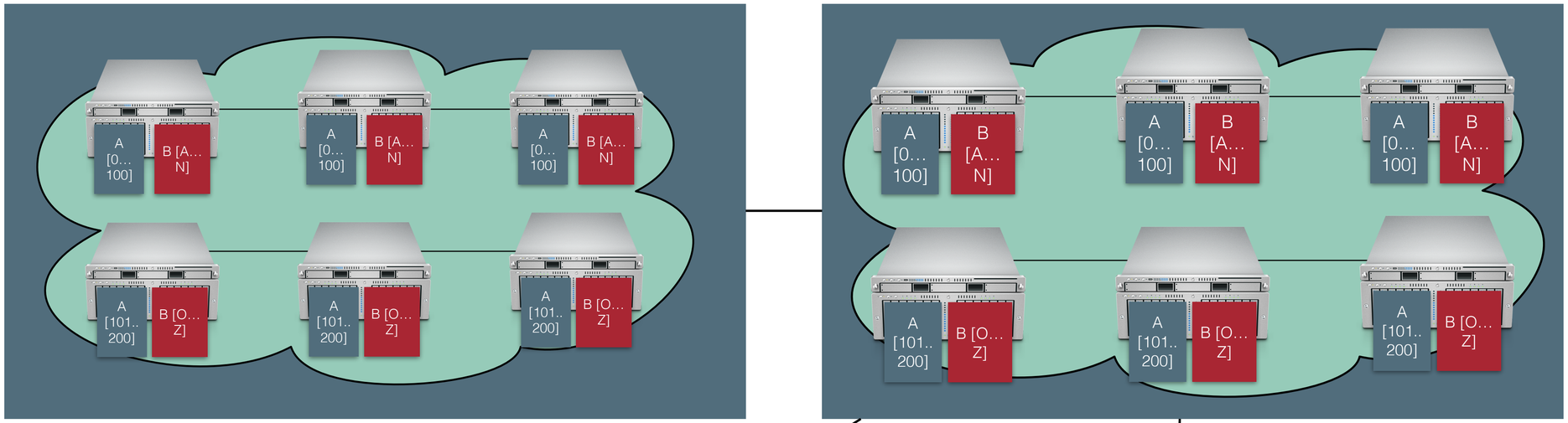
Partitioning + Replication



Partitioning + Replication

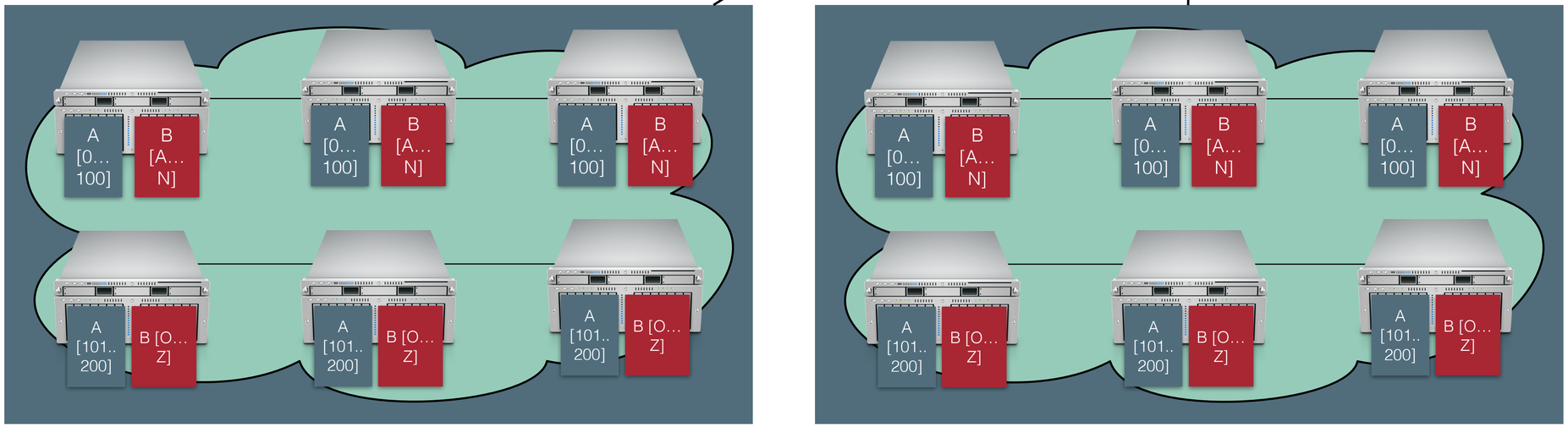


