Distributed Computing and the Internet

15-110 – Monday 03/18

Announcements

- Hw4 was due **today**
  - How did it go?

- Check5/Hw5 available
  - Note that Check5 only has a written component – no programming part
  - Can do half of the Hw5 programming component now; the other half requires helper functions

Announcements

- **Exam2** takes place **next Wednesday!**
  - Content covers Unit 2: Lists and Methods – Tractability
    - Unit 1 material also eligible, but not the focus
  - Same logistics as Exam1

- Review Sessions: TBA
  - Other review materials available on the Assessments page: https://www.cs.cmu.edu/~110/assessments.html
  - Vote on topics on Piazza!

Learning Goals

- Recognize and define the following keywords: **distributed computing, cloud computing, browsers, routers, ISPs, IP addresses, DNS servers, protocols,** and **packets**.

- Use the **MapReduce pattern** to design parallelized algorithms for distributed computing

- Understand at a high level the **internet communication process** that happens when you click on a link to a website in your browser.

Multiprocessing vs Distributed Computing

In the previous lecture, we discussed how we can run multiple programs at the same time on a single computer using **multiprocessing**.

This is useful, but you're still limited by the number of CPUs you can fit on a single machine. To get real efficiency gains, we'll need to use **multiple computers** instead.

Distributed Computing and MapReduce

Distributed Computing Uses Many Machines

Distributed Computing is a concurrency approach where you network multiple computers (each with its own set of CPUs) together and use them all to perform advanced computations. This can be done by assigning different processes to different computers.

The more computers you have available to network together, the faster your algorithm can be!

Scheduling with Distributed Computing

Each computer in the network can take a single task, break it up into further subtasks, and assign those subtasks to its cores. This makes it possible for us to solve problems quickly which would take a long time to solve on a single processor.

Task 1

Core 1	Subtask 1-1
Core 2	Subtask 1-2
Core 3	Subtask 1-3
Core 4	Subtask 1-4

Task 3

Core 1	Subtask 3-1
Core 2	Subtask 3-2
Core 3	Subtask 3-3
Core 4	Subtask 3-4

Task 2

Core 1	Subtask 2-1
Core 2	Subtask 2-2
Core 3	Subtask 2-3
Core 4	Subtask 2-4

Task 4

Core 1	Subtask 4-1
Core 2	Subtask 4-2
Core 3	Subtask 4-3
Core 4	Subtask 4-4

Cloud Computing

When a company says that they store things in "the cloud", they're referring to **other computers that are connected to the internet**. This is just distributed computing!

Companies use the cloud because it makes storage **cheap,** can **scale at need**, and is **available on demand**. It's generally easier to access computers that are provided by another company than to maintain a set of servers yourself.

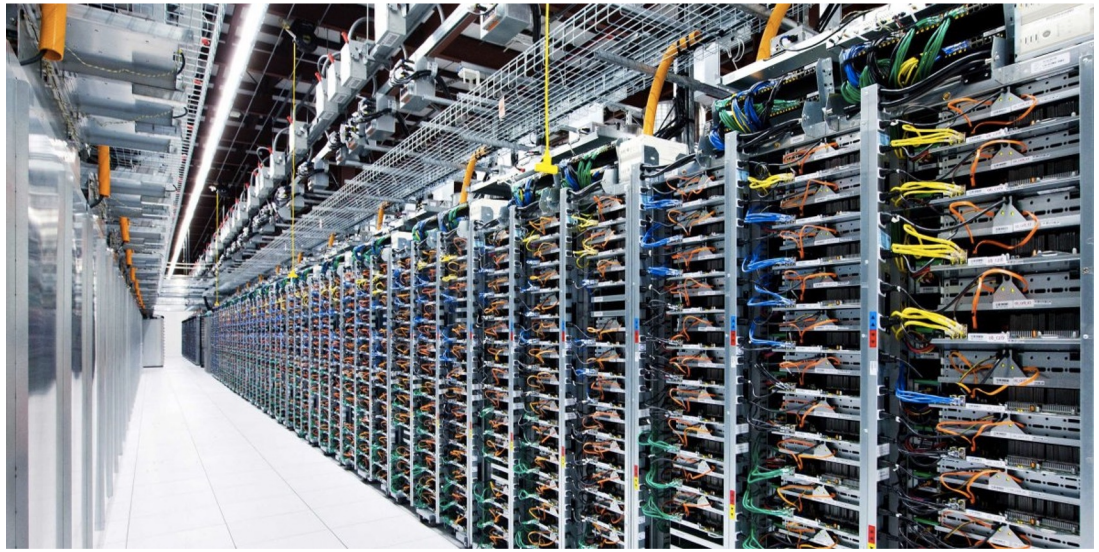
You probably use the cloud too. If you store data online (like in Google Drive, or Instagram), you're storing data in the cloud.

Server Farms and Supercomputers

Distributed computing is used by big tech companies (like Google and Amazon) to provide cloud computing, to manage thousands of customers simultaneously, and to process complex actions quickly.

This is where the term 'server farm' comes from- these companies will construct large buildings full of thousands of computers which are all networked together and ready to process information.

A **supercomputer** is very similar to distributed computing. It's a computer with a *huge* number of processors connected together. The main difference is that all the processors are located in the same place.



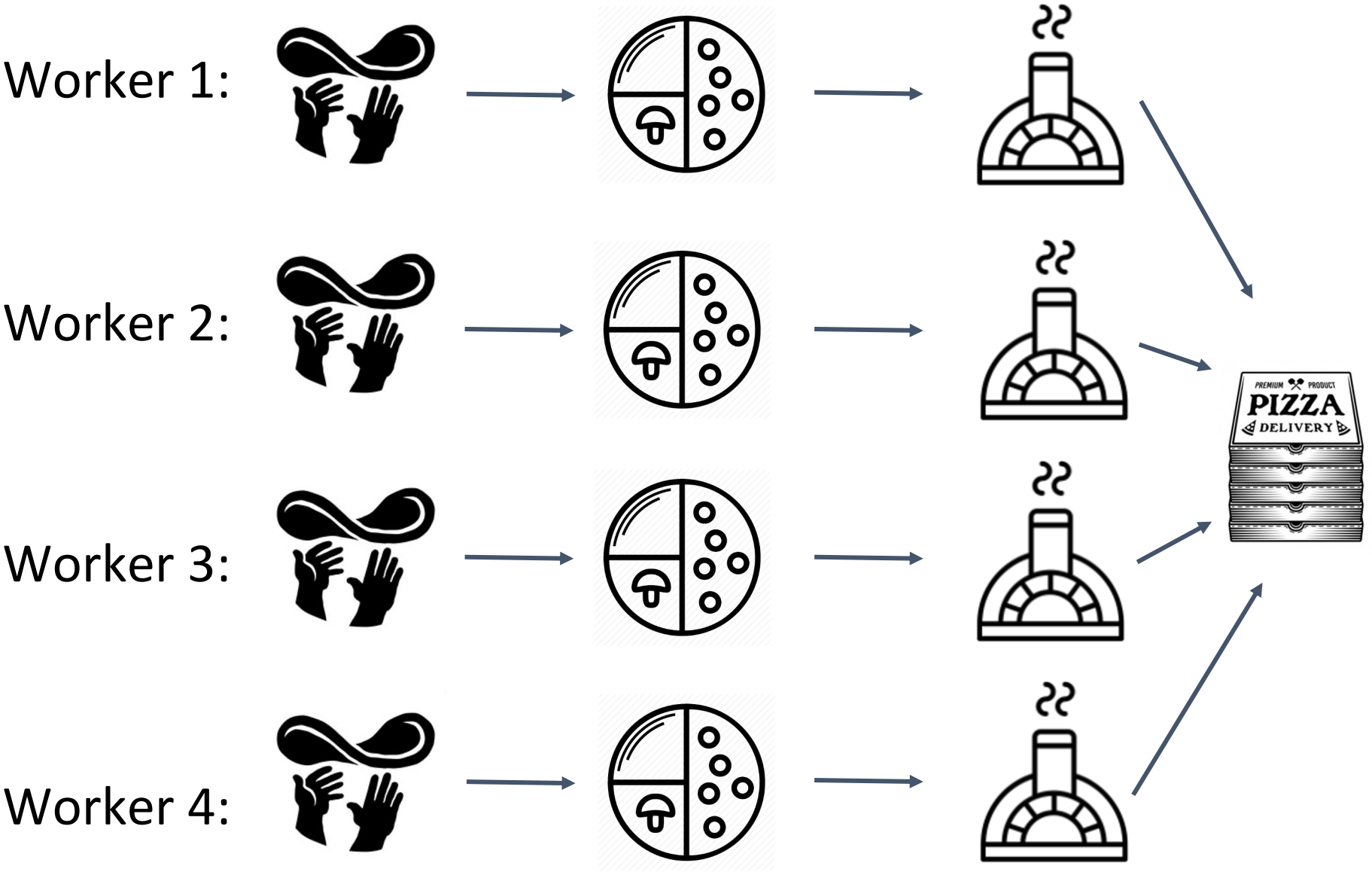
MapReduce Organizes Broad Concurrency

We still need a good approach to split algorithms across computers in order to take advantage of distributed computing. Pipelining can be useful here, but it only works for processes that have a large number of steps – if you have a simple three-step process and want to run a billion data points through it, it will still take a long time.

A different popular algorithm for organizing parallelized programs is called **MapReduce**. Instead of breaking up a procedure's steps across different cores, this algorithm takes a large data set and breaks up **the data itself** across the cores.

