

CS 4530 & CS 5500 **Software Engineering**

Lecture 9.1: Why Engineer Distributed Software?

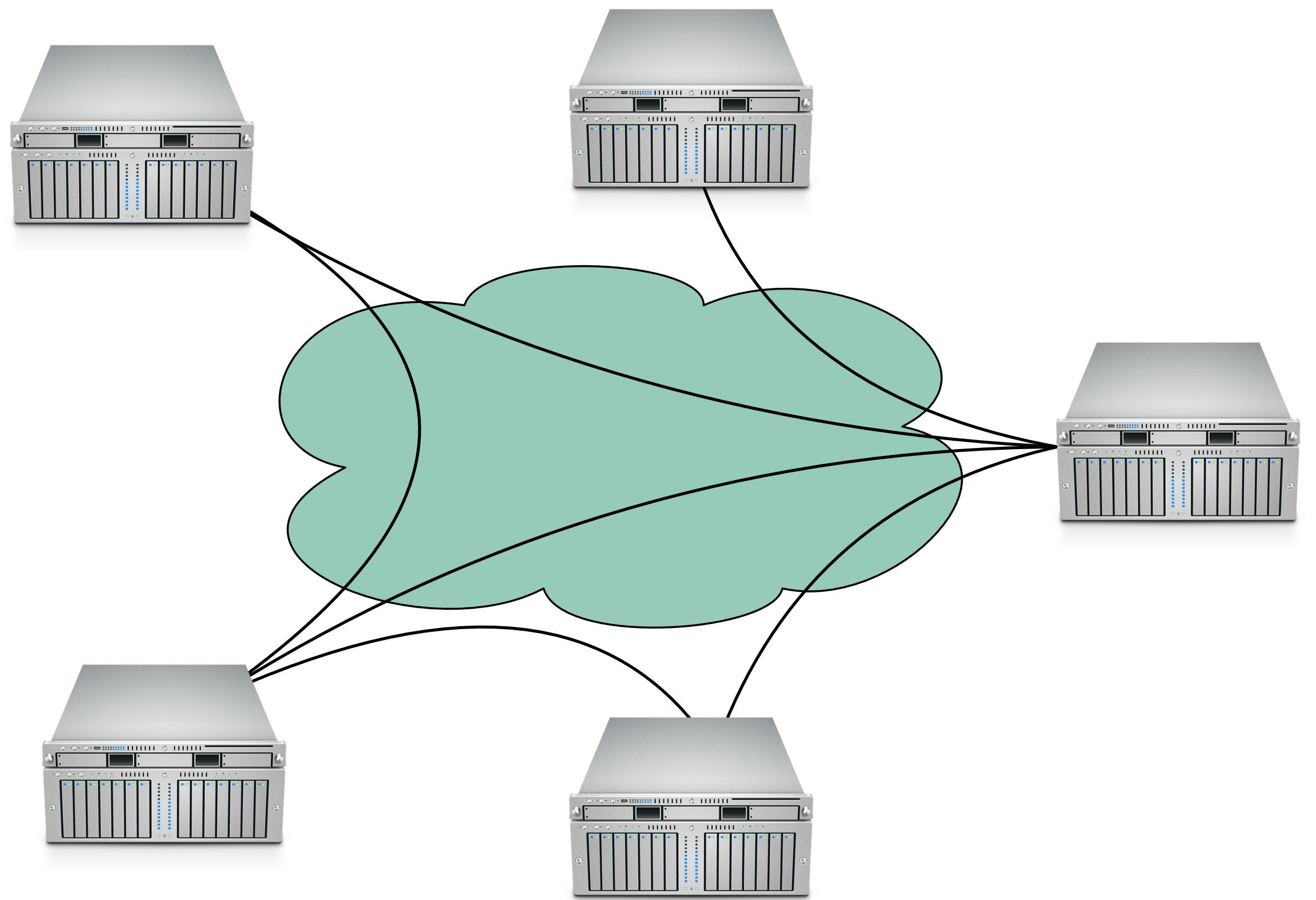
Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences
© 2021, released under [CC BY-SA](#)

