CS118:
Computer Network Fundamentals

Instructor: Alex Afanasyev (aa@cs.ucla.edu)

Class Website: http://web.cs.ucla.edu/classes/spring17/cs118/

- Syllabus
- Project description

CCLE: https://ccle.ucla.edu/course/view/17S-COMSCI118-1

- Homework assignments

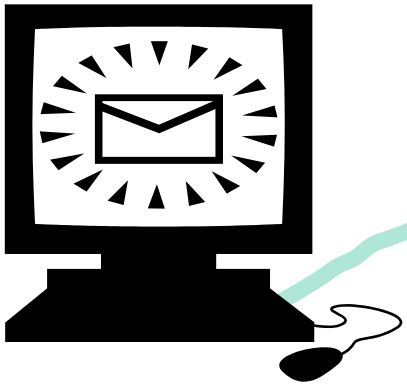
Gradescope: https://gradescope.com/courses/6565

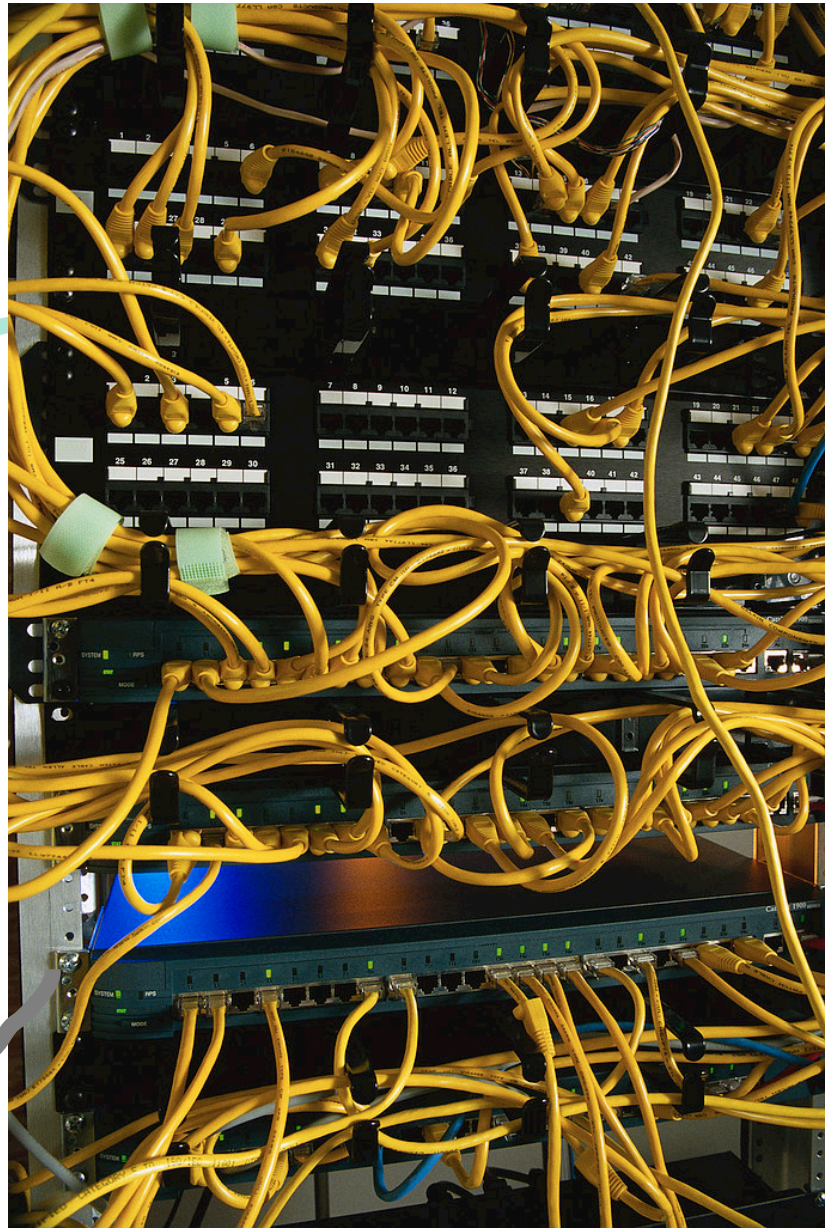
- Homework and project submission

Course Info Summary

Lectures	Mondays/Wednesdays 8am-9:50am **ROLFE 1200**
Discussion Sections	DIS 1: Fridays / 10:00am-11:50am, **BH 5264** (Seungbae Kim) DIS 2: Fridays / 12:00pm-1:50pm, **BH 5264** (Haitao Zhang) DIS 3: Fridays / 2:00pm-3:50pm, **BH 5249** (Zengwen Yuan) DIS 4: Fridays / 8:00am-9:50am, **GEOLOGY 4660** (Pranav Sodhani)
Office hours	Wednesday 5:30pm-6:30pm (BH 4809), other times by appointment
TAs	Seungbae Kim (sbkim at cs.ucla.edu): Fridays / 10:00am-11:50am, **BH 5264** Haitao Zhang (haitao at cs.ucla.edu): Fridays / 12:00pm-1:50pm, **BH 5264** Zengwen Yuan (zyuan at cs.ucla.edu): Fridays / 2:00pm-3:50pm, **BH 5249** Pranav Sodhani (sodhanipranav at cs.ucla.edu): Fridays / 8:00am-9:50am, **GEOLOGY 4660**

What is a Computer Network?





CS118: Explain to You How Internet Works

- Divide-and-conquer
  - Internet: a very large and complex system
  - First: figure out how many major parts
  - Then: Learn one part at a time
- Your job:
  - Read the textbook, think through
  - Ask questions
  - Practice from doing your homeworks and projects

Course Workload and Grading

- Weekly homeworks
  - 8 homeworks total
- Midterm Exam
- Final Exam
- Three programming projects
  - Acio (individual)
  - Confundo (team 2-3)
  - Riddikulus (team 2-3)
- **Strict Grading Policy**
  - No credit for late homework. No exceptions
  - No credit for late projects. No extensions.
  - No make-up exams

Homeworks	16%
Programming Projects	40% (8 / 16 / 16)
Midterm exam	22%
Final exam	22%
Extra Credits	0-5%

