CS 4530 Software Engineering

Lecture 9.1: Why Engineer Distributed Software?

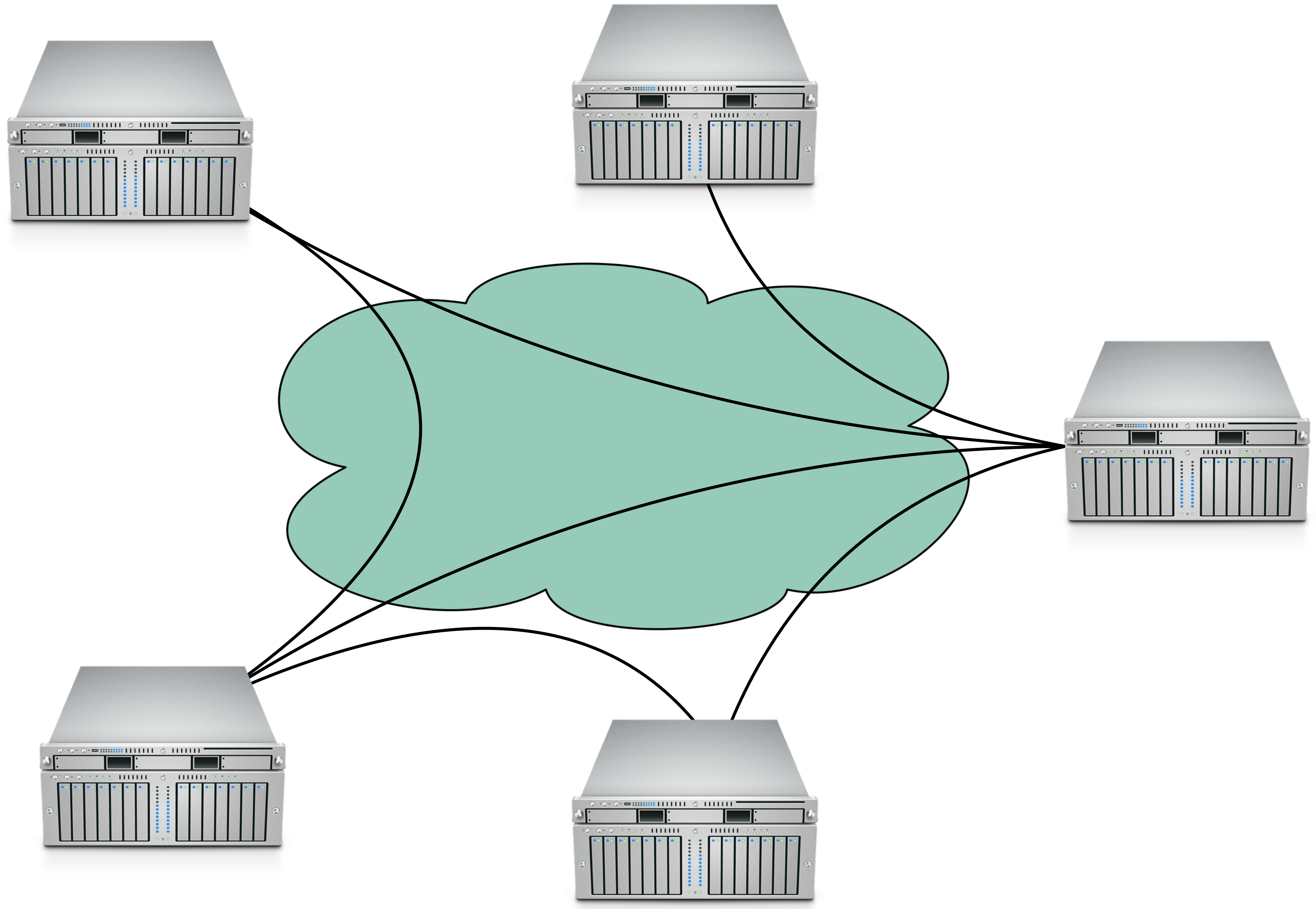
Jonathan Bell, Adeel Bhutta, Ferdinand Vesely, Mitch Wand
Khoury College of Computer Sciences

