# Distributed Computing and the Internet

15-110 – Monday 10/28

# 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, 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** executes algorithms over multiple computers.

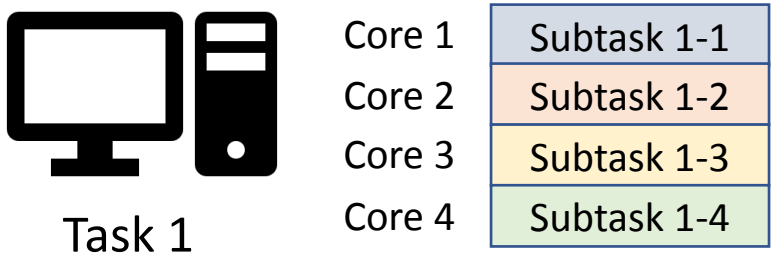
Distributed Computing is a concurrency approach where you **network multiple computers (each with its own set of CPUs) together** and use them all to perform advanced computations. This can be done by assigning different processes to different computers.

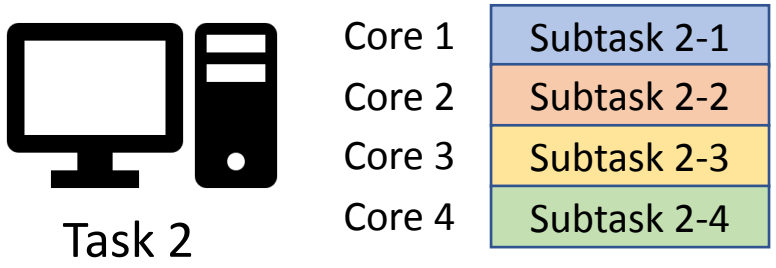
The more computers you have available to network together, the **faster your algorithm can* be**!

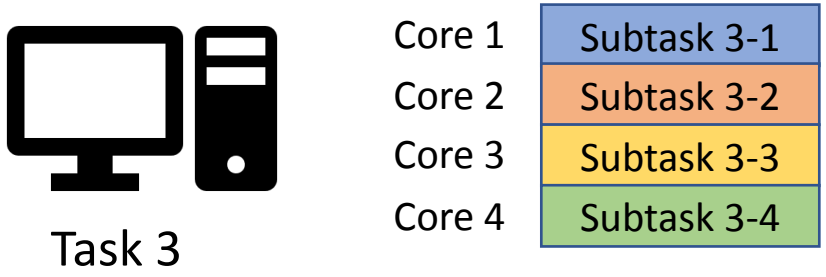
*However: Designing algorithms to get the most out of distributed computing is even harder than designing algorithms for multiprocessing.

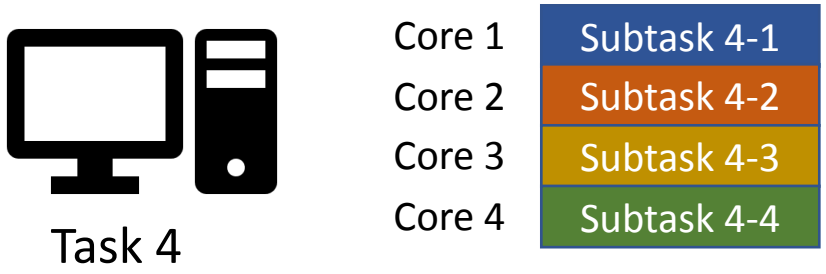
# Distributed computing **scheduling requires breaking up tasks** to work across multiple computer.

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.



# Datacenters 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 **'datacenter'** 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* (ex: 164,000) number of processors connected together. The main difference is that all the processors are located in the same place.

**MapReduce** is a popular framework for implementing distributed algorithms across computers.

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 is also a great approach for any problem over **big data** – giant data sets that are too large to process using typical software on one machine.

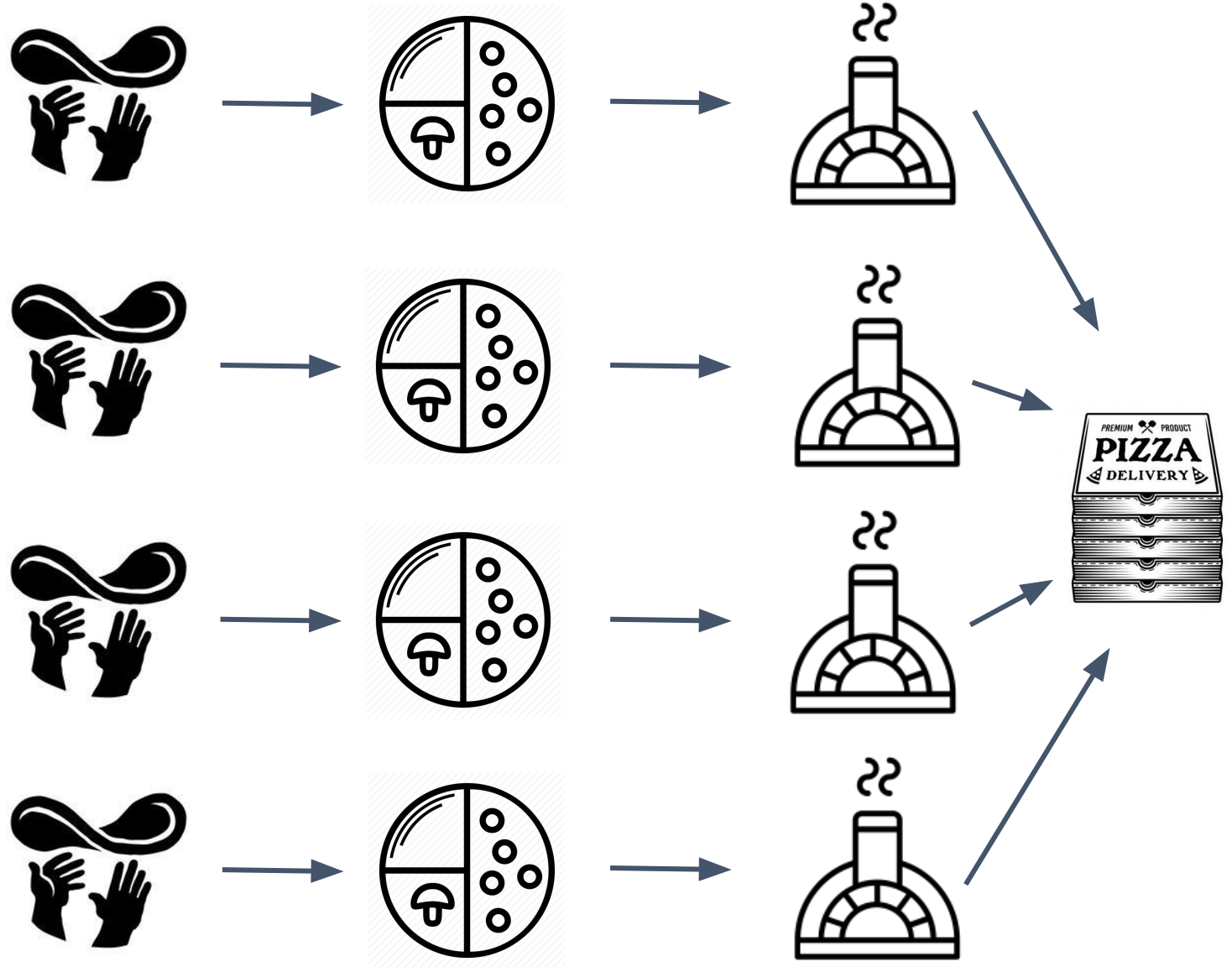
# MapReduce - 4 workers, 4 ovens, 3 steps

Worker 1:

Worker 2:

Worker 3:

Worker 4:




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!

A MapReduce algorithm is composed of **three parts**:

**mapper:** takes a piece of data, processes it, and finds a partial result

**reducer:** takes a set of results and combines them together

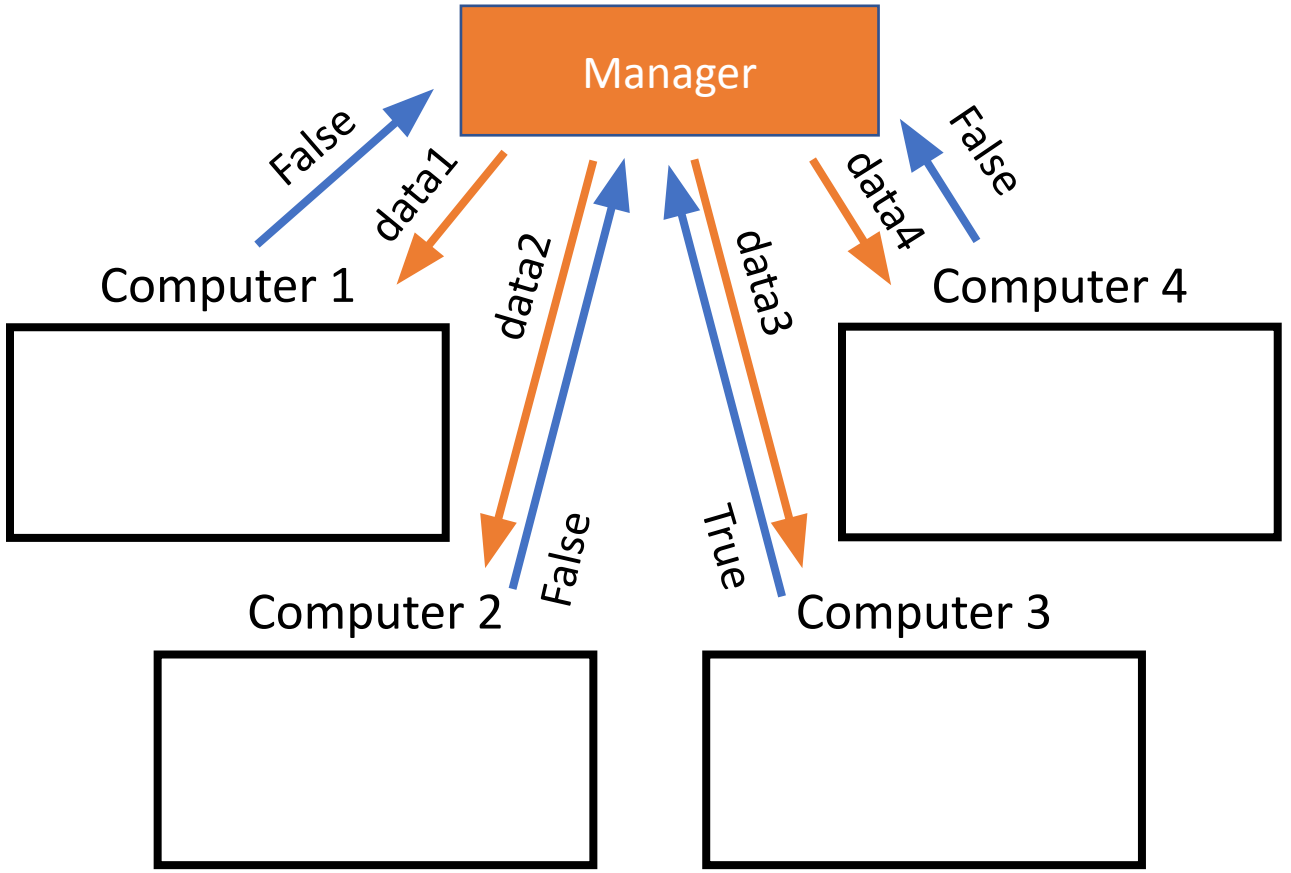
**manager:** moves data through the process and outputs the final result