Start forming teams NOW

Course Schedule

Midterm	Wednesday, May 3, 2017, 8am-9:50am, **ROLFE 1200**
Final	Monday, June 12, 2017, 6:30pm-9:30pm, **TBD**
Homeworks due	Wednesday of the week following the assignment, 11pm
Project 1 due	Sunday, April 23, 2017, 11pm PDT
Project 2 due	Sunday, May 21, 2017, 11pm PDT
Project 3 due	Sunday, June 11, 2017, 11pm PDT

(Tentative) Schedule of the Quarter

Week:	1	2	3	4	5
Mon	4/3 Course intro 1	4/10 HTTP , HTTP2.0	4/17 DNS	4/24 Reliable data delivery	5/1 Congestion Control
Wed	BW& delay, socket programming 2	Email, P2P	Transport Protocols 3	TCP	**5/3 Midterm**

	6	7	8	9	10	
Mon	5/8 Internet Protocol (IP) 4	5/15 Routing algorithms & protocols	5/22 Multicast routing	5/29 Memorial Day	6/5 Network Security 8	**6/12: Final exam**
Wed	Forwarding and Addressing	Routing in the Internet	Link Layer (Ethernet) 5	5/31 Hubs and switches 6	Review	

The big yellow numbers indicate the chapter numbers in the textbook.

Hints for Getting Good Grade

- Read the text before coming to lecture
- Ask questions!
  - In class and come to office hours
- Use piazza

  https://piazza.com/class/j08uupsnixp5z2
- Make use of the course webpage
  - Lecture slides uploaded after lectures
    - CCLE
  - Homework assignment
    - CCLE
  - Project details and hints
    - http://web.cs.ucla.edu/classes/spring17/cs118/
  - Homework and Project submission
    - https://gradescope.com/courses/6565

A Few More Hints

- You can earn extra credit for in-class participation
  - Make sure you come to me after class and that I have recorded your name in ”the secret book”
- You can earn extra credit for extra tasks in projects
- If you think you may need a recommendation letter
  - Make sure I get to know you
  - Make sure I know you beyond the class

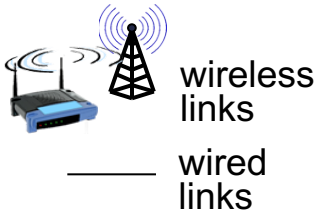
Class Policy

- Posting/sharing/selling class material with or without answers to students outside the class (during or after the class) is strictly prohibited
- Using old homeworks/midterm/finals, except ones provided by the instructor or TAs is strictly prohibited
- Making your project code publicly available during or after the class is strictly prohibited (i.e., you are prohibited to use public GitHub repositories; use private ones either on GitHub or GitLab)
- You must sign and adhere to UCLA Academic Integrity policy and UCLA Code of conduct.

Let's Start

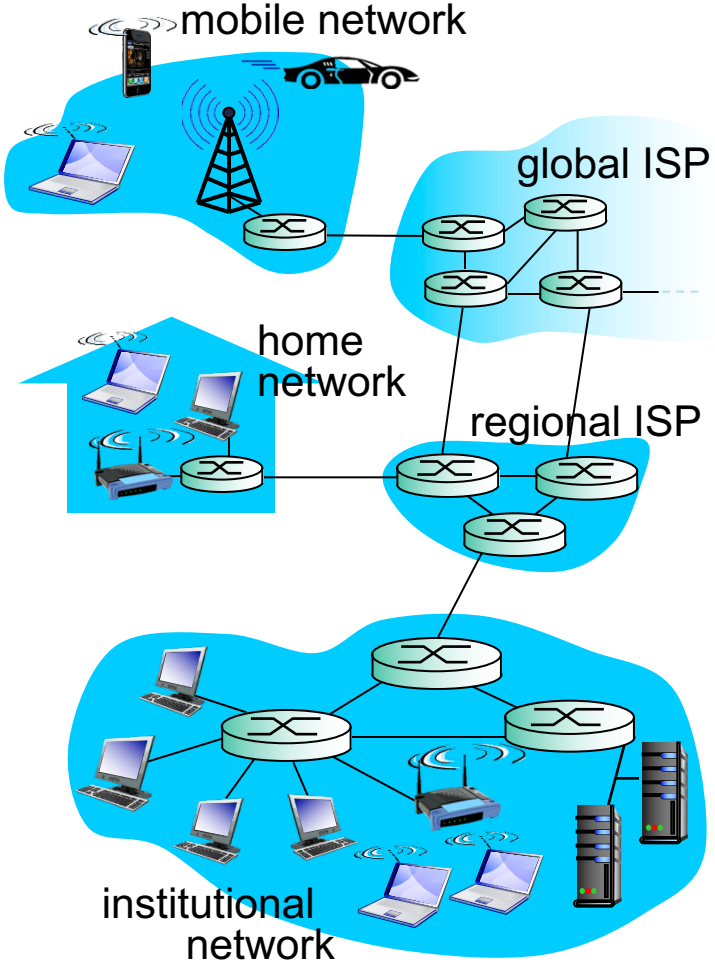
First step: Big Picture & Terminology

- millions of connected computing devices:
  - *hosts* = *end systems*
  - running *network apps*



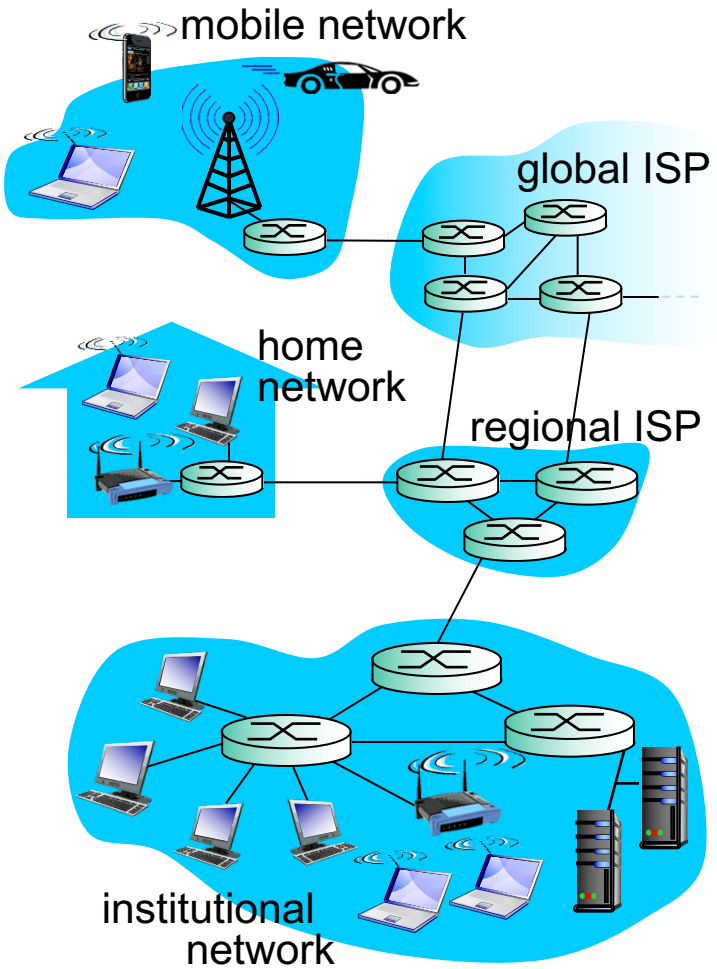
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate: *bandwidth*