This is a really effective approach if you have a lot of cores to work with. It's also a great approach for any problem over **big data** – that is, giant data sets that are too large to process using typical software.

MapReduce - 4 workers, 4 ovens, 3 steps



Each worker makes one pizza instead of doing one task repeatedly.

If we have infinite ovens and infinite workers, we can make as many pizzas as we want in just 3 time-steps!

Making MapReduce Algorithms

A MapReduce algorithm is composed of three parts.

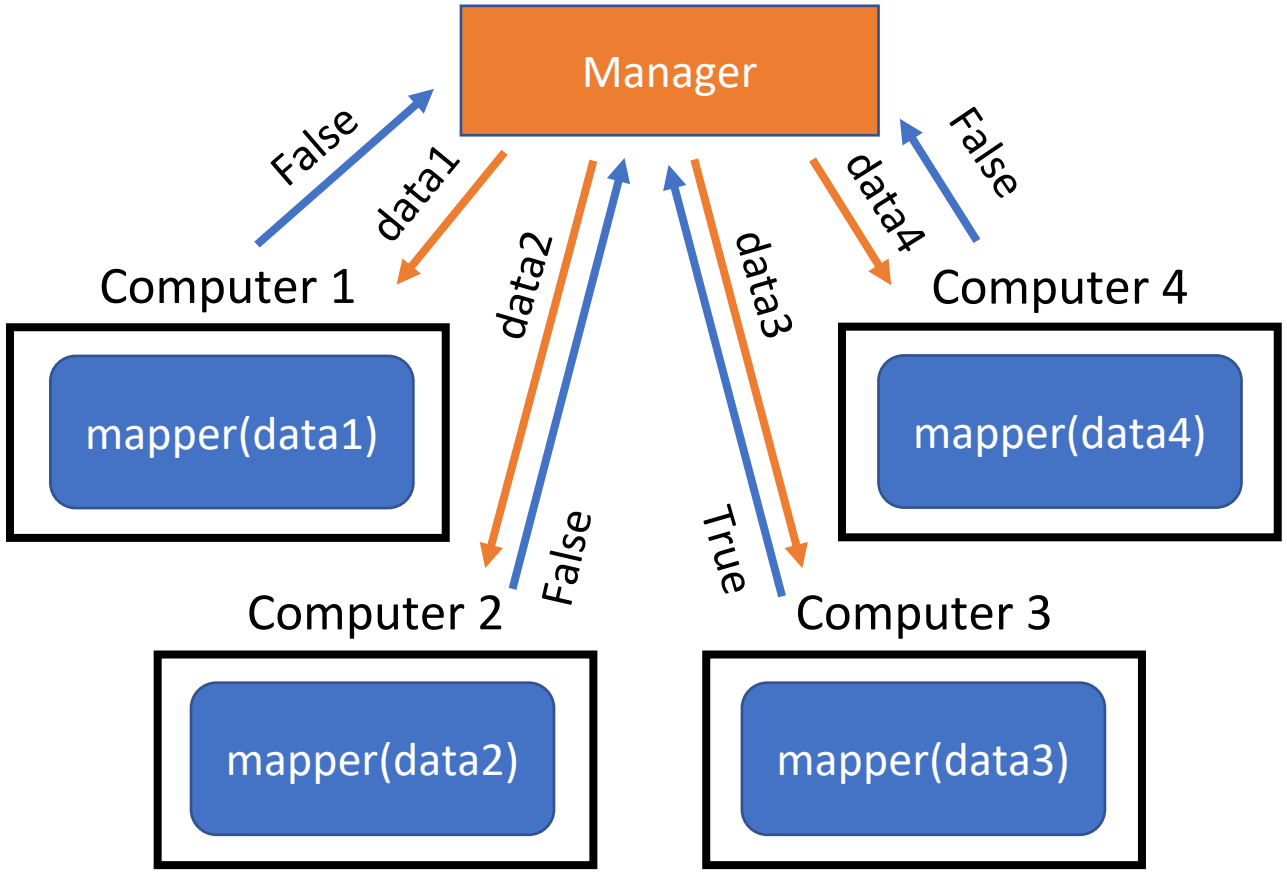
- The **mapper** takes a piece of data, processes it, and finds a partial result
- The **reducer** takes a set of results and combines them together
- The **manager** moves data through the process and outputs the final result
  - Splits up data, sends to mappers, get results back
  - Combines results together, sends to the reducer
  - Gets the final result, outputs it

MapReduce Example: Search – Mapper

Let's say we want to search a book for a specific word. How can we split up this task?

First, the **manager** divides the book into many small parts- maybe one page per part. It sends each page to a different computer.

Each computer runs its copy of the **mapper** on its page. It returns `True` if it finds the result, and `False` otherwise. These results are sent back to the **manager**.

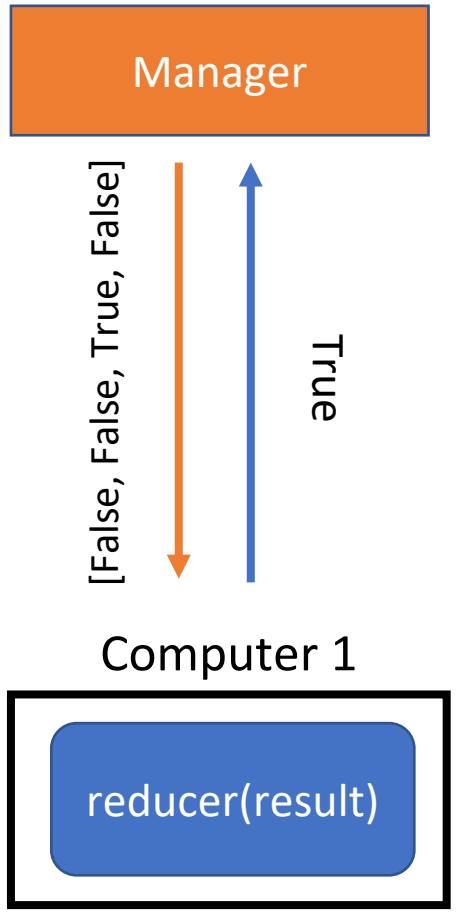


MapReduce Examples: Search – Reducer

Once all the mappers have returned their results the **manager** puts them all in a list and sends that list to the **reducer(s)**. The reducer combines the results together in some way.

There can be more than one reducer if there are lots of results to combine or if we're checking multiple things (like searching for more than one word). For now, we'll just use one.

Our reducer will check all of the results and send `True` back to the **manager** if any of them are `True`.

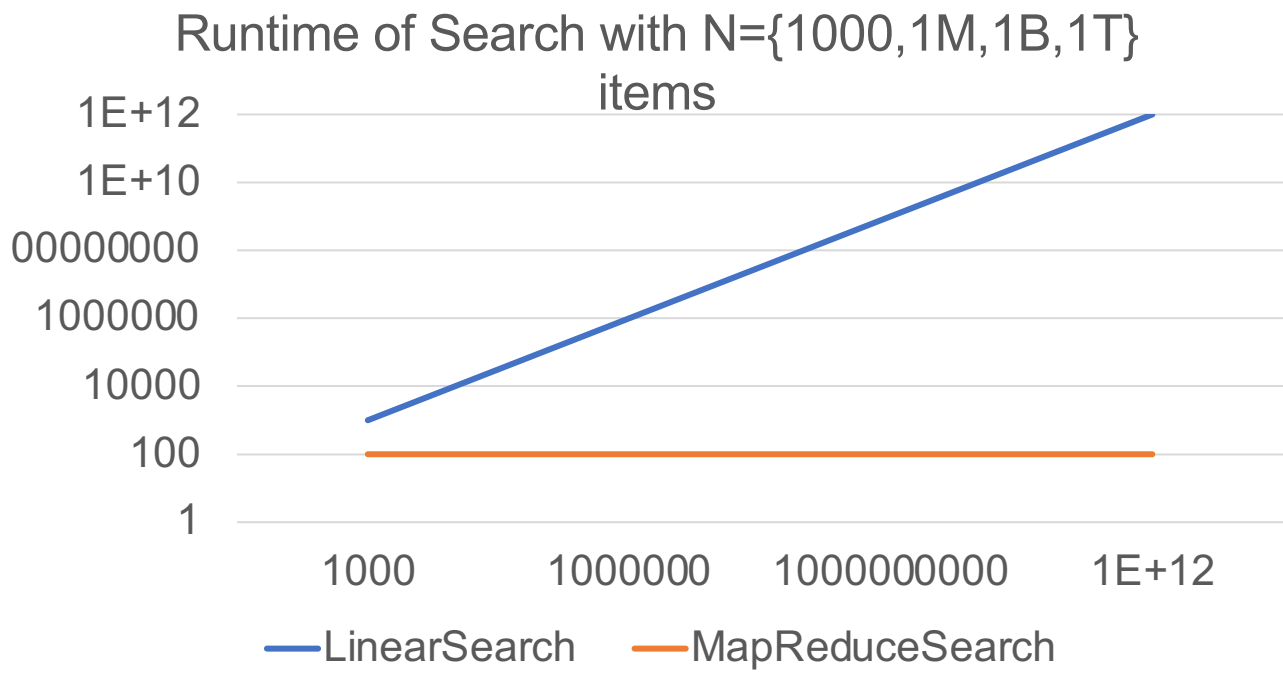


MapReduce Efficiency

MapReduce can process huge data sets and get results quickly because it takes a list of length N and breaks it up into **constant-size parts**.