Learning Objectives for this Lesson

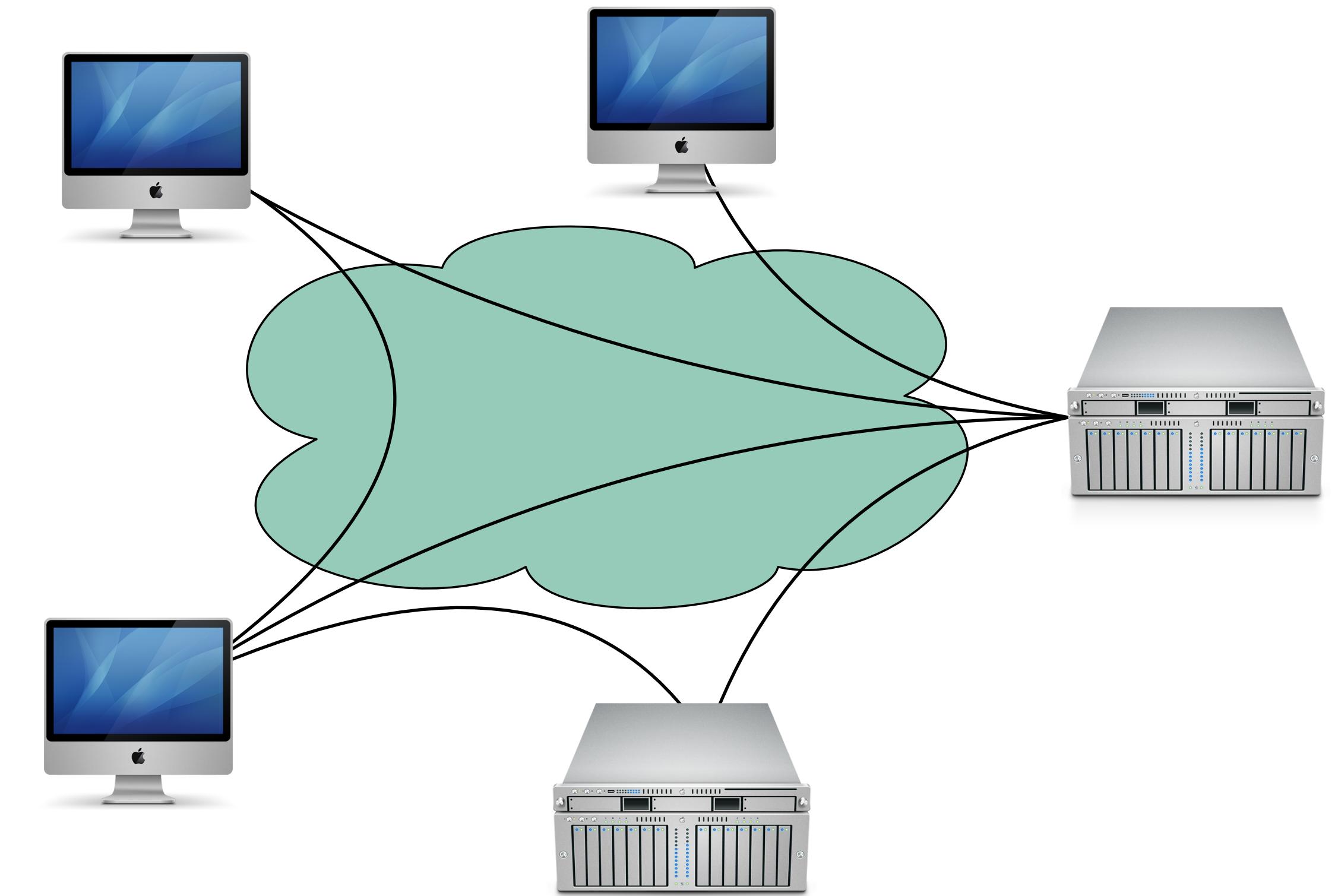
By the end of this lesson, you should be able to...