Learning Objectives for this Lesson

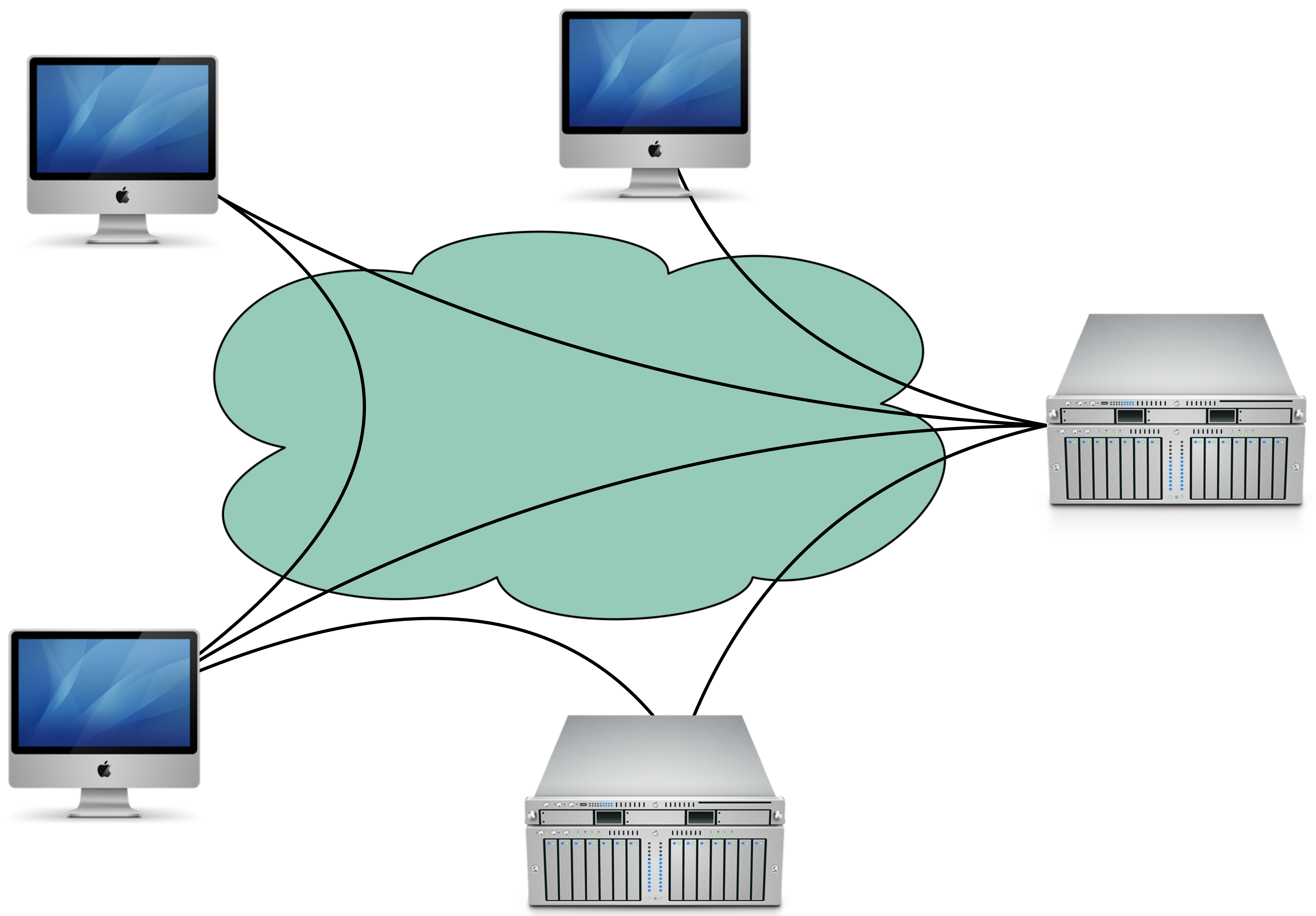
By the end of this lesson, you should be able to…

- Decide why would you want to build your system as a distributed system
- Describe 5 key goals 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

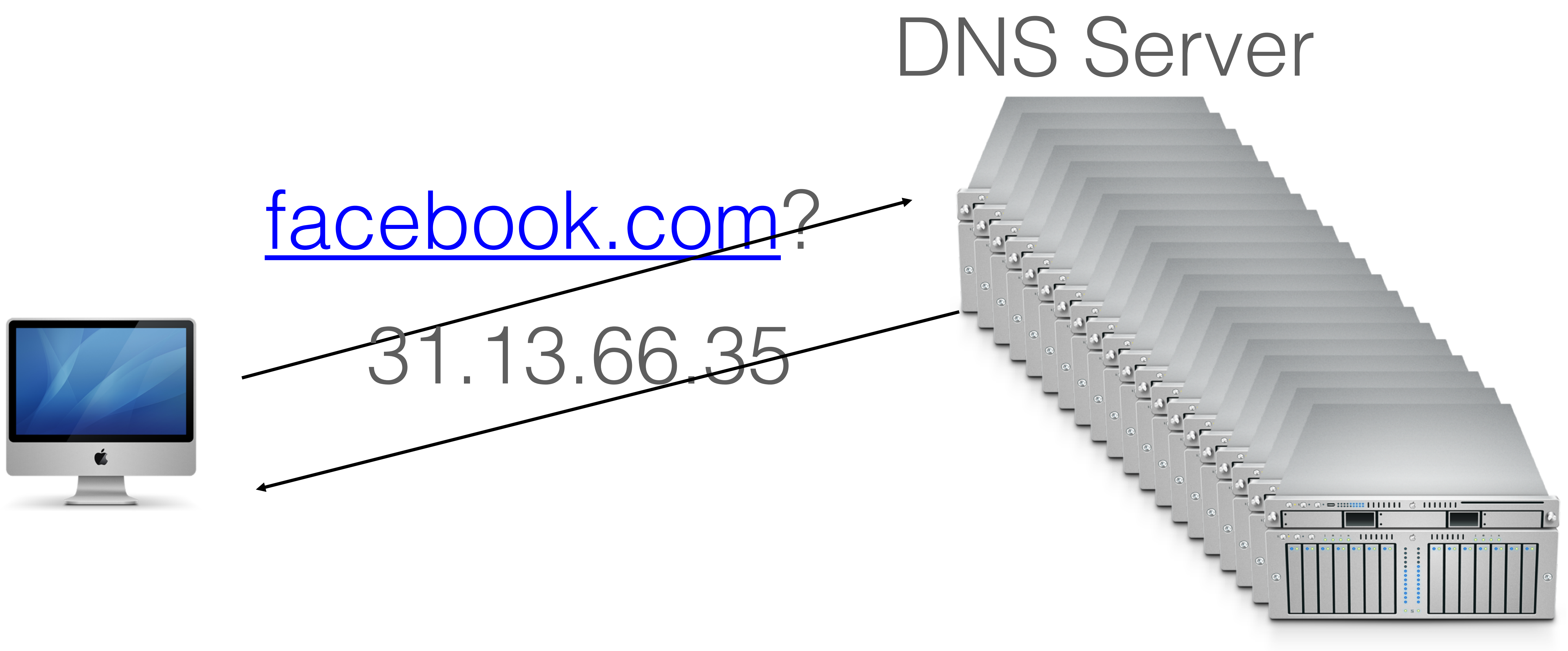
Example: Domain Name System (DNS)

Problem Statement

- Nodes (hosts) on a network are identified by IP addresses
- E.g.: 142.251.41.4
- We humans prefer something easier to remember: calendar.google.com, facebook.com, www.khoury.northeastern.edu
- We need to keep a directory of domain names and their addresses
- We also need to make sure everybody gets directed to the correct host

Example: Domain Name System (DNS)

- Need to handle millions of DNS queries per second
- Not immediately obvious how to scale: how do we maintain replication, some measure of consistency?