The core assumption is that we have enough computers to make the data pieces really small. If we process 1 million data points with 100,000 computers, each computer only needs to handle 100 data points.

This is similar to the logic behind hashing!



Another Example: Find Max

What if we instead wanted to find the website you visit most often based on your entire browser history?

First, the **manager** breaks up the data- maybe the log for each day goes to a different computer.

The **mapper** can take the log for a single day and create a dictionary that maps each URL to a count of the number of times it was visited.

The **manager** takes a set of dictionaries and puts them into a list.

The **reducer** takes the list of dictionaries and merges them together. A second reducer could then take the merged-dictionary and find the most-visited website.

Uses of Distributed Computing

Distributed computing is incredibly useful for large-scale data processing! But that's not all it can do...

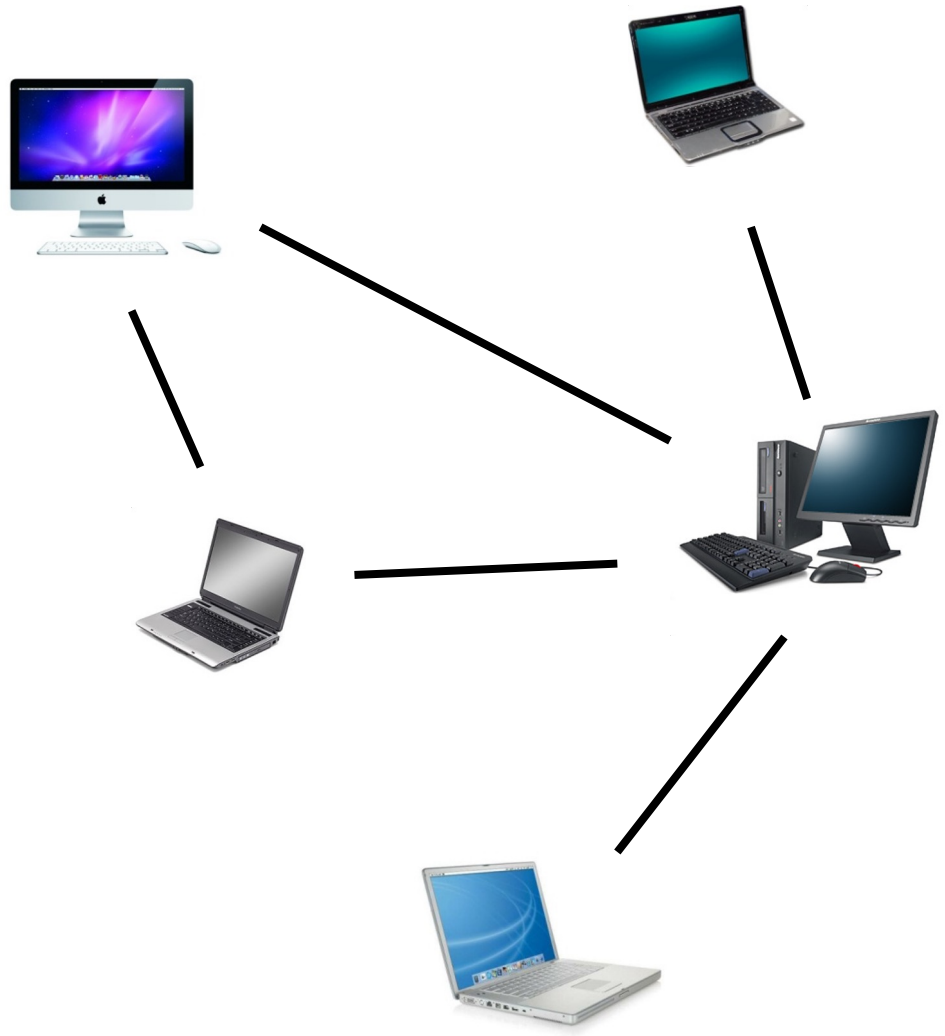
Distributed computing is also used to support **the internet itself!**

The Internet

What is the Internet?

The Internet is a network of computer networks all across the world that are connected (distributed computing!). The purpose of the internet is to send data between different computers in a manner that is **decentralized**- no one person has control over the whole thing.

It's like a **graph** where the **nodes** are computers and the **edges** are different methods of transmitting information.

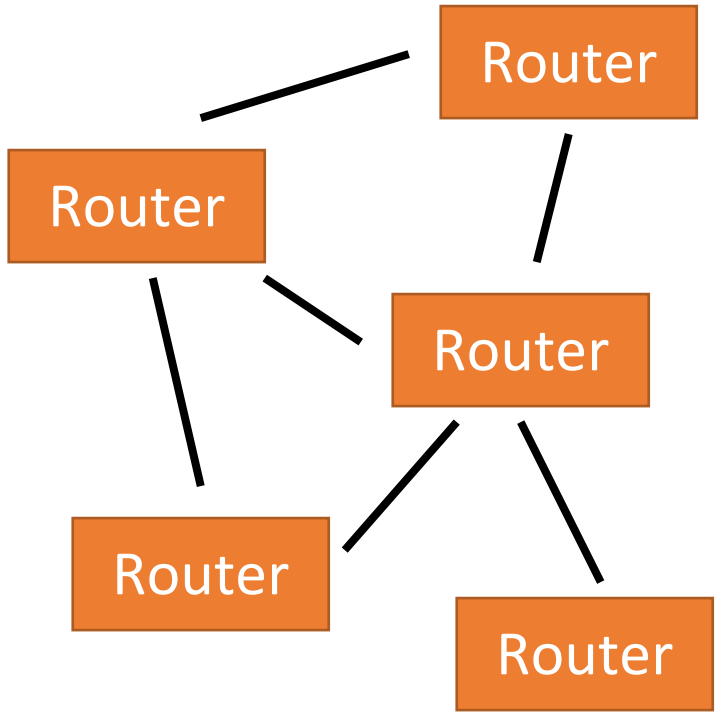


Routers are the Core

The core of the internet is a collection of devices called **routers**.

These devices act like switches – they take in data and send it to one of many possible locations based on the end destination of the data and the current connections on the internet.

There are thousands of routers spread across the world to help move data around from one place to another.

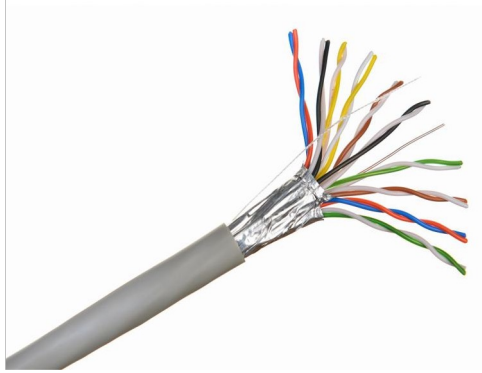


Connections Between Routers

Routers are commonly connected by **cables**, which are used to send data across a long distance. That data is usually represented using **bits**.

Cables range from telephone wires to coaxial cable to fiberoptic cable. All of these systems convert bits to different real-world representations (analog signal, electricity, light, etc.).

Computers can also send data to routers over **Wi-Fi**. In this connection, data is sent over a short distance via radio waves.



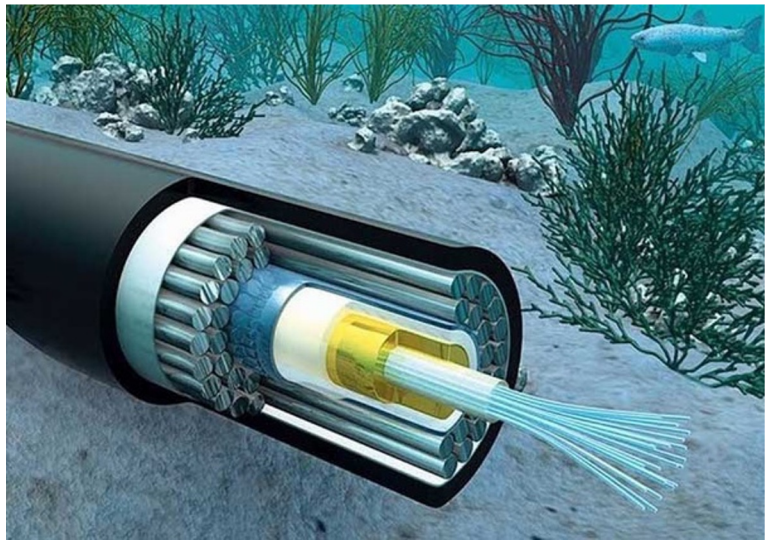
Sidebar: International Internet

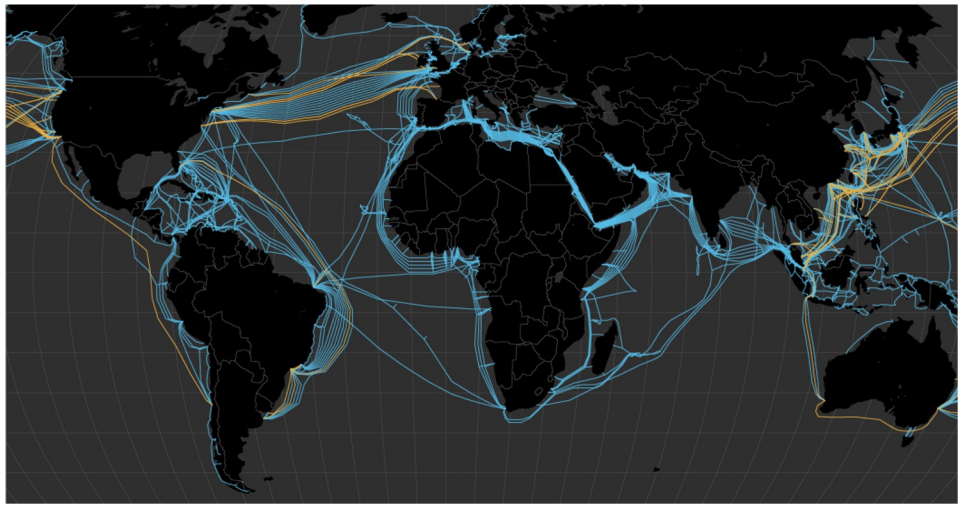
How does the internet connect across continents?

