Distributed Systems

15-440 / 15-640

Fall 2021

Welcome! Course Staff

Yuvraj Agarwal

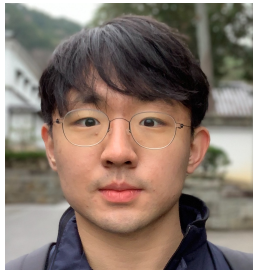
Rashmi Vinayak

Instructors

… and a team of 13 Awesome TA's

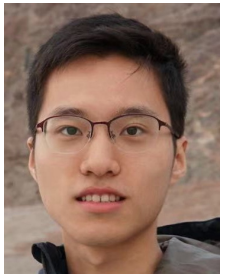
Qifei Dong

Jack Cameron



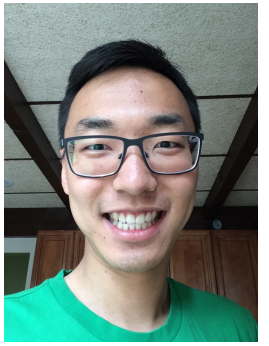
Young Hun Oh

William Foy



Yiwen Song (Victor)

Emma Cohron

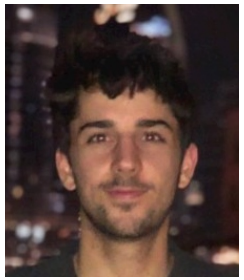


Han Zhang

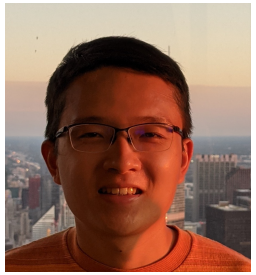
Kishan Patel

Eunice Chen

Crystal Jin



Riccardo Santoni



Karl Xiao

Nirav Atre

Outline

- Course logistics
- Waitlist
- Recitations & Office hours
- Course goals
- Course format
- Course policies
- Introduction to the course

Course Website

- Course website:

https://www.synergylabs.org/courses/15-440/

- The *tentative* schedule for the course is up
  - Always check on the website to see latest updates!

Communication: Piazza

- Join the class Piazza:
https://www.piazza.com/cmu/fall2021/1544015640/home

- No private posts allowed on Piazza

- For any private discussion, email us at the address below
  - Recipients: Yuvraj, Rashmi, PhD TAs (Han Zhang and Nirav Atre)
  - Email: ds-staff-f21-private@lists.andrew.cmu.edu

- Emails to individual instructors may not be answered
  - Please use Piazza (for all non-private questions) and the email list (only for something private)

Video Recording: Canvas

- Join the class on Canvas
- Recorded lecture videos will be posted on Canvas
  - Note: Linked to Panopto, but links posted on Canvas.
- If you are not able to attend the lecture in real time, must watch the video before the next lecture
- Note: We are making recordings available based on the current COVID posture, will re-evaluate over the Fall

Processing WL/Enrollment

- Unable to take the class if (no exceptions):
  - If not taken 213/513 at CMU *before*
  - If you are an UGRAD and lower than a "C" in 213
  - If you are a Grad and lower than a "B-" in 213/513
  - If you have schedule conflict for lecture time

- Priority order
  - Required: CS UGrad, MS in SCS (MSCS, MSDC, MITS,..)
- ... Use WL rank + 213 Grade for ECE and INI students

- Questions?
  - Mary Widom (ugrad) <marwidom@andrew.cmu.edu>
  - Angy Molly (grad) <amalloy@andrew.cmu.edu>

Waitlist

- Class capacity is limited by room size GHC4401 (237)
  - As of now, 237 enrolled in 15-440 and 15-640 (!!)
  - Current WL is ~75 students: 15-440 (5), 15-640 (70).
  - We expect some churn in first few weeks, typically 10-15 drops

- The advice: If you are interested but still on WL:
  - Watch the lecture, do P0, P1, but you **CANNOT** attend in person class
  - Reality: 10-15 more students from the WL *may* be admitted
  - If you your WL # is higher than 5, no chance of being enrolled

- The plea: Not serious? PLEASE DROP SOON.

- Questions?
  - Mary Widom (ugrad) <marwidom@andrew.cmu.edu>
  - Angy Molly (grad) <amalloy@andrew.cmu.edu>

Recitations & TA office hours

- Recitations
  - Please see S3 for the latest schedule for recitations
  - All recitations are in-person, Wednesdays 7-8pm and 8pm-9pm.
  - Please attend the section in which you are enrolled. (No exceptions; need to be strictly adhered to)

- Recitations (6 or 7) primarily to support Projects
  - Will be updated on the course website and announced in class
  - Introduction to GO (2nd week of class)
  - Introduction to P0, P1, P2, P3 + Discussion after projects due
  - Lead by TAs, are not meant to go over class lectures
  - No recitations this (first) week

Recitations & TA office hours

- TA Office Hours (7 days / week, spread out during the day)
  - Will be updated on the website + course Google Calendar
  - Will include instructor office hours
  - No office hours first (this) week
  - Managed using OH Queue (link on the website)
  - No office hour help on debugging 24 hours before projects are due
- TA office hours: Practical issues for implementing projects
- Instructor office hours: general questions and discussion of course content

Course Goals

- Systems requirement:
  - Learn something about distributed systems in particular;
  - Learn general systems principles (modularity, layering, naming, security, ...)
  - Practice implementing real, larger systems; in teams; must run in nasty environment;

- One consequence: Must pass homeworks, exams, and projects independently as well as in total.
  - Note, if you fail either you will not pass the class

Course Format

- ~24 lectures: Tu/Th 10:10am – 11:30am in GHC 4401

- 4 projects; 2 solo (p0, p2), 2 person team (p1,p3)
  - P0 – warm up project, learn syntax of GO
  - P1: Distributed (internet-wide) bitcoin miner
  - P2: Project with distributed systems concepts like distributed commit/consensus (e.g. PAXOS/RAFT)
  - P3: Building Tribbler