Partitioning + Replication



DC

NYC

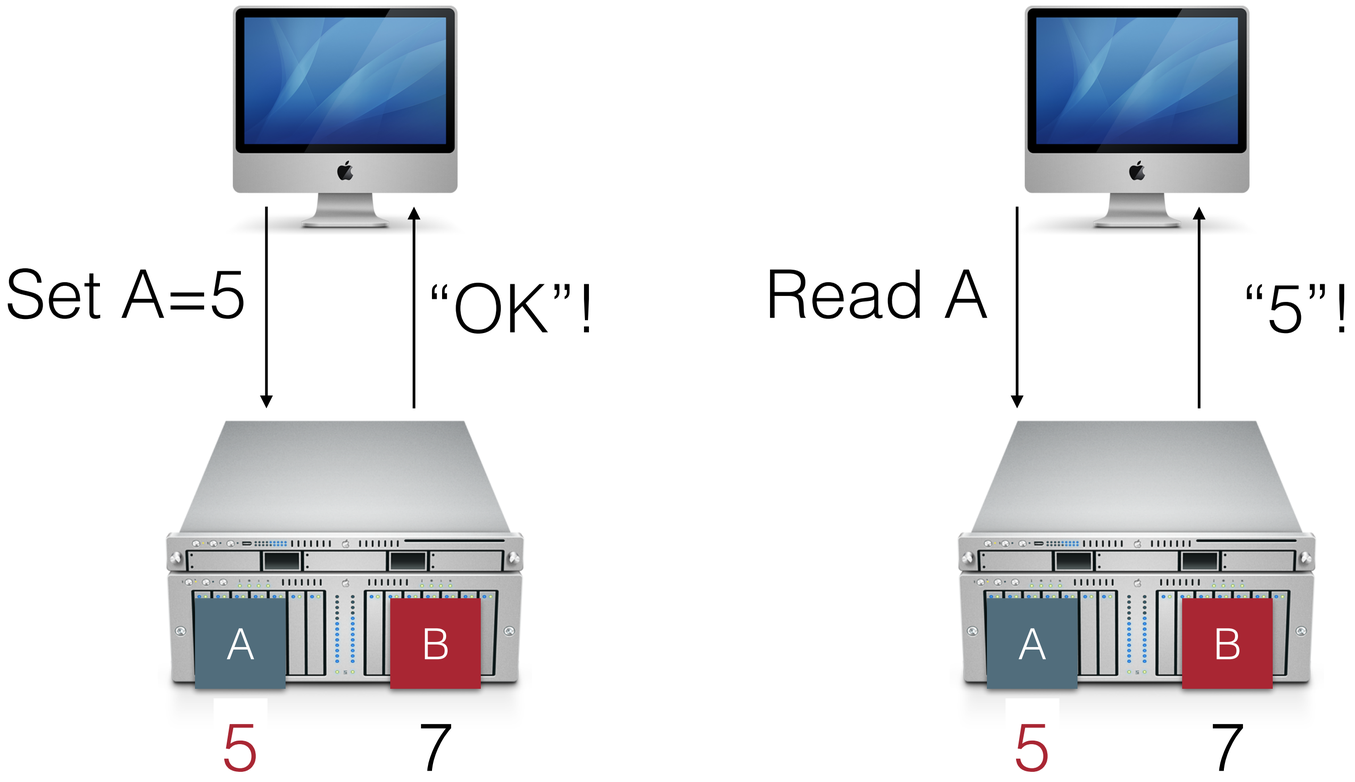


SF

London

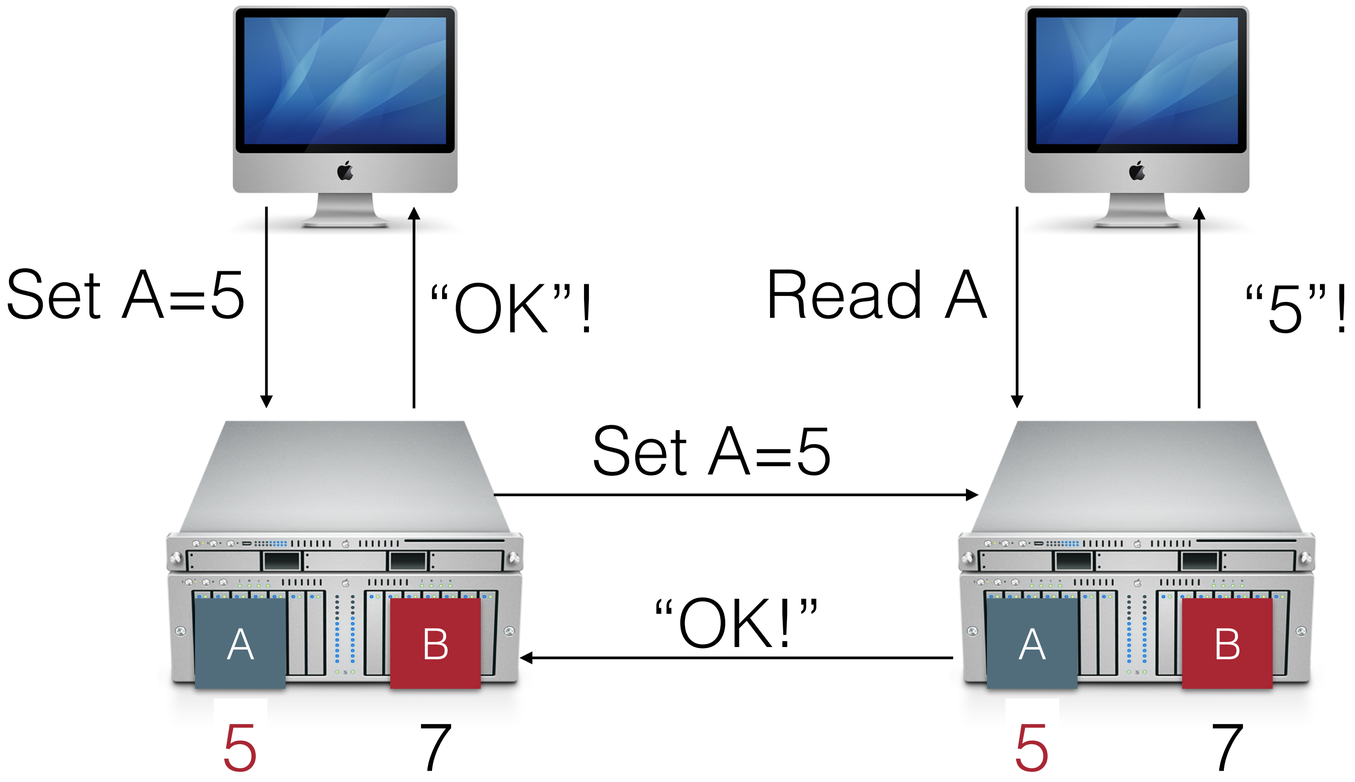
Replication Problem: Consistency

