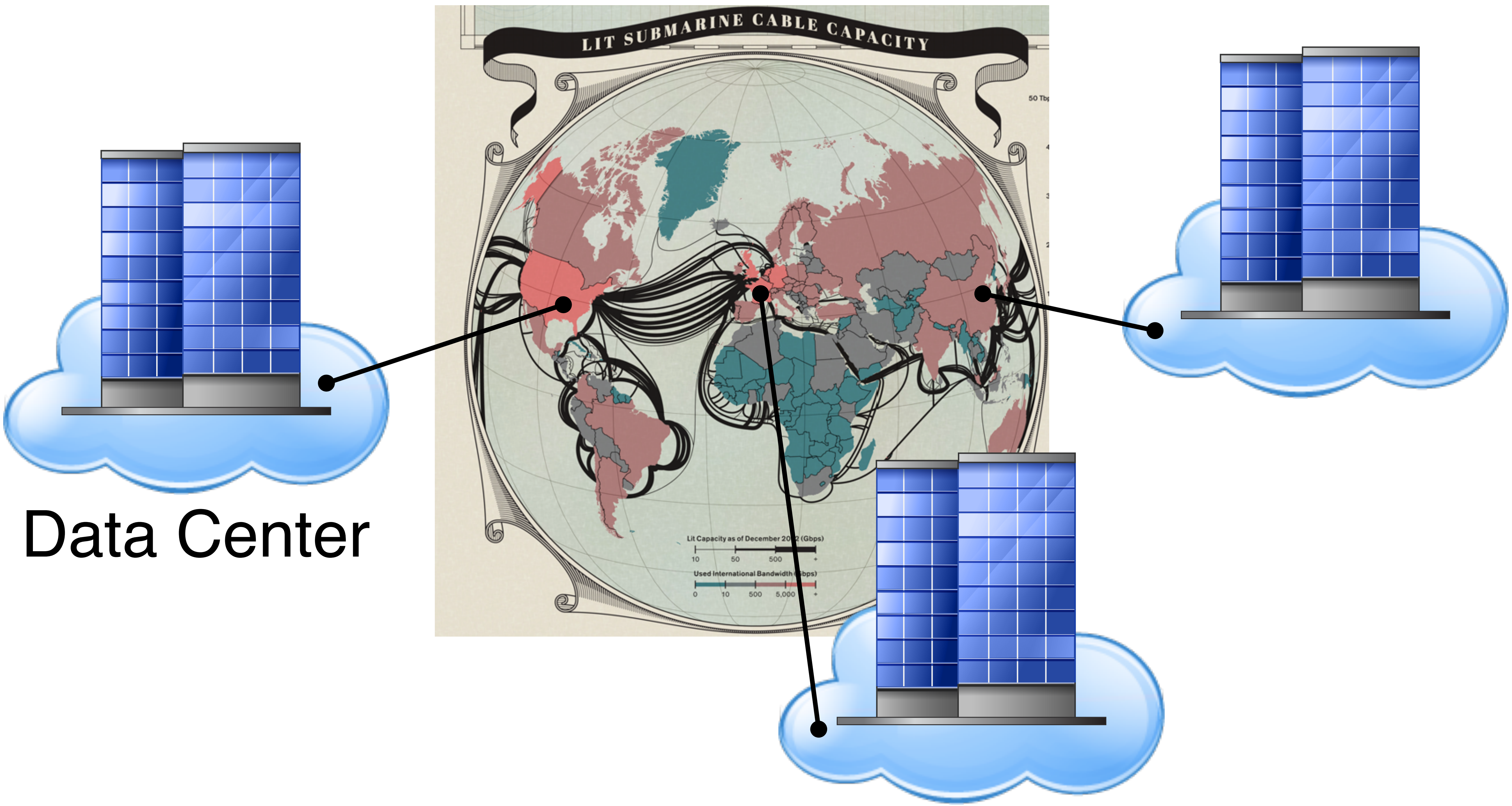
Distributed Systems CS6421

Storage: Scale and Fault Tolerance

Prof. Tim Wood

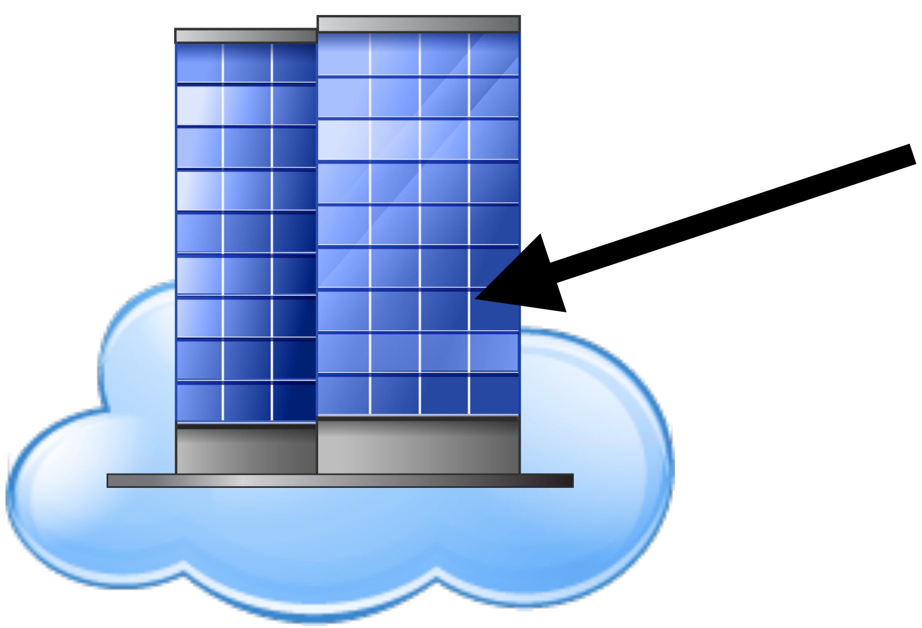
Our picture of the cloud…

Data centers connected around the world…



Our picture of the cloud…

Filled with lots of servers…