- 4 Homeworks, cover course lecture topics

Course Format – Midterms

Everything on this slide is tentative, check website

- Current Plan: Two Mid-terms
  - *Tentative*: Oct 12 and Dec 2, subject to change
  - *Typically* : During scheduled class time

- Registrar: Please do not make any plans to leave for winter break before the final exam schedule is out.
  - If you must, then earliest day you could leave: Dec 15th
  - Announce in class and also update the course website

Course Format – Grading

- 45% Projects
- 15% Homeworks
- 20% Midterm 1
- 20% Midterm 2

Collaboration

- Working together important
  - Discuss course material
  - Work on problem debugging
- Parts must be your own work
  - Homeworks, midterm, final, solo projects

- What we hate to say: we run cheat checkers...
- Please *do not* put code on *public* repositories

Collaboration

- Team projects: both students should understand entire project

- Try to identify partners on your own
- If you are not paired by end of week 2, we will help

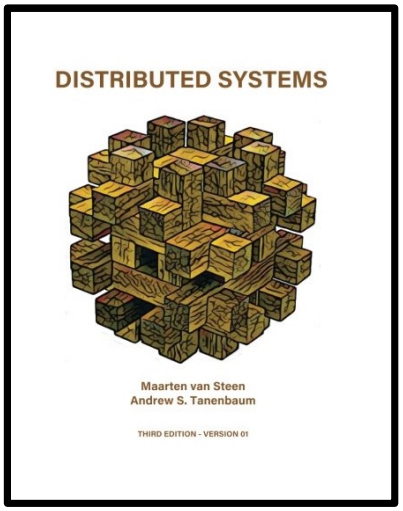
- Partner problems: *Please address them early*

Late Work

- 10% penalty per day
- Cannot be more than 2 days late
  - (no exceptions after 48 hours of due date/time)
  - No TA help after the official deadline (i.e., for late days)
- Usual exceptions:
  - documented medical, emergency, etc.
- Talk to us early if there's a problem!
- Regrade requests in writing to course admin

Study Material

- Slides and notes on course website
  - Not identical to prior 15-440 instances
- Distributed Systems 3.0.1 (2017)
  - Free download
  - Link to purchase ($25) from syllabus page
- Several useful references on web page
- Go work through the Tour of Go!
  - https://tour.golang.org/welcome/1



About Projects

- Projects are in Go
- Systems programming somewhat different from what you've done
  - Low-level (C / GO)
  - Designed to run indefinitely (error handling must be rock solid)
  - Must be secure - horrible environment
  - Concurrency
  - Interfaces specified by documented protocols

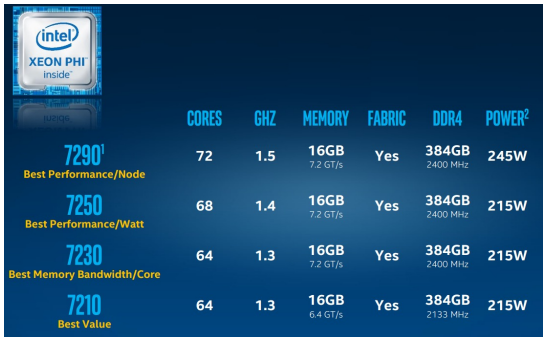
- TA office Hours & "System Hacker's View of Software Engineering"
  - Practical techniques designed to save you time & pain

- WARNINGS
  - Many students dropped during Project 1 (started too late!)
  - No TA help and office hours on the day before the project deadline

Questions?

Why take this course?

- Huge amounts of computing are now distributed...
  - A few years ago, Intel threw its hands up in the air: couldn't increase GHz much more without CPU temperatures reaching solar levels
  - But we can still stuff more transistors (Moore's Law)
  - Result: Multi-core and GPUs.
  - Result 2: Your computer has become
  - a parallel/distributed system



- Oh, yeah, and that whole Internet thing...
  - my phone syncs its calendar with google, which I can get on my desktop with a web browser, ...
    - (That phone has the computing power of a desktop from 10 years ago and communicates wirelessly at a rate 5x faster than the average american home could in 1999.)
  - Stunningly impressive capabilities now seem mundane. But lots of great stuff going on under the hood...
  - Most things are distributed, and more each day

If you find yourself ...

- In Hollywood....
  - ... rendering videos on clusters of 100s of 10,000s of nodes?
  - Or getting terabytes of digital footage from on-location to post-processing?
- On Wall Street...
  - tanking our economy with powerful simulations running on large clusters of machines
  - For 11 years, the NYSE ran software from Cornell systems folks to update trade data
- In biochem...
  - using protein folding models that require supercomputers to run
- In gaming...
  - Writing really bad distributed systems to enable MMOs to crash on a regular basis
- Not to mention the obvious places (Internet-of-Things Anyone?)

What Is A Distributed System?

"A collection of independent computers that appears to its users as a single coherent system."

- Features:
  - No shared memory – message-based communication
  - Each runs its own local OS
  - Heterogeneity
- Ideal: to present a single-system image:
  - The distributed system "looks like" a single computer rather than a collection of separate computers.

Characteristics of a DS

- Present a single-system image
  - Hide internal organization, communication details
  - Provide uniform interface
- Easily expandable
  - Adding new servers is hidden from users
- Continuous availability
  - Failures in one component can be covered by other components
- Supported by middleware