We probably want our system to work like this



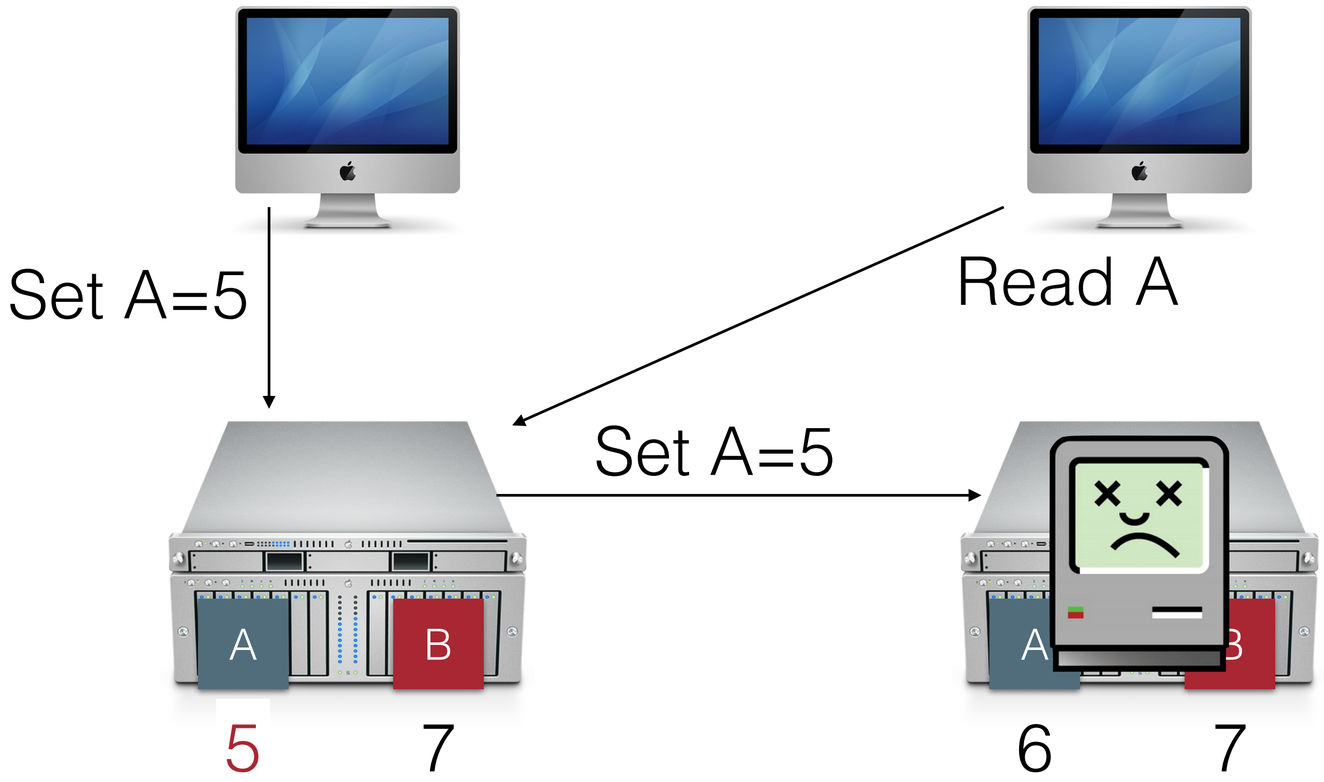
Sequential Consistency