- Describe 5 key goals of distributed systems
- Describe 8 key common fallacies of distributed systems
- Analyze a system's requirements and determine if it should be implemented as a distributed system or not

What is a distributed system?



Model:
Many servers talking through a network



Model:
Many servers and clients talking through a network

Why expand to distributed systems?

- Scalability
- Performance
- Latency
- Availability
- Fault Tolerance

“Distributed Systems for Fun and Profit”, Takada

Distributed Systems Goals

- **Scalability**
- Performance
- Latency
- Availability
- Fault Tolerance

“the ability of a system, network, or process, to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth.”

Distributed Systems Goals

- Scalability
- **Performance**
- Latency
- Availability
- Fault Tolerance

“is characterized by the amount of useful work accomplished by a computer system compared to the time and resources used.”

Distributed Systems Goals

- Scalability
- Performance
- **Latency**
- Availability
- Fault Tolerance

“The state of being latent; delay, a period between the initiation of something and the it becoming visible.”

Distributed Systems Goals

- Scalability
- Performance
- Latency
- **Availability**
- Fault Tolerance

“the proportion of time a system is in a functioning condition. If a user cannot access the system, it is said to be unavailable.”

Availability = uptime / (uptime + downtime).

Often measured in “nines”

Availability %	Downtime/year
90%	>1 month
99%	< 4 days
99.9%	< 9 hours
99.99%	<1 hour
99.999%	5 minutes
99.9999%	31 seconds

Distributed Systems Goals

- Scalability
- Performance
- Latency
- Availability
- **Fault Tolerance**

Disks fail

Power supplies fail

“ability of a system to behave in a well-defined manner once faults occur”

What kind of faults?