Distributed System Layer

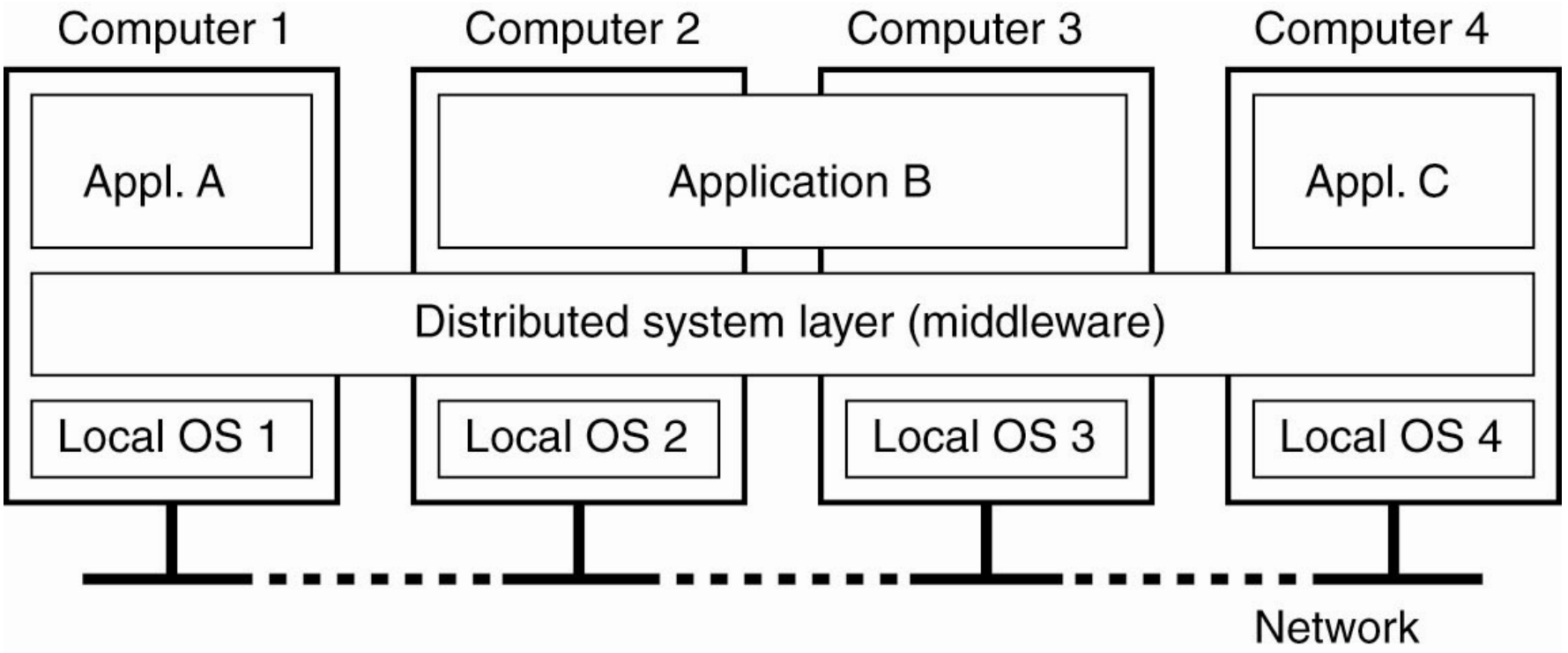


Figure 1-1. A distributed system organized as middleware. The middleware layer runs on all machines, and offers a uniform interface to the system

Goal 1 – Resource Availability

- Support user access to remote resources (printers, data files, web pages, CPU cycles) and the fair sharing of the resources
- Economics of sharing expensive resources
- Performance enhancement – due to multiple processors; also due to ease of collaboration and info exchange – access to remote services
- Resource sharing introduces security problems.

Goal 2 – Transparency

- Software hides some of the details of the distribution of system resources.
  - Makes the system more user friendly.
- A distributed system that appears to its users & applications to be a single computer system is said to be transparent.
  - Users & apps should be able to access remote resources in the same way they access local resources.
- Transparency has several dimensions.

Types of Transparency

Transparency	Description
Access	Hide differences in data representation & resource access (enables interoperability)
Location	Hide location of resource (can use resource without knowing its location)
Migration	Hide possibility that a system may change location of resource (no effect on access)
Replication	Hide the possibility that multiple copies of the resource exist (for reliability and/or availability)
Concurrency	Hide the possibility that the resource may be shared concurrently
Failure	Hide failure and recovery of the resource. How does one differentiate betw. slow and failed?
Relocation	Hide that resource may be moved during use

Transparency to Handle Failures?

The Joys of Real Hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
~1 network rewiring (rolling ~5% of machines down over 2-day span)
~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
~5 racks go wonky (40-80 machines see 50% packetloss)
~8 network maintenances (4 might cause ~30-minute random connectivity losses)
~12 router reloads (takes out DNS and external vips for a couple minutes)
~3 router failures (have to immediately pull traffic for an hour)
~dozens of minor 30-second blips for dns
~1000 individual machine failures
~thousands of hard drive failures

slow disks, bad memory, misconfigured machines, flaky machines, etc.

slide from Jeff Dean, Google

Goal 3 - Openness

- An **open distributed system** "…offers services according to standard rules that describe the syntax and semantics of those services." In other words, the interfaces to the system are clearly specified and freely available.
  - Compare to network protocols, Not proprietary
- **Interface Definition/Description Languages (IDL)**: used to describe the interfaces between software components, usually in a distributed system
  - Definitions are language & machine independent
  - Support communication between systems using different OS/programming languages; e.g. a C++ program running on Windows communicates with a Java program running on UNIX
  - Communication is usually RPC-based.