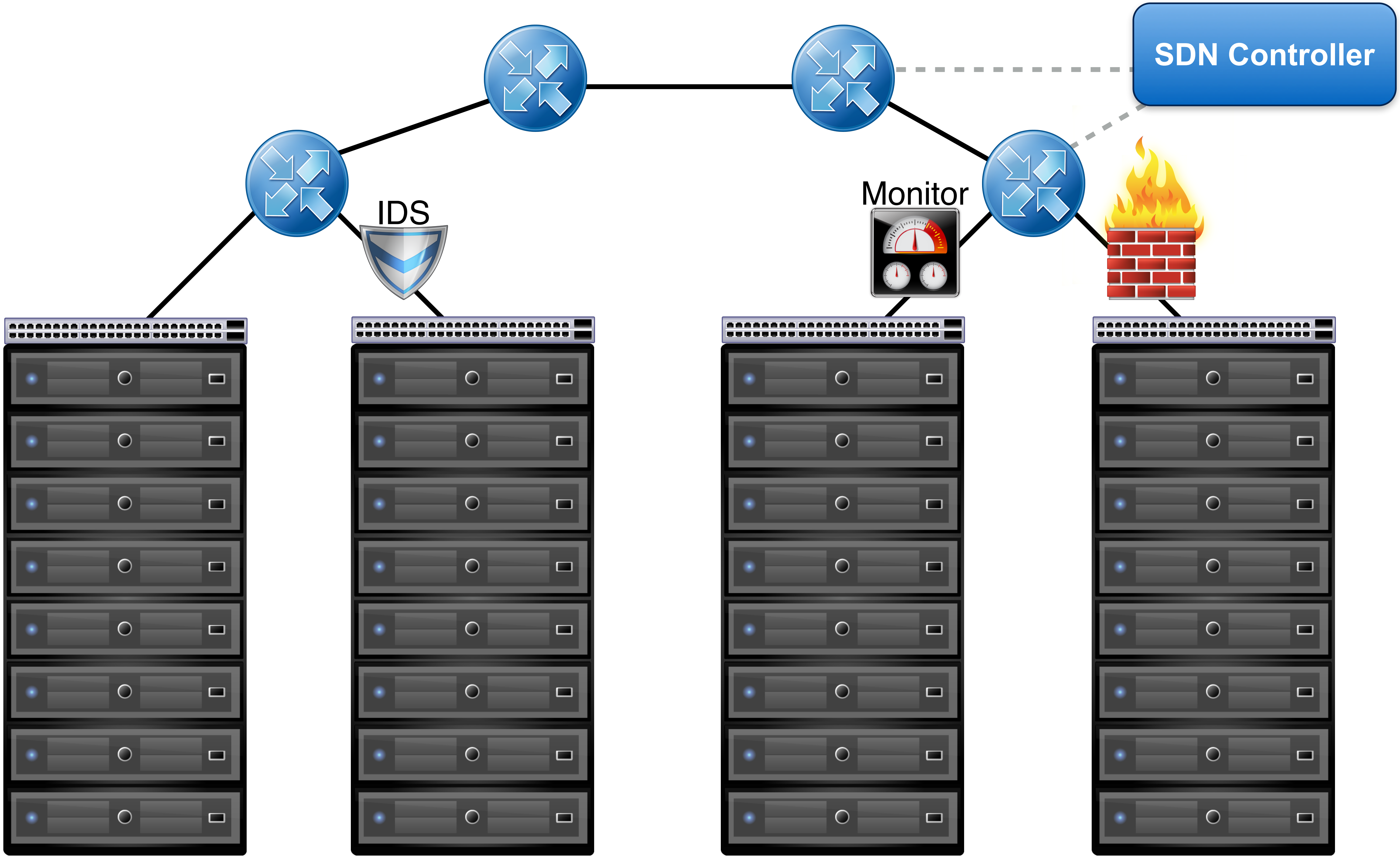


Data Center

Racks of servers

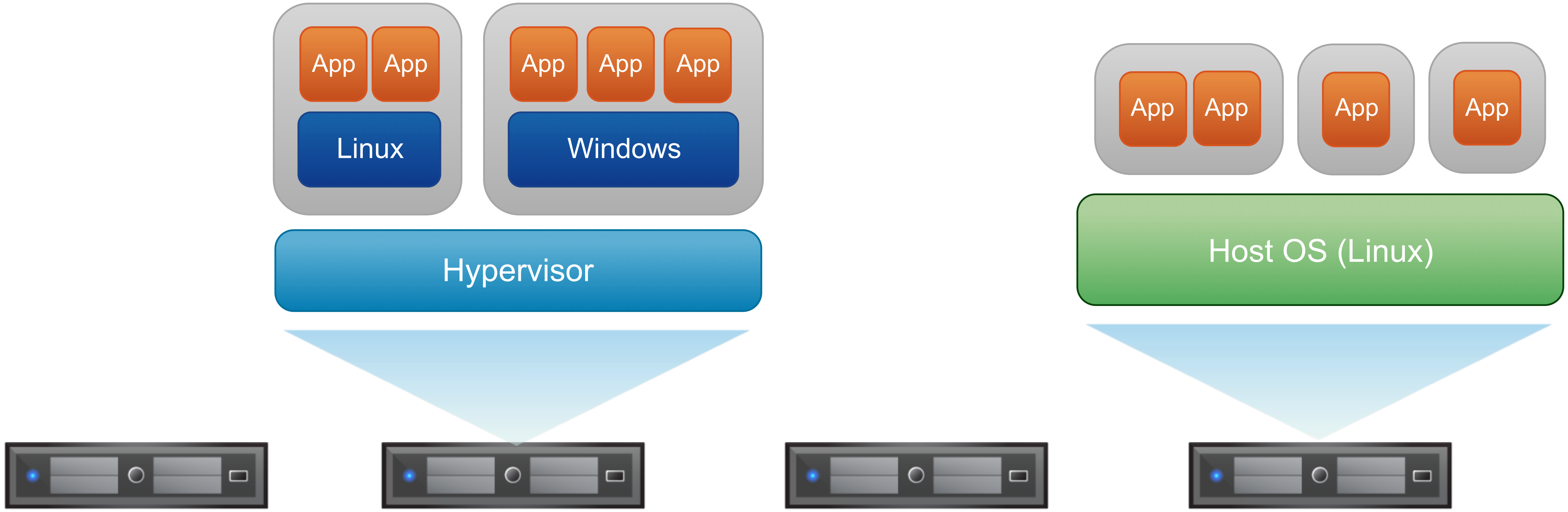
Our picture of the cloud…

Connected with routers, switches, and middle boxes



Our picture of the cloud…

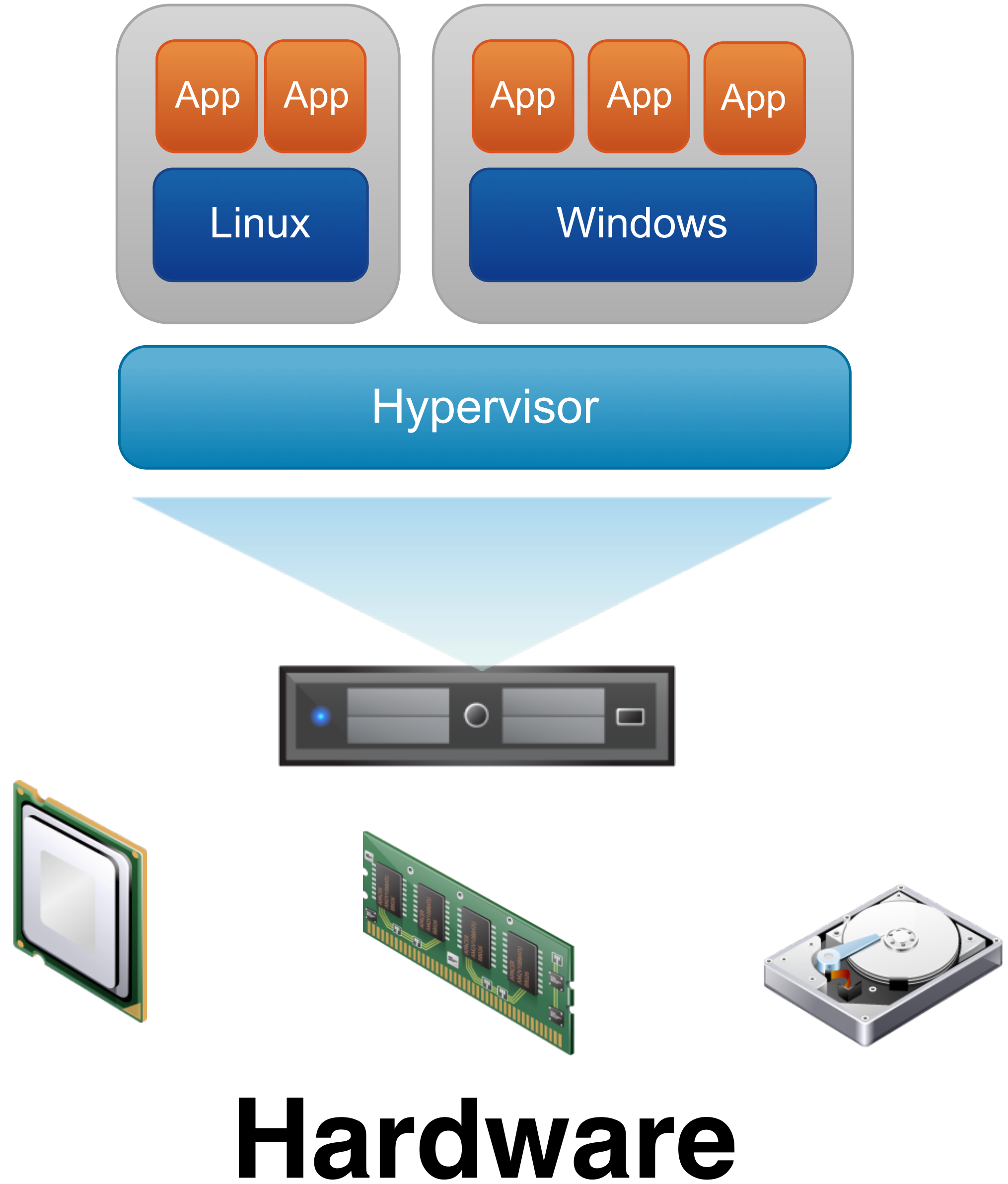
Each hosting VMs or containers



Resources?

