Internet Services

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Introduction

Lecture 1

Srinivas Narayana

http://www.cs.rutgers.edu/~sn624/553-S25

RUTGERS
UNIVERSITY | NEW BRUNSWICK

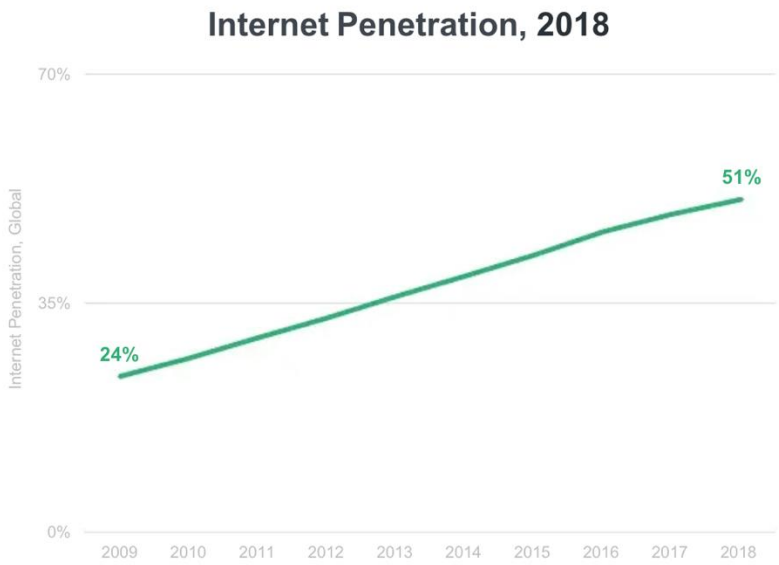
Our life on the Internet



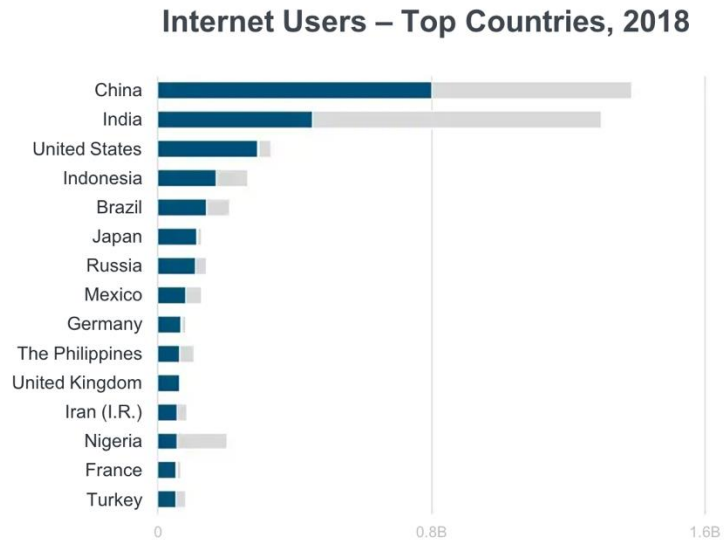


Internet users are everywhere

Global Internet Users =
3.8B >50% of Population

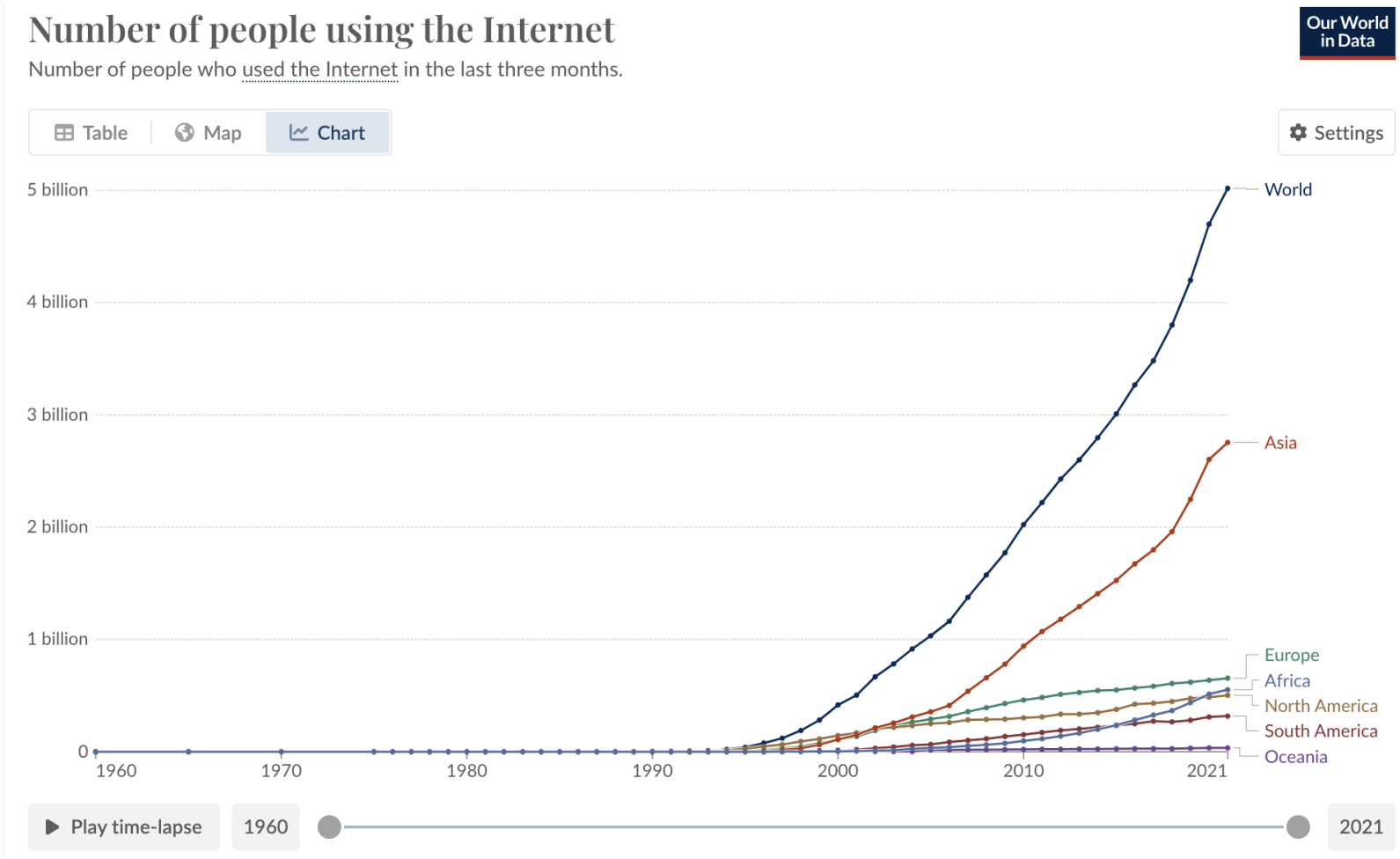


Global Internet Users =
China @ 21% of Total...India @ 12%...USA @ 8%