Open Systems Support …

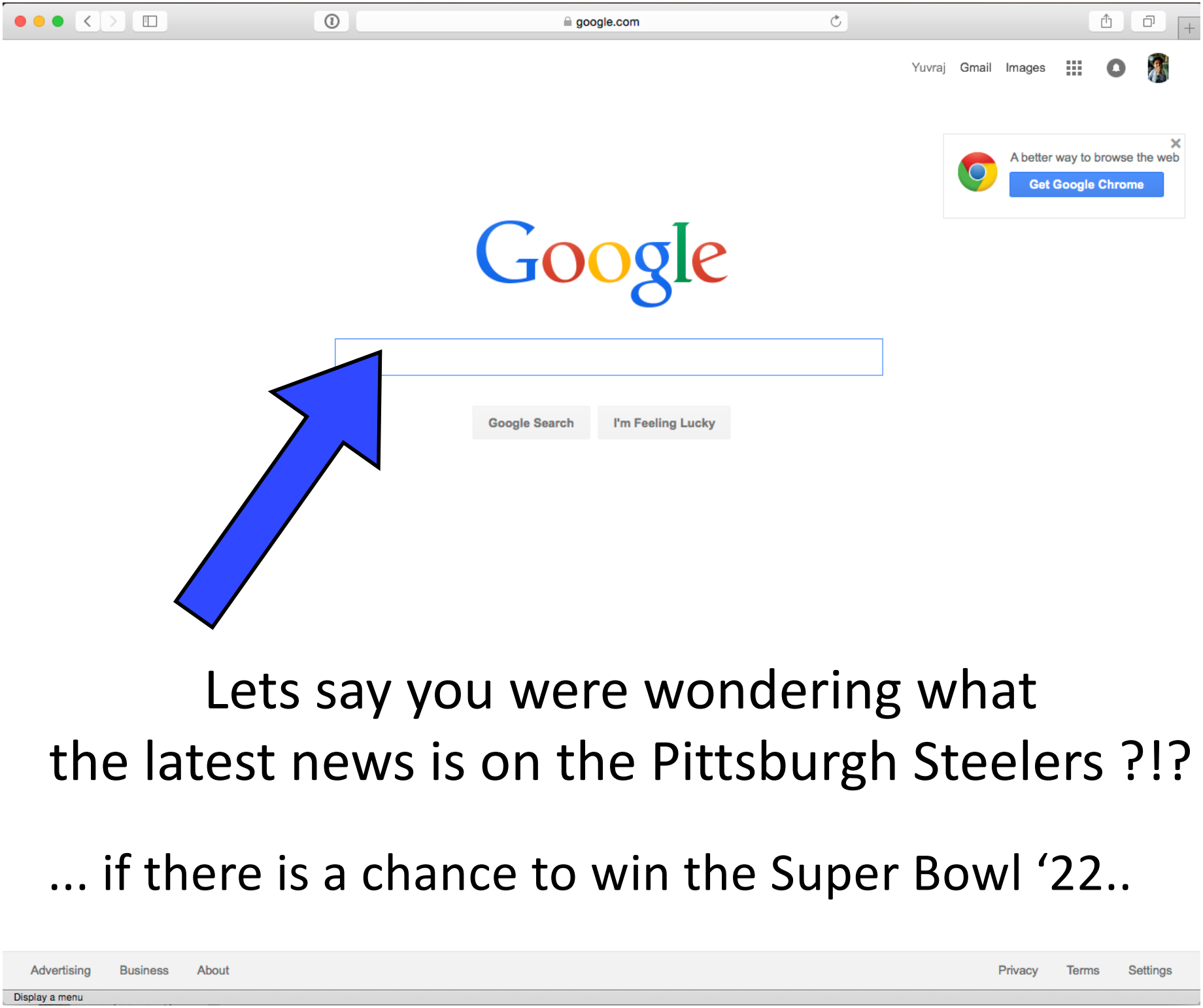
- **Interoperability**: the ability of two different systems or applications to work together
  - A process that needs a service should be able to talk to any process that provides the service.
  - Multiple implementations of the same service may be provided, as long as the interface is maintained
- **Portability**: an application designed to run on one distributed system can run on another system which implements the same interface.
- **Extensibility**: Easy to add new components, features

Goal 4 - Scalability

- Dimensions that may scale:
  - With respect to size
  - With respect to geographical distribution
  - With respect to functionality
  - With respect to administration
  - …
- A scalable system still performs well as it scales up along any of the dimensions.