Giant fiberoptic cables have been laid on the ocean floor. Most international internet traffic is transmitted through these cables.

Read more:

https://www.nytimes.com/interactive/2019/03/10/technology/internet-cables-oceans.html





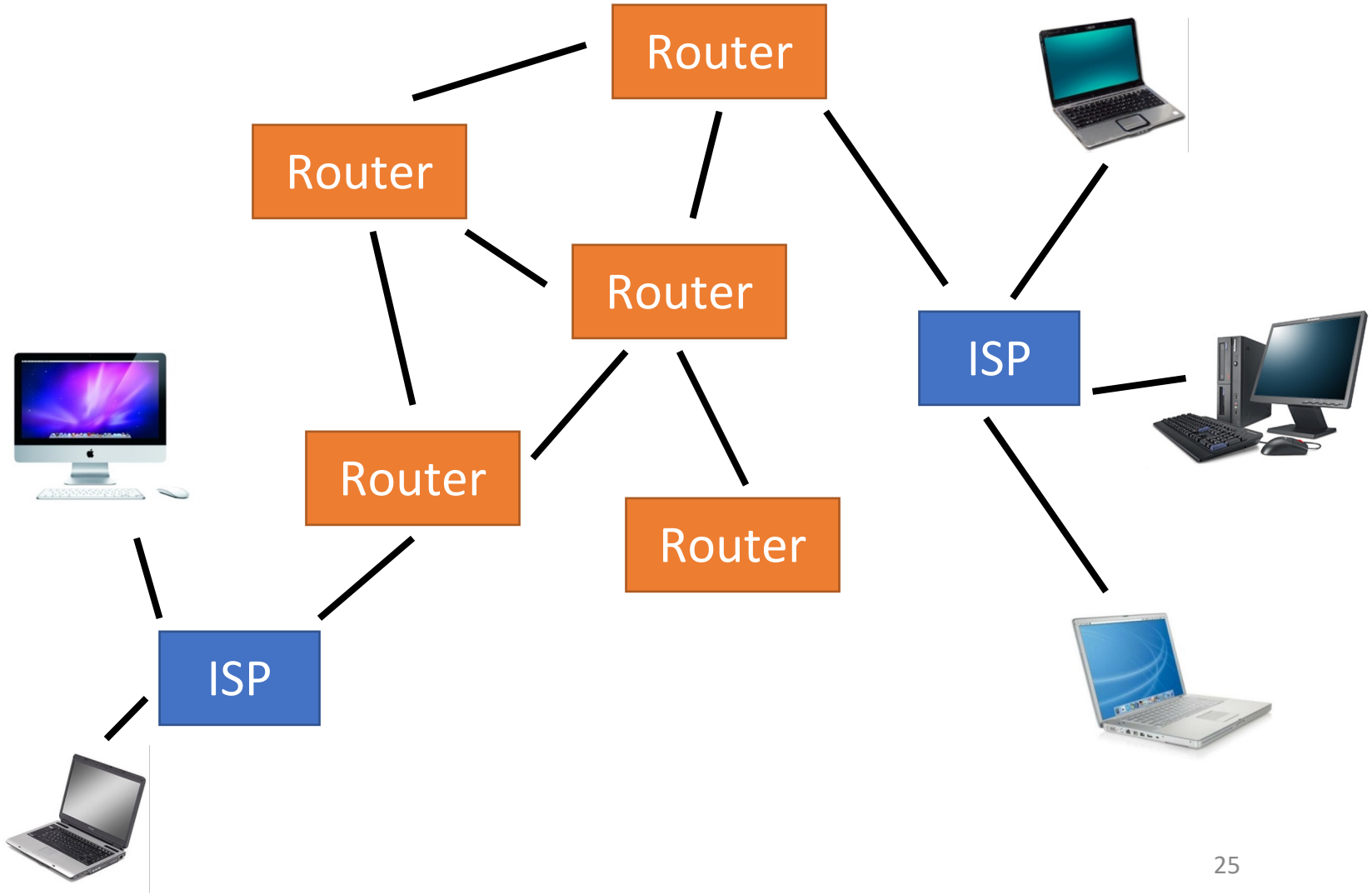
ISPs Manage Connections

Internet Service Providers (ISPs) connect a user's computer to the core of the internet.

Verizon, Comcast, etc. are all ISPs.

Organizations can be their own ISP – CMU is its own ISP, for example.

Some ISPs, called backbone ISPs, manage the routers at the core of the internet.



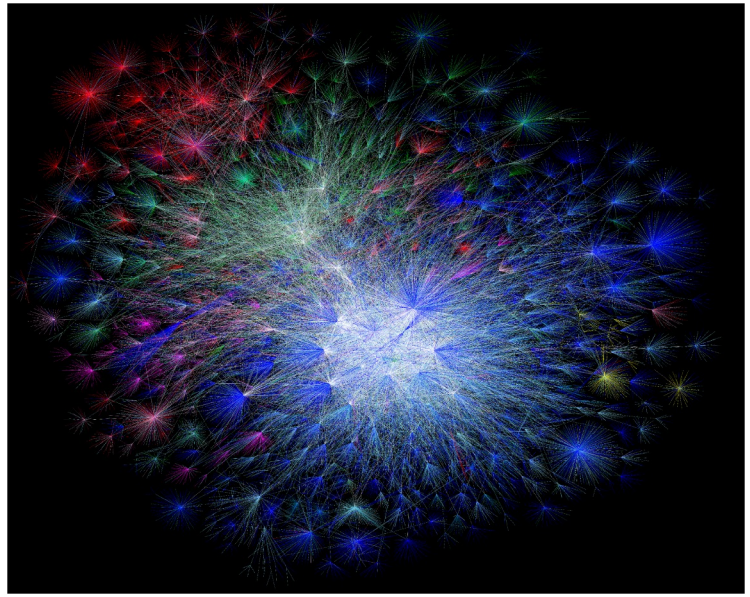
The Modern Internet is Huge!

The internet today is used widely across the world and contains millions of computers and connections.

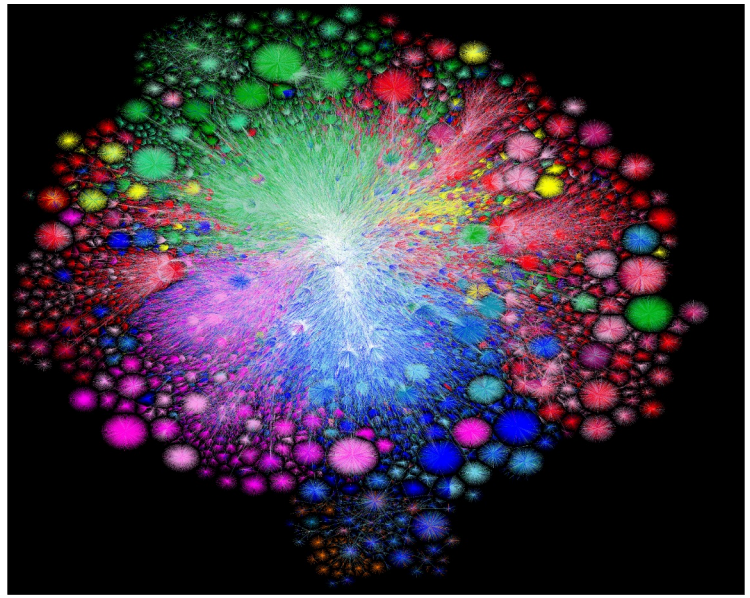
The pictures to the right (from the Opte Project) illustrate the computers connected to the internet around the world. The internet has grown massively over the years!

How is it possible for us to make a request for a specific website in this massive web and get the result back so quickly?

2000



2020



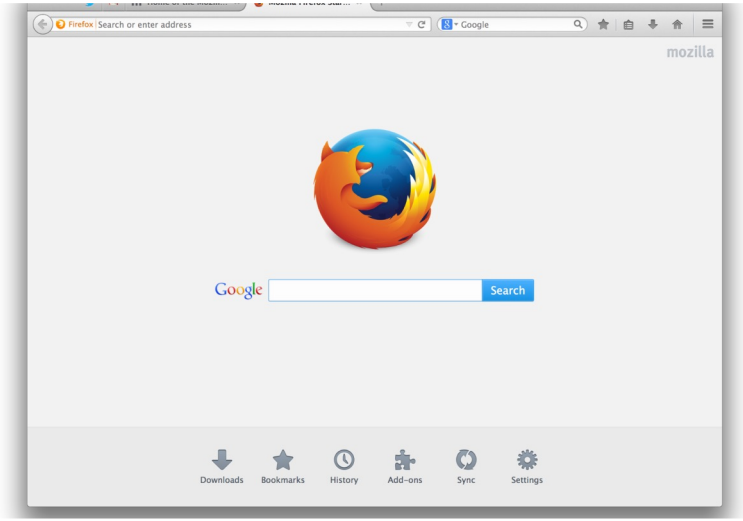
The Internet:
Journey of a Website

Browsers Display Data

Your **browser** (Firefox, Chrome, Safari, etc.) is an application that receives data from the internet and organizes it into a webpage that you can read.