Domain Name System

- Obvious solution: Use a Local file
  - Keep local copy of mapping from all hosts to all IPs (e.g., /etc/hosts)
  - Hosts change IPs regularly: Download file frequently
  - Lot of constant internet bandwidth use
  - IPv4 space is now full
    - 32-bits: 4,294,967,296 addresses
    - At 1 byte per address, file would be 4GB
    - Not a lot of disk space (now, DNS introduced in the late 80s)

Domain Name System

- Obvious solution: Use a Local file
  - Keep local copy of mapping from all hosts to all IPs (e.g., /etc/hosts)
  - Hosts change IPs regularly: Download file frequently
  - Lot of constant internet bandwidth use
  - IPv4 space is now full
    - 32-bits: 4,294,967,296 addresses
    - At 1 byte per
    - Not a lot of c ed

We need 200x of these to hold 4GB: $270K+

your Tandy 1000s/3000/4000 or PC Compatible on a user-installable card. Includes manual and software for easy installation. The easy way to get hard disk power for your computer system. 25-4059 .... 799.00

Add an External Hard Disk Drive

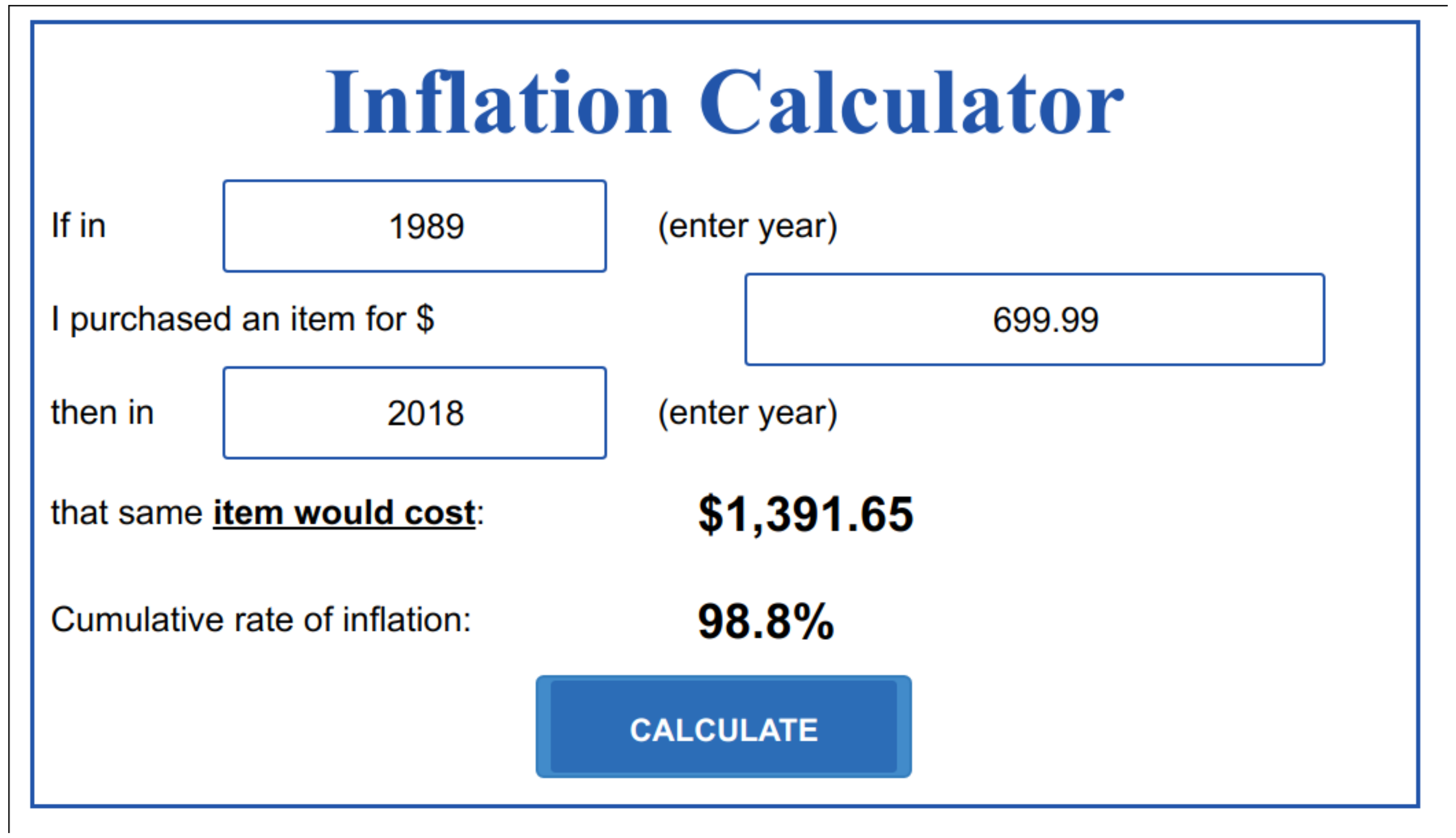
699⁰⁰ Low As $40 Per Month*

- Alternative Expansion Option
- Allows Greater Data Storage

20-Megabyte External Hard Disk Drive. (Cable Kit and installation required for secondary unit.) Requires Hard Disk Controller Board (25-1007). 25-1041 .... 699.00