1. Splits up data, sends to mappers, get results back
2. Combines results together, sends to the reducer
3. Gets the final result, outputs it

# MapReduce Example: **Searching** a book for a specific word.

How can we **split up this task**?

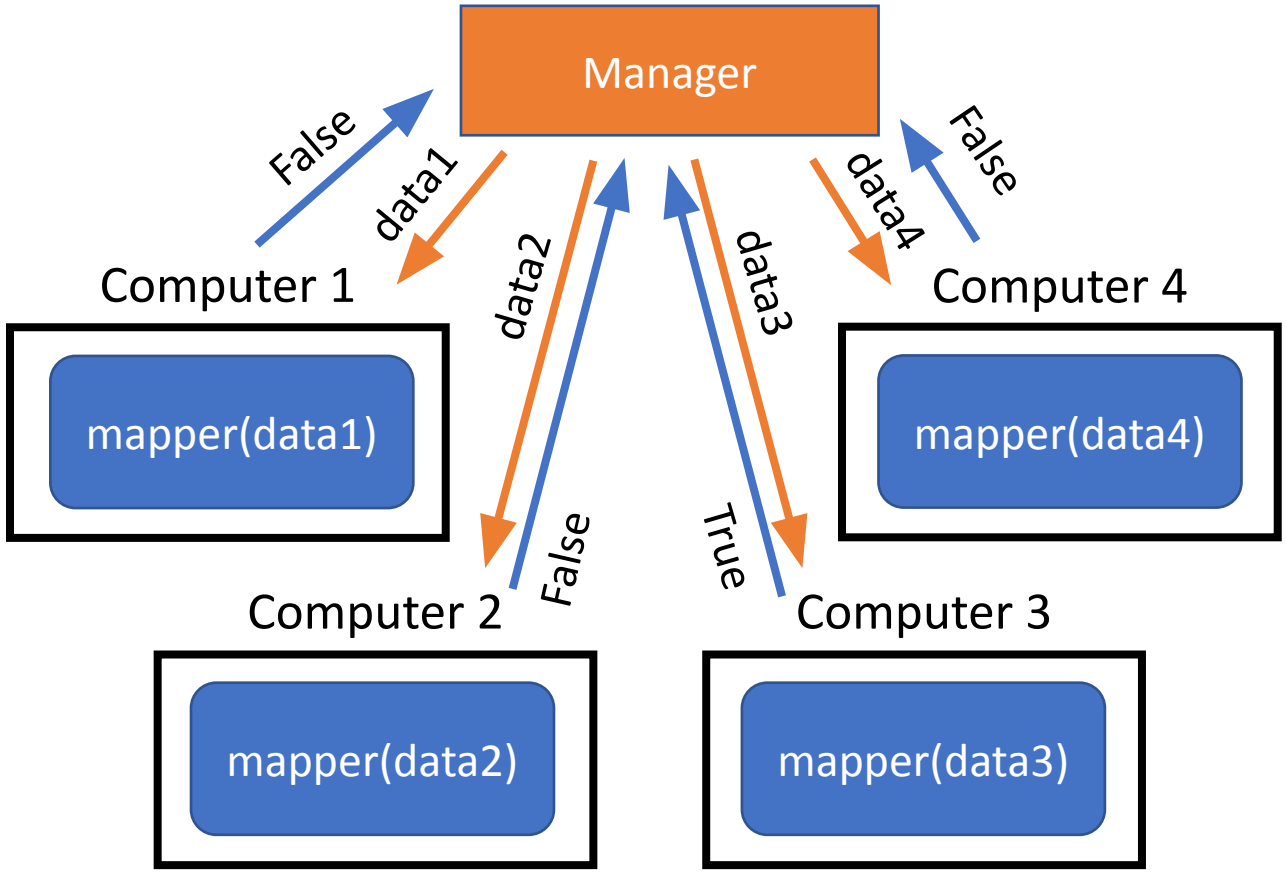
1. The **manager** divides the book into many small parts- example: one page per part. It sends each page to a different computer.



# MapReduce Example: **Searching** a book for a specific word.

How can we **split up this task**?

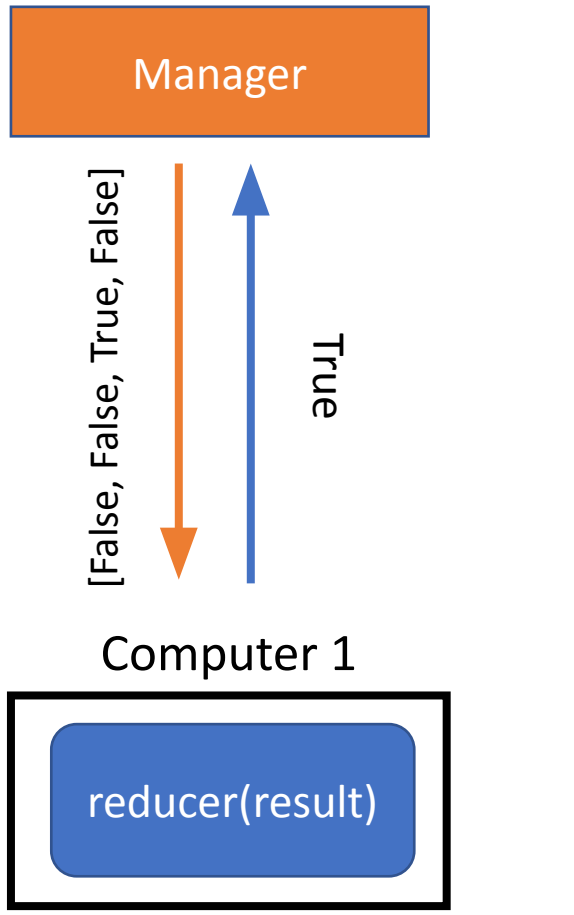
1. The **manager** divides the book into many small parts- example: one page per part. It sends each page to a different computer.
2. Each computer runs its copy of the **mapper** on its page. It returns `True` if it finds the result, and `False` otherwise. These results are sent back to the **manager**.



# MapReduce Example: **Searching** a book for a specific word.

How can we **combine the results**?

1. Once all the mappers have returned their results the **manager** puts them all in a list and sends that list to the **reducer(s)**.
2. Our reducer will check all of the results and send `True` back to the **manager** if any of them are `True`.




Note: Depending on the problem, there can be more than one reducer.

MapReduce can process huge data sets and **get results quickly**.

MapReduce 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.

Another example: Finding the **most visited website** (max) based on 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 of dictionaries.

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!**

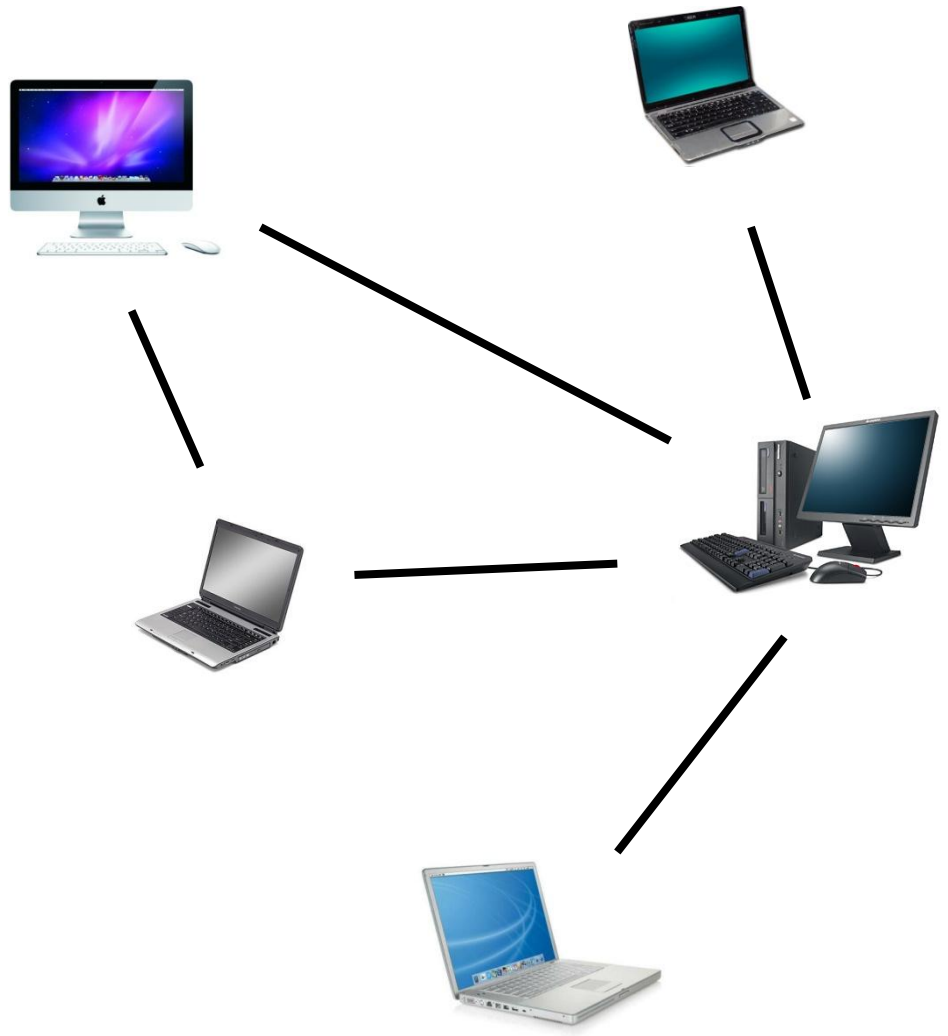
# The Internet

# The Internet is a **network of computer networks** all across the world that are connected.

A **computer network** is a group of connected networks that can send data to each other (distributed computing!).