CPU, RAM, and Disks

- How is storage different?

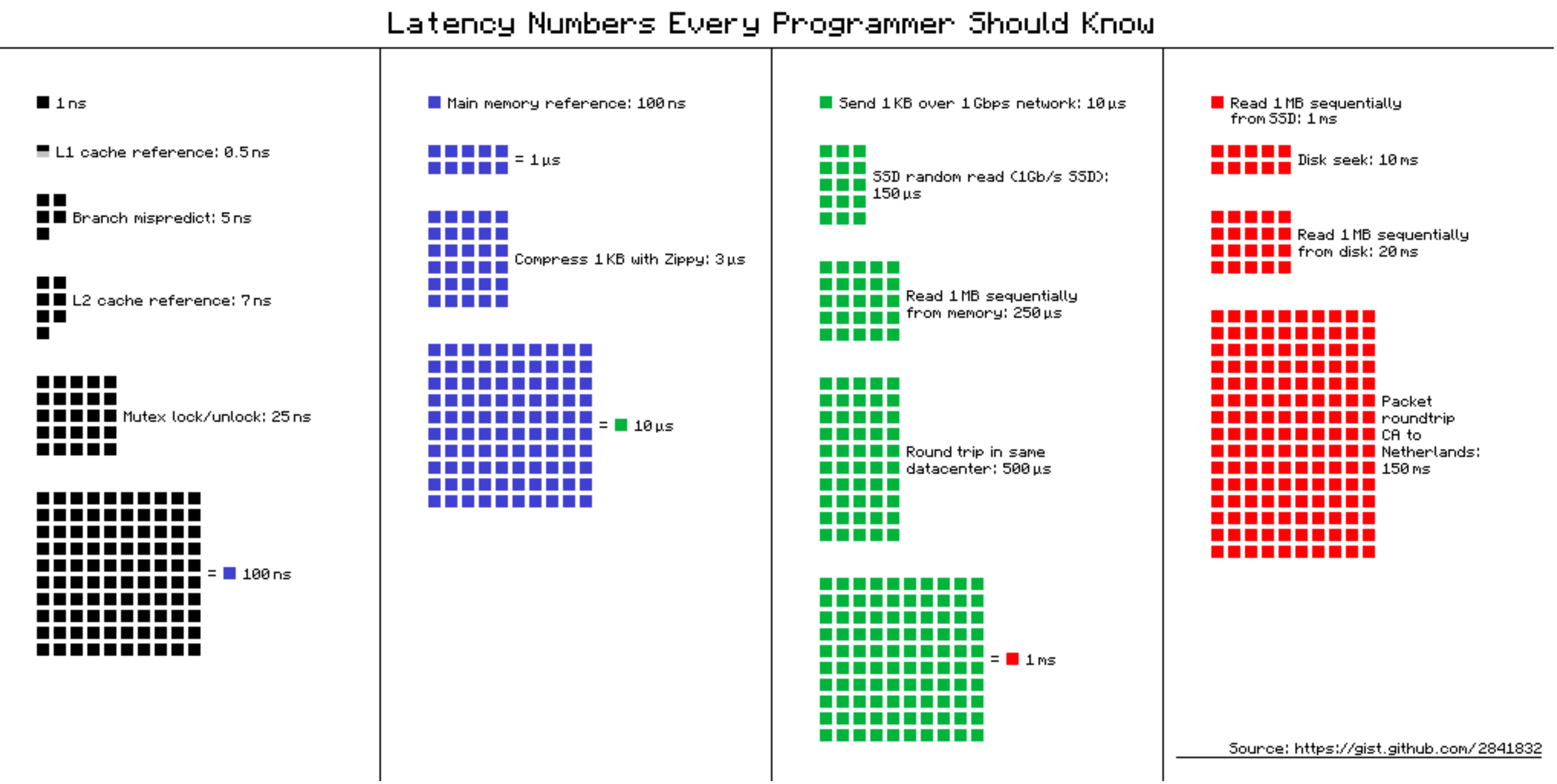


Numbers you should know…

How long to…

```
L1 cache reference ......................... 0.5 ns
Branch mispredict ............................ 5 ns
L2 cache reference ........................... 7 ns
Mutex lock/unlock ........................... 25 ns
Main memory reference ...................... 100 ns
Compress 1K bytes with Zippy ............. 3,000 ns  =   3 µs
Send 2K bytes over 1 Gbps network ....... 20,000 ns  =  20 µs
SSD random read ........................ 150,000 ns  = 150 µs
Read 1 MB sequentially from memory ..... 250,000 ns  = 250 µs
Round trip within same datacenter ...... 500,000 ns  = 0.5 ms
Read 1 MB sequentially from SSD* ..... 1,000,000 ns  =   1 ms
Disk seek ........................... 10,000,000 ns  =  10 ms
Read 1 MB sequentially from disk .... 20,000,000 ns  =  20 ms
Send packet CA->Netherlands->CA .... 150,000,000 ns  = 150 ms
```

Numbers you should know…



also: https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html

Scale it up…

After multiplying everything by a billion…

L1 cache reference	0.5 s	One heart beat (0.5 s)
L2 cache reference	7 s	Long yawn
Main memory reference	100 s	Brushing your teeth
Send 2K bytes over 1 Gbps network	5.5 hr	From lunch to end of work day
SSD random read	1.7 days	A normal weekend
Read 1 MB sequentially from memory	2.9 days	A long weekend
Round trip within same datacenter	5.8 days	A medium vacation
Read 1 MB sequentially from SSD	11.6 days	Waiting 2 weeks for a delivery
Disk seek	16.5 weeks	A semester in university
Read 1 MB sequentially from disk	7.8 months	Almost producing a new human being
The above 2 together	1 year	
Send packet CA->Netherlands->CA	4.8 years	Going to university for BS degree

???

Accessing 1MB data

Data in RAM:

- 3 days (250 **micro**seconds)
- 60 GBps

Data in SSD:

- 2 weeks (1 **milli**second)
- 1 GBps

Data in HDD:

- a year (20 **milli**seconds)
- 100 MBps

Send 1MB over 10Gbps network:

- length of this class (2 **milli**seconds)
- 1.25 GBps

Networked Storage

The added cost of using the network is relatively low

What are the benefits of using remote storage instead of local?

Storage Services

Block storage (EBS)

- Access to the raw bytes of a remote disk
- Unit of access: disk block (4KB)
- Mount as the disk for a VM
- Pros/Cons?
  - Different types of underlying storage (SSD vs HDD)
  - Pay per GB but I need to reserve the space in advance, number of IOPs?
  - Limited to 16TB per disk

Storage Services

Block storage (EBS)

Object storage (S3, DynamoDB)

- Get and Put “objects” in remote storage
- Unit of access: a full object
- Store static web content, data sets
- Pros/Cons?
  - Pay per object based on size, also per request, net BW?
  - Limit 5 TB per object

Storage Services

Block storage (EBS)

Object storage (S3, DynamoDB)

Database (RDS)

- Store structured rows and columns of data
- Unit of access: SQL query
- Web applications requiring transaction support
- Pros/Cons?
  - AWS handles management