AKA: Behaves like a single machine would



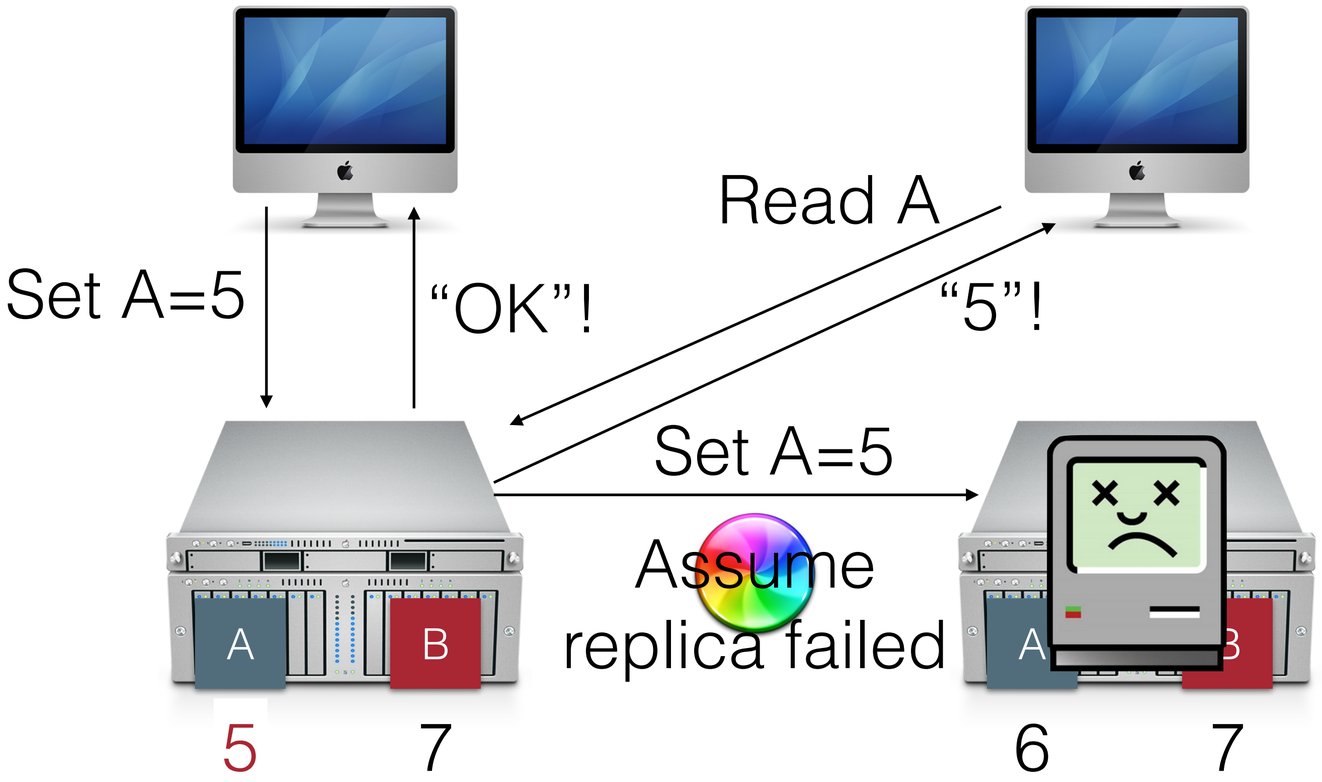
Availability

If at least one node is online, can we still answer a request?