The Internet is **decentralized**: computers can send data between different computers without one person/machine having control over the whole thing.

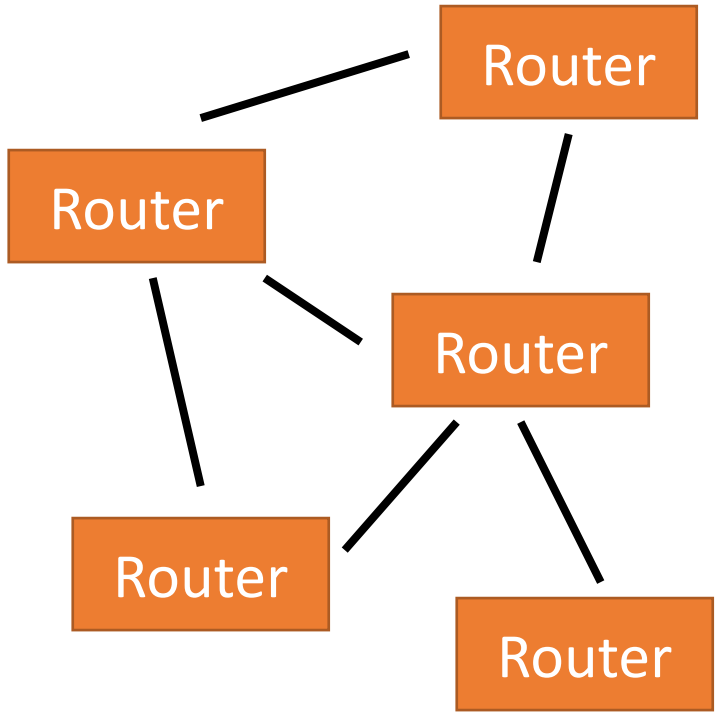
It is like a **graph** where the **nodes** are computers and the **edges** are different methods of transmitting information.



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

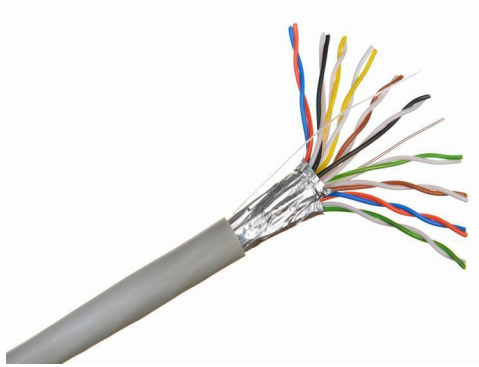
Routers receive data and **send it to one of many possible locations** based on the end destination of the data and the current connections on the internet (routes).

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



Routers are commonly **connected by cables**, which are used to send data as bits over long distances.

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.

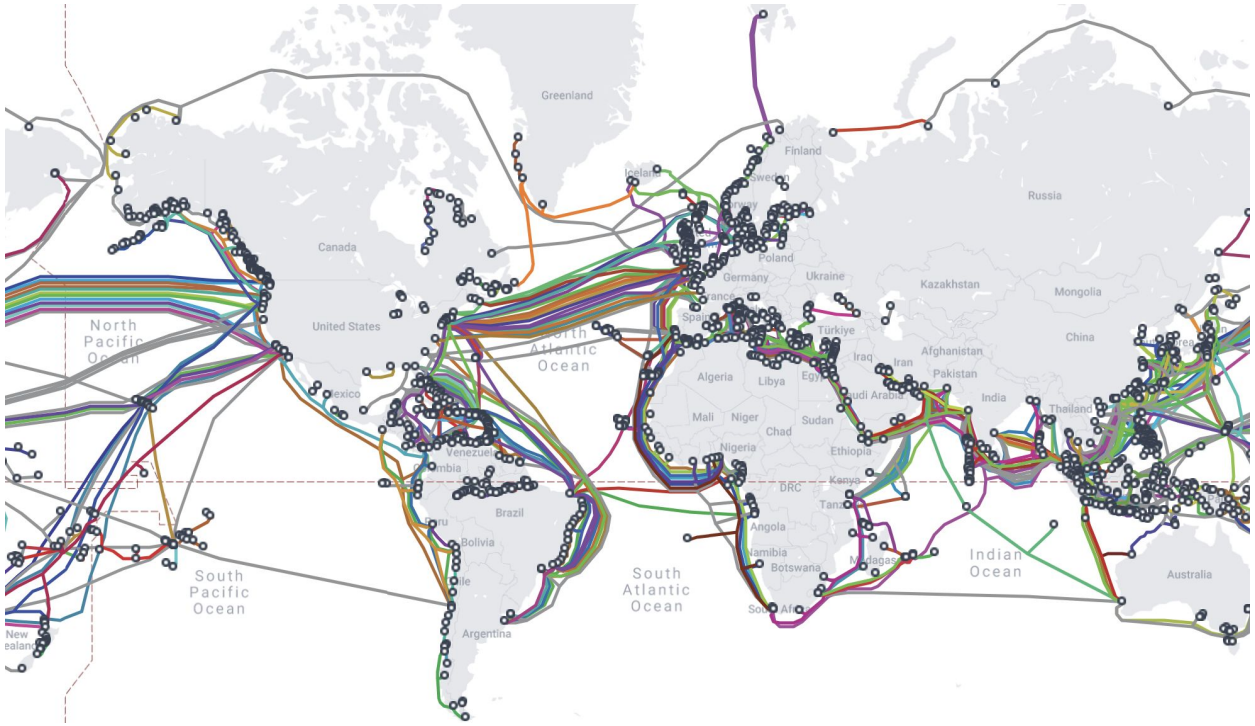
# Sidebar: International Internet

How does the internet connect across continents?

**Giant fiberoptic cables are 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

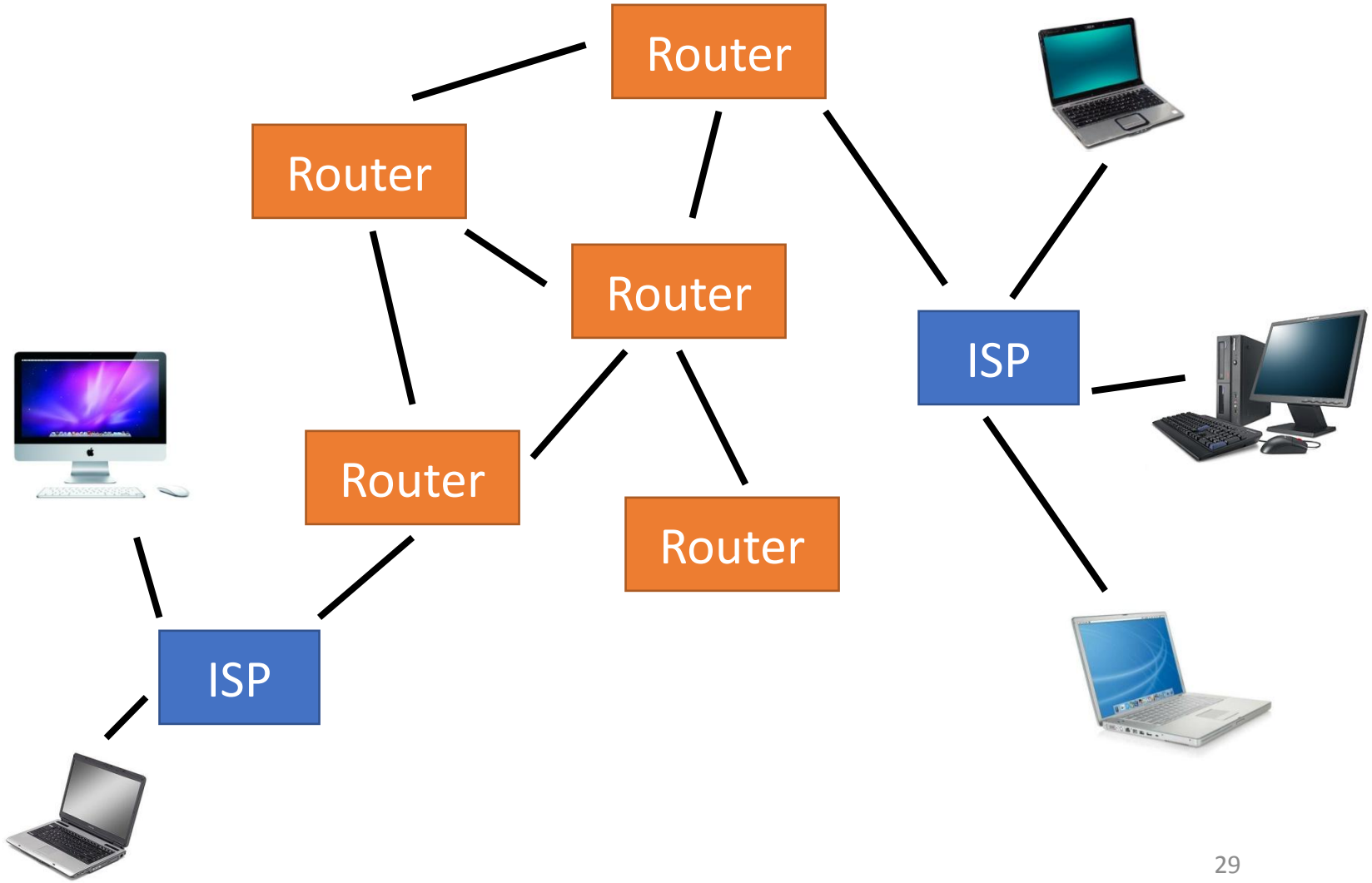



Current submarine cable map https://www.submarinecablemap.com/

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

Verizon, Comcast, etc. are all ISPs.