Browsers receive webpages as **text** and turn that text into visual content using a **protocol** called **HTML** (HyperText Markup Language).

You can view the HTML of any webpage by right-clicking and selecting 'View Page Source'.




```
</nav>

<div class="container">

<h2>15-110: Principles of Computing</h2>

<br>
<p><b><font color="#990099">Due to the COVID-19 Epidemic, all classes and office hours from 03/16 onwards will be conducted remotely. Please
refer to the class Piazza for links to the Zoom class sessions.</font></b></p><br>

Principles of Computing (15110) is a course in fundamental computing principles for students with little to no computing background. Programming
constructs: sequencing, selection, iteration, and recursion. Data organization: arrays and lists. Use of abstraction in computing: data
representation, computer organization, computer networks, functional decomposition, and application programming interfaces for graphics. Use of
computational principles in problem-solving: divide and conquer, randomness, and concurrency. Classification of computational problems based on
complexity, non-computable functions, and using heuristics to find reasonable solutions to complex problems. Social, ethical and legal issues
associated with the development of new computational artifacts will also be discussed. Prerequisites: none. <br><br>

<h4>Meeting Times</h4>
<table class="table table-striped">
  <thead><td><b>Session</b></td><td><b>Instructor(s)</b></td><td><b>Time</b></td><td><b>Location</b></td></thead>
  <tr><td><b>Lecture 1</b></td><td><b>Kelly Rivers (krivers)</b></td><td><b>MWF 2:30-3:20pm</b></td><td><b>DH 2210</b></td></tr>
  <tr><td>Recitation A</td><td>Diaj (dtoussai) and Enock (emaburi)</td><td>R  9:30-10:20am</td><td>GHC 5207</td></tr>
  <tr><td>Recitation B</td><td>Amanda (lianglij) and Neeraj (neerajsa)</td><td>R 10:30-11:20am</td><td>GHC 5207</td></tr>
  <tr><td>Recitation C</td><td>Mahima A. (mahimaa) and Rachel (rachelt1)</td><td>R 11:30-12:20pm</td><td>GHC 5207</td></tr>
  <tr><td>Recitation D/J</td><td>Frank (frankh) and Mahima S. (mshanwar)</td><td>R 12:30-1:20pm</td><td>GHC 5207</td></tr>
  <tr><td>Recitation E</td><td>Andrea (arestrad) and Emily (eding)</td><td>R 1:30-2:20pm</td><td>GHC 5207</td></tr>
  <tr><td>Recitation F</td><td>Meghan (mamcgraw) and Rishabh (rishabhc)</td><td>R 2:30-3:20pm</td><td>GHC 5207</td></tr>
  <tr><td>Recitation G</td><td>Elyana (erhurst) and Iris (ilu1)</td><td>R 3:30-4:20pm</td><td>GHC 5207</td></tr>
  <tr><td><b>Lecture 2</b></td><td><b>Margaret Reid-Miller (mr54)</b></td><td><b>MWF 3:30-4:20pm</b></td><td><b>DH 2210</b></td></tr>
  <tr><td>Recitation H</td><td>Jonan (jseeley) and Laura (lkoye)</td><td>R  9:30-10:20pm</td><td>GHC 5210</td></tr>
  <tr><td>Recitation I</td><td>Lauren (leheller) and Rhea (rkudtar1)</td><td>R 10:30-11:20am</td><td>GHC 5210</td></tr>
  <tr><td>Recitation D/J</td><td>Frank (frankh) and Mahima S. (mshanwar)</td><td>R 12:30-1:20pm</td><td><b>GHC 5207</b></td></tr>
```

URLs are Website Nicknames

Find www.google.com

At the beginning of the process, you have to make a request to access a specific website.

You generally do this by clicking on a link on a webpage or typing out a **URL** (Uniform Resource Locator). The URL is like a nickname for the website you want to access.

IP Addresses are Real Names

`Find www.google.com`

If a URL is a nickname for a website, an **IP Address** is its real name.

`Google`

`172.217.9.206`

Every computer on the internet is assigned a series of numbers, like `172.217.9.206`. That series of numbers uniquely identifies the computer that hosts a website.

The first step in finding a website is to translate the URL into the equivalent IP Address.

IP Address Assignment

IP Addresses aren't a core part of a computer; they aren't built into the hardware or software. But they aren't entirely random either.

An organization called ICANN (Internet Corporation for Assigned Names and Numbers) assigns groups of addresses to different organizations (like ISPs and companies). The organizations then assign their numbers to individual computers when they connect to the internet.

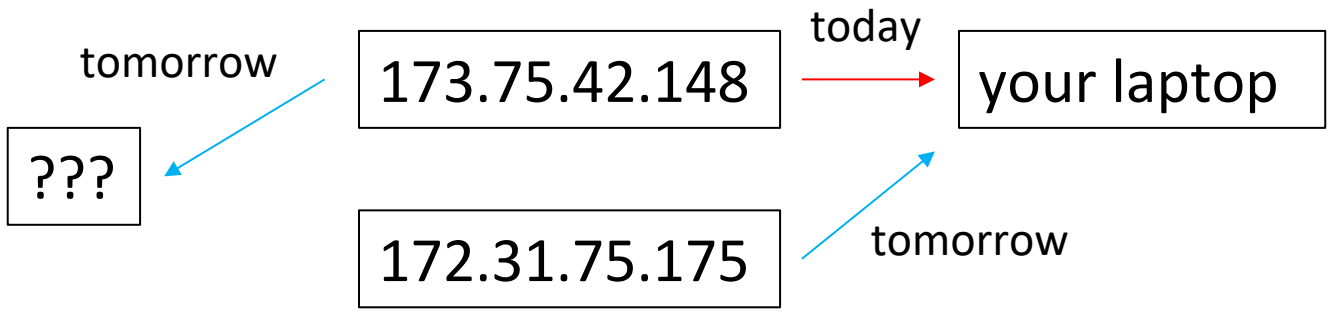
The core standard for IP Addresses consists of four numbers, each between 0-255. In other words, each number is a **byte**. You can look up an IP address to see which organization owns it.

IP Addresses are Static or Dynamic

Some IP Addresses are **static**. Many of these are the addresses of specific websites (like Google, or CMU).

128.2.42.10 -> CMU

Other IP Addresses are **dynamic.** They get assigned to different computers at different times. This is used for computers that go online and offline regularly (like your computer).



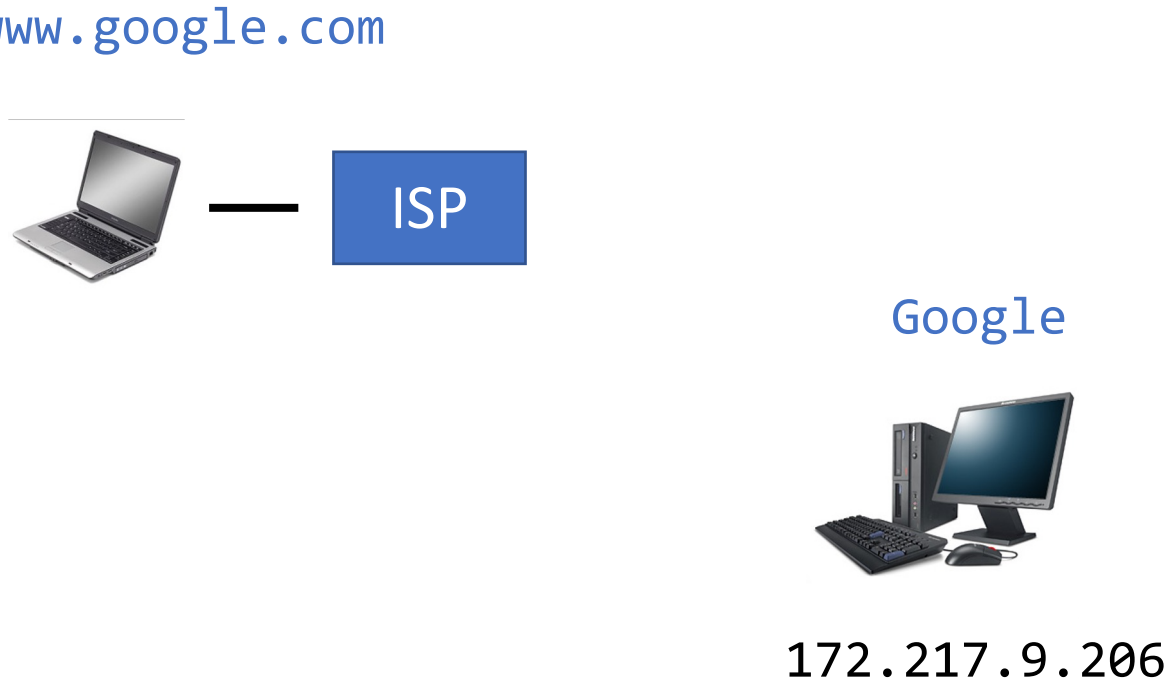
Note that a dynamic IP Address does **not** say who owns the associated computer, or what kind of machine it is. This makes it possible for internet communication to be private to outside observers (though it is not private to the ISP).

Finding IP Addresses

How do we go from a URL to an IP Address?

DNS (**Domain Name System)** is the system to look up a URL. Your computer knows the IP address of a DNS server that it can ask, usually run by your ISP.