- *Packet switches:* forward packets (chunks of data)
  - *routers* and *switches*



"Nuts and Bolts"

- *Internet*: "network of networks"
  - Interconnected ISPs
- *Protocols* control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, 802.11
- *Internet standards*
  - RFC: Request for comments
    - https://www.rfc-editor.org/rfc-index.html
    - IETF: Internet Engineering Task Force
  - IEEE Standards
  - W3C
  - and others



What's a Protocol?

human protocols:

- "how're you doing?"
- "what's the time?"
- "I have a question"

… specific msgs sent

… specific actions taken when msgs received, or other events

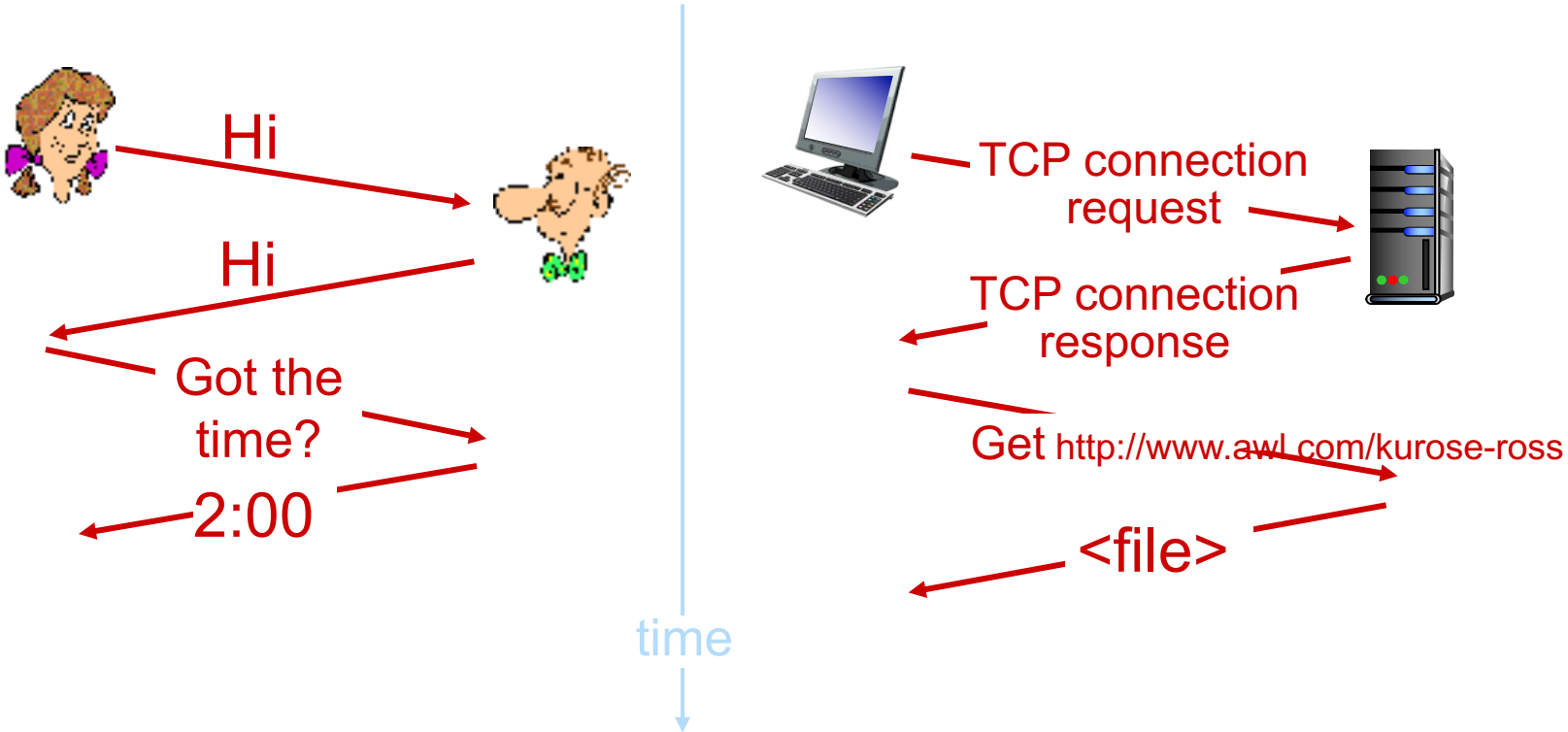
network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt

What's a Protocol?

a human protocol and a computer network protocol:

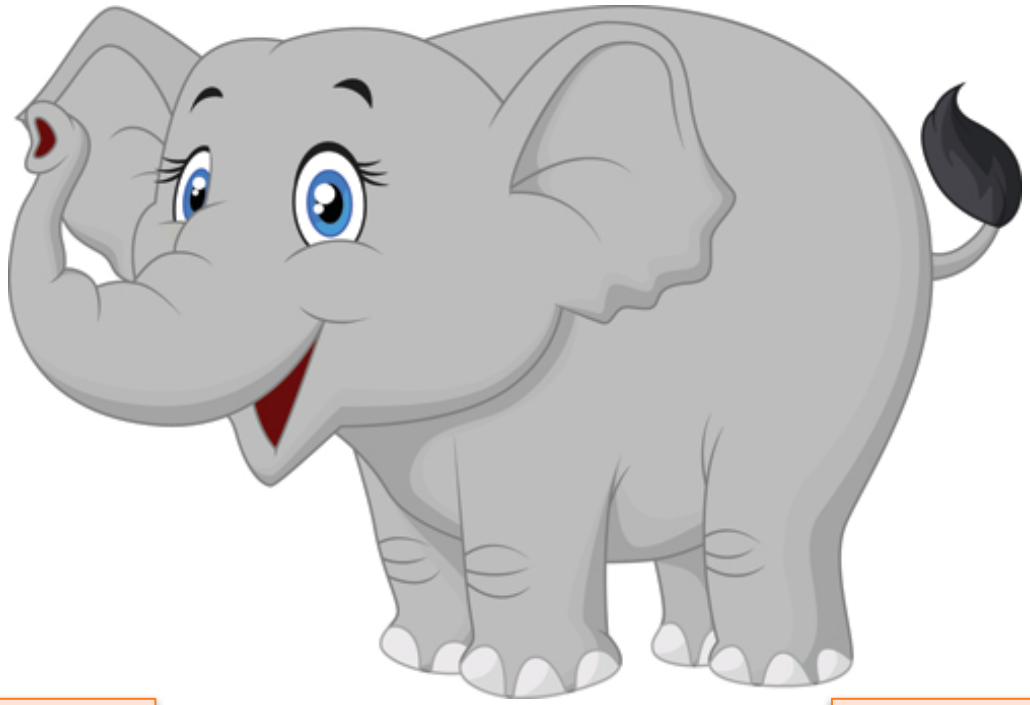


Q: other human protocols?

Different Views on Networking

Application View

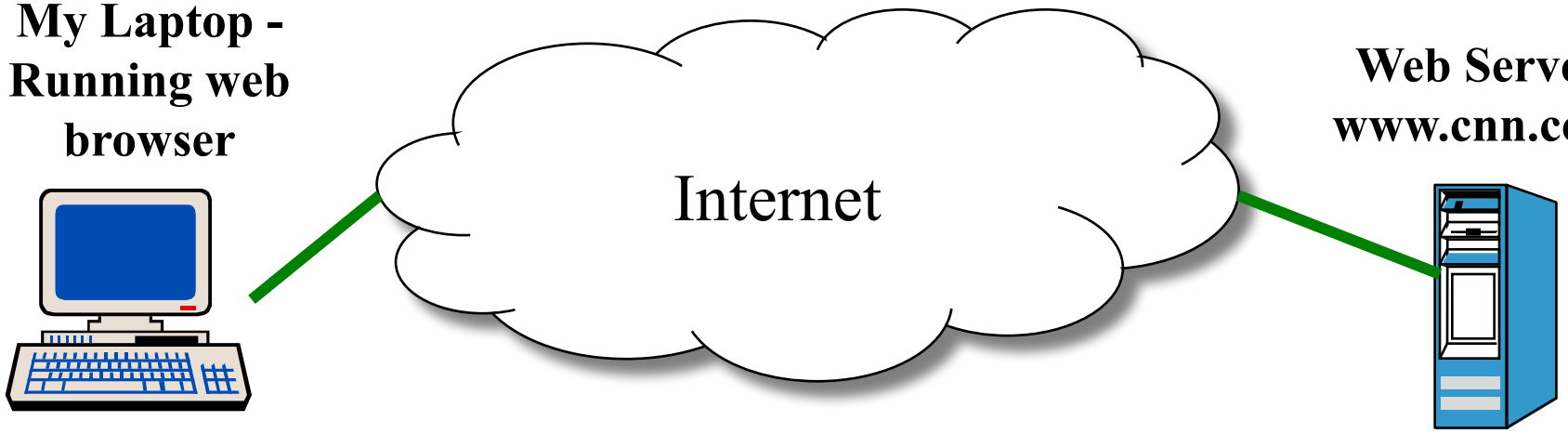
Transport View



Network-Level View

Link-Level View

Application View

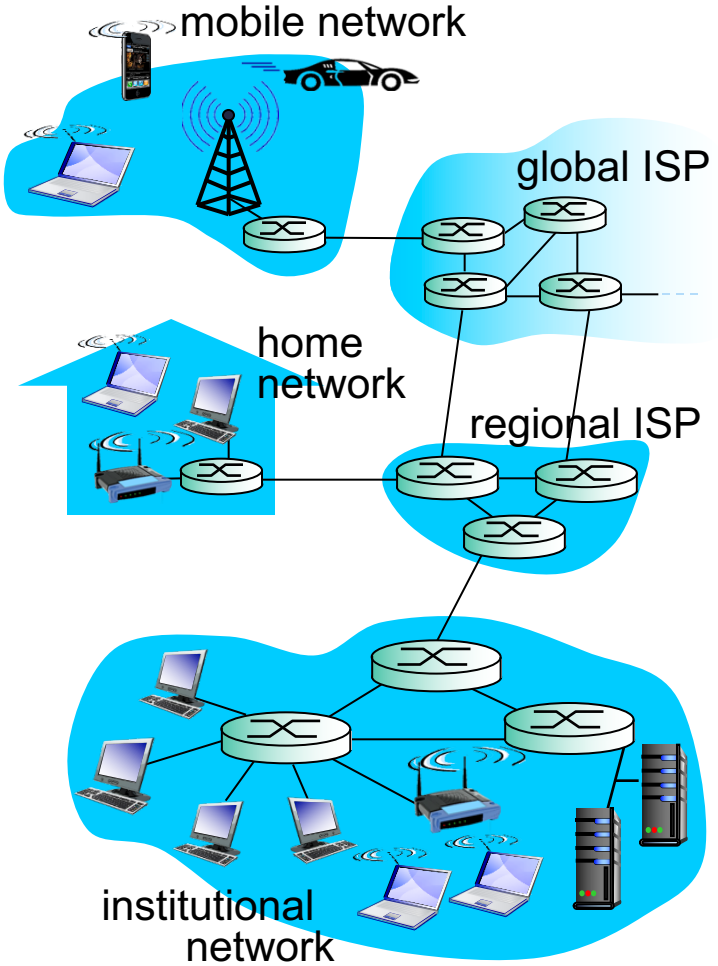


Application protocols

- Assume network provides a way to send data to any hosts on the Internet
- Don't know *or care* how the data is sent; do care whether it is delivered reliably
- Runs on top of transport protocols who take care of how data gets sent