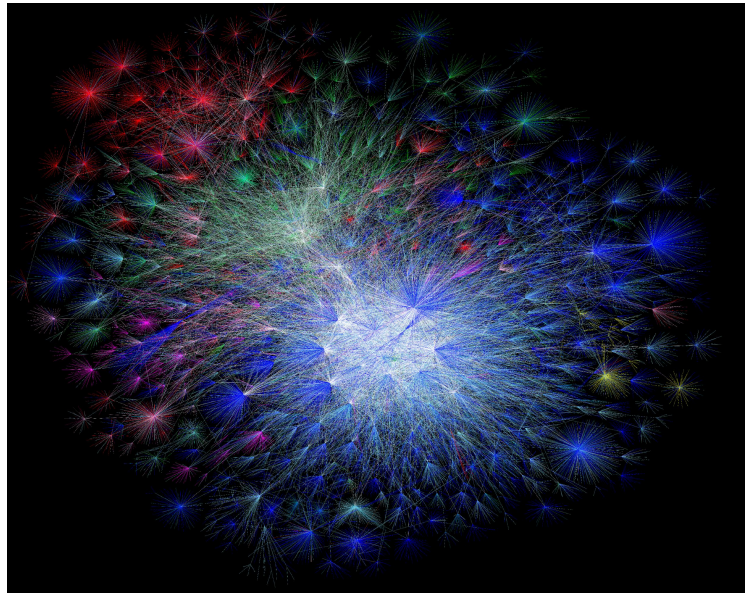
Some ISPs, called "backbone ISPs", manage the routers at the core of the internet.



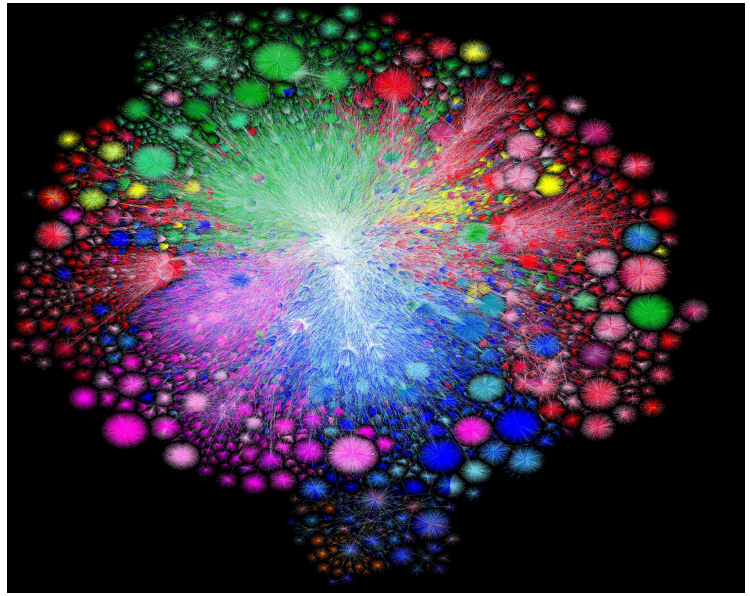
The Internet today is used widely across the world and contains millions of computers and connections.

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

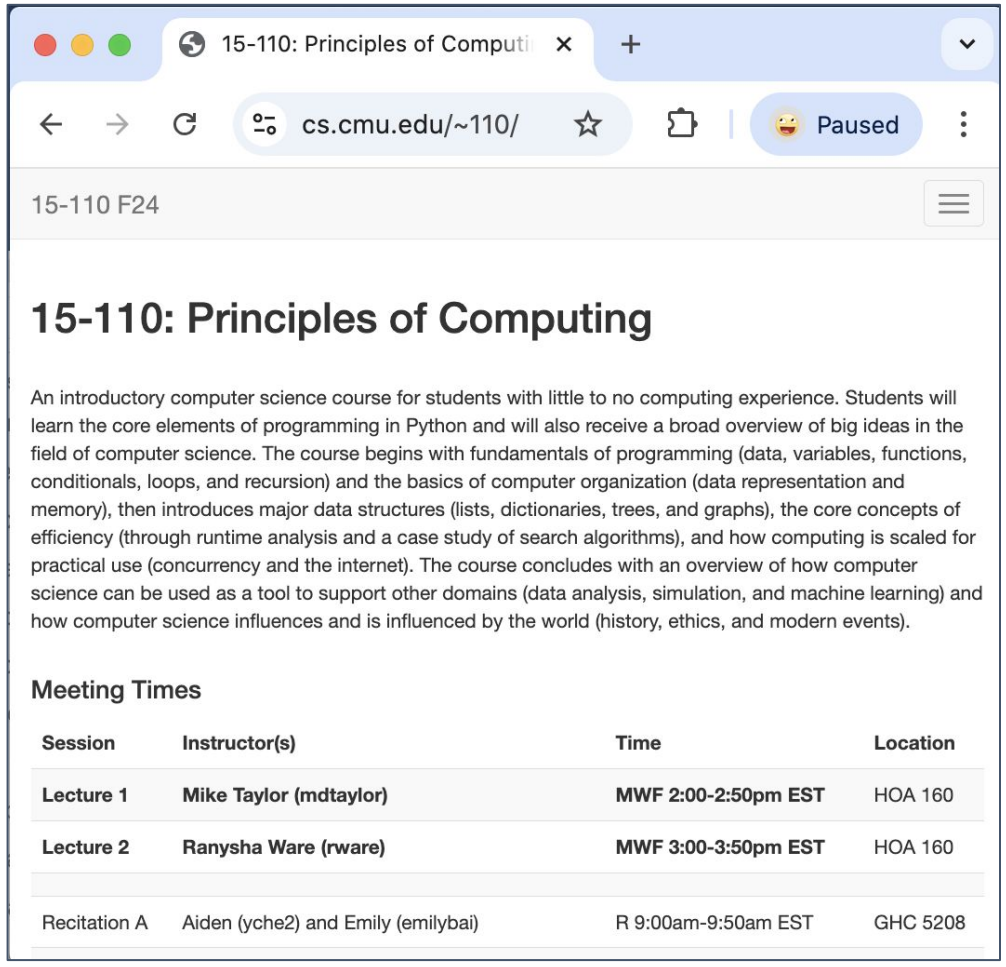
Data is sent across the Internet using **packets** and **protocols**.

Data is broken up into smaller pieces called **packets.** Packets include data and information about the data like its source address and destination address.

**Protocols** are standards for doing certain actions and formatting data so that different devices are able to communicate with and understand each other.

# Example: What happens when you type cs.cmu.edu/~110/ into your browser and hit enter?

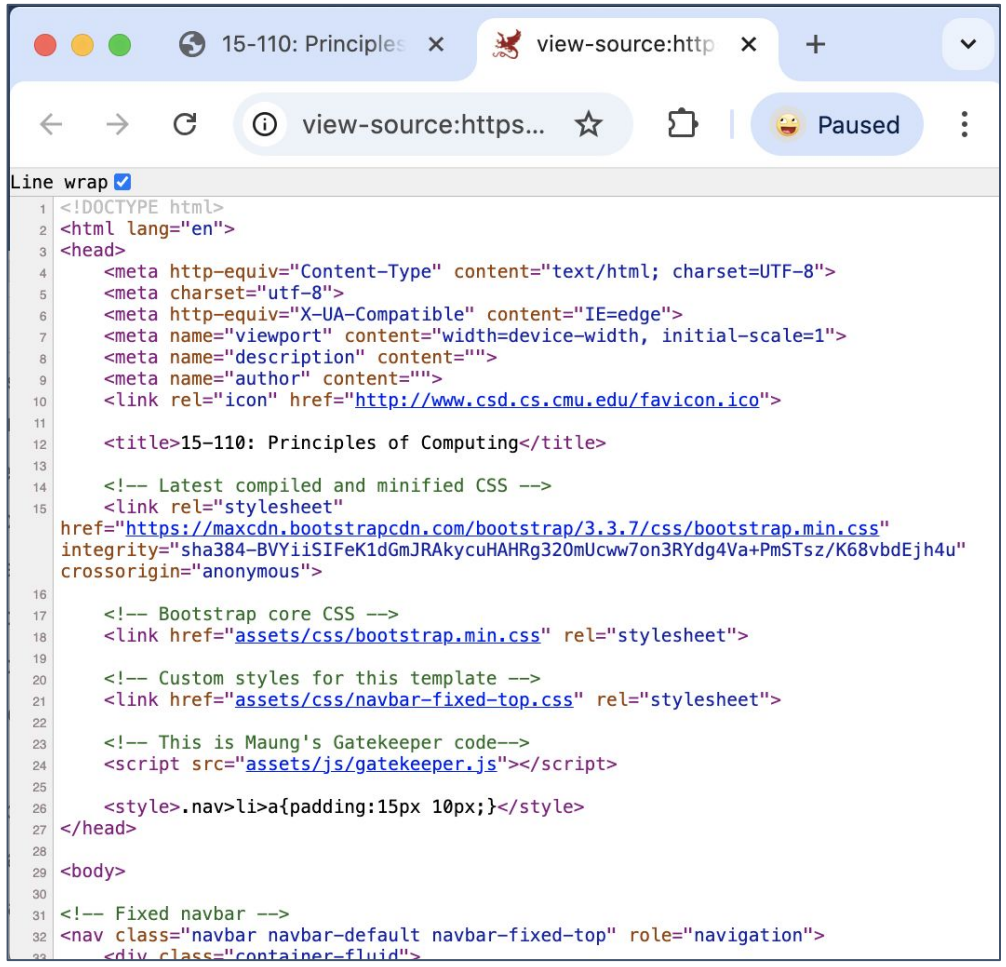
A **browser** (Firefox, Chrome, Safari, etc.) is an application that receives data from the internet and organizes it into a webpage that you can read.