Your computer sends a request to that server. It keeps a list of recently-requested websites, so it might be able to send the answer back right away.

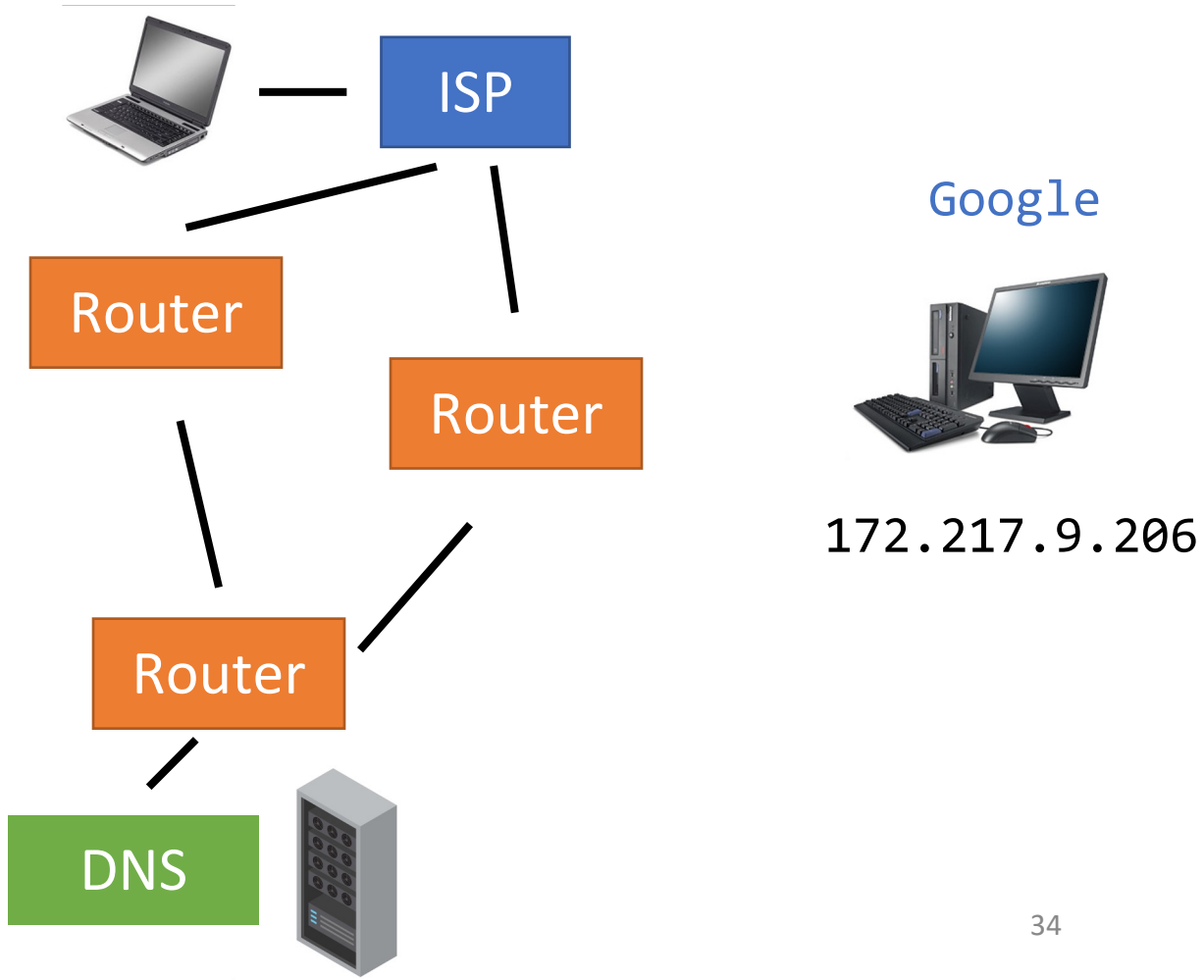


Finding IP Addresses

If your ISP's DNS server doesn't know the IP Address, it sends your request on to another DNS server.

Your request may need to pass through several **routers** to get to a DNS server.

Eventually, your request reaches a DNS server that knows the answer, and that answer is sent back to your computer.

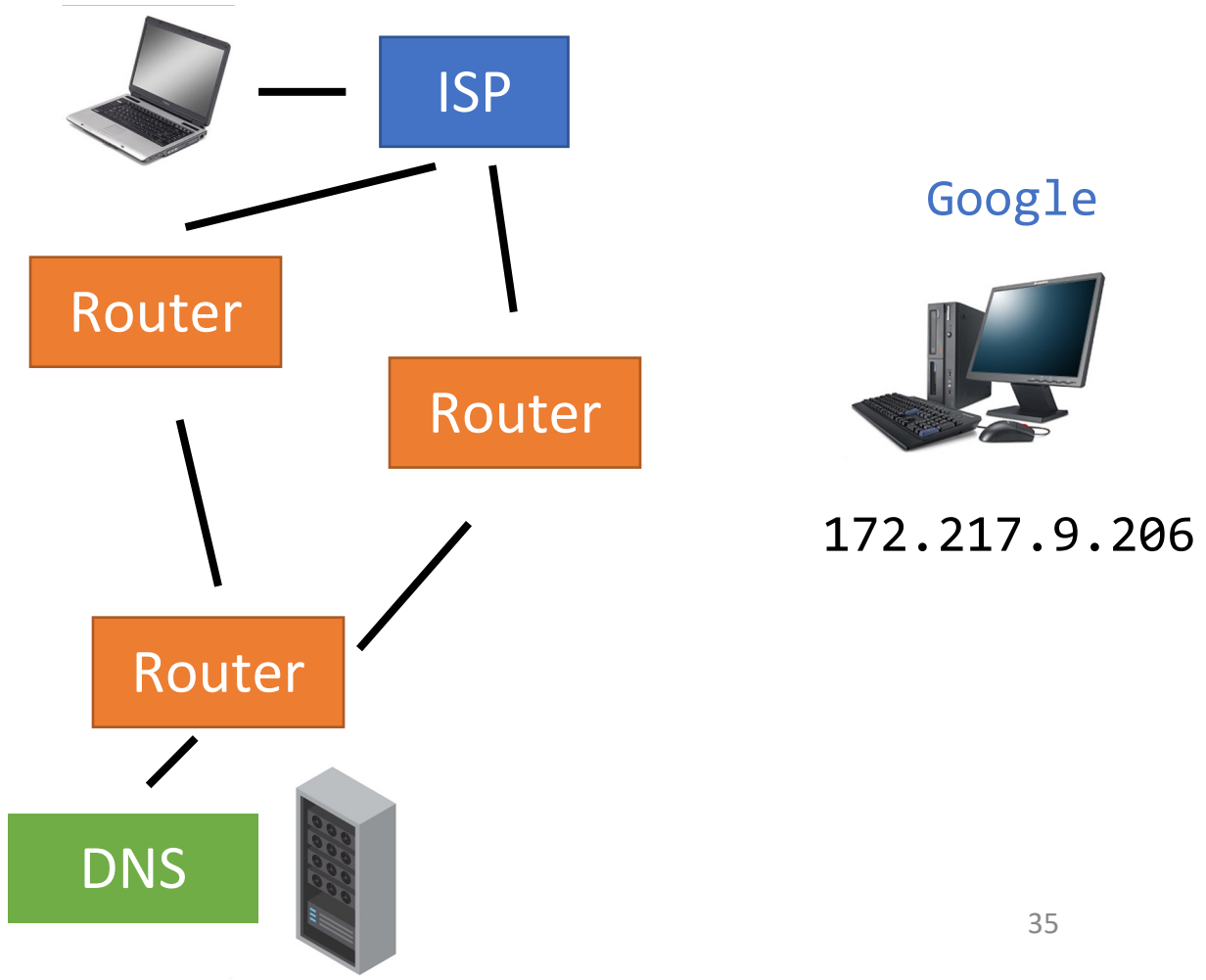


Requesting a Website

Once your computer knows what the IP Address is, it sends a **request** for a specific page to the IP Address.

The request is structured to match a certain **protocol**. For example, HTTP (HyperText Transfer Protocol) is a standard that describes how to request information from a website.

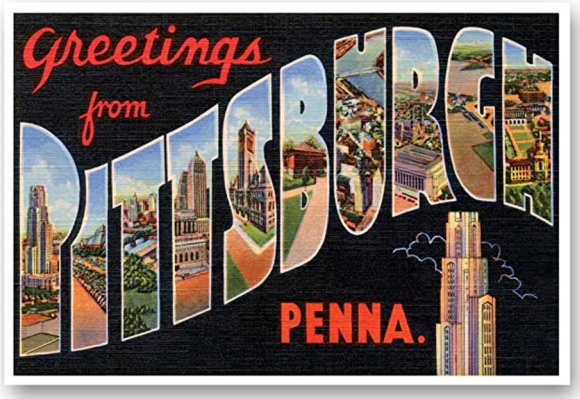
This request is sent using something called a **packet**.

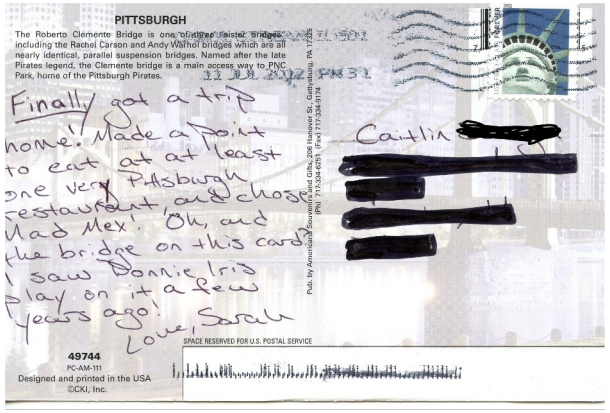


Packets Store Data

A **packet** is a small message that is sent to a particular IP Address.