Networking fails

Security breached

Power goes out Datacenter goes offline

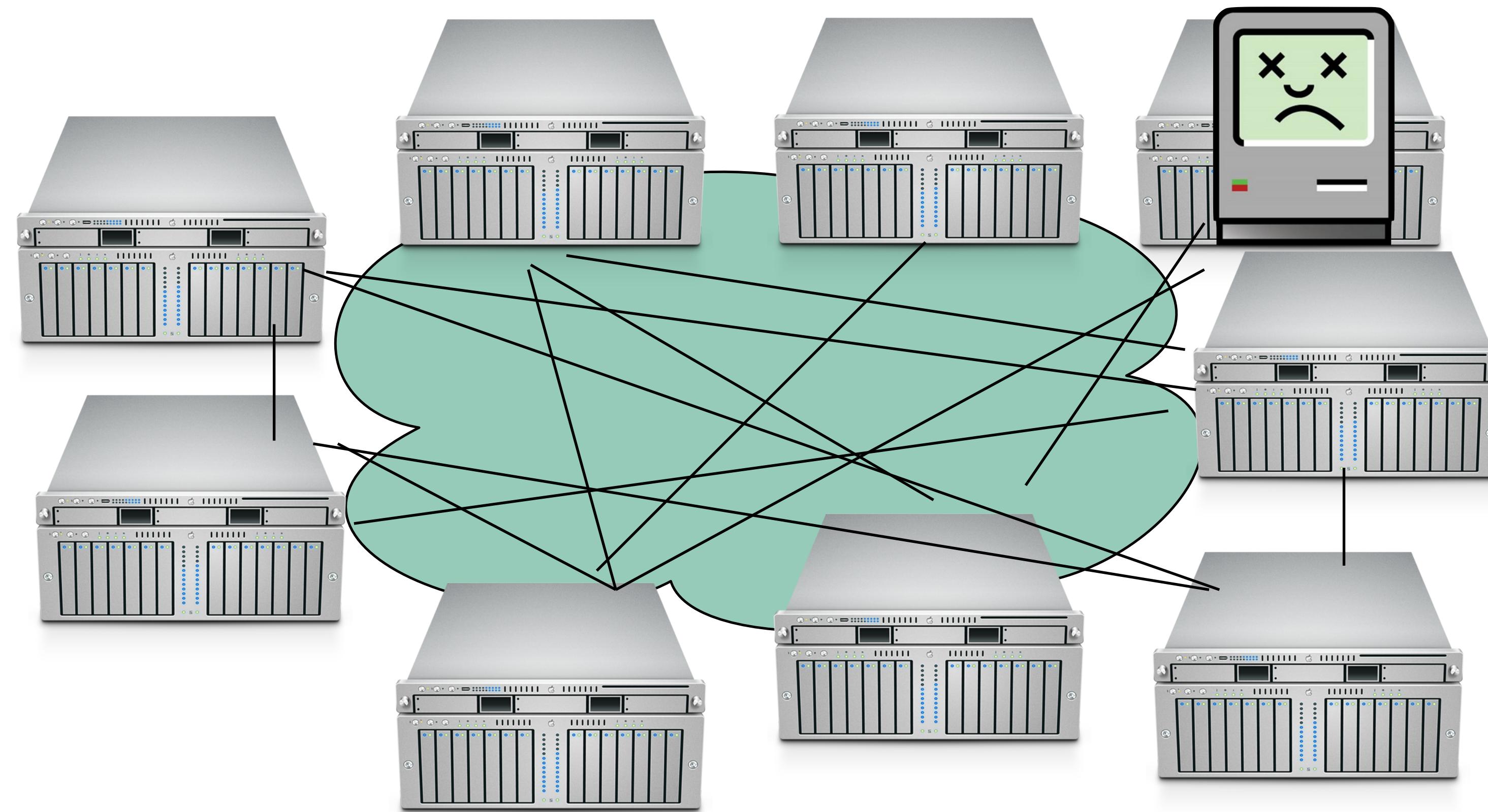
Distributed Systems Challenges

More machines, more problems

- Say there's a 1% chance of having some hardware failure occur to a machine (power supply burns out, hard disk crashes, etc)
- Now I have 10 machines
 - Probability(at least one fails) = $1 - \text{Probability}(\text{no machine fails}) = 1 - (1 - .01)^{10}$
= 10%
- 100 machines -> 63%
- 200 machines -> 87%