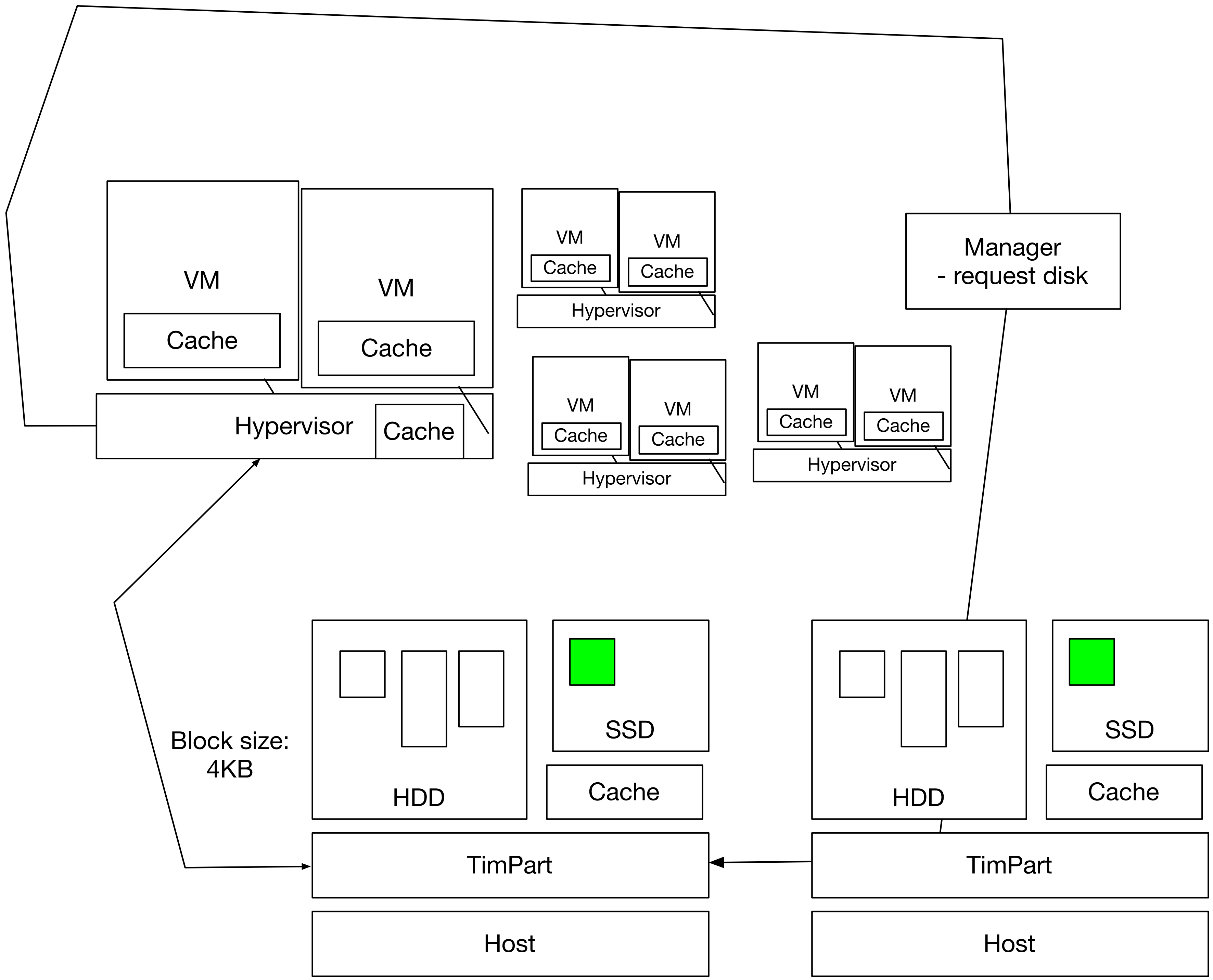
Implementation?

How would you build a Block/Object store?

- What abstraction layers?
- What should the interface look like?
- What traits do you optimize for?

What price could you sell it for?

Our Block Store



Replication Improves

Performance

- Access the least loaded or most geographically close replica

Reliability

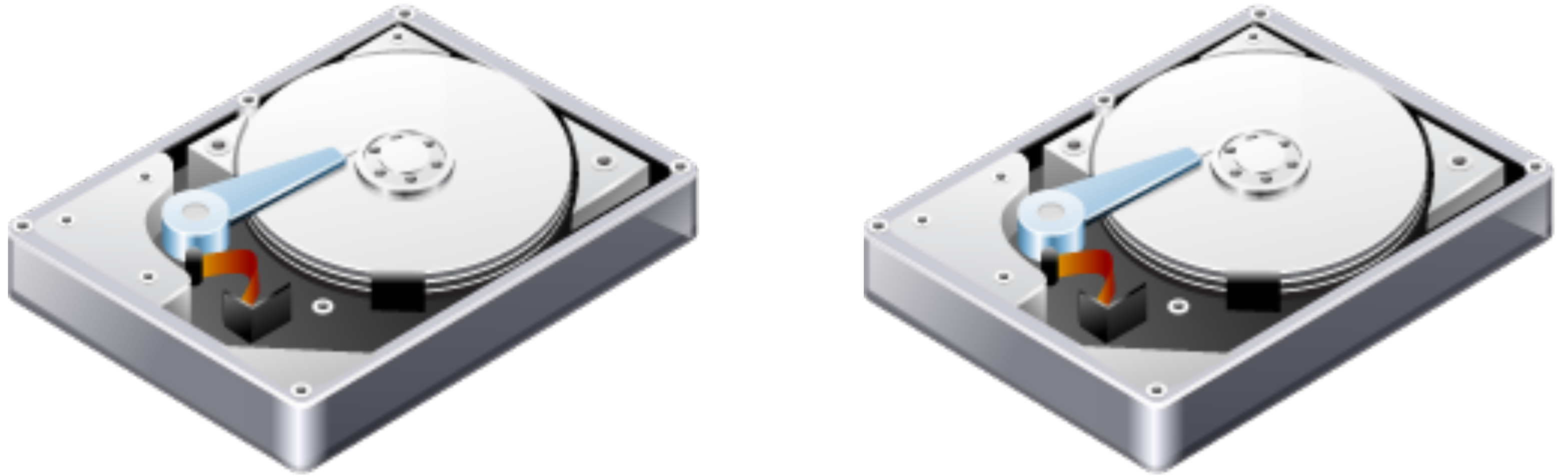
- Failure only impacts one copy, can recover from others

How many replicas?

What failures can I handle?

How many replicas?

What failures can I handle?



How many replicas?

What failures can I handle?

How many replicas?

What failures can I handle?

Types of Faults

Crash failure

- power outage, hardware crash

Content failure

- incorrect value returned
- could be permanent or transient
- could be independent or coordinated

In practice, we make assumptions about how many failures we expect to occur at once

- "At most 1 node will crash at a time"
- "At most 3 nodes will be corrupted at the same time"

Fault Tolerance through Replication

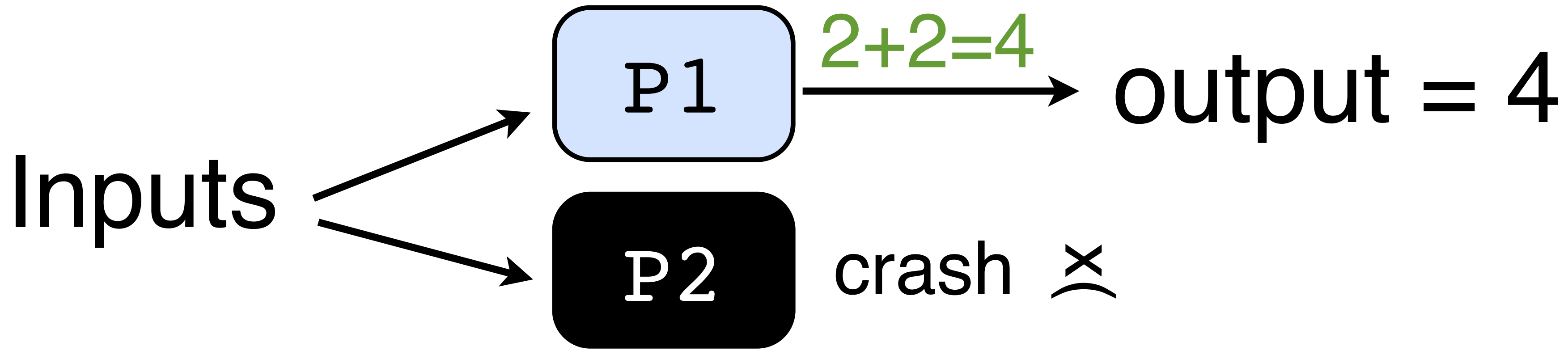
How to tolerate a **crash** failure?