Application View

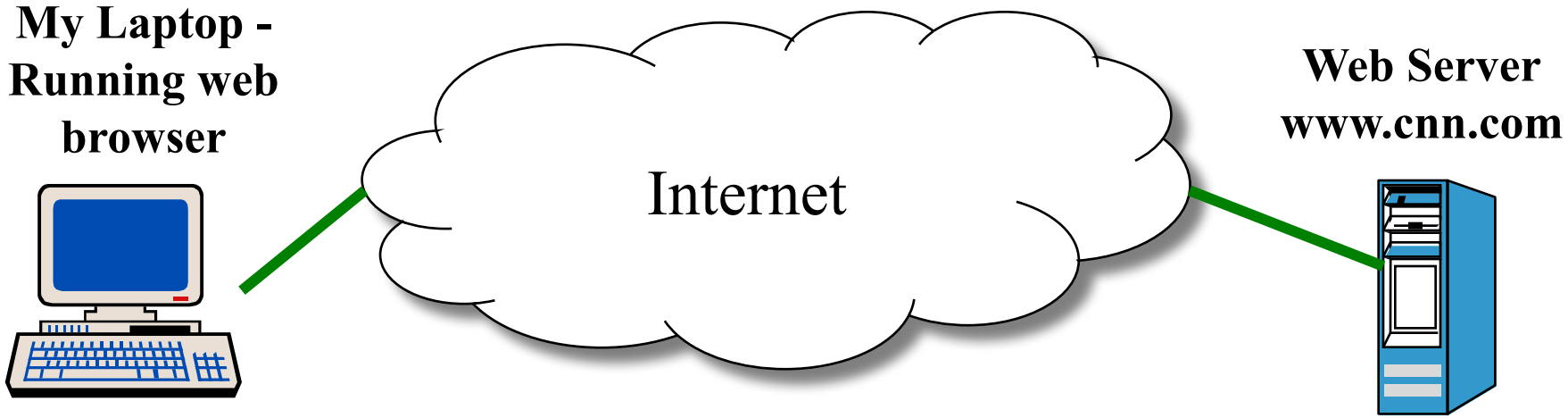
- *Infrastructure that provides services to applications:*
  - Web, VoIP, email, games, e-commerce, social nets, …
- *Provides programming interface to apps*
  - hooks that allow sending and receiving „app programs to "connect" to Internet
  - provides service options, analogous to postal service



Application Layer Protocols

- Covered in Chapter 2 of the textbook
- Basic objective: understand common application protocols:
  - Web: Hyper-Text Transfer Protocol (HTTP), Secure Hyper-Text Transfer Protocol (HTTPS)
  - Email: Simple Mail Transport Protocol (SMTP)
  - Domain Name System (DNS)
- More important objective: design issues
  - What kinds of services required from the network?
  - How does the choice of services impact application design?

Transport View



- Assuming **application protocols** take care of **data content**
- **Transport protocol**'s job: **delivering data** between communicating ends
- *Don't know or care about which paths data may traverse through the network*
- Do care about (1) delivering data to the right application process, (2) delivery reliability, (3) congestion control

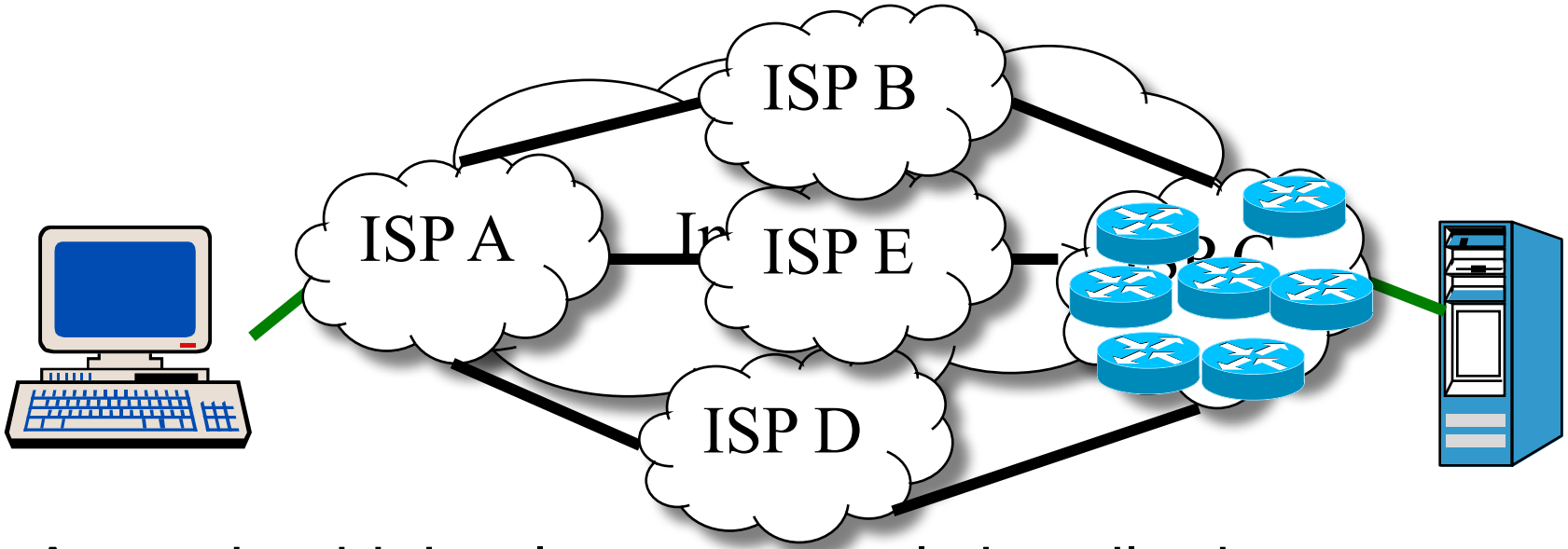
Transport Layer Protocols (Chapter 3)

- Unreliable data delivery: UDP (User Datagram Protocol)
- Reliable data delivery: TCP (Transport Control Protocol)
  - Reliable delivery over potentially unreliable network
  - Understanding and managing network delays
  - Coping with Congestion

But transport protocols don't really do the delivery!