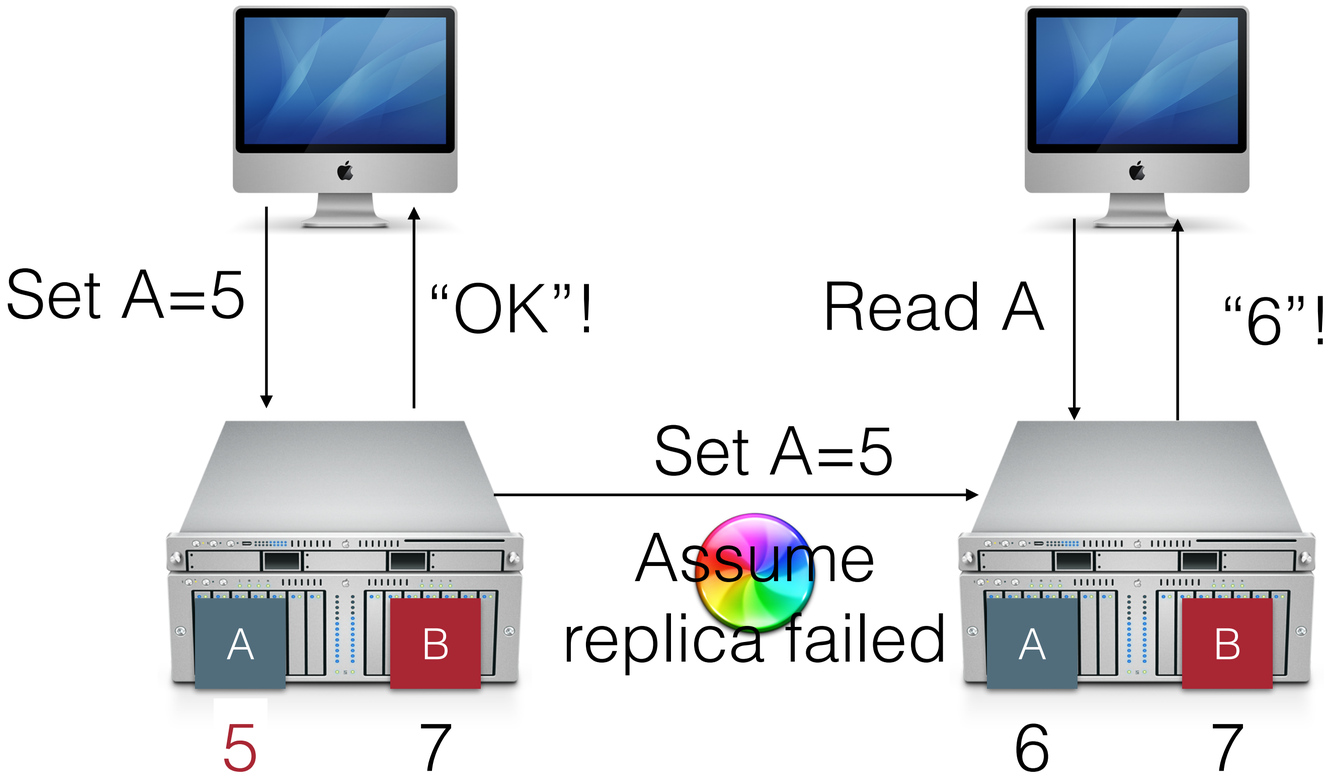
Consistent + Available

On timeout, assume node is crashed



Recall: Fallacies of Distributed Systems

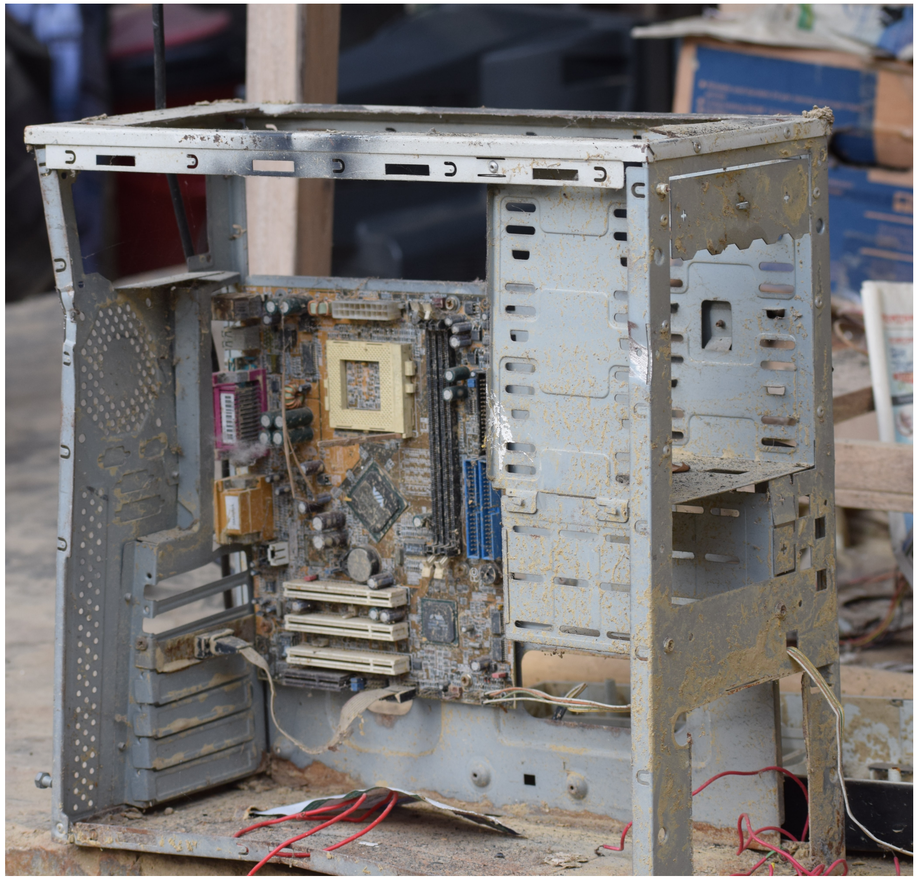
What if *the network* fails?