- A server suddenly disappears

Fault Tolerance through Replication

How to tolerate a **crash** failure?

- A server suddenly disappears

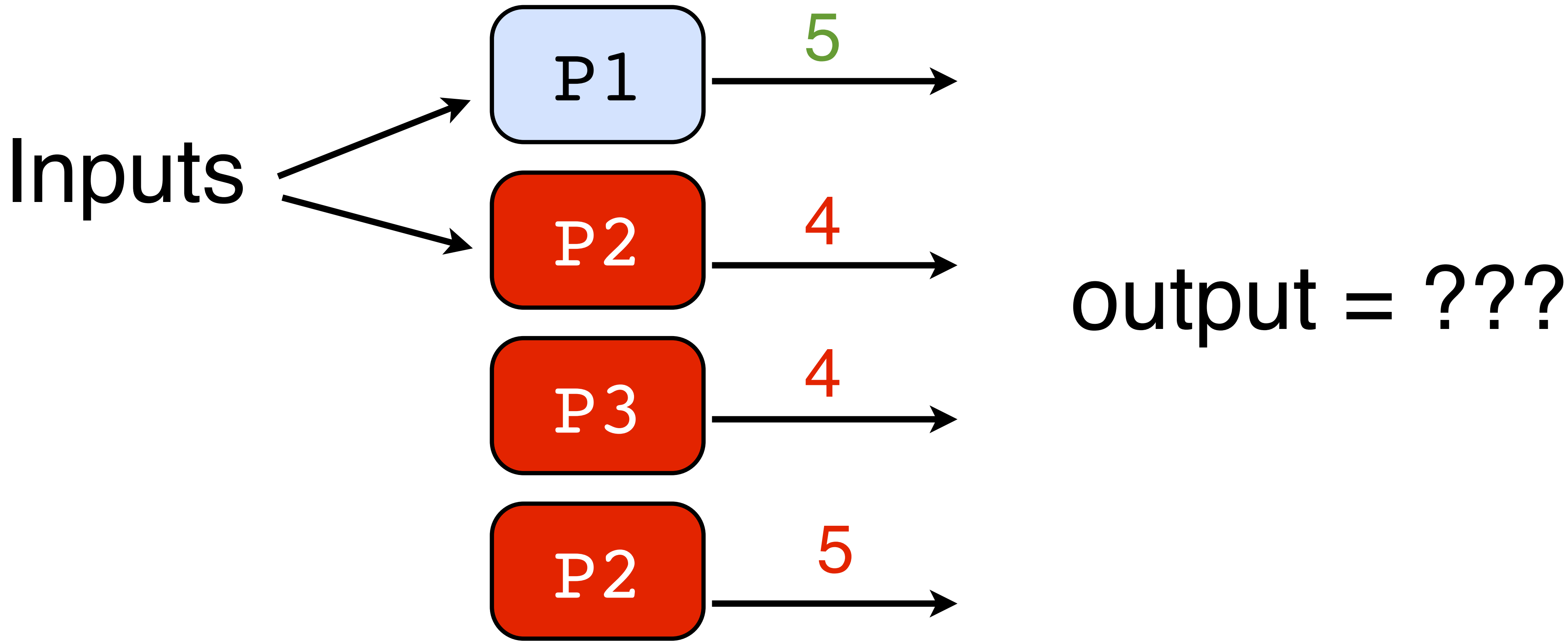


If we expect at most *f* faulty nodes, how many servers do we need total?

Fault Tolerance through Replication

How to tolerate a **content** failure?

- A server starts producing an incorrect result



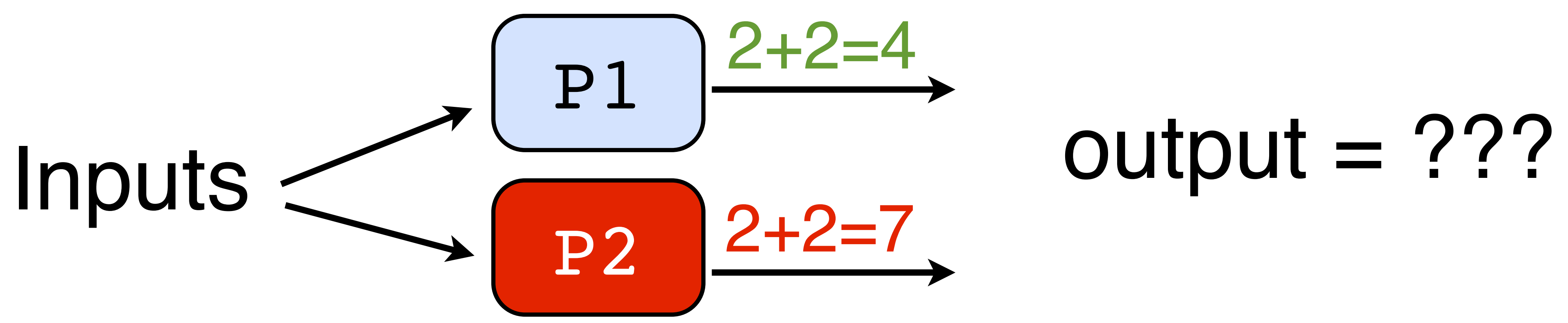
f=2.   need f+2

Fault Tolerance through Replication

How to tolerate a **content** failure?

- A server starts producing an incorrect result

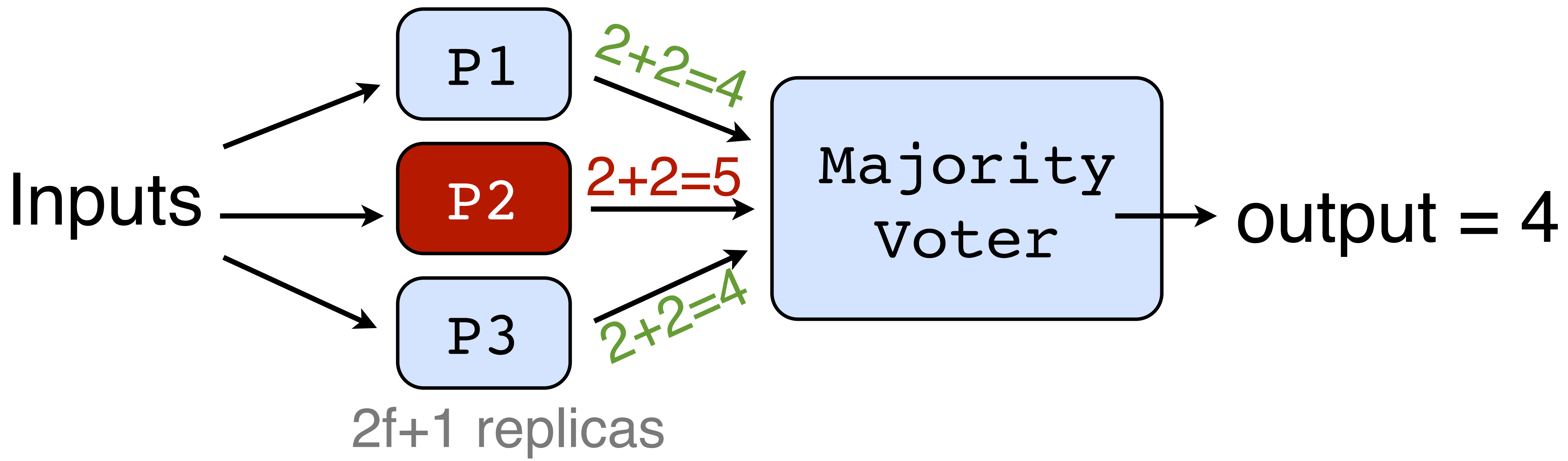
Will the same solution work?