Hard Disk Controller Board. For Tandy 1000 SX/SL and original Tandy 1000 only. Allows you to add hard disk drives for up to 40 million characters of storage. Includes cable for use with 10 or 20-megabyte hard disks. 25-1007 .... 299.95

180 PRICES APPLY AT PARTICIPATING RA

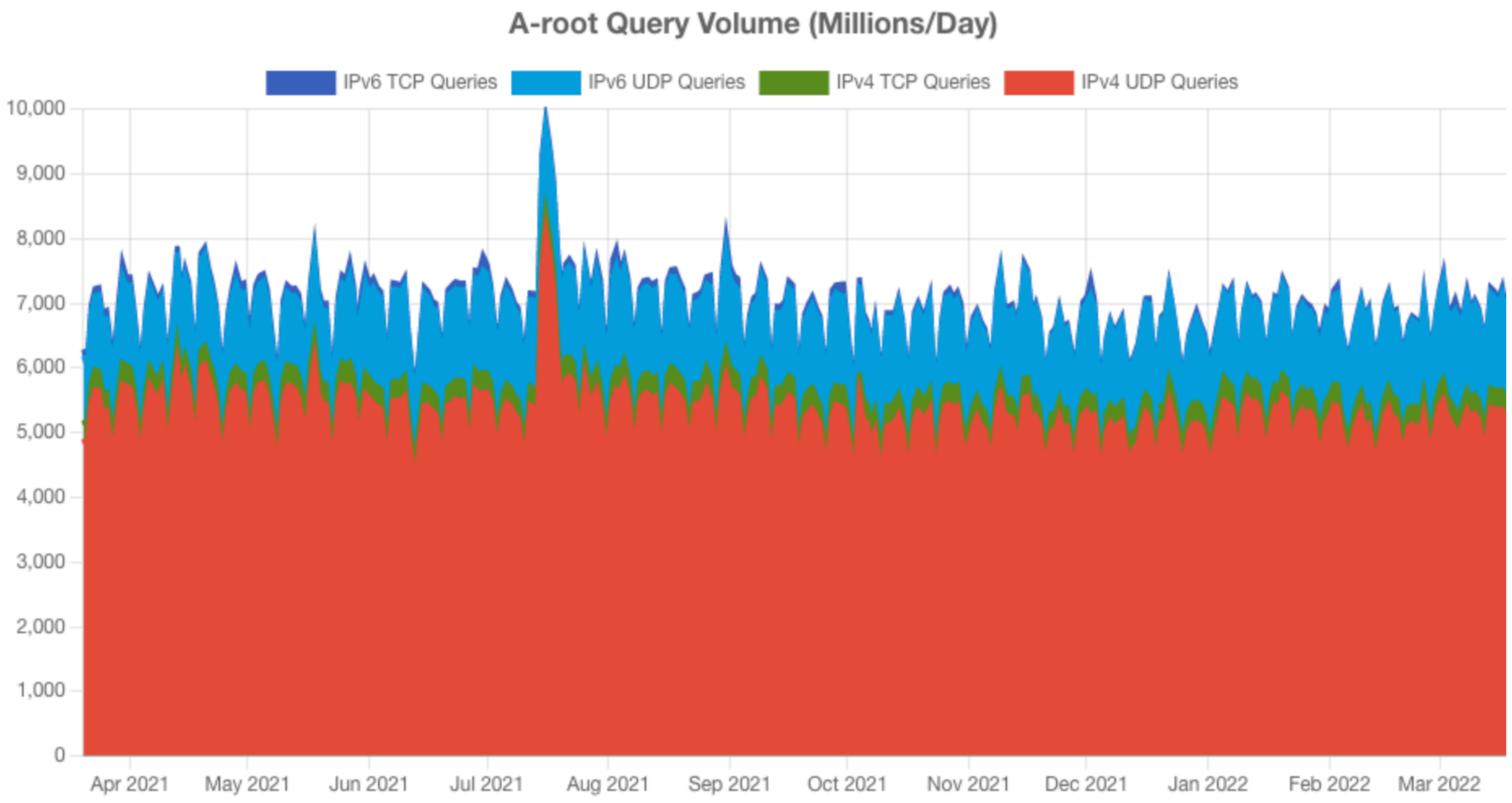


Domain Name System

- Obvious solution: Use a Local file
  - Keep local copy of mapping from all hosts to all IPs (e.g., /etc/hosts)
  - Hosts change IPs regularly: Download file frequently
  - IPv4 space is now full
    - 32-bits: 4,294,967,296 addresses
    - At 1 byte per address, file would be 4GB
    - Not a lot of disk space (now, DNS introduced in the late 80s)
    - But a lot of constant internet bandwidth
  - More names than IPs
    - Aliases
  - **Not scalable!**

Domain Name System

- Another Obvious Solution: Well-known centralized server
  - Single point of failure
  - Traffic volume
  - Access time
  - Ultimately, **not scalable**!



https://a.root-servers.org/metrics

DNS as a distributed system

- We need a **scalable** solution
  - New hosts keep being added
  - Number of users increases
  - Need to maintain speed/responsiveness
- We need our service to be **available** and **fault tolerant**
  - It is a crucial basic service
  - A problematic node shouldn’t “crash the internet”
- Parts of the system should be maintainable independently
  - E.g., national domains
  - Maintaining it shouldn’t add significant amount of traffic
- Global in scope
  - Domain names mean the same thing everywhere

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 for Fun and Profit”, Takada

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 for Fun and Profit”, Takada]

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 for Fun and Profit”, Takada]

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

"Dis ada

Distributed Systems Goals

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

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

What kind of faults?