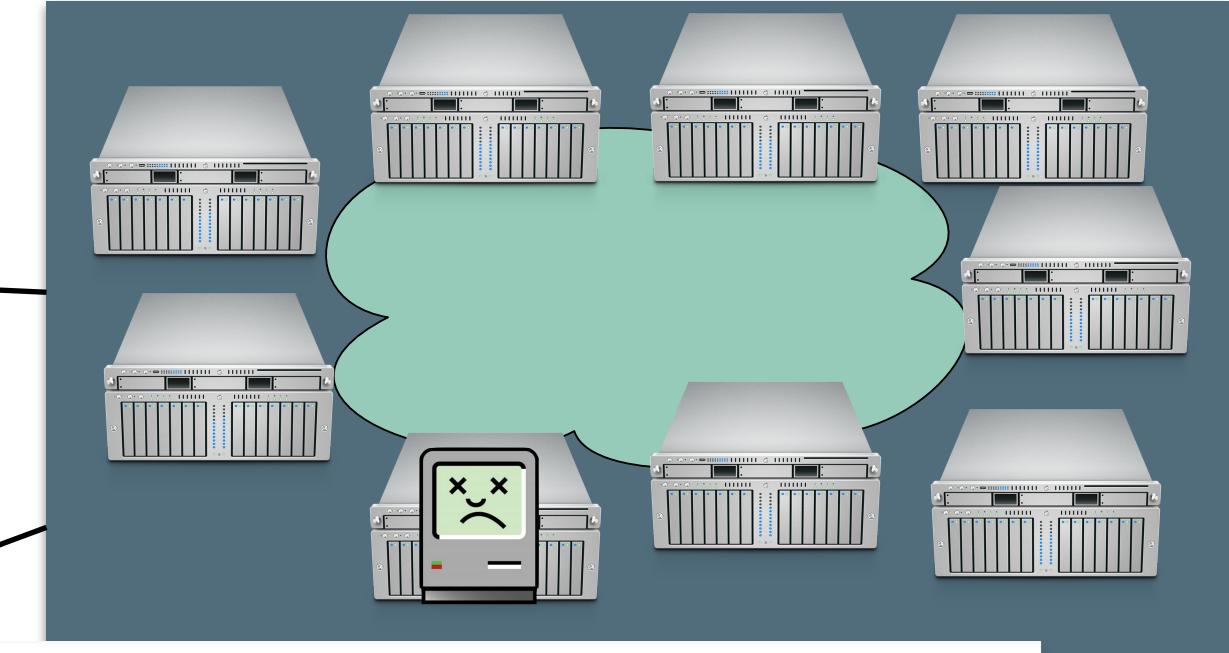
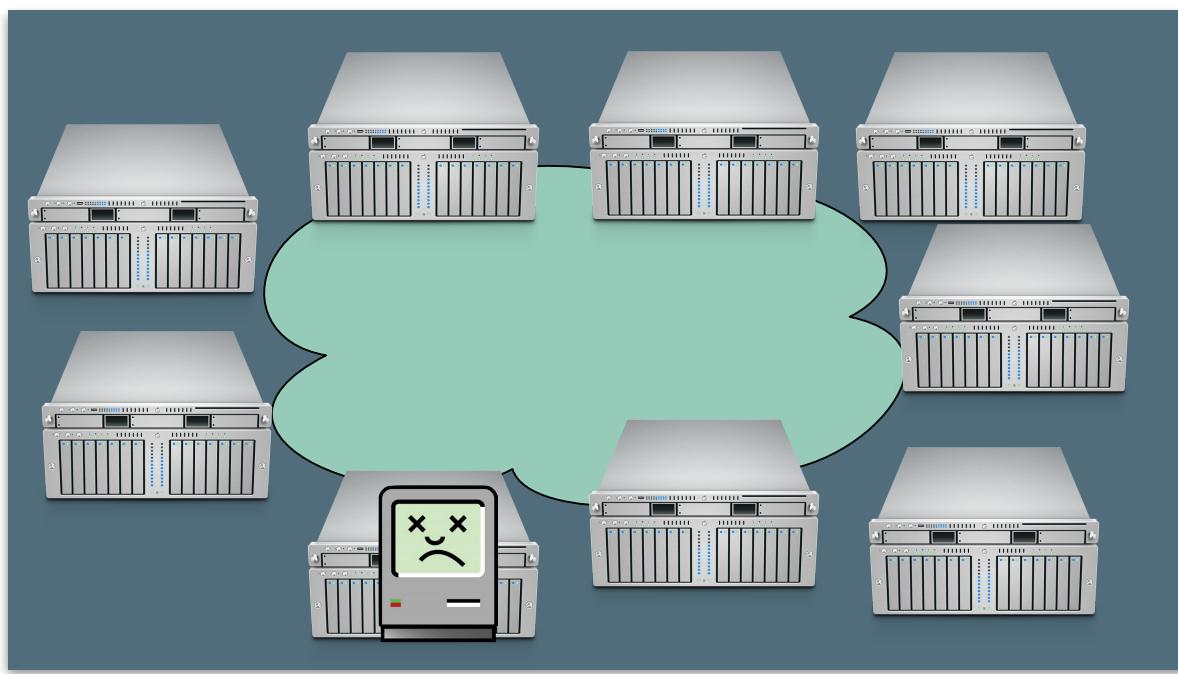
Distributed Systems Challenges

Number of nodes + distance between them



Distributed Systems Challenges

Number of nodes + distance between them



Even if cross-city links are fast and cheap (are they?)
Still that pesky speed of light...