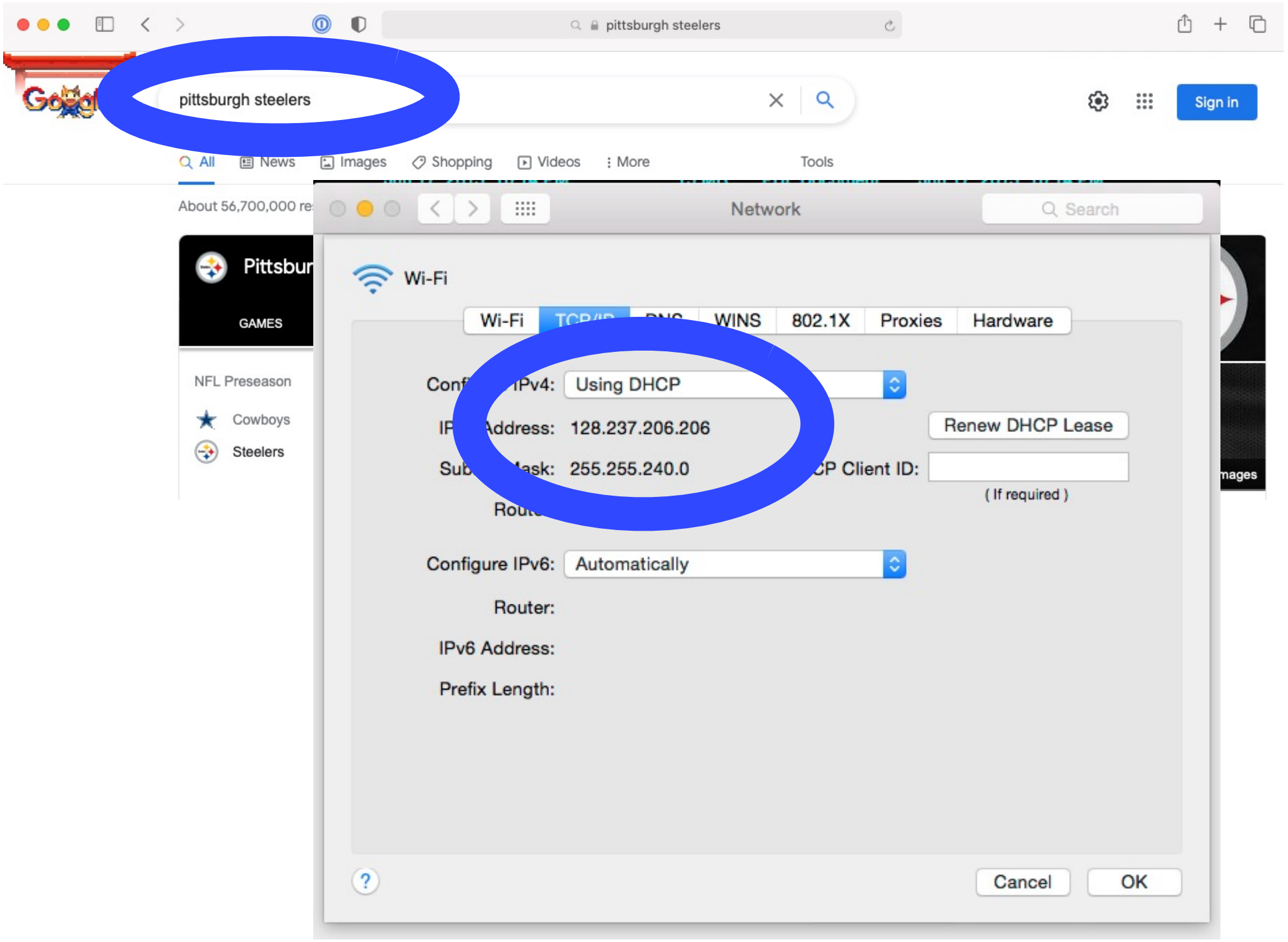
Enough advertising

- Let’s look at one real distributed system
- That’s drastically more complex than it might seem from the web browser...

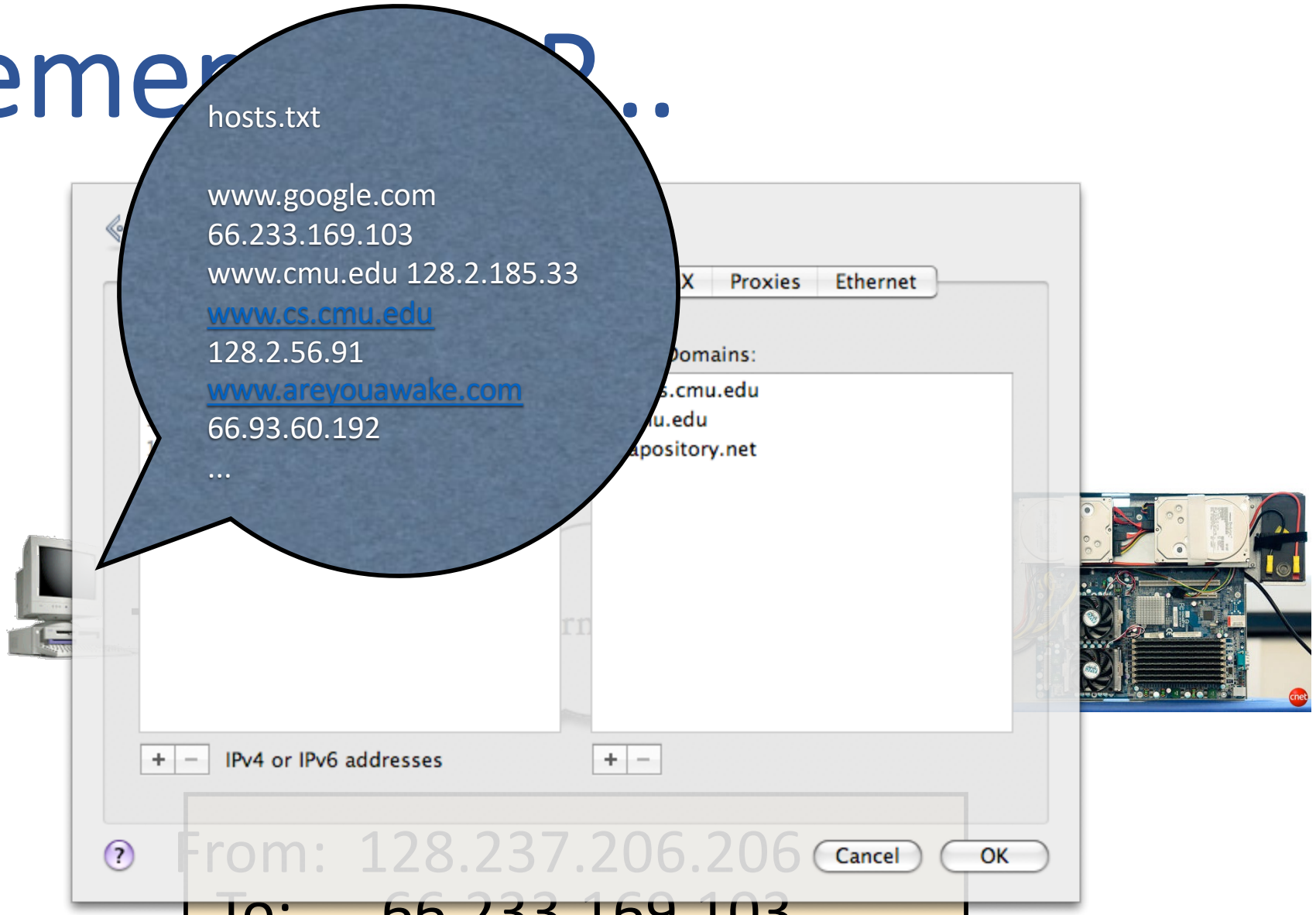


Lets say you were wondering what
the latest news is on the Pittsburgh Steelers ?!?

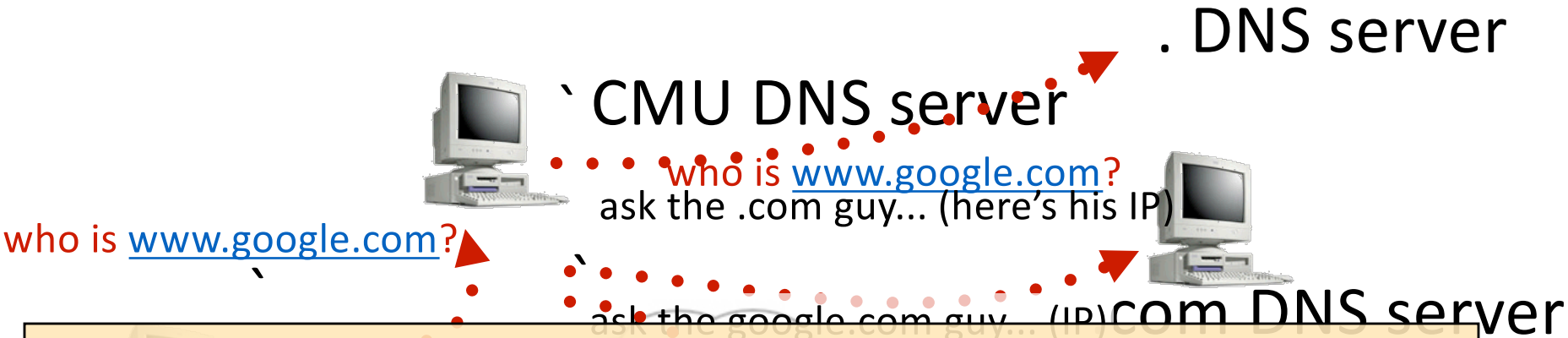
... if there is a chance to win the Super Bowl ‘22..