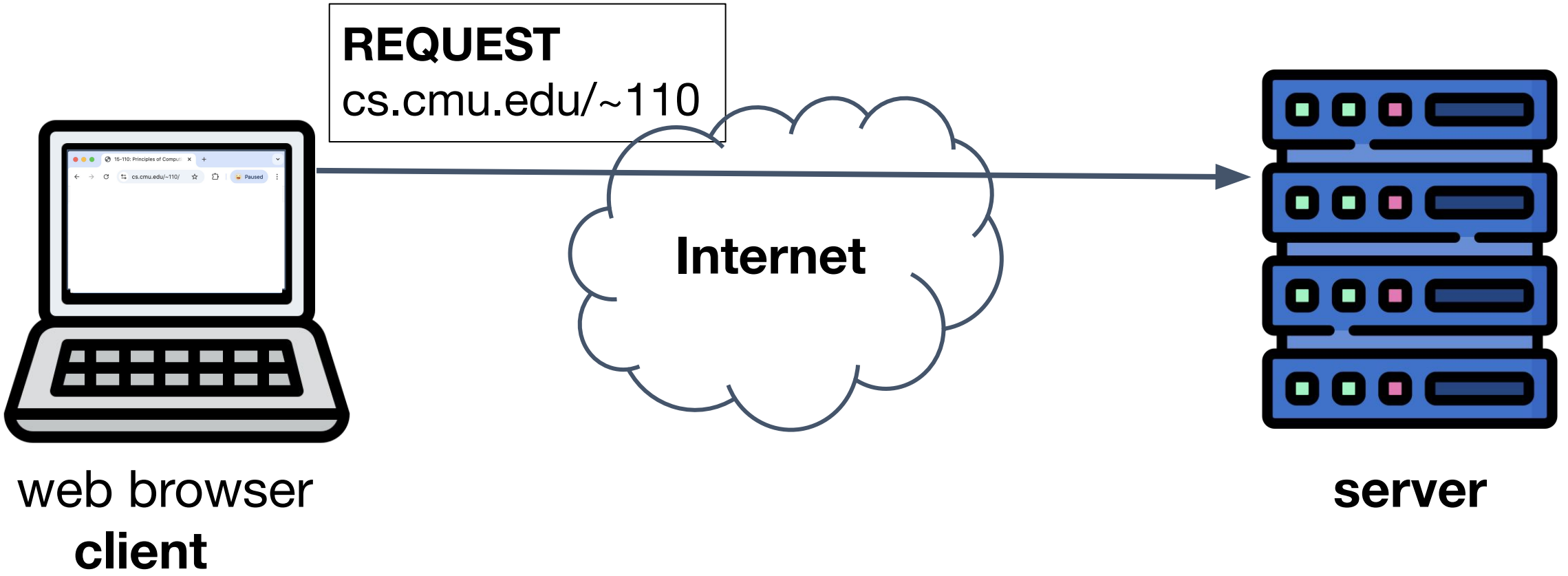
Example: What happens when you type cs.cmu.edu/~110/ into your browser and hit enter?

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

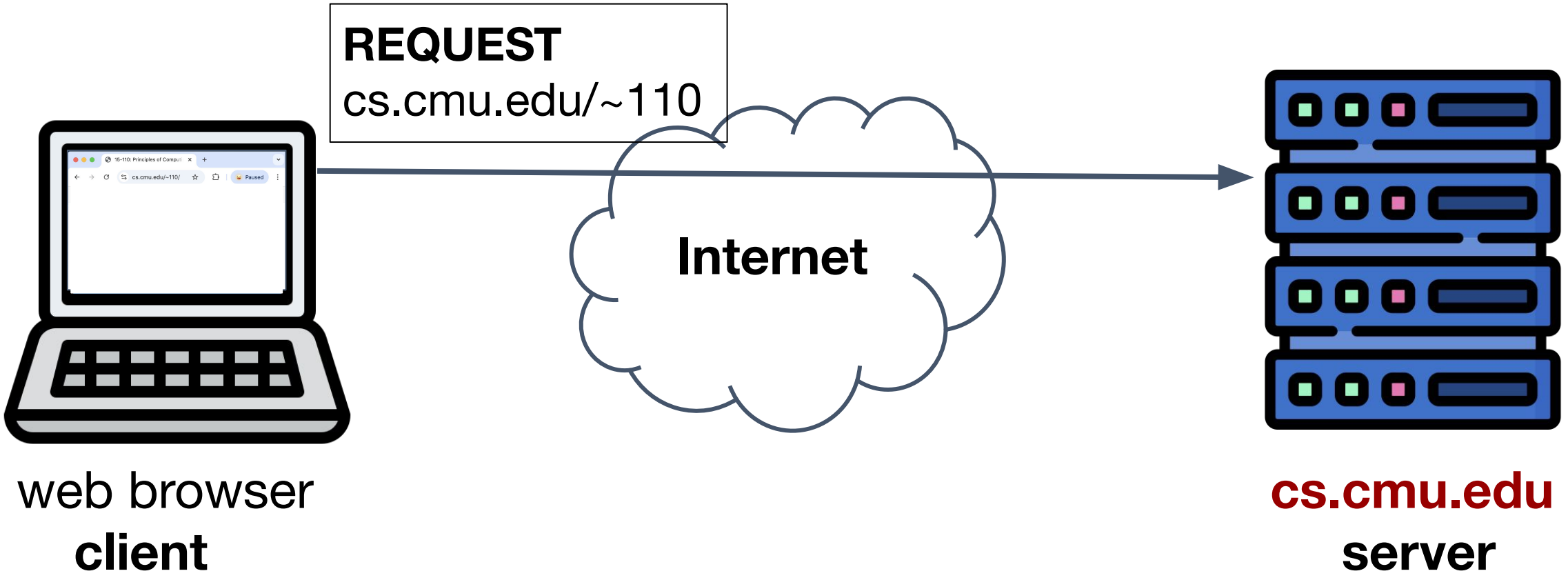
You can view the HTML of any webpage by right-clicking and selecting 'View Page Source'.



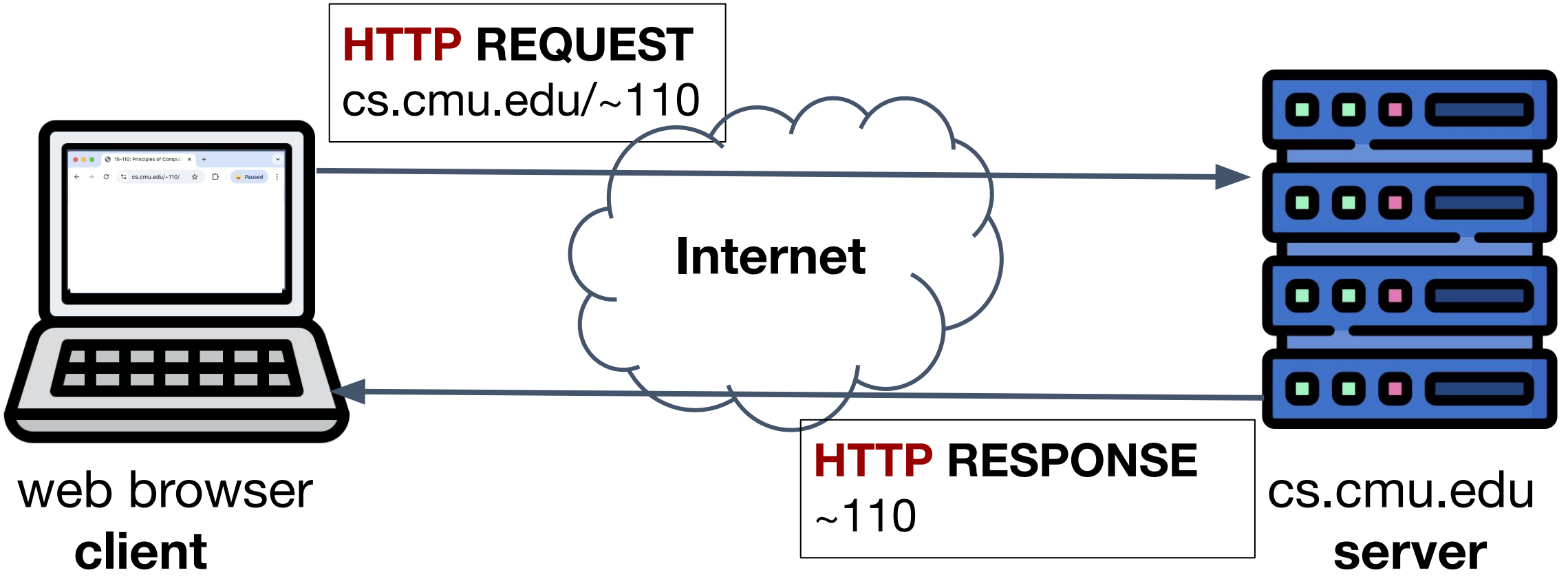
To get the webpage data, our computer needs to **request it from the web server** storing that data.