It's similar to a postcard – it has a message (the data), a destination address (IP address), and a return/sender address (IP address).





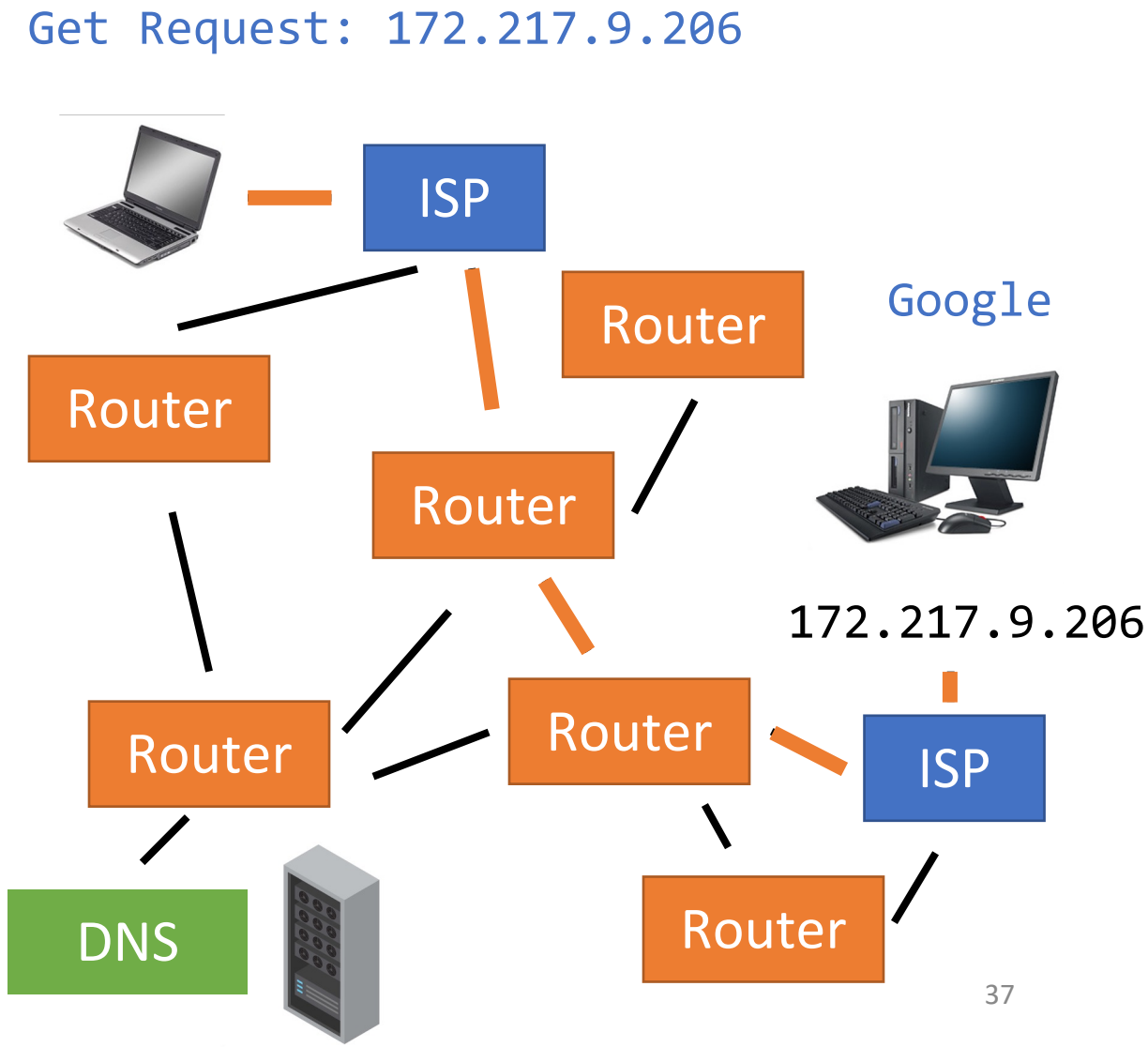
Because a packet is small, it can be sent along a wire very quickly.

A Packet Can Take Many Paths

Sending a packet across the internet is like sending a postcard through the mail.

You don't tell the post office which roads to take; you just tell it the destination, and the post finds a route to get it there.

Similarly, you don't tell the internet which routers to visit; the internet figures it out.

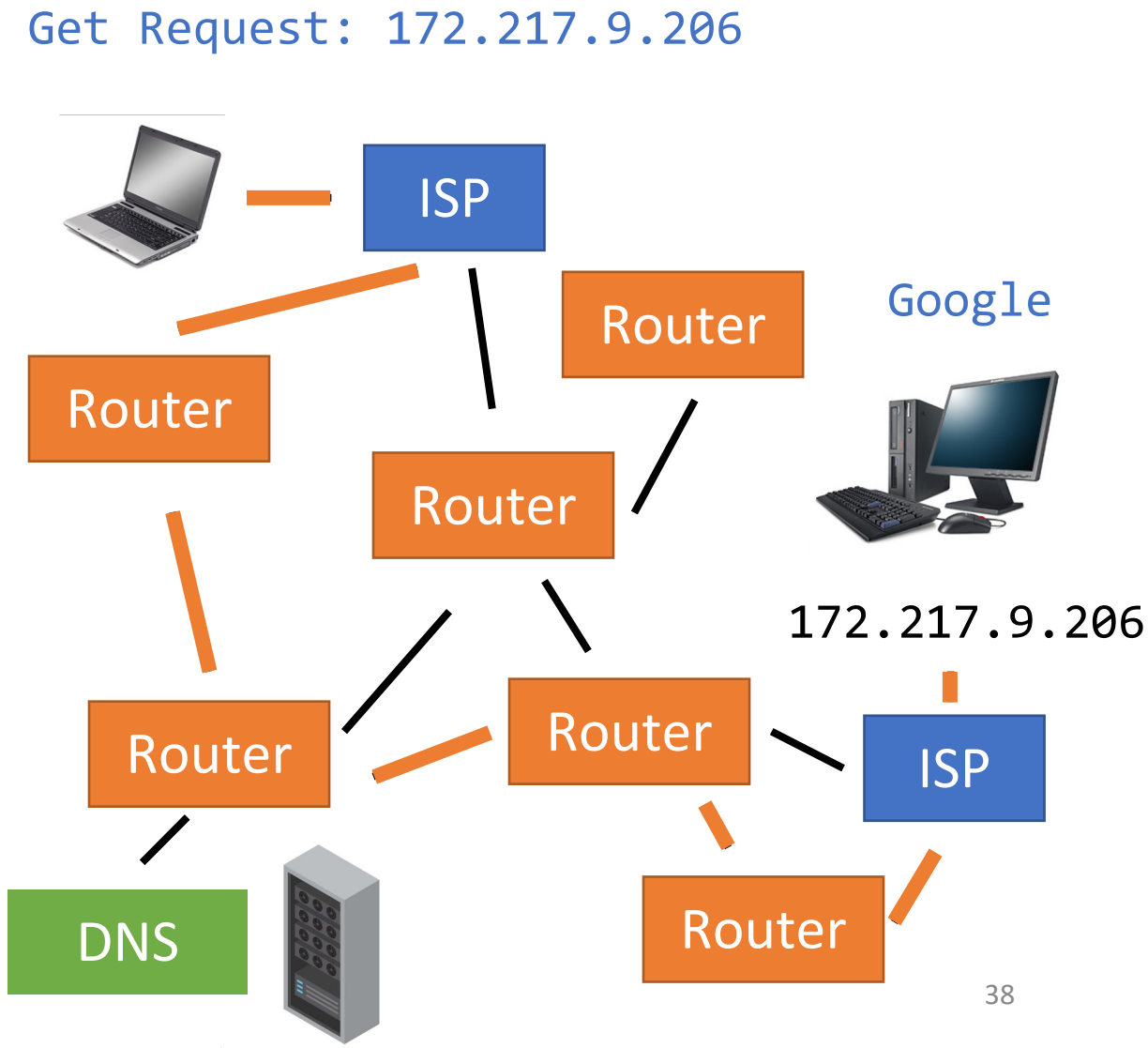


A Packet Can Take Many Paths

Sending a packet across the internet is like sending a postcard through the mail.

You don't tell the post office which roads to take; you just tell it the destination, and the post finds a route to get it there.

Similarly, you don't tell the internet which routers to visit; the internet figures it out.