Shared Fate

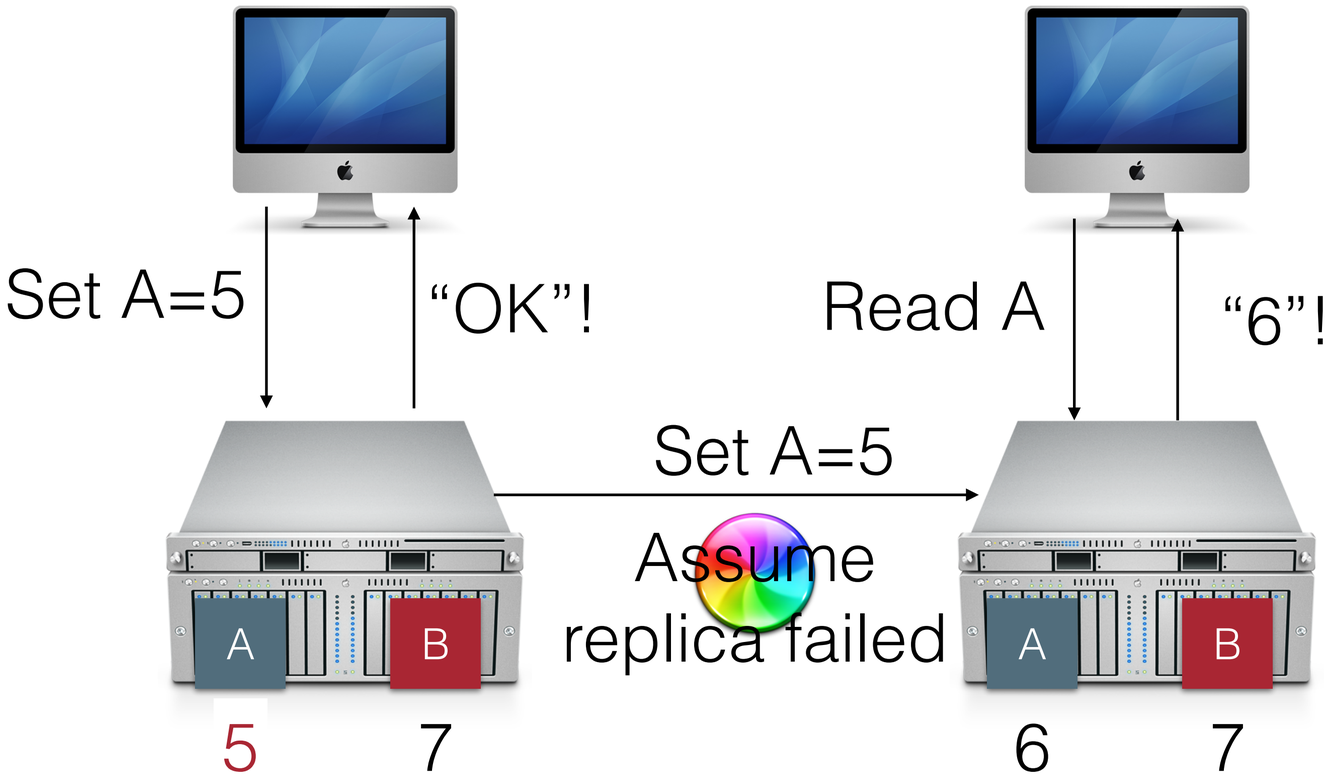
Are you still there?

- Two methods/threads/processes running on the same computer generally have **shared fate** [Crashed/not]
- When two machines in a distributed system can't talk to each other, how do we know if the other is crashed?
- We call this a **split brain** problem