f+1 replicas

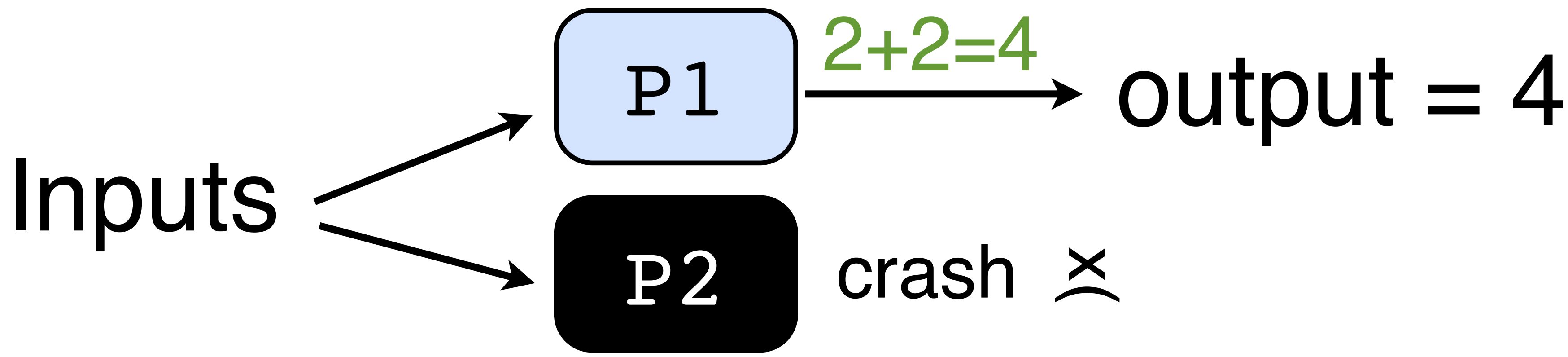
Fault Tolerance through Replication

How to tolerate a **content** failure?

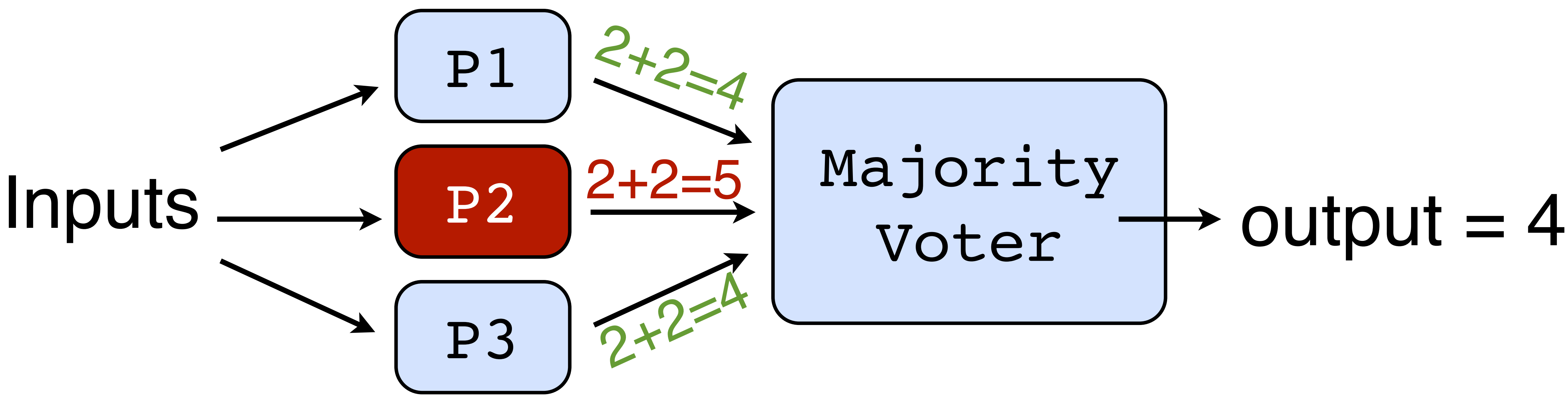


Fault Tolerance through Replication

How to tolerate a **crash** failure? **f+1 replicas**



How to tolerate a **content** failure? **2f+1 replicas**



Detection is Hard

Or maybe even impossible

How long should we set a timeout?

How do we know heart beat messages will go through?

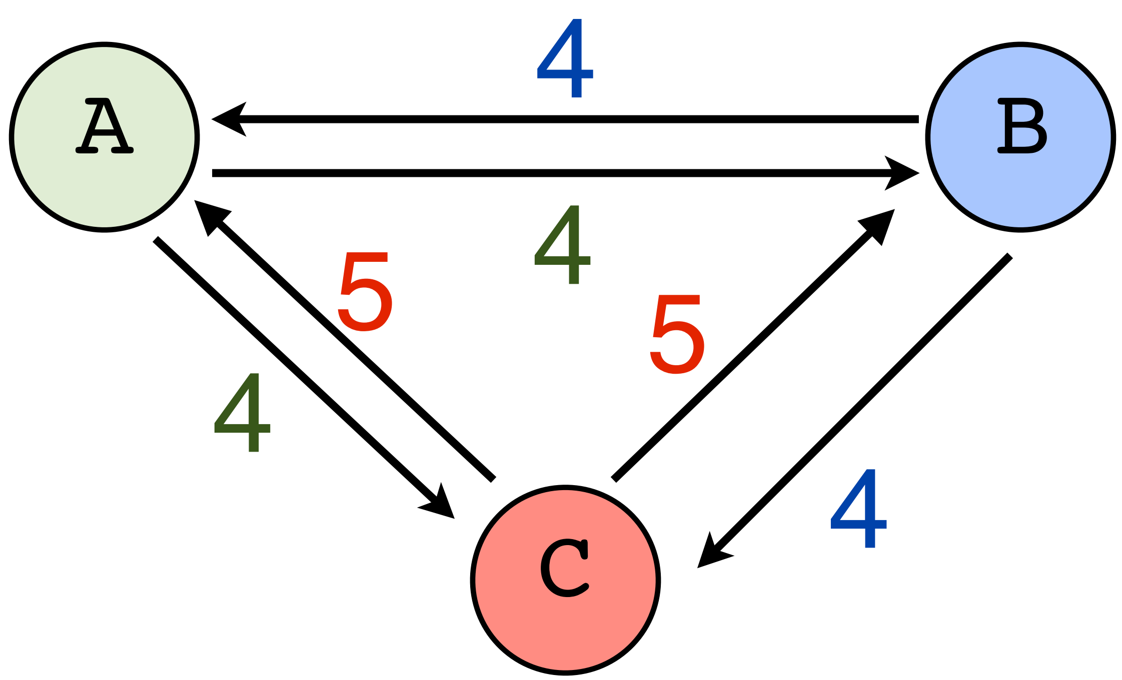
What if the voter is wrong?

Agreement without Voters

We can't always assume there is a perfectly correct voter to validate the answers

Better: Have replicas reach **agreement** amongst themselves about what to do

- Exchange calculated value and have each node pick winner



Replica	Receives	Action
A	4, 4, 5	= 4
B	4, 4, 5	= 4
C	4, 4, 5	= 4?