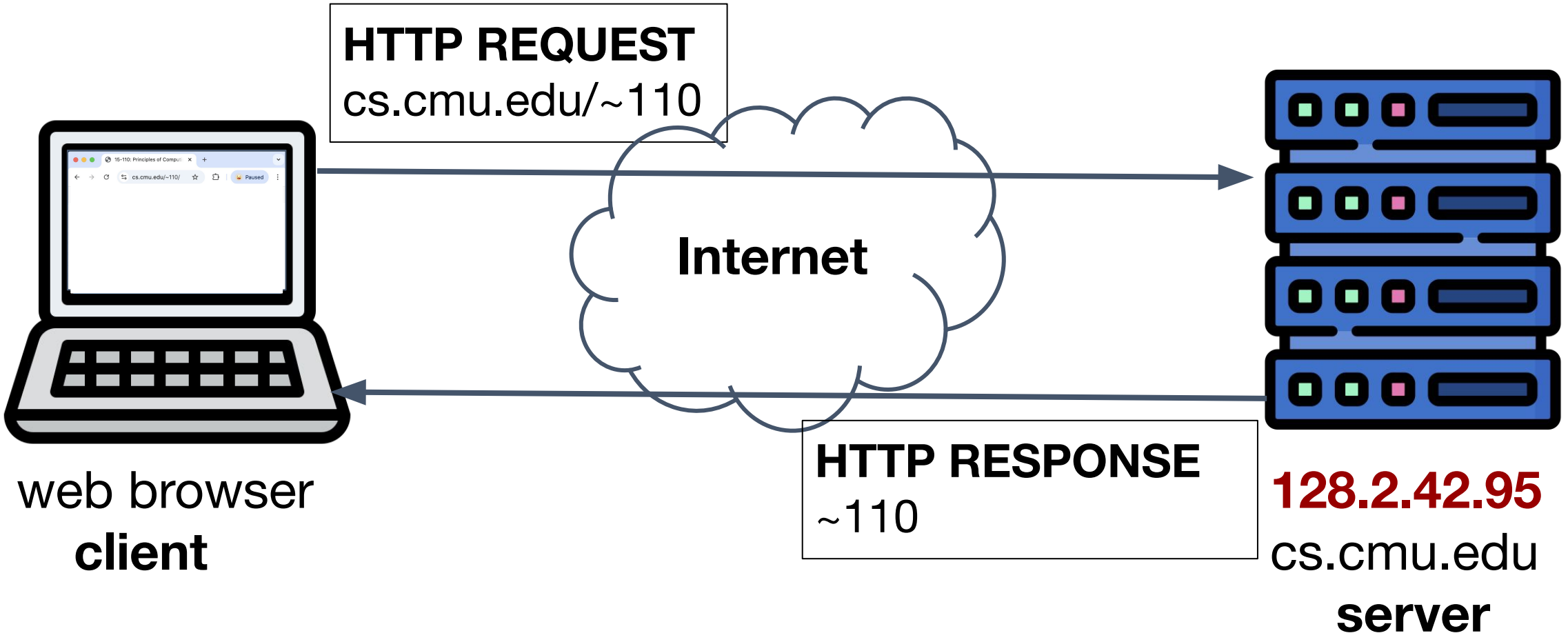
# A **URL** is a nickname for a web server.



The webpage request to a web server is structured to match **HTTP (HyperText Transfer Protocol)**, a standard that describes how to request information from a website.



To know where to send the request, the client needs the web servers' **IP address**, the unique numerical name for an Internet-connected device.



The **IP (Internet Protocol)** manages routing data between computers based on IP addresses.

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.

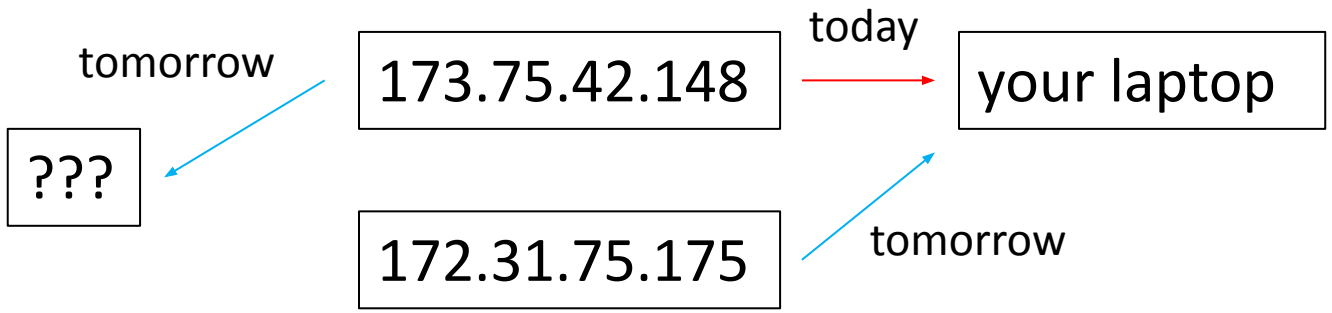
The IPv4 standard consists of four numbers, each between 0-255. In other words, each number is a **byte**. You can look up an IP address to see which organization owns.

# 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

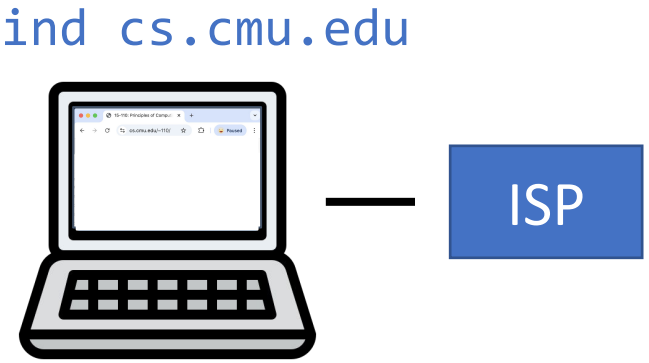
Other IP Addresses are **dynamic.** They get assigned to different computers at different times. This is used for computers that go online and offline regularly (like your computer).

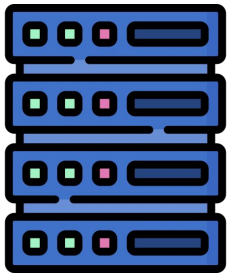


To find the web server's IP address, the client needs to convert the URL to an IP address using **Domain Name System (DNS)**.

DNS is the **phone book of the Internet**.

There are special local **DNS servers**, typically at your ISP, that store mappings from recently-requested names to IP addresses.



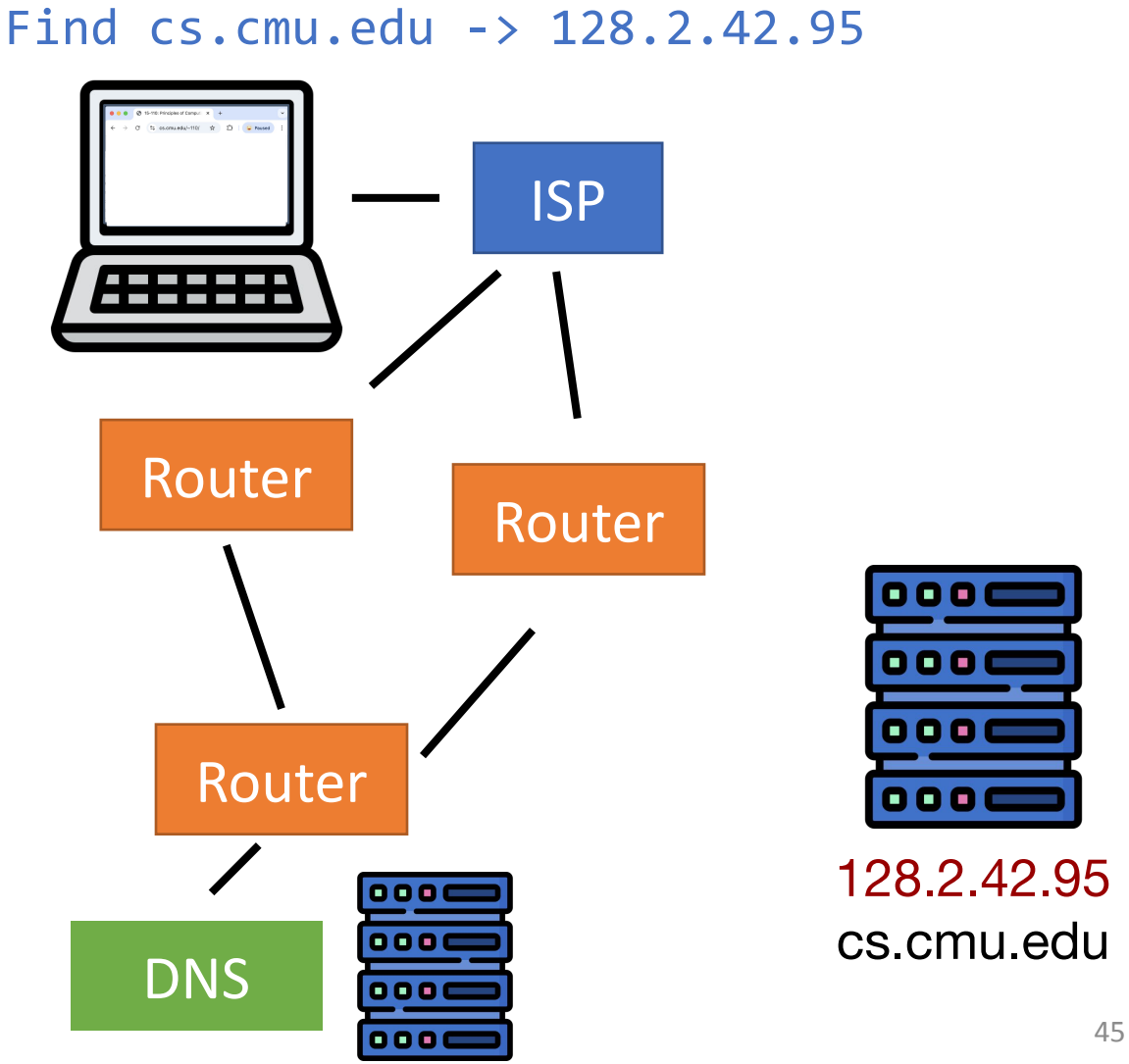
128.2.42.95
cs.cmu.edu

To find the web server's IP address, the client needs to convert the URL to an IP address using **Domain Name System (DNS)**.