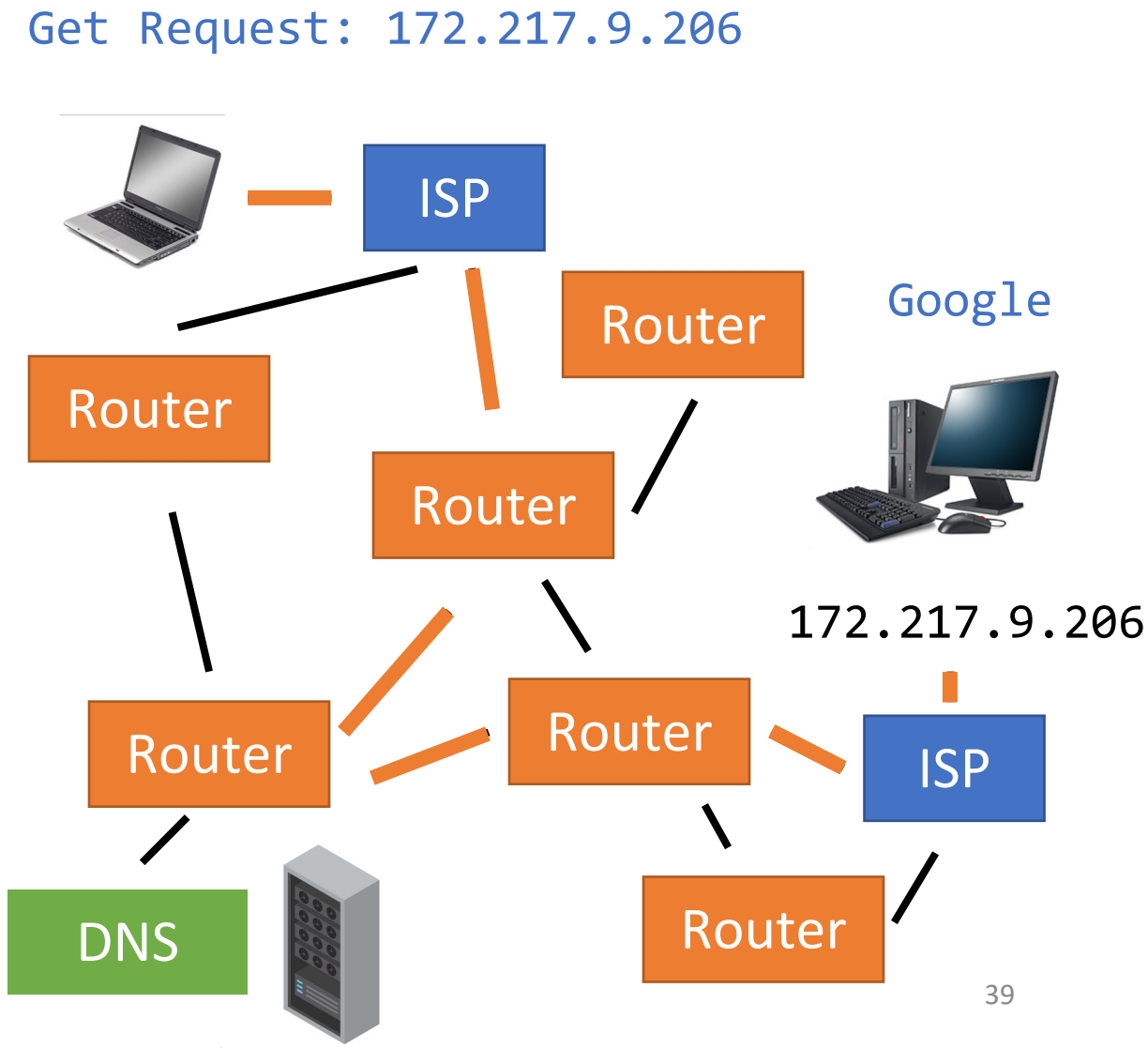


A Packet Can Take Many Paths

Sending a packet across the internet is like sending a postcard through the mail.

You don't tell the post office which roads to take; you just tell it the destination, and the post finds a route to get it there.

Similarly, you don't tell the internet which routers to visit; the internet figures it out.

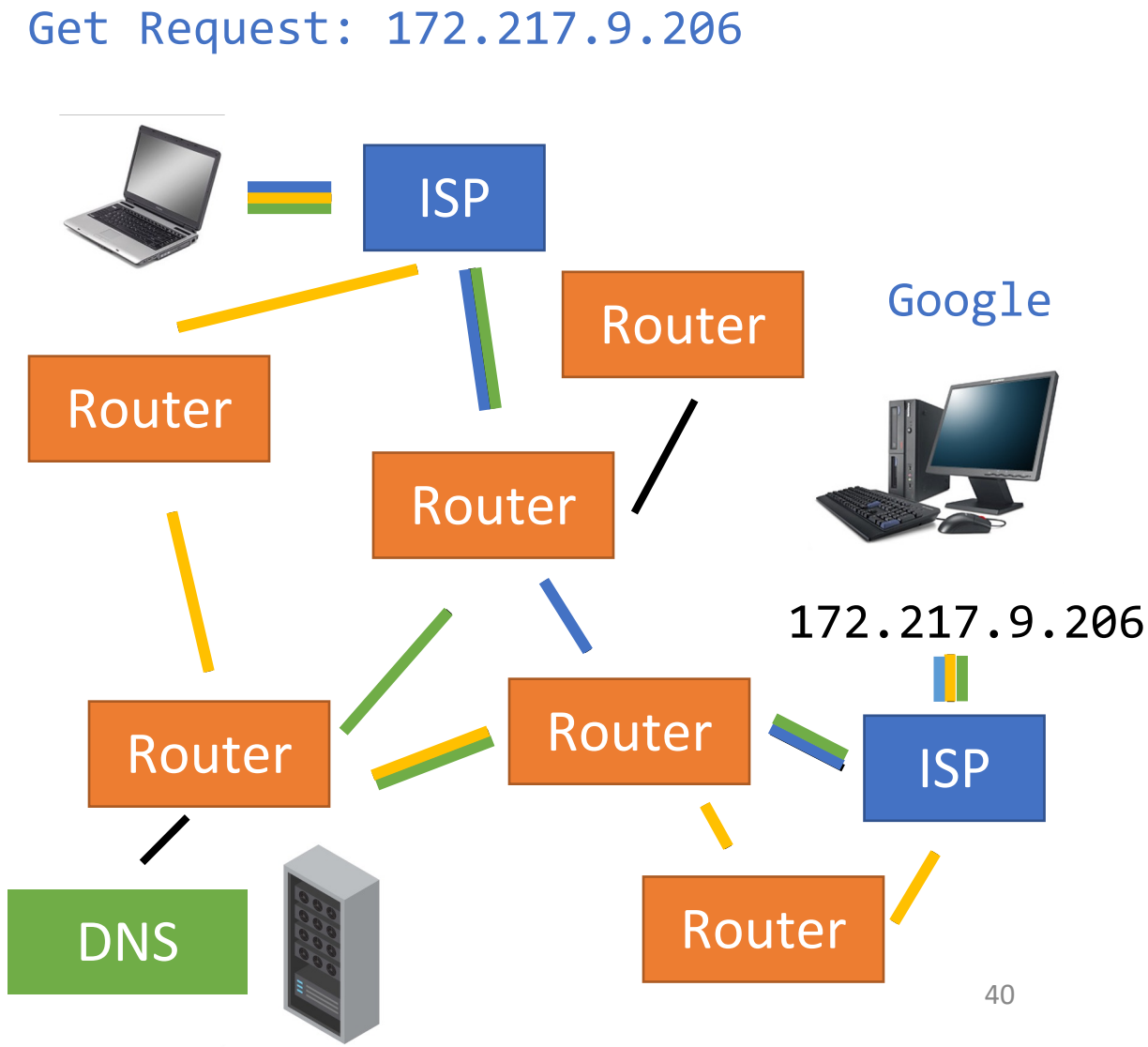


Webpages are also Packets

When the website gets your request, it might need to send back a response (like a webpage).

Since webpages are generally large, the page is **split into multiple packets** and the packets are sent back through the routers to your computer.

When all the packets get to your computer, the browser assembles them to produce the HTML of a website. It's like putting together a jigsaw puzzle.



Demo: Classroom Internet

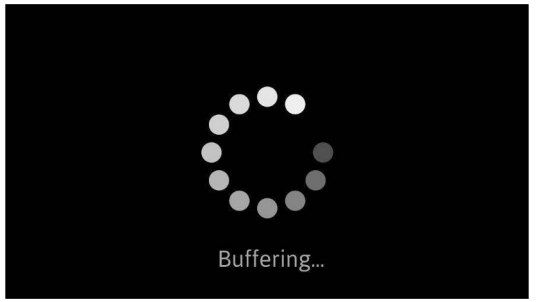
Let's model the process of requesting a website via the internet with our classroom!

The **instructor** is the person requesting a specific website.

Three other people will serve as the **ISP's DNS server**, a **DNS server with the answer**, and the **requested website's server**.

Everyone else will be a **router**! As a router, you can accept individual packets (index cards) from other routers and pass individual packets along to other routers as well.

Sidebar: Buffering



Some webpages need a **lot** of packets. For example, a video takes a lot of data to render. Packets may take a long time to reach the browser, which can cause lag.

Your browser uses **buffering** to show you part of a website while the rest of it loads. Buffering occurs when the browser receives enough of the early packets to pre-load the initial content onto your computer. While you read or watch the content, the browser silently loads the rest of the content as it arrives.

If a buffer pauses for a long time, your browser is probably waiting for a few packets that are still missing.

[if time] Discuss: Internet Culture

We have shown that the internet was designed to be **decentralized**, with no single router that can control the whole thing.

Do you think that's still true? If yes, how has this has affected the way internet culture has evolved? If no, what changed?

Learning Goals

- Recognize and define the following keywords: **distributed computing, cloud computing, browsers, routers, ISPs, IP addresses, DNS servers, protocols,** and **packets**.

- Use the **MapReduce pattern** to design parallelized algorithms for distributed computing

- Understand at a high level the **internet communication process** that happens when you click on a link to a website in your browser.