- Pass data to network protocols to do the job

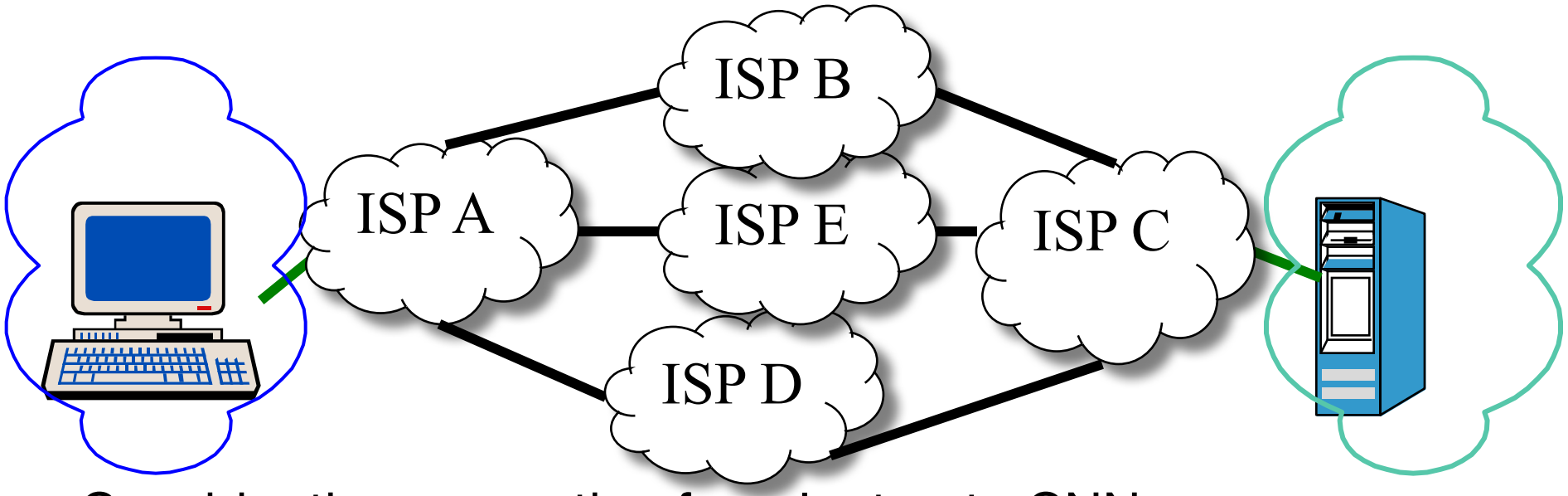
Network Layer View



- Assuming higher layer protocols handle data content, reliability, congestion
- Network's job: forward data from source to destination
- Do care about: which way to forward data at each step?
- Why is this hard?

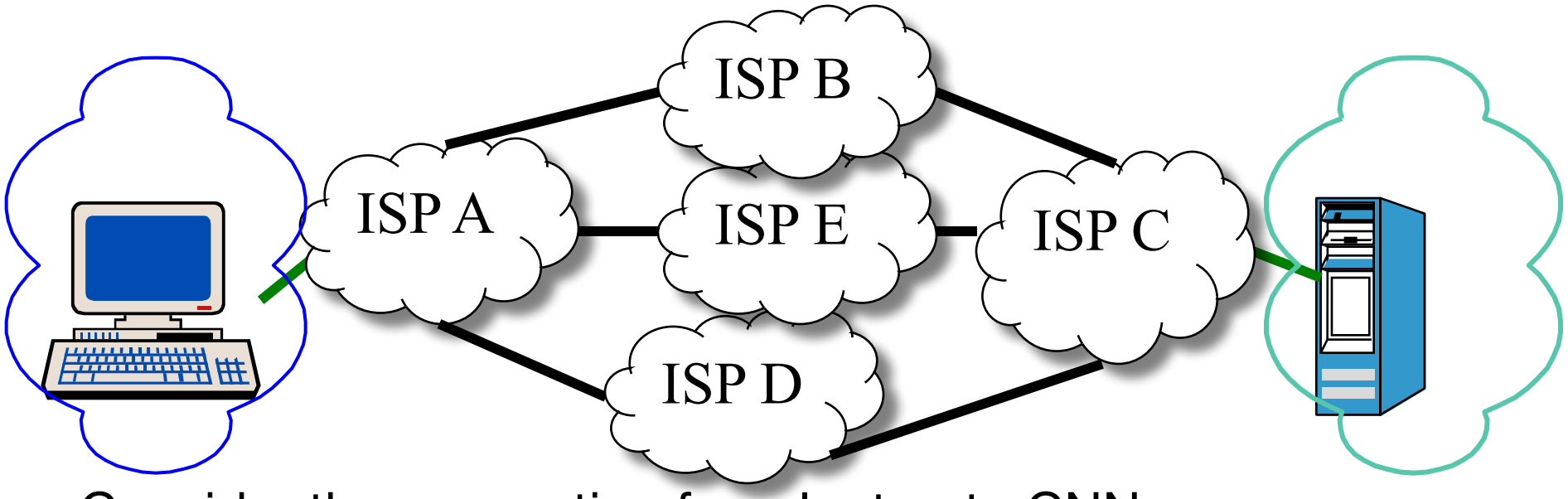
Because the Internet is huge!

Internet connectivity



- Consider the connection from laptop to CNN.com:
  - WiFi → campus backbone → ISP → other ISP → CNN website
- Access Networks
  - Connect end system to local network
  - Some local network router connects to ISP router
- ISP interconnect with each other to form the Internet
  - Each ISP consists of many links and routers

Internet connectivity



- Consider the connection from laptop to CNN.com:
  - WiFi → campu
- Access Networ
  - Connect end s
  - Some local net
- ISP interconnec
  - Each ISP cons

```
✔  10:24 ~ $ traceroute google.com
traceroute to google.com (172.217.2.238), 64 hops max, 52 byte packets
 1  cisco196-1.cs.ucla.edu (131.179.196.3)  3.207 ms  2.544 ms  2.538 ms
 2  border1.cs.ucla.edu (131.179.244.3)  1.262 ms  1.233 ms  1.236 ms
 3  169.232.12.154 (169.232.12.154)  1.032 ms  0.926 ms  0.878 ms
 4  dr01f2.csb1--cr01f1.anderson.ucla.net (169.232.4.54)  1.057 ms  1.046 ms
1.096 ms
 5  cr01f1.anderson--bd11f1.anderson.ucla.net (169.232.4.7)  1.533 ms  1.366
ms  1.365 ms
 6  lax-agg6--ucla-10g.cenic.net (137.164.24.134)  3.395 ms  2.105 ms  1.957
ms
 7  74.125.49.165 (74.125.49.165)  1.573 ms  1.782 ms  1.670 ms
 8  64.233.174.238 (64.233.174.238)  1.725 ms  3.783 ms  2.074 ms
 9  209.85.250.245 (209.85.250.245)  2.036 ms  2.005 ms  1.960 ms
10  lax02s19-in-f14.1e100.net (172.217.2.238)  2.500 ms  1.611 ms  1.774 ms
```