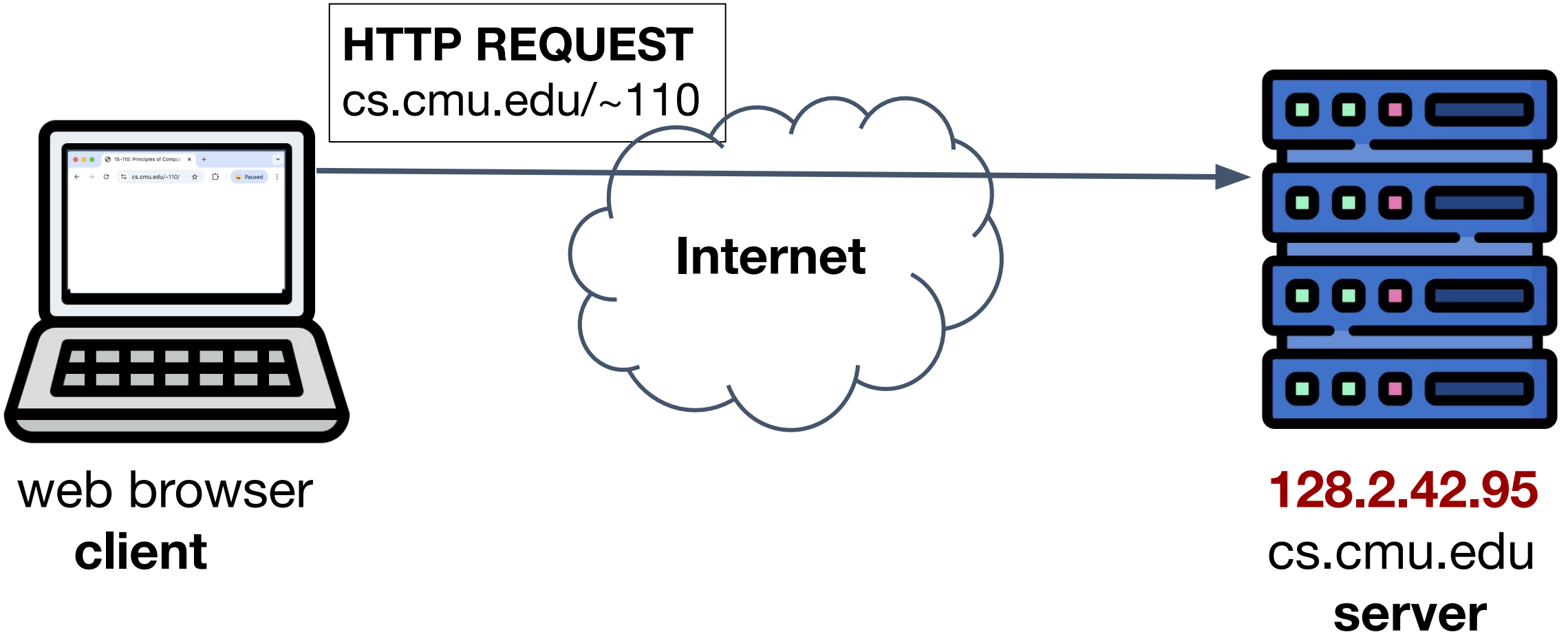
If your ISP's local DNS server doesn't know the IP Address, **it sends your request on to another DNS server**.

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

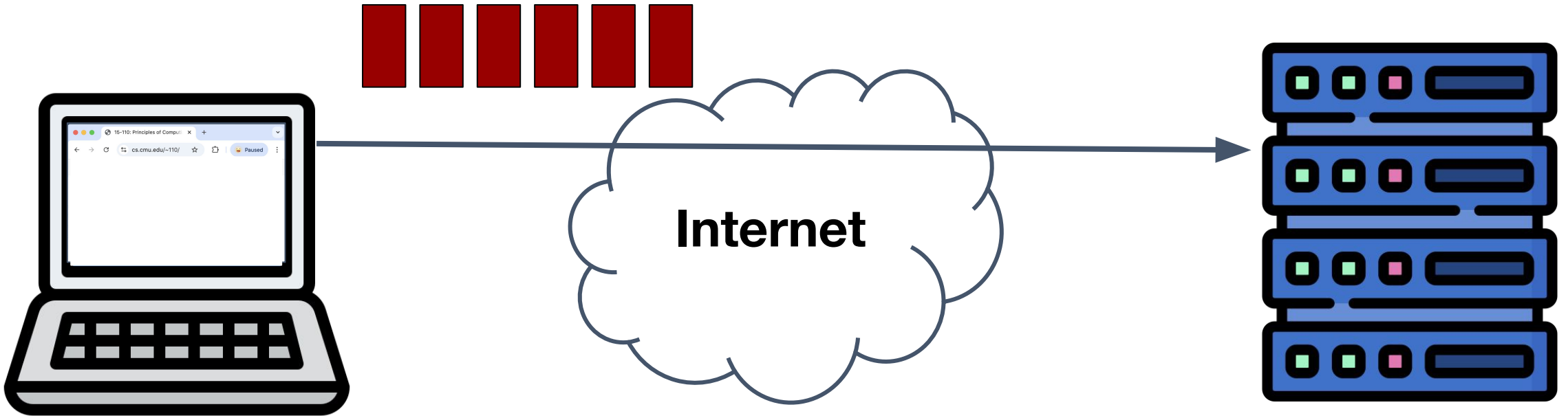
Eventually, **your request reaches a DNS server that knows** the answer, and that answer is sent back to your computer.



Once the client knows the server's IP address, it will send an HTTP Request. The request is **broken into packets** and each packet is separately sent to the address.



Once the client knows the server's IP address, it will send an HTTP Request. The request is **broken into packets** and each packet is separately sent to the address.

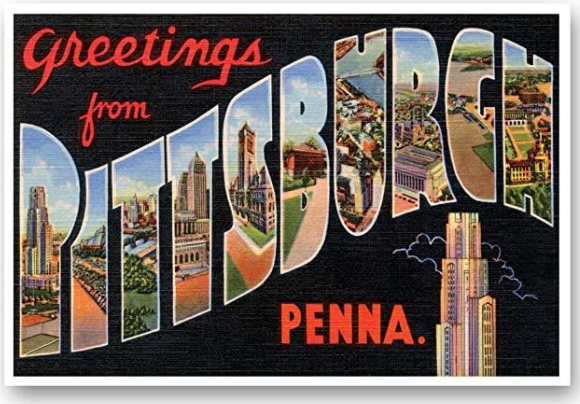


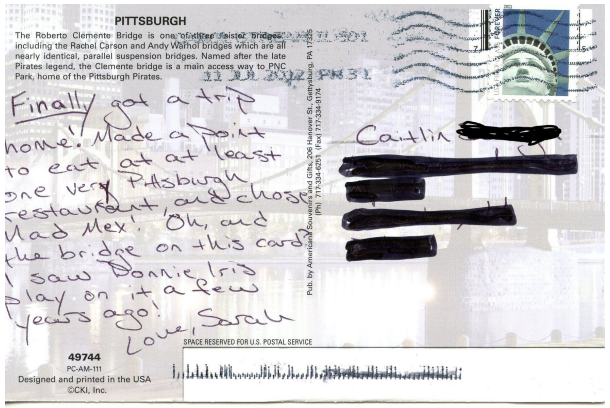

web browser
**client**

**128.2.42.95**
cs.cmu.edu
**server**

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

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



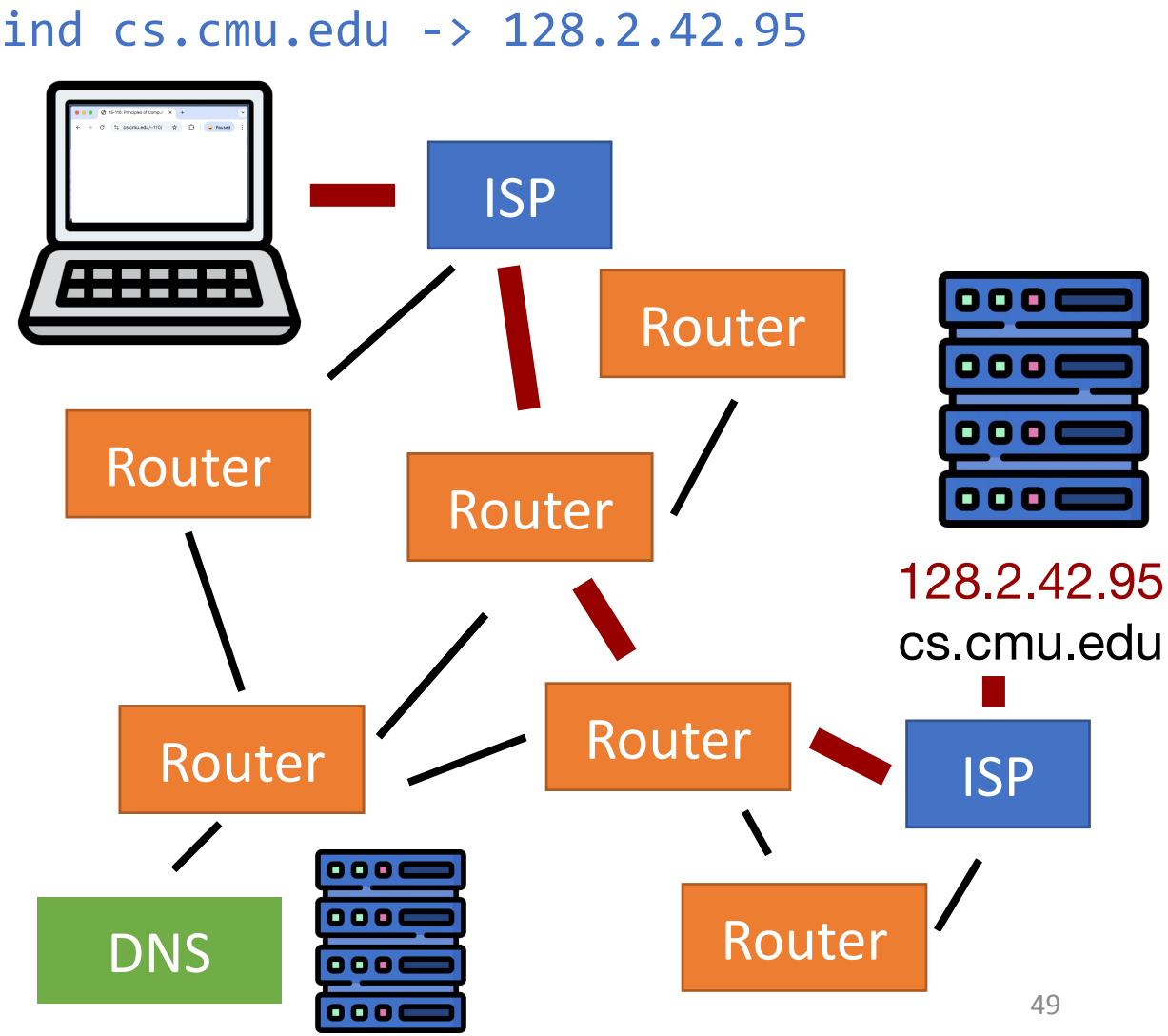

Because a packet is small, it can be sent along a wire very quickly.

A packet can take **many possible paths** from a source to a destination.

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.