pittsburgh steelers
pittsburgh steelers
Sign in
All
News
Images
Shopping
Videos
More
Tools
About 56,700,000 re
Pittsbur
GAMES
NFL Preseason
Cowboys
Steelers
Network
Search
Wi-Fi
Wi-Fi
TCP/IP
DNS
WINS
802.1X
Proxies
Hardware
Conf IPv4: Using DHCP
IP Address: 128.237.206.206
Renew DHCP Lease
Sub Mask: 255.255.240.0
CP Client ID:
(If required)
Rout
Configure IPv6: Automatically
Router:
IPv6 Address:
Prefix Length:
Cancel
OK

Remember...


hosts.txt
www.google.com
66.233.169.103
www.cmu.edu 128.2.185.33
www.cs.cmu.edu
128.2.56.91
www.areyouawake.com
66.93.60.192
...
X
Proxies
Ethernet
Domains:
s.cmu.edu
u.edu
apository.net
IPv4 or IPv6 addresses
Cancel
OK
From: 128.237.206.206
To: 66.233.169.103
<packet contents>

Domain Name System

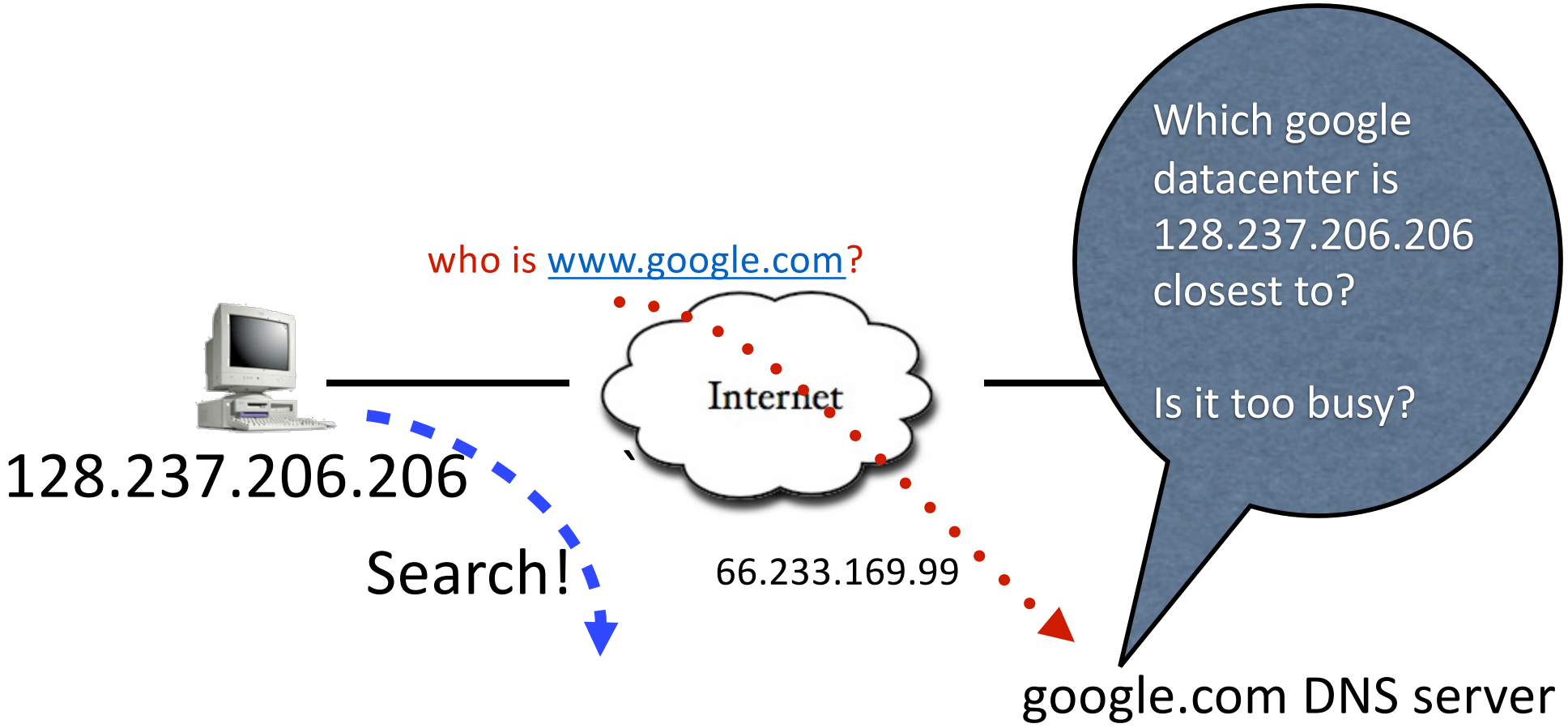


Decentralized - admins update own domains without coordinating with other domains

Scalable - used for hundreds of millions of domains

Robust - handles load and failures well

But there's more...