Disks fail

Networking fails

Power supplies fail

Security breached

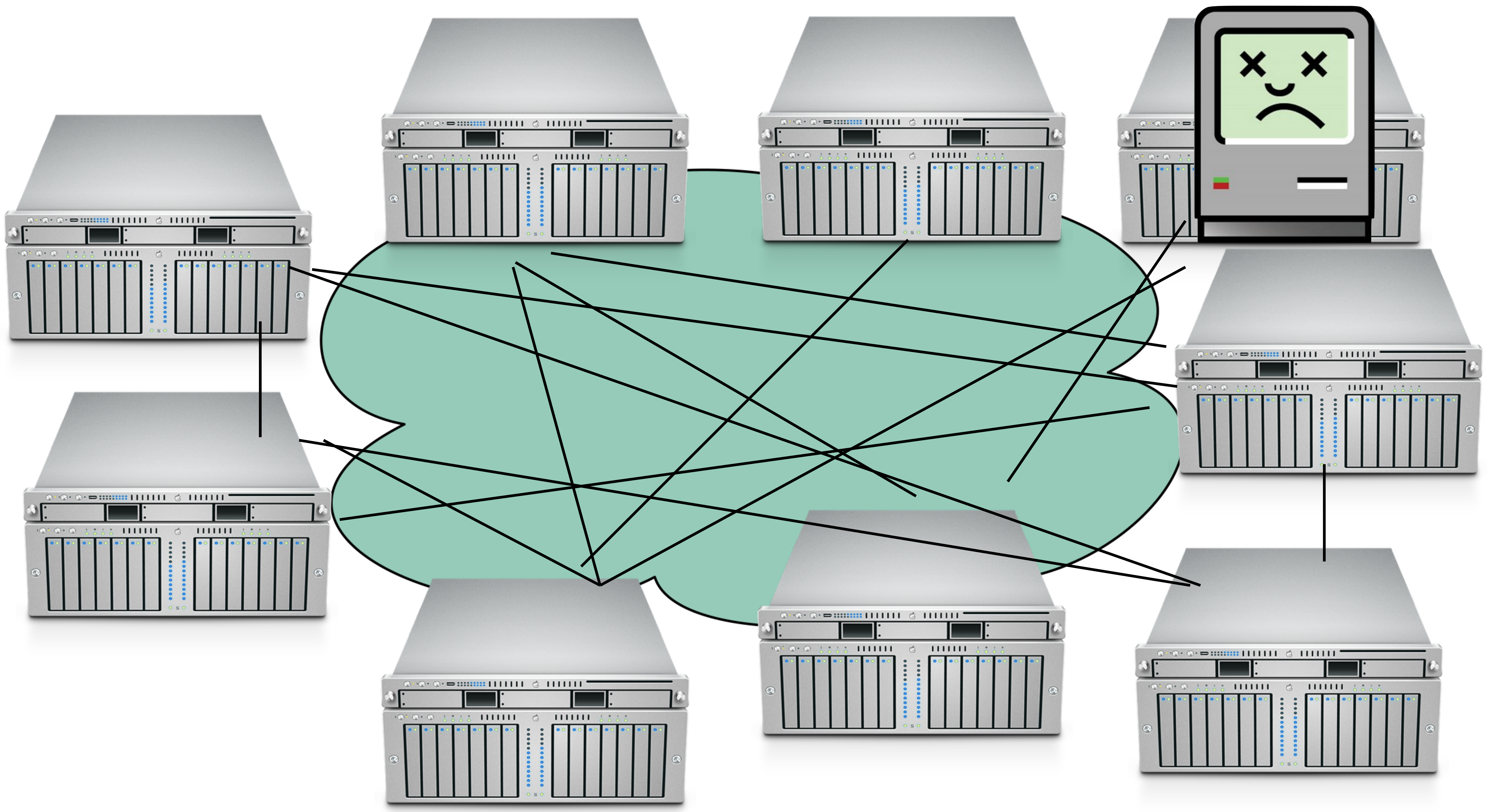
Datacenter goes offline

Power goes out

[“Distributed Systems for Fun and Profit”, Takada]()