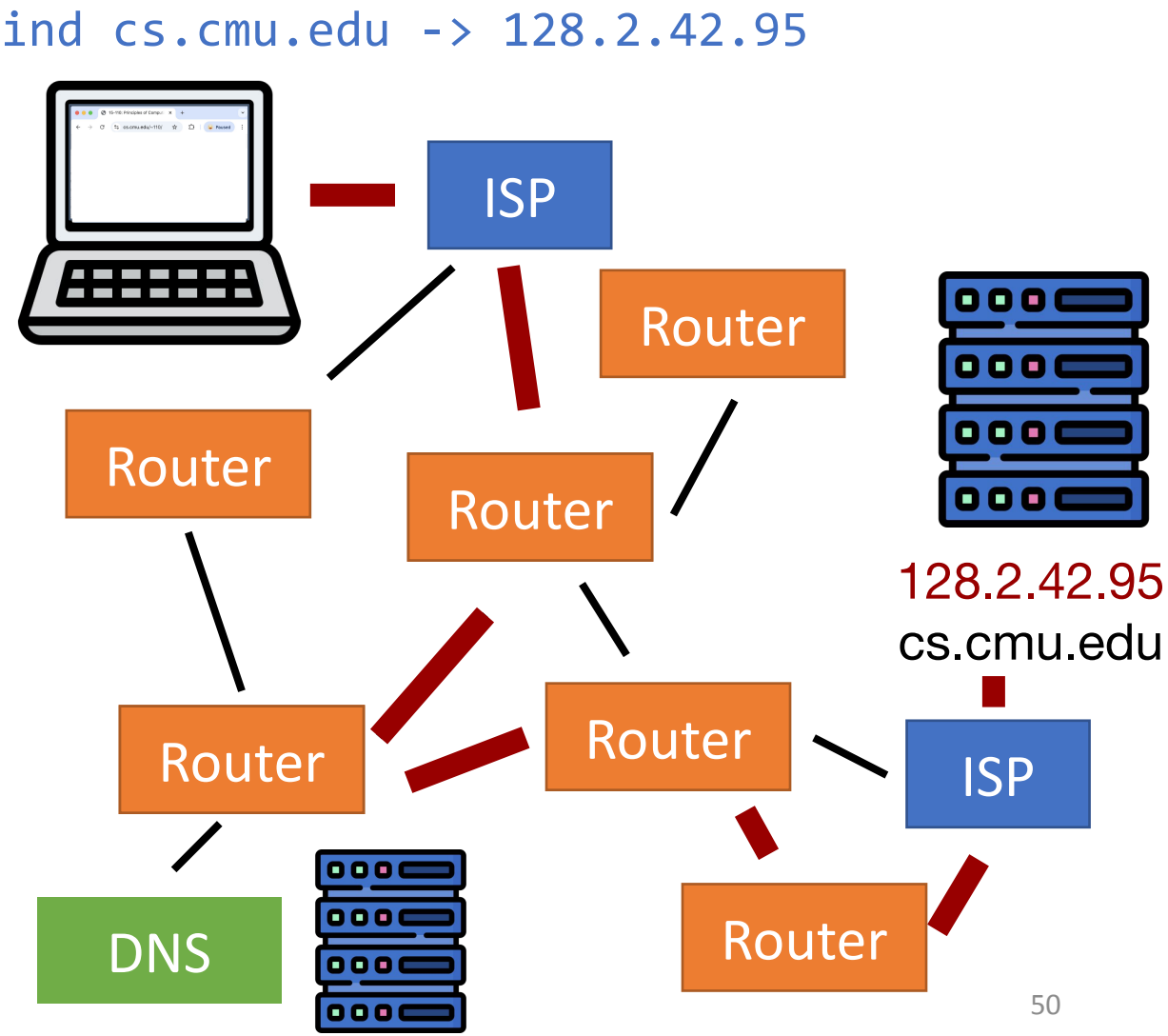
Similarly, you don't tell the internet which routers to visit; the internet figures it out.



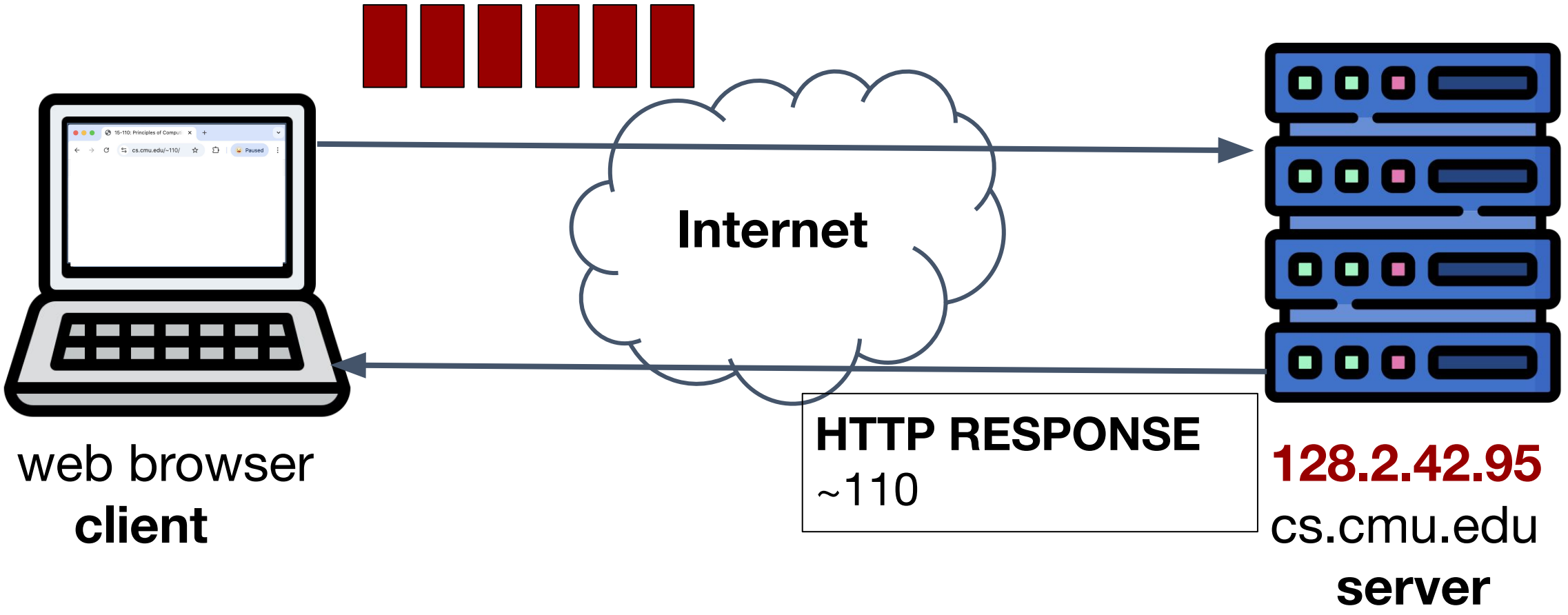
A packet can take **many possible paths** from a source to a destination.

It is not actually always what is the shortest path, there are also business agreements between different organizations that also determine where packets will be routed!

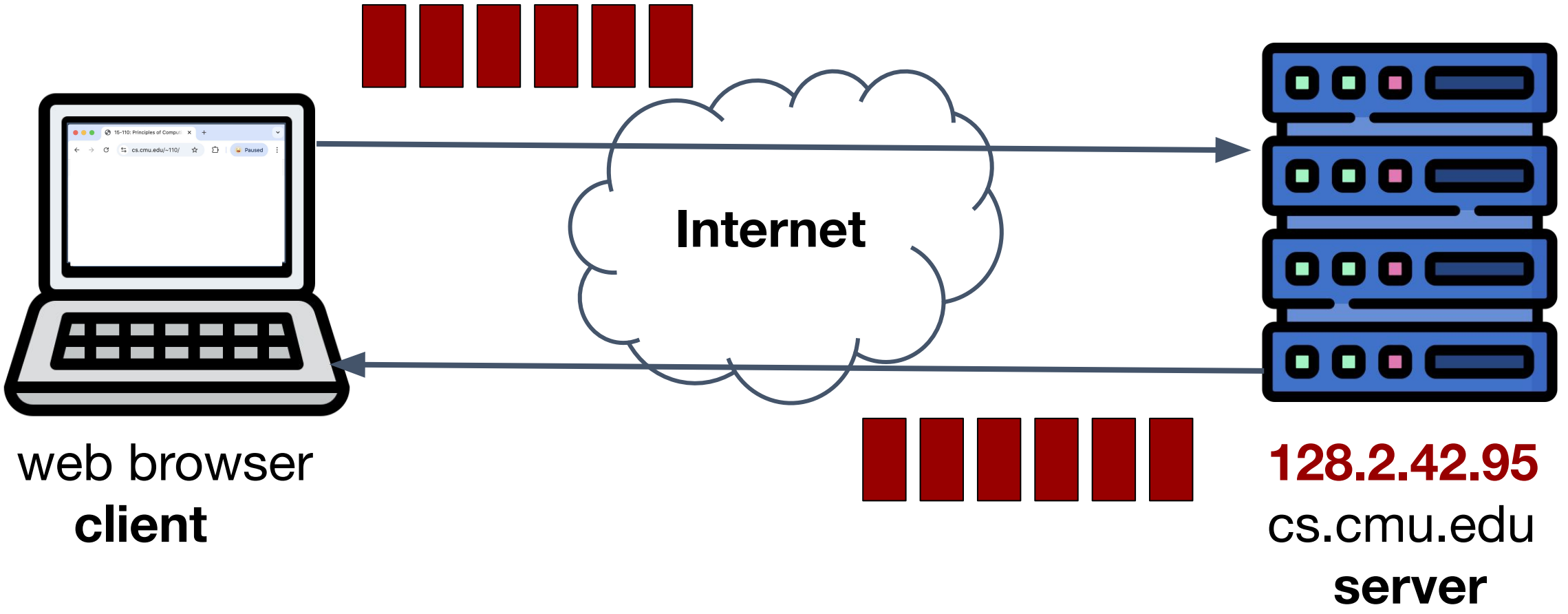
Even though the Internet is decentralized, everyone has to implement protocols correctly for everything to work. Websites may go offline when things are not configured correctly!



HTTP Responses, like **web pages, are also broken into packets** and packet is separately sent back to the client.



HTTP Responses, like **web pages, are also broken into packets** and packet is separately sent back to the client.



# [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, 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.