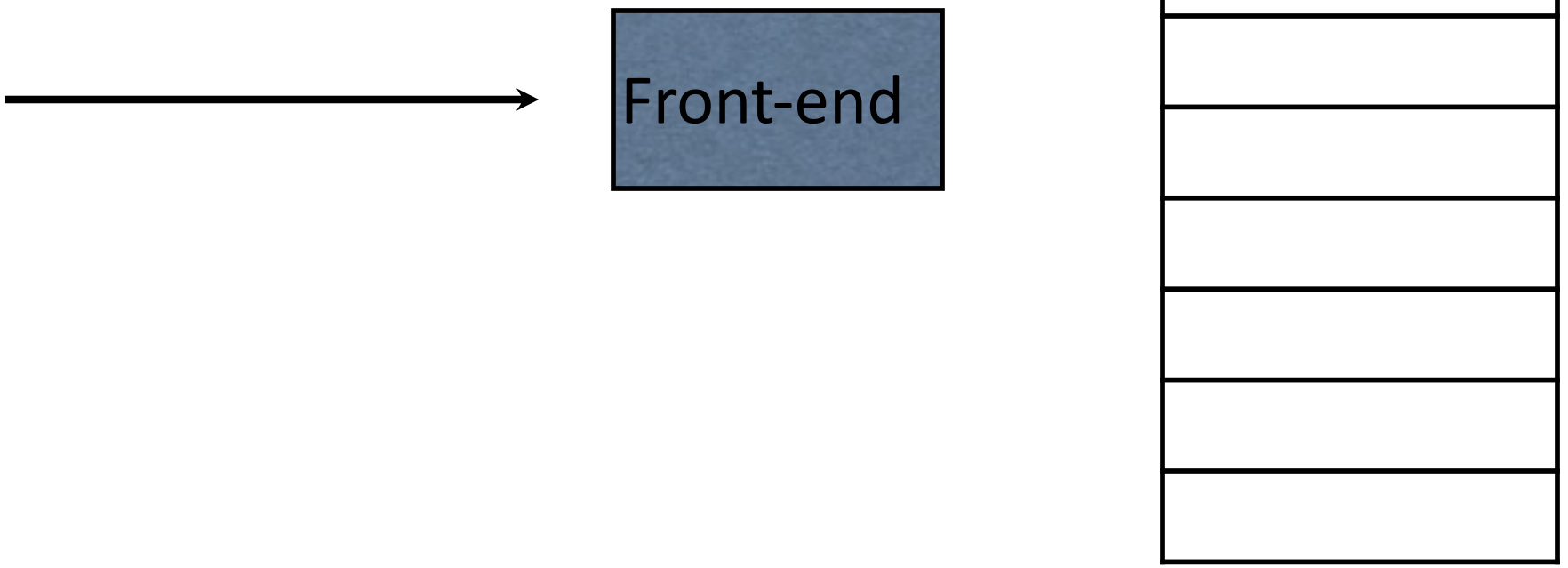
A Google Datacenter

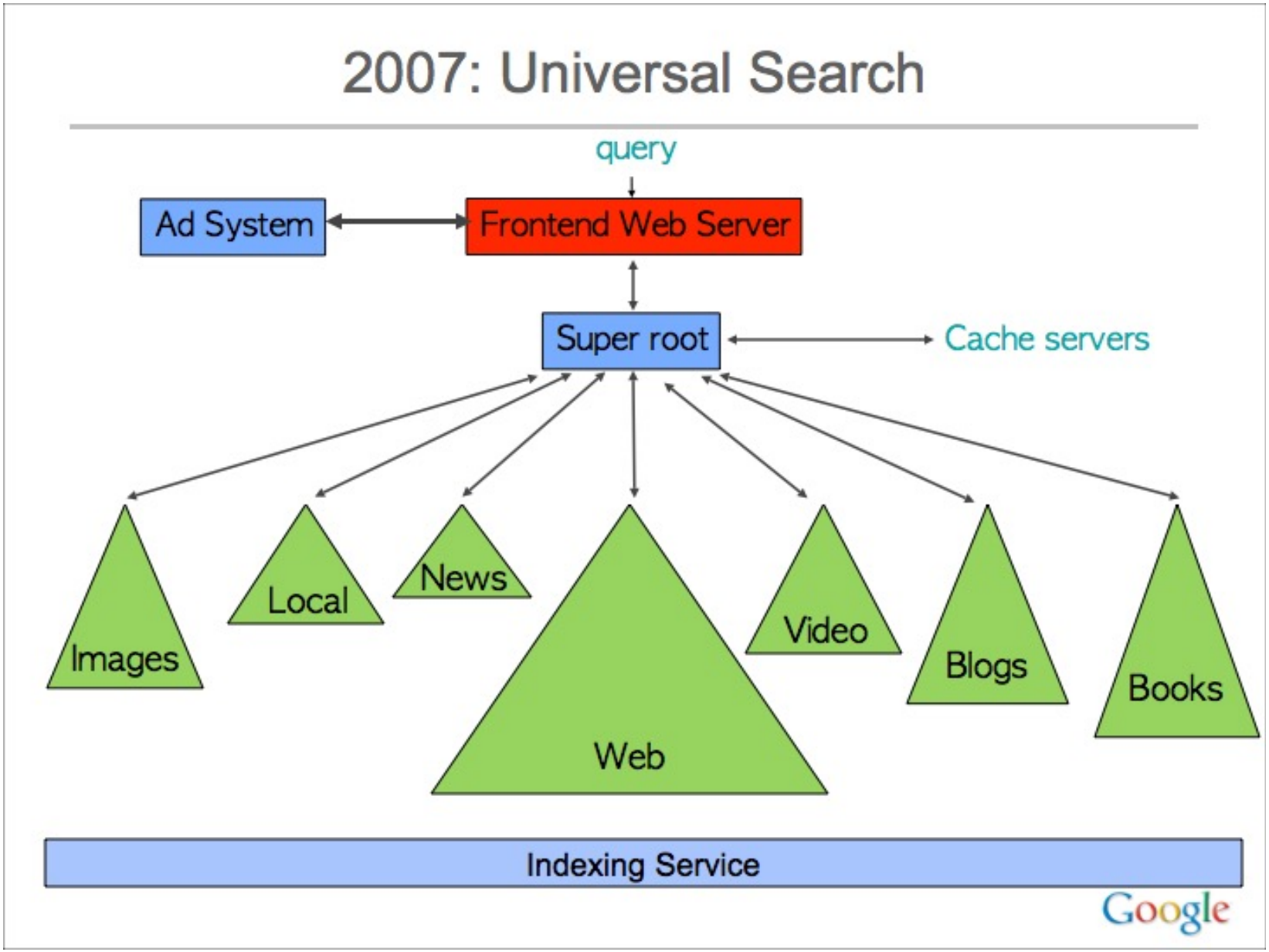
How big? Perhaps one million+ machines

but it's not that bad...