Network Partitions

The network might fail *arbitrarily!*

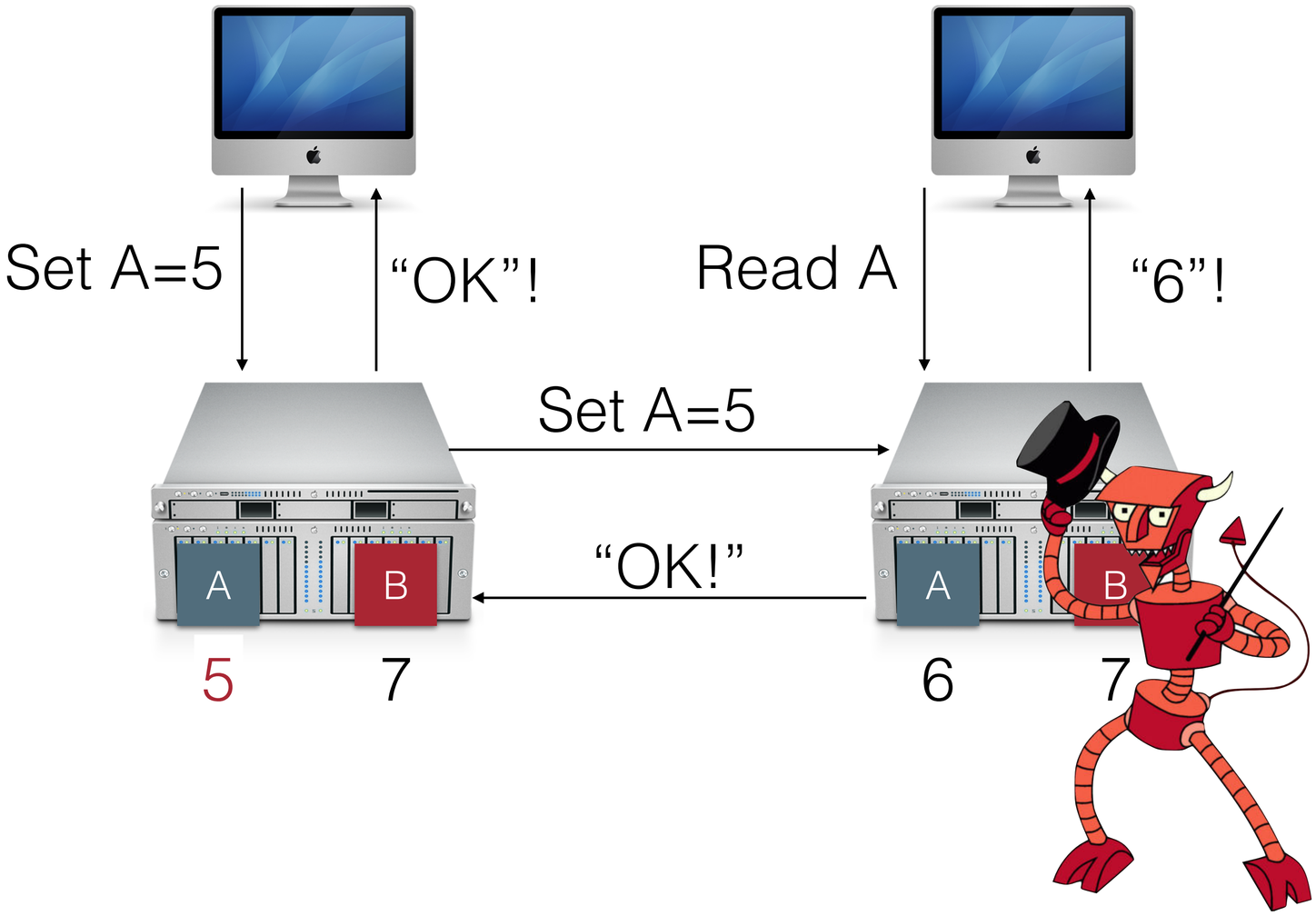


CAP Theorem

- Pick two of three:
  - Consistency: All nodes see the same data at the same time (strong consistency)
  - Availability: Individual node failures do not prevent survivors from continuing to operate
  - Partition tolerance: The system continues to operate despite message loss (from network and/or node failure)

Byzantine Faults

Unfortunately, still more things can go wrong



Distributed Software Engineering Abstractions

Key Question: Consistency vs Availability

- Distributed system will never match exact semantics of non-distributed system
- For replication do we value more: guaranteed consistency (looks like a single machine) or guaranteed availability (sometimes read stale data)?
  - For a lock server?
  - For the order of tweets on twitter?
- For partitioning: Where can we draw the line?

Distributing Covey.Town

Partitioning: Separate responsibilities by town

TownID	TownController
Town1	A
Town2	A
Town3	B
Town4	C

CoveyTownsStore

CoveyTownController "A"