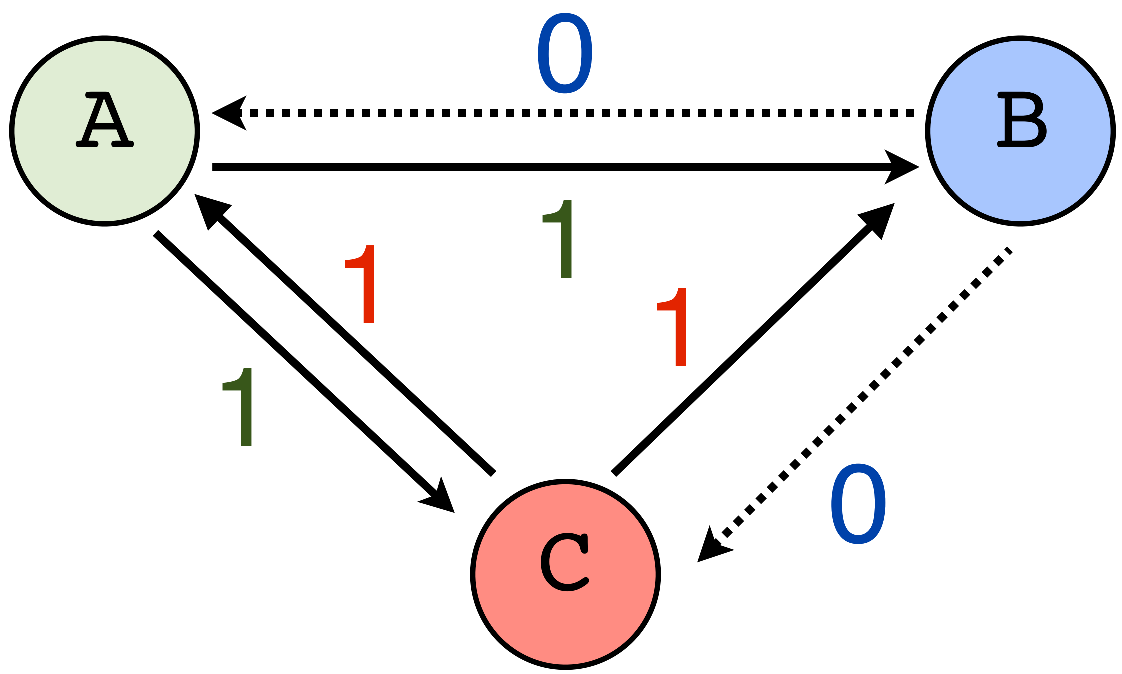
Reaching Agreement

3 Armies gather to attack a city

- But one is led by a traitor!
- We have 2f+1 replicas

The assault will only succeed if at least 2 armies attack at the same time

- I think we should… 1 = attack, 0 = retreat!



Replica	Receives	Action
A	1, 0, 1	Attack!
B	1, 0, 1	Attack!
C	1, 0, 1	???

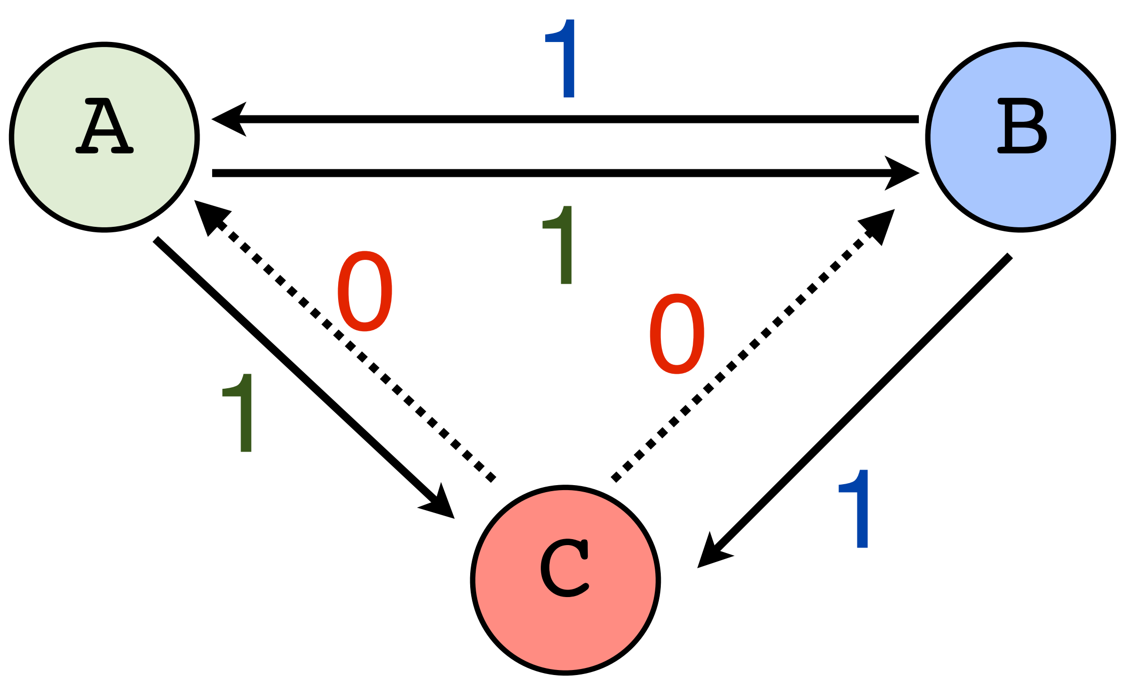
Reaching Agreement

3 Armies gather to attack a city

- But one is led by a traitor!
- We have 2f+1 replicas

The assault will only succeed if at least 2 armies attack at the same time

- I think we should… 1 = attack, 0 = retreat!



Replica	Receives	Action
A	`1, 1, 0`	`Attack!`
B	`1, 1, 0`	`Attack!`
C	`1, 1, 0`	`???`

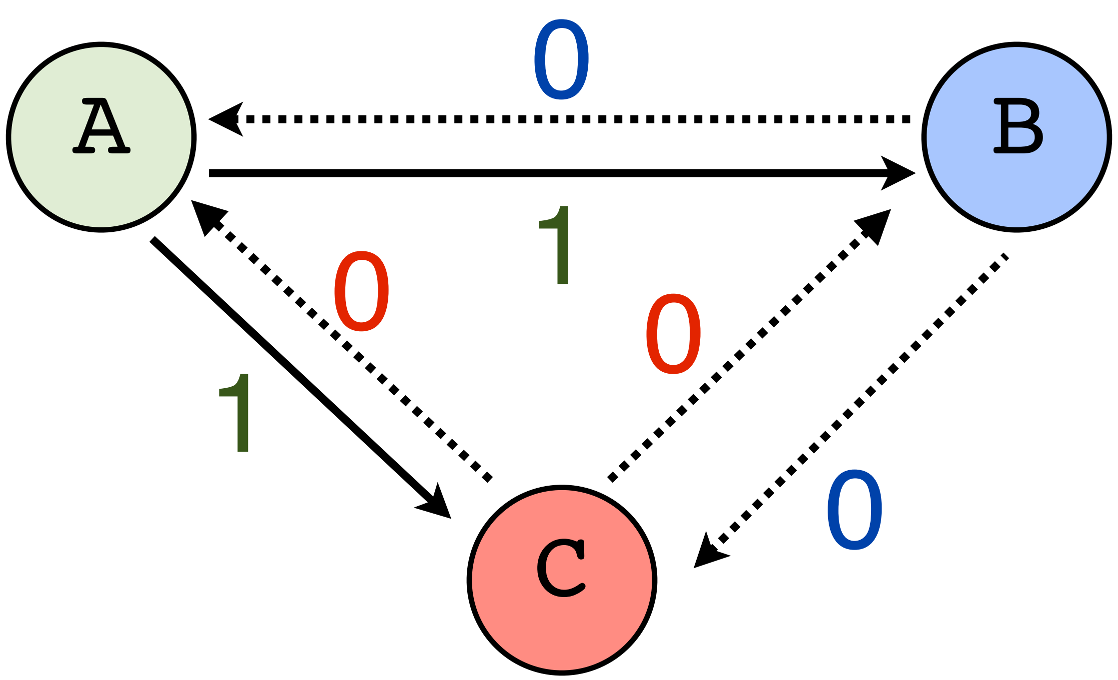
Reaching Agreement

3 Armies gather to attack a city

- But one is led by a traitor!
- We have 2f+1 replicas

The assault will only succeed if at least 2 armies attack at the same time

- I think we should… 1 = attack, 0 = retreat!