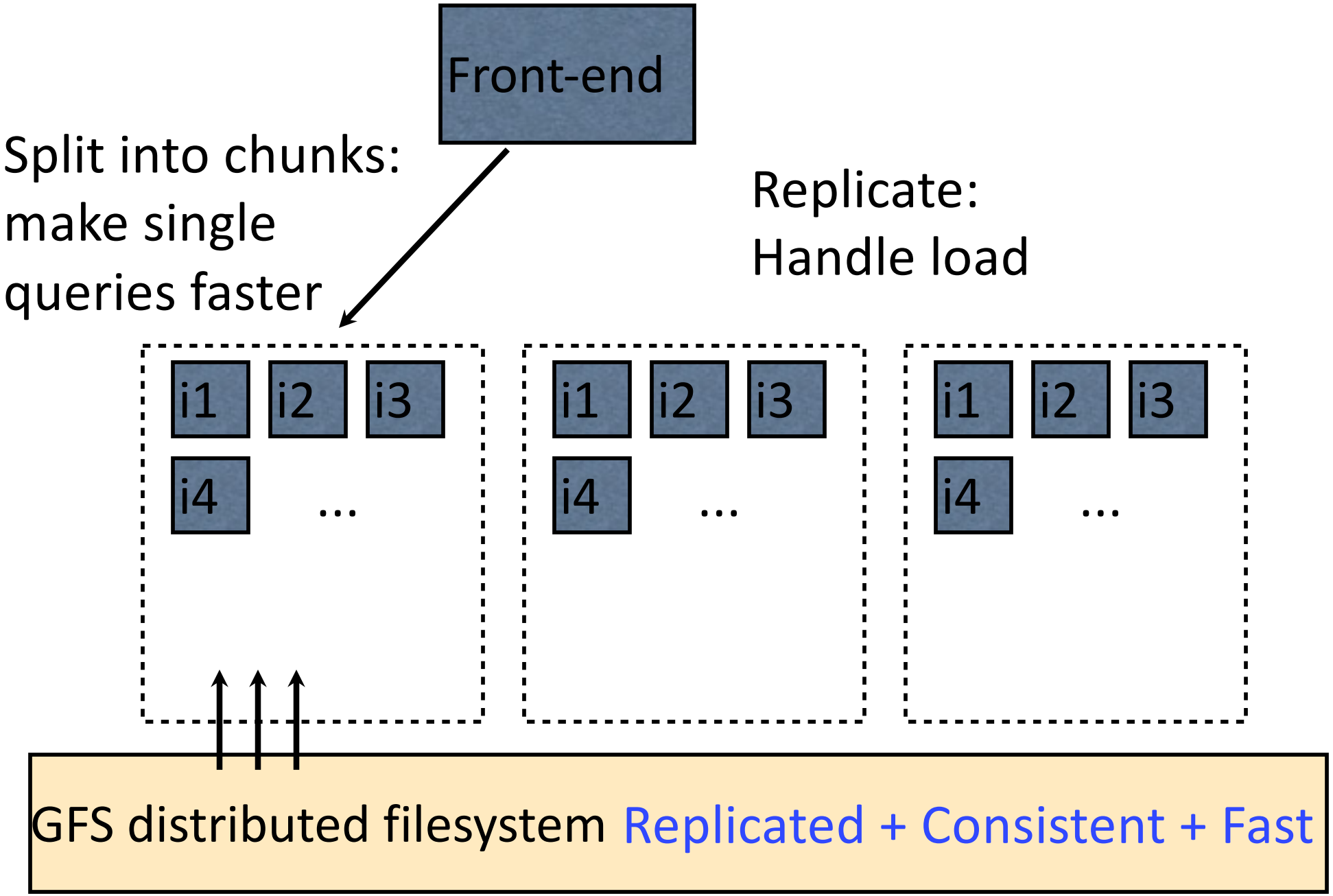
usually don't use more than **20,000** machines to accomplish a single task. [2009, probably out of date]

Search for “Steelers”





slide from Jeff Dean, Google


Front-end
Split into chunks:
make single
queries faster
Replicate:
Handle load
i1
i2
i3
i4
...
i1
i2
i3
i4
...
i1
i2
i3
i4
...
GFS distributed filesystem Replicated + Consistent + Fast

How do you index the web?

- Get a copy of the web.
- Build an index.
- Profit.

There are over 1 trillion unique URLs

Billions of unique web pages

Hundreds of millions of websites

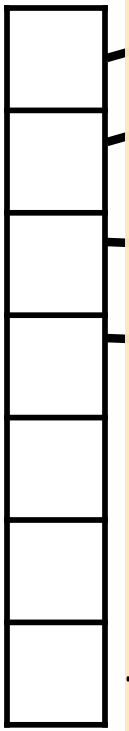
30?? terabytes of text

- Crawling -- download those web pages
- Indexing -- harness 10s of thousands of machines to do it
- Profiting -- we leave that to you.

- “Data-Intensive Computing”

MapReduce / Hadoop

Data Chunks



Storage

Why? Hiding details of programming 10,000 machines!

Programmer writes two simple functions:

map (data item) -> list(tmp values)
reduce (list(tmp values)) -> list(out values)

MapReduce system balances load, handles failures, starts job, collects results, etc.

Data Transformation

Data Aggregation

All that…

- Hundreds of DNS servers
- Protocols on protocols on protocols
- Distributed network of Internet routers to get packets around the globe
- Hundreds of thousands of servers
- … to find out what’s the deal with Pittsburgh Steelers!

All excited to learn how to distributed systems work and how to build them?

See you in the next class!

Questions?