Challenges

More Machines, more problems



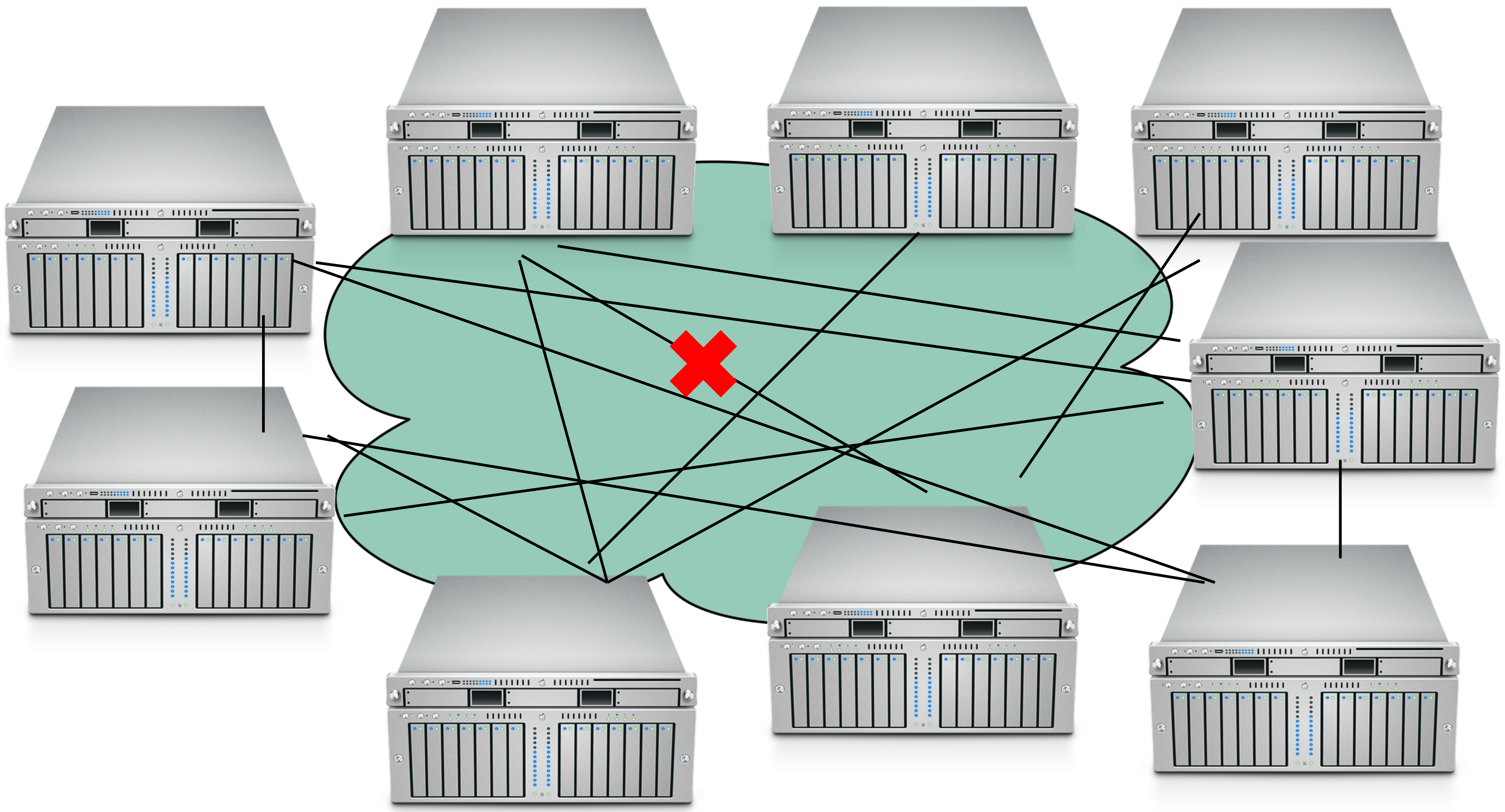
Challenges

More machines, more problems

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

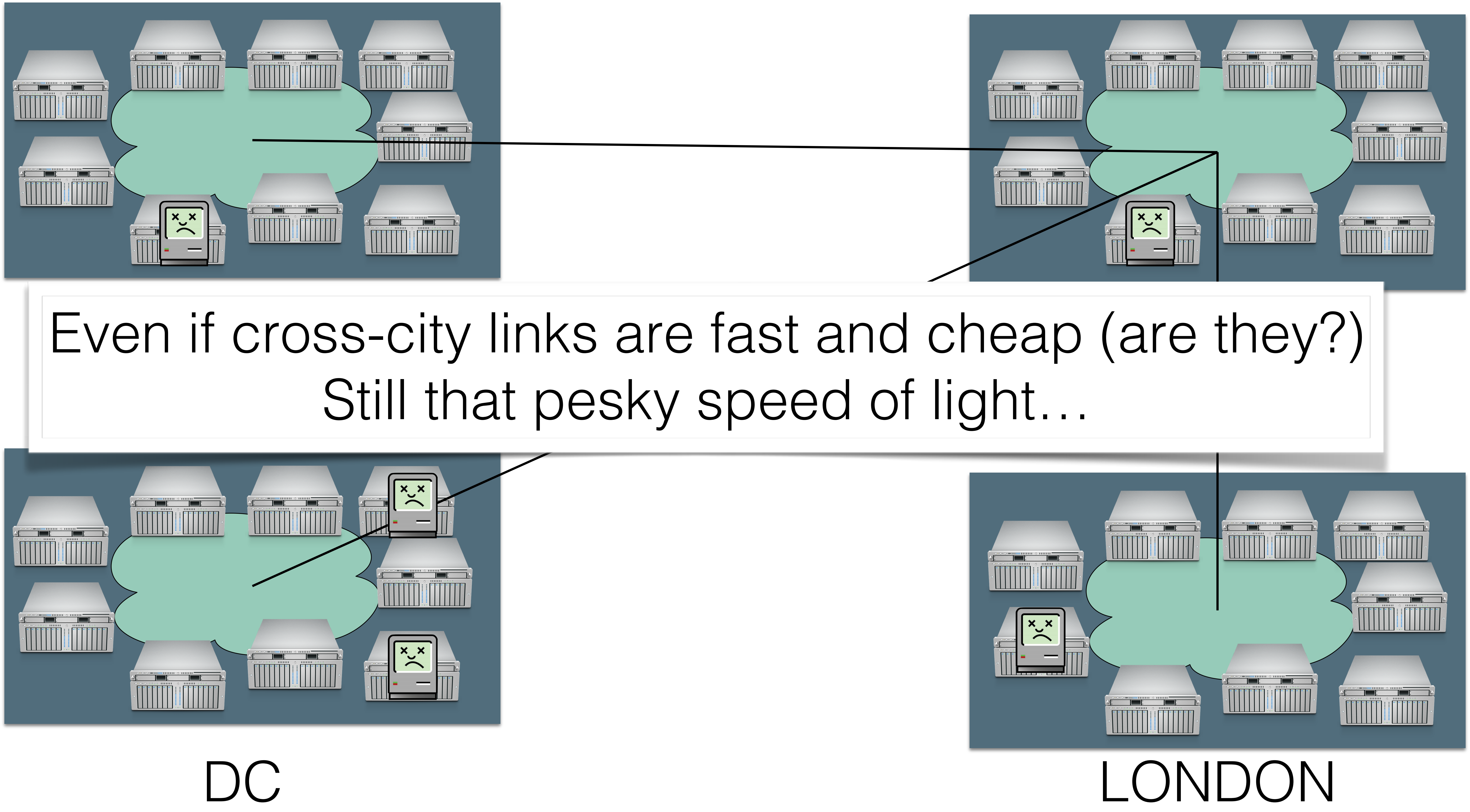
Challenges

Number of nodes + distance between them



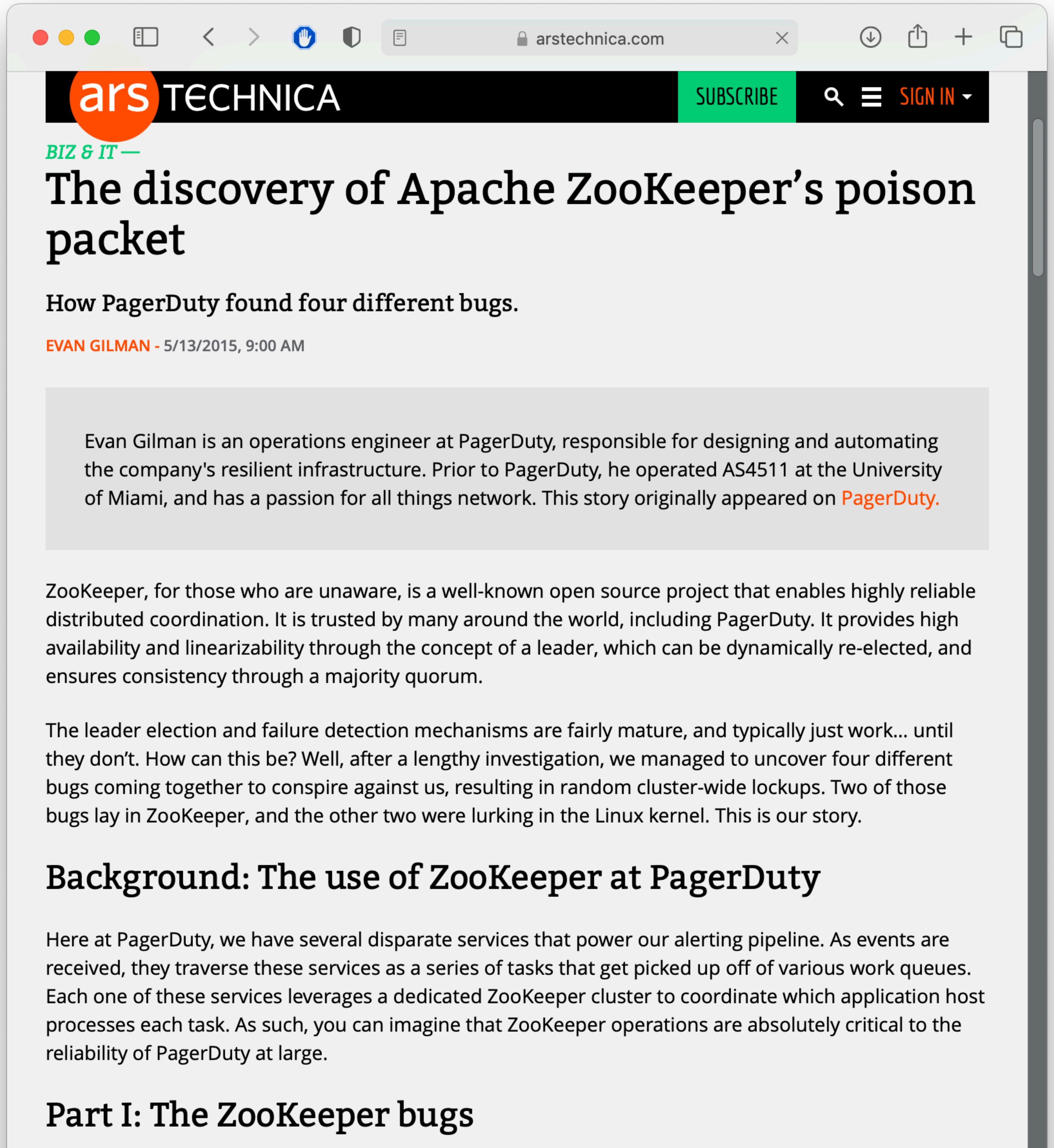
Challenges

Number of nodes + distance between them



More challenges:

Networks still fail, intermittently and for prolonged periods



arstechnica.com

ars TECHNICA SUBSCRIBE SIGN IN

BIZ & IT —

The discovery of Apache ZooKeeper's poison packet

How PagerDuty found four different bugs.

EVAN GILMAN - 5/13/2015, 9:00 AM

Evan Gilman is an operations engineer at PagerDuty, responsible for designing and automating the company's resilient infrastructure. Prior to PagerDuty, he operated AS4511 at the University of Miami, and has a passion for all things network. This story originally appeared on PagerDuty.