Replica	Receives	Action
A	`1, 0, 0`	`Retreat!`
B	`1, 0, 0`	`Retreat!`
C	`1, 0, 1`	`???`

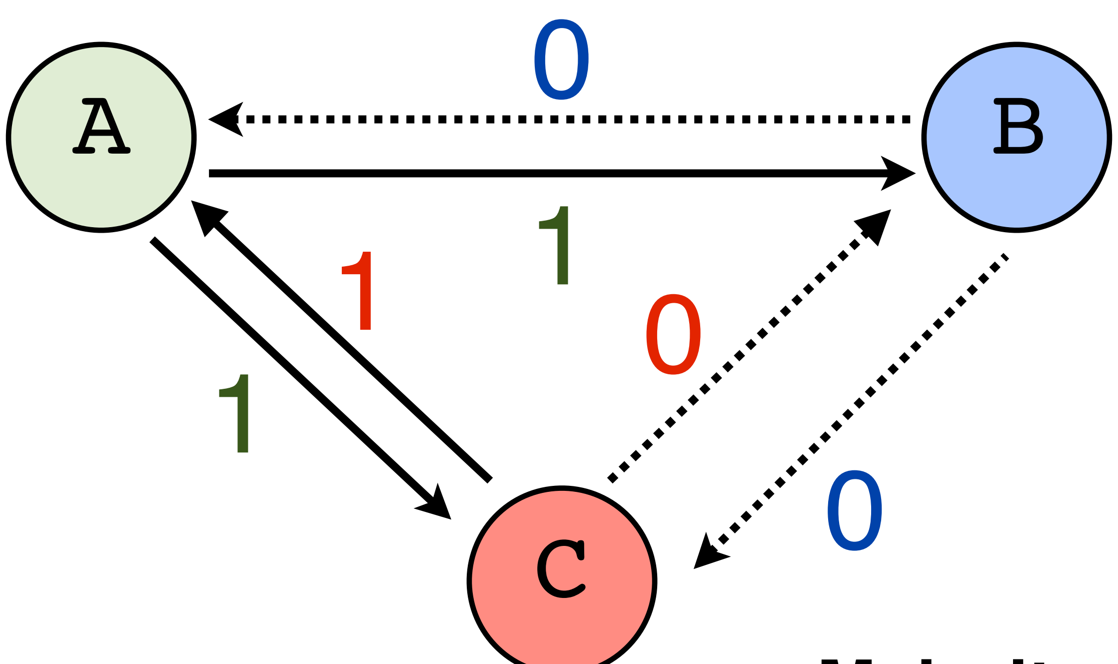
Byzantine Generals Problem

3 Armies gather to attack a city

- But one is led by a traitor!
- We have 2f+1 replicas

The assault will only succeed if at least 2 armies attack at the same time

- I think we should… 1 = attack, 0 = retreat!



Replica	Receives	Action
A	1, 0, 1	Attack!
B	1, 0, 0	Retreat!
C	1, 0, 1	???

Majority voting doesn't work if a replica lies!

Byzantine Generals Solved*!

Need more replicas to reach consensus

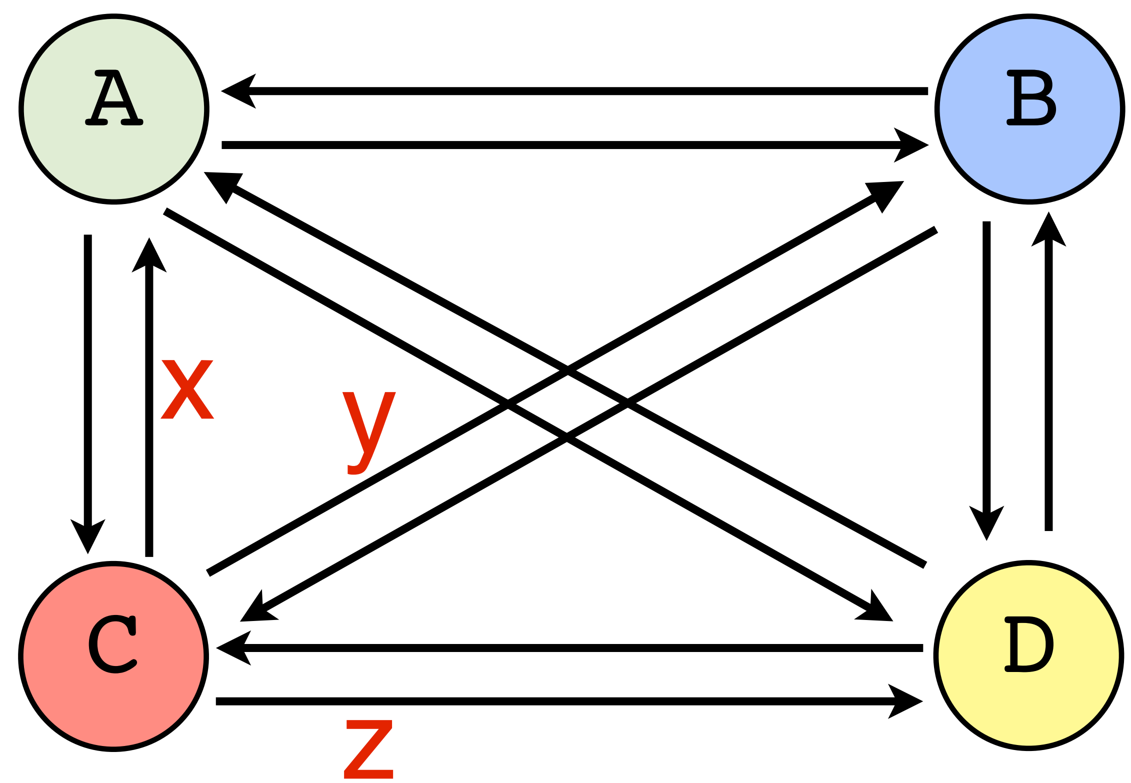
- Requires ***3f+1*** replicas to tolerate ***f*** byzantine faults

Step 1: Send your plan to everyone

Step 2: Send learned plans to everyone

Step 3: Detect conflicts and use majority

Replica	Receives	Majority
A	**A: (1,0,1,1)** **B: (1,0,0,1)** **C: (1,1,1,1)** **D: (1,0,1,1)**	**A: 1** **B: 1** **C: 1** **D: 1**
B	**A: (1,0,1,1)** **B: (1,0,0,1)** **C: (0,0,0,0)** **D: (1,0,0,1)**	**A: 1** **B: 1** **C: 0** **D: 1**



Quorum Based Systems

Quorum: a set of responses that agree with each other of a particular size

Crash fault tolerance: Need a quorum of **1**

- **f** others can fail (thus need f+1 total replicas)

Data fault tolerance: Need a quorum **f+1**

- **f** others can fail (thus need 2f+1 total replicas)
- Need a majority to determine correctness

Quorum

4 Replicas

- Some nodes might be temporarily offline

How many replicas to send to for a read or write?

- Must wait for a response from each one

Dynamo DB

Object Store from Amazon

- Technical paper at SOSP 2007 conference (top OS conference)

Stores **N** replicas of all objects

- But a replica could be out of date!
- Might saved across multiple data centers
- Gradually pushes updates to all replicas to keep in sync

When you read, how many copies, **R**, should you read from before accepting a response?

When you write, how many copies, **W**, should you write to before confirming the write?

Dynamo DB

Read and Write Quorum size:

R=1 — fastest read performance, no consistency guarantees

W = 1 — fast writes, reads may no be consistent

R = N/2+1 (reading from majority)

R=1, W = N - slow writes, but reads are consistent

R=N, W=1 - slow reads, fast writes, consistent

Standard: N=3, R=2, W=2

- if F=1,

Quorum

How do N, R, and W affect:

Performance:

Consistency:

Durability:

Availability:

DynamoDB lets the user tune these for their needs

Quorum

How do N, R, and W affect:

Performance:

- low R or W -> higher performance
- for a fixed R or W: higher N gives higher performance
- higher N means more synchronization traffic

Consistency:

- R + W > N — guarantees consistency
- R+w << N — much less likely to be consistent

Durability:

- N=1 vs N=100, more N = more durability

Availability:

- Higher N or W => higher availability