Network Layer Protocols

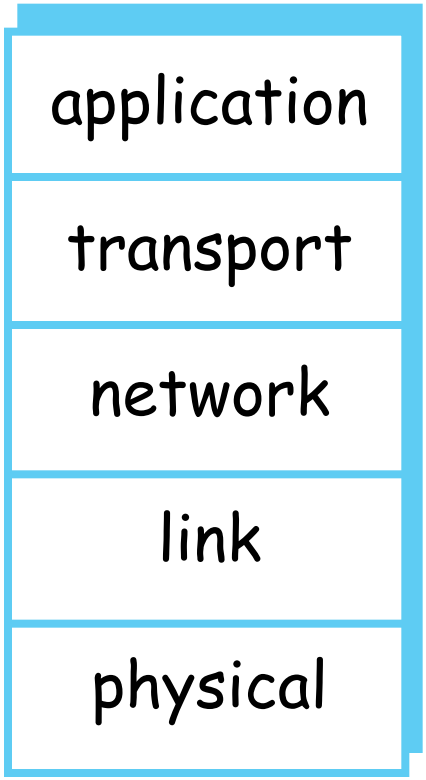
- Covered in Chapter 4 of the textbook
- There are different types of networks
  - Circuit-switched versus packet-switched
- Internet: packet-switched networks
  - Network layer provides best effort delivery of packets
- Don’t care exactly how a packet is delivered from one node to next
  - That's the job for link layer protocols

Link Layer View

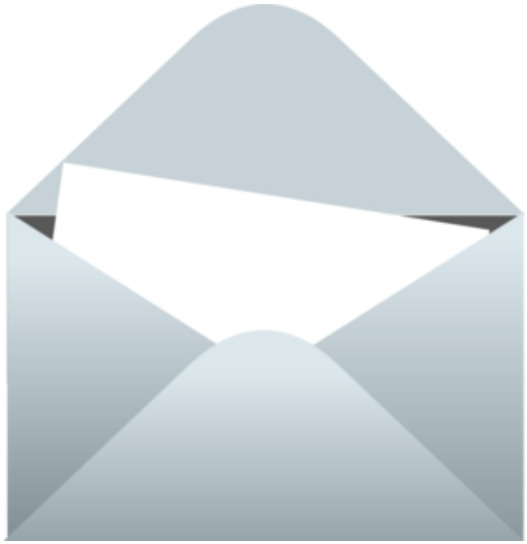
- Link can be twisted pair, coaxial cable, fiber optic, or wireless (multiple types)
- Link layer job: Get a packet sent across some medium
  - Different medium → different link layer protocol
- Covered in Chapter 5 and 6.1-6.3 of the text.
  - Borders on Electrical Engineering: running on top of physical layer
- Our objective is to understand technology
  - How the network is built
  - How do link layer features impact higher layers designs?

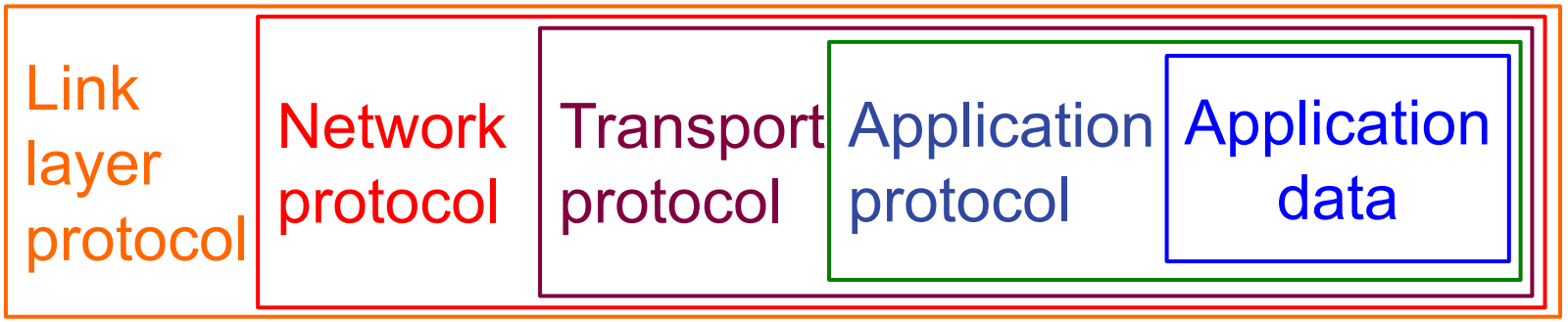
Internet protocol stack

- Application layer
  - Support data exchange between app. processes
  - Example: ftp, smtp, http
- Transport layer
  - handling delivery reliability, multiplex within a host
  - Example: TCP, UDP
- Network layer:
  - forward packets from source to destination
  - IP, routing protocols
- Link layer:
  - transfer data between directly connected network elements
  - Ethernet protocol
- physical: bits “on the wire”