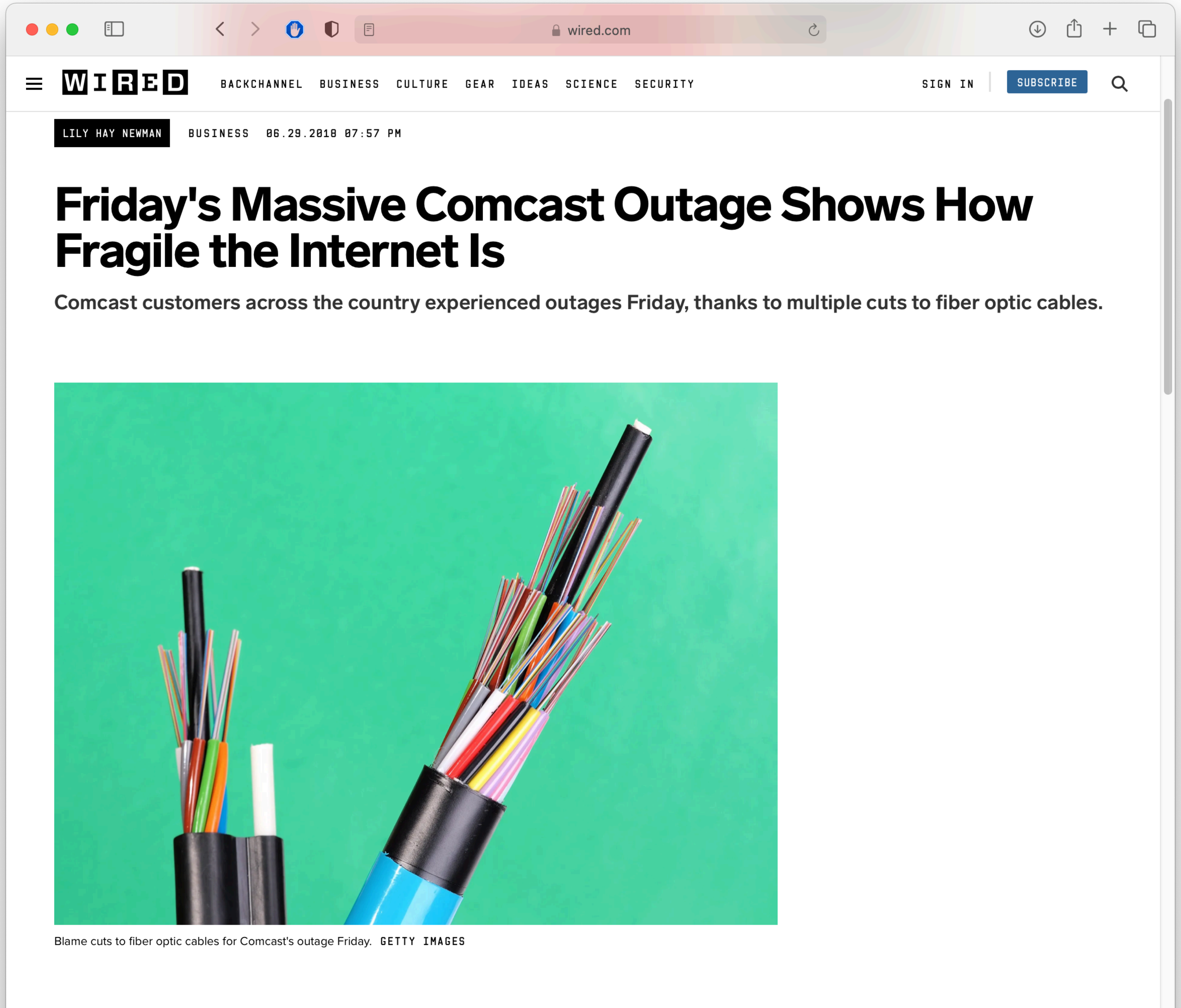
ZooKeeper, for those who are unaware, is a well-known open source project that enables highly reliable distributed coordination. It is trusted by many around the world, including PagerDuty. It provides high availability and linearizability through the concept of a leader, which can be dynamically re-elected, and ensures consistency through a majority quorum.

The leader election and failure detection mechanisms are fairly mature, and typically just work... until they don't. How can this be? Well, after a lengthy investigation, we managed to uncover four different bugs coming together to conspire against us, resulting in random cluster-wide lockups. Two of those bugs lay in ZooKeeper, and the other two were lurking in the Linux kernel. This is our story.

Background: The use of ZooKeeper at PagerDuty

Here at PagerDuty, we have several disparate services that power our alerting pipeline. As events are received, they traverse these services as a series of tasks that get picked up off of various work queues. Each one of these services leverages a dedicated ZooKeeper cluster to coordinate which application host processes each task. As such, you can imagine that ZooKeeper operations are absolutely critical to the reliability of PagerDuty at large.

Part I: The ZooKeeper bugs



wired.com

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY SIGN IN SUBSCRIBE

LILY HAY NEWMAN BUSINESS 06.29.2018 07:57 PM

Friday's Massive Comcast Outage Shows How Fragile the Internet Is

Comcast customers across the country experienced outages Friday, thanks to multiple cuts to fiber optic cables.

Blame cuts to fiber optic cables for Comcast's outage Friday. GETTY IMAGES

And still more challenges

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

Comment

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

aws.amazon.com

aws

Contact Sales Support English My Account Sign In to the Console

Products Solutions Pricing Documentation Learn Partner Network AWS Marketplace Customer Enablement Events Explore More

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 still 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?