DC

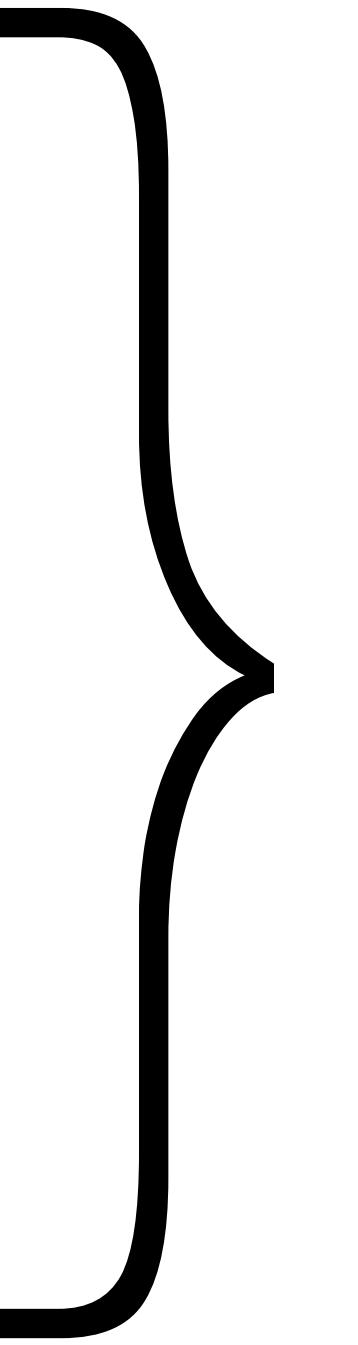


LONDON

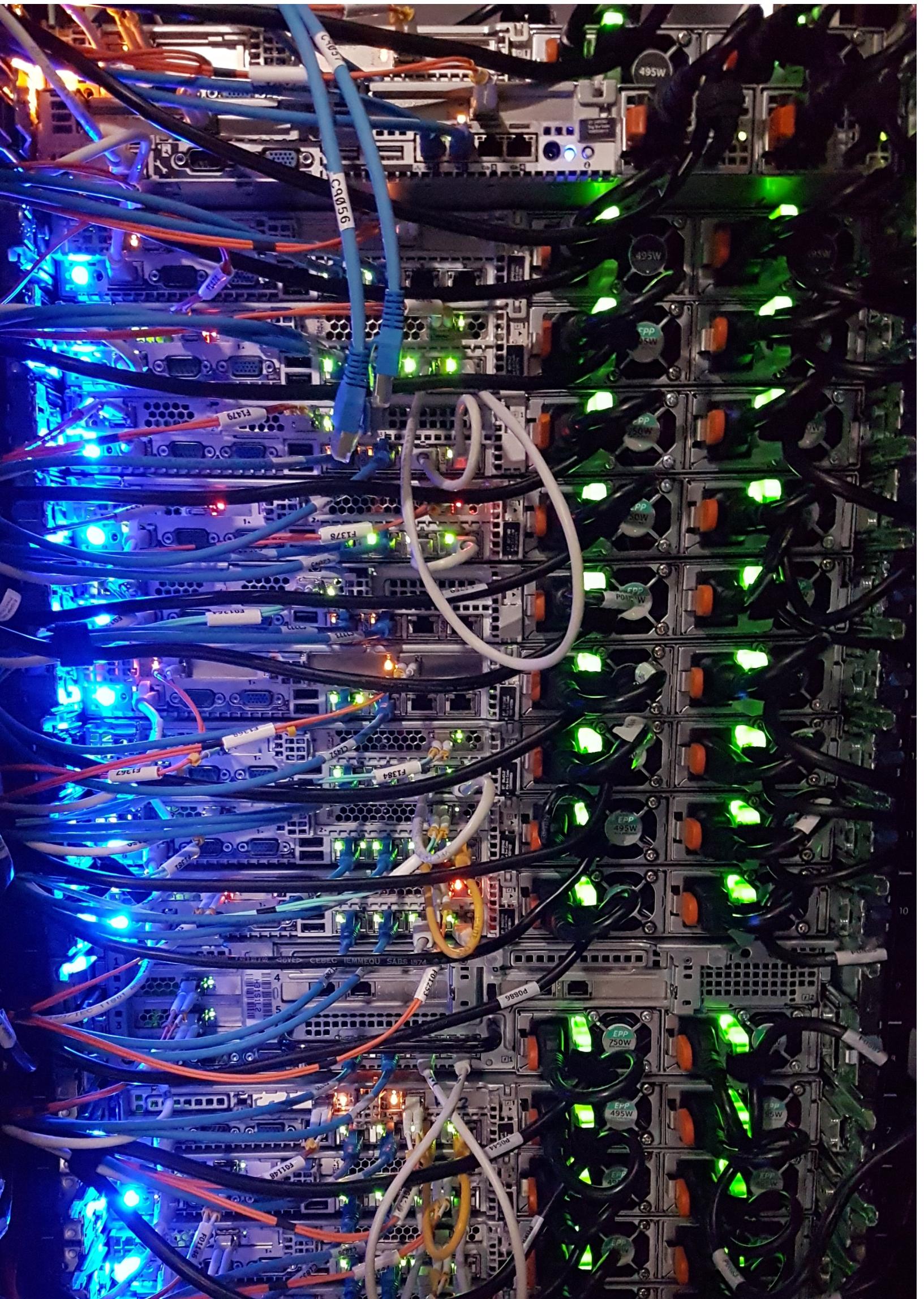
8 Fallacies of Distributed Computing

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



Do these fallacies still hold?

Networks still fail, intermittently and for prolonged periods

The screenshot shows a web browser window for arstechnica.com. The page title is "The discovery of Apache ZooKeeper's poison packet". Below the title, it says "How PagerDuty found four different bugs." and credits "EVAN GILMAN - 5/13/2015, 9:00 AM". A bio for Evan Gilman follows, and then a paragraph about ZooKeeper. The text continues with a section titled "Background: The use of ZooKeeper at PagerDuty" and ends with "Part I: The ZooKeeper bugs".

The screenshot shows a web browser window for wired.com. The page title is "Friday's Massive Comcast Outage Shows How Fragile the Internet Is". It includes a subtitle: "Comcast customers across the country experienced outages Friday, thanks to multiple cuts to fiber optic cables." Below the text is a large image of several fiber optic cables fanning out against a green background. At the bottom of the image, the caption reads "Blame cuts to fiber optic cables for Comcast's outage Friday. GETTY IMAGES".

Do these fallacies still hold?

We still rely on other administrators, who are not infallible

**Amazon Web Services
outage takes a portion of
the internet down with it**

Zack Whittaker

@zackwhittaker / 12:32 PM EST • November 25, 2020



 **Image Credits:** David Becker / Getty Images

Amazon Web Services is currently having an outage, taking a chunk of the internet down with it.

Several AWS services were experiencing problems as of early Wednesday, according to [its status page](#). That means any app, site or service that relies on AWS might also be down, too. (As I found out the hard way this morning when



Comment

The screenshot shows a web browser window for aws.amazon.com. The header includes the AWS logo, navigation links like Products, Solutions, Pricing, Documentation, Learn, Partner Network, AWS Marketplace, Customer Enablement, Events, Explore More, and a Sign In to the Console button. The main content is titled "Summary of the Amazon Kinesis Event in the Northern Virginia (US-EAST-1) Region" and dated "November, 25th 2020". It discusses a service disruption in the Northern Virginia region on November 25th, 2020. The text explains that Amazon Kinesis enables real-time processing of streaming data and is used by several other AWS services. It details the impact of a capacity addition to the front-end fleet, mentioning shard-map, microservice, and DynamoDB interactions. It also describes the diagnosis work, which involved removing new capacity and investigating errors, leading to a full restart of the front-end fleet at 7:51 AM PST. The text concludes with a confirmation of a root cause at 9:39 AM PST, noting memory pressure as the driver.