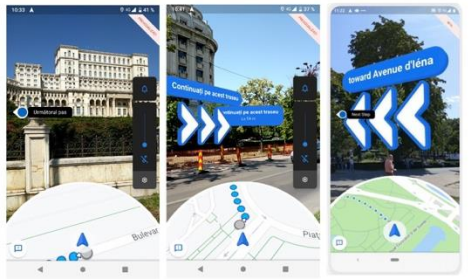
Source: Mary Meekers, 2019 Internet trends

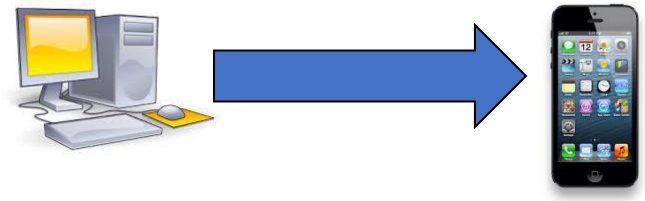
Internet users



https://ourworldindata.org/grapher/number-of-internet-users

Evolution of Internet services





2010-2020

2020--

1992	1996	2000	2004	2008
ftp Web email	chat Games IM Yahoo!	news Blog Search amazon	Music itunes Games search Google	Wikipedia Craiglist Youtube

UBER

airbnb

NETFLIX

Spotify

NVIDIA

OMNIVERSE

FORTNITE

MINECRAFT

Gather