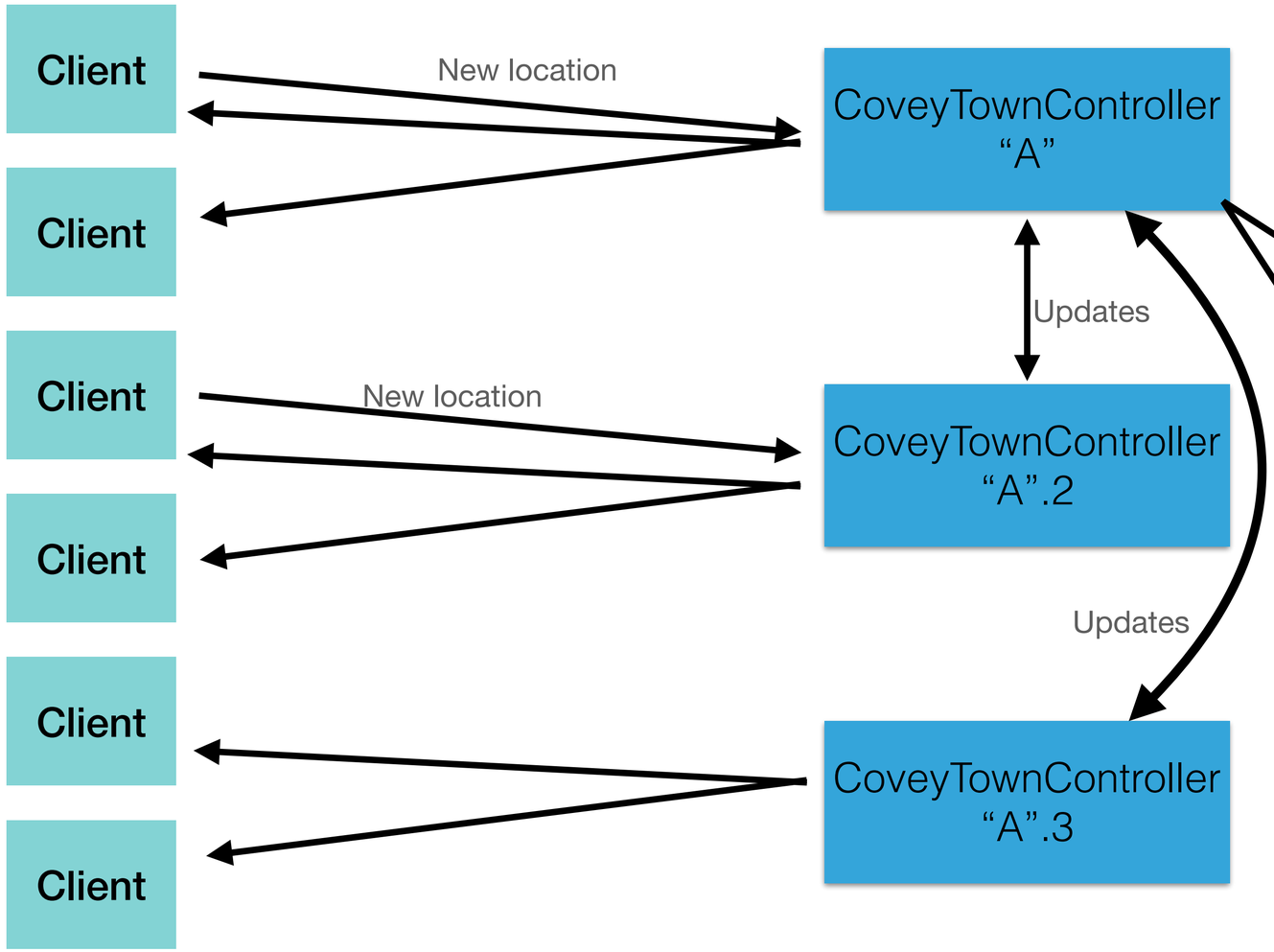
CoveyTownController "B"

CoveyTownController "C"

TownID	Players
Town1	[player1, player2, player3]
Town2	[player12, player26, player34]

Distributing Covey.Town

Replicating the Town Controller



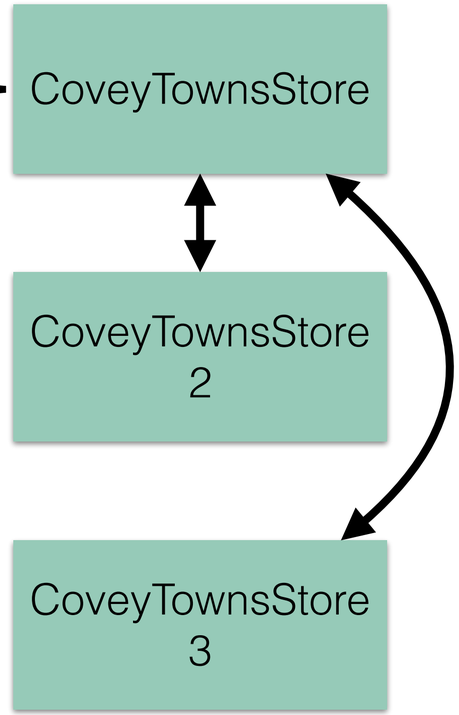
To be consistent: Everyone must see each player updates in same order (as if just 1 TownController) - but we don't need that, right?

TownID	Players
Town1	[player1, player2, player3]
Town2	[player12, player26, player34]

Distributing Covey.Town

Replicating the Town Store

TownID	TownController
Town1	A
Town2	A
Town3	B
Town4	C



What are our consistency challenges?

- Each town must have a unique ID
- Ideally - distribute towns to controllers evenly?

Should we even do this?

Learning Objectives for this Lesson

By the end of this lesson, you should be able to…

- Characterize the benefits of replication and partitioning in distributed systems
- Evaluate the tradeoffs between consistency and availability in distributed systems

This work is licensed under a Creative Commons Attribution-ShareAlike license