Summary of the Amazon Kinesis Event in the Northern Virginia (US-EAST-1) Region

November, 25th 2020

We wanted to provide you with some additional information about the service disruption that occurred in the Northern Virginia (US-EAST-1) Region on November 25th, 2020.

Amazon Kinesis enables real-time processing of streaming data. In addition to its direct use by customers, Kinesis is used by several other AWS services. These services also saw impact during the event. The trigger, though not root cause, for the event was a relatively small addition of capacity that began to be added to the service at 2:44 AM PST, finishing at 3:47 AM PST. Kinesis has a large number of “back-end” cell-clusters that process streams. These are the workhorses in Kinesis, providing distribution, access, and scalability for stream processing. Streams are spread across the back-end through a sharding mechanism owned by a “front-end” fleet of servers. A back-end cluster owns many shards and provides a consistent scaling unit and fault-isolation. The front-end’s job is small but important. It handles authentication, throttling, and request-routing to the correct stream-shards on the back-end clusters.

The capacity addition was being made to the front-end fleet. Each server in the front-end fleet maintains a cache of information, including membership details and shard ownership for the back-end clusters, called a shard-map. This information is obtained through calls to a microservice vending the membership information, retrieval of configuration information from DynamoDB, and continuous processing of messages from other Kinesis front-end servers. For the latter communication, each front-end server creates operating system threads for each of the other servers in the front-end fleet. Upon any addition of capacity, the servers that are already operating members of the fleet will learn of new servers joining and establish the appropriate threads. It takes up to an hour for any existing front-end fleet member to learn of new participants.

At 5:15 AM PST, the first alarms began firing for errors on putting and getting Kinesis records. Teams engaged and began reviewing logs. While the new capacity was a suspect, there were a number of errors that were unrelated to the new capacity and would likely persist even if the capacity were to be removed. Still, as a precaution, we began removing the new capacity while researching the other errors. The diagnosis work was slowed by the variety of errors observed. We were seeing errors in all aspects of the various calls being made by existing and new members of the front-end fleet, exacerbating our ability to separate side-effects from the root cause. At 7:51 AM PST, we had narrowed the root cause to a couple of candidates and determined that any of the most likely sources of the problem would require a full restart of the front-end fleet, which the Kinesis team knew would be a long and careful process. The resources within a front-end server that are used to populate the shard-map compete with the resources that are used to process incoming requests. So, bringing front-end servers back online too quickly would create contention between these two needs and result in very few resources being available to handle incoming requests, leading to increased errors and request latencies. As a result, these slow front-end servers could be deemed unhealthy and removed from the fleet, which in turn, would set back the recovery process. All of the candidate solutions involved changing every front-end server’s configuration and restarting it. While the leading candidate (an issue that seemed to be creating memory pressure) looked promising, if we were wrong, we would double the recovery time as we would need to apply a second fix and restart again. To speed restart, in parallel with our investigation, we began adding a configuration to the front-end servers to obtain data directly from the authoritative metadata store rather than from front-end server neighbors during the bootstrap process.

At 9:39 AM PST, we were able to confirm a root cause, and it turned out this wasn’t driven by memory pressure. Rather, the new capacity had caused all of the servers in the fleet to exceed the maximum number of threads allowed by an operating system configuration. As this limit was being exceeded,

Should we make {our software} distributed?

Reflecting on goals + challenges

- Do we need to store more data than one computer can store?
- Do we need to process requests faster than one computer can?
- Are we willing and able to take on these additional complications?
- Next lesson: what tools do we have at our disposal?

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

- This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>
- You are free to:
 - Share — copy and redistribute the material in any medium or format
 - Adapt — remix, transform, and build upon the material
 - for any purpose, even commercially.
- Under the following terms:
 - Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
 - ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
 - No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.