What “layer” means to a packet



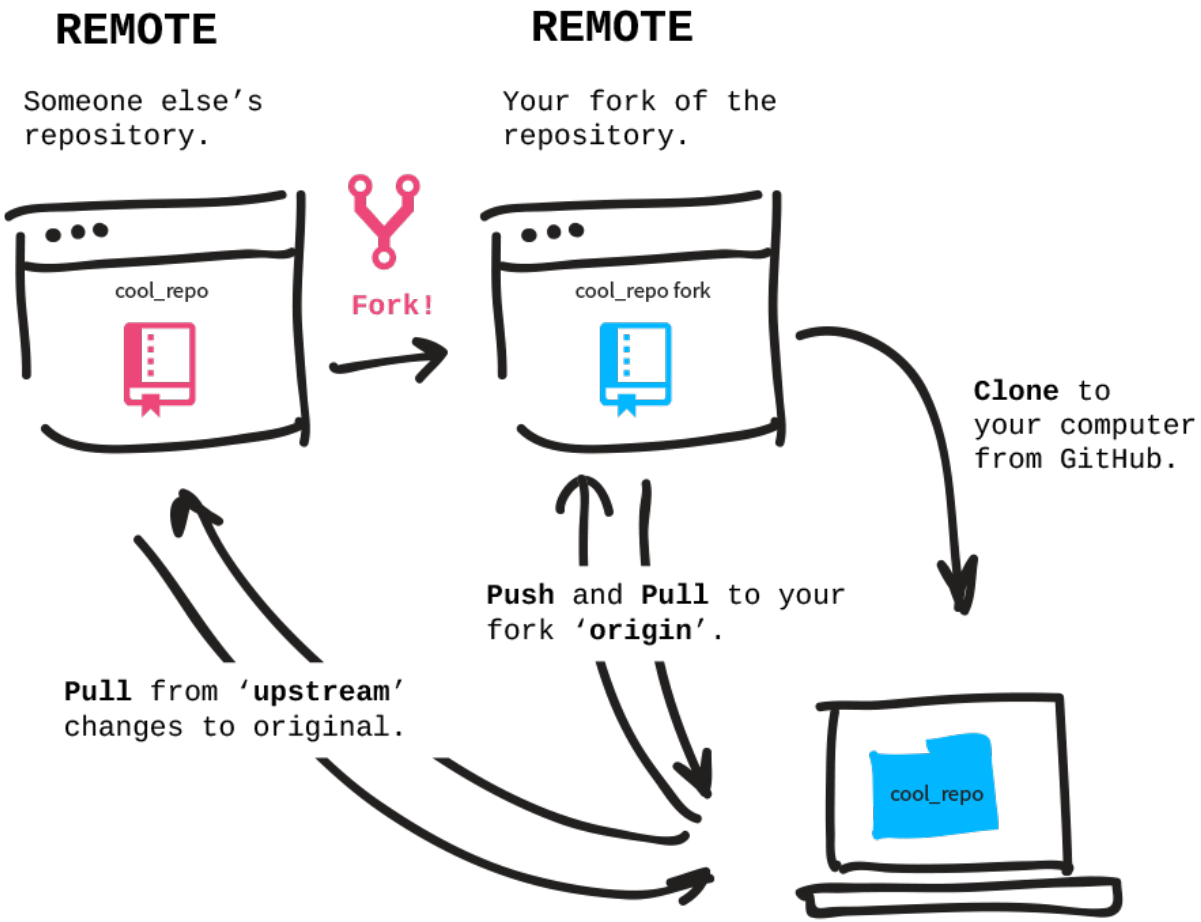


Introduction to Collaboration and Networking Tools

In This Class You Will Need / Learn: git

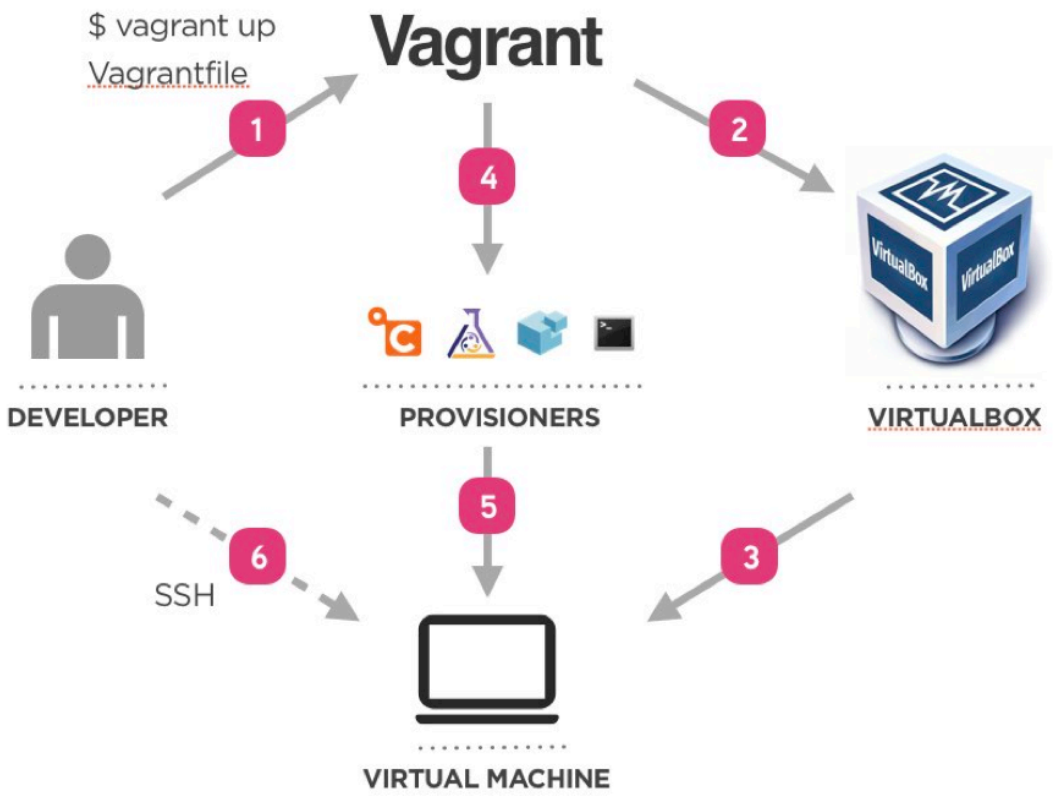
- Distributed version control system
- https://git-scm.com/
- ! Make sure you use **private** repositories for your code (free on GitLab, free with student account on GitHub)
- If you new to git
  - http://rogerdudler.github.io/git-guide/
- If you think you know git:
  - Play this game http://learngitbranching.js.org/index.html
  - How your commit messages look? https://chris.beams.io/posts/git-commit/

Working with Remote GIT Repositories



In This Class You Will Need / Learn: Vagrant

- A tool for building and managing virtual machine environments
  - https://www.vagrantup.com/
  - https://vagrantcloud.com/



In This Class You May Learn: Docker

- A platform for software containers
  - Light-weight VM to run a specific task
  - Container itself should be stateless, so it can easily be migrated to a different host
    - Can reference network storage
  - https://www.docker.com/
    - https://docs.docker.com/docker-for-mac/
    - https://docs.docker.com/docker-for-windows/
  - https://hub.docker.com/
- https://www.youtube.com/watch?v=YFl2mCHdv24
  - 12 minutes intro to Docker

Demo time

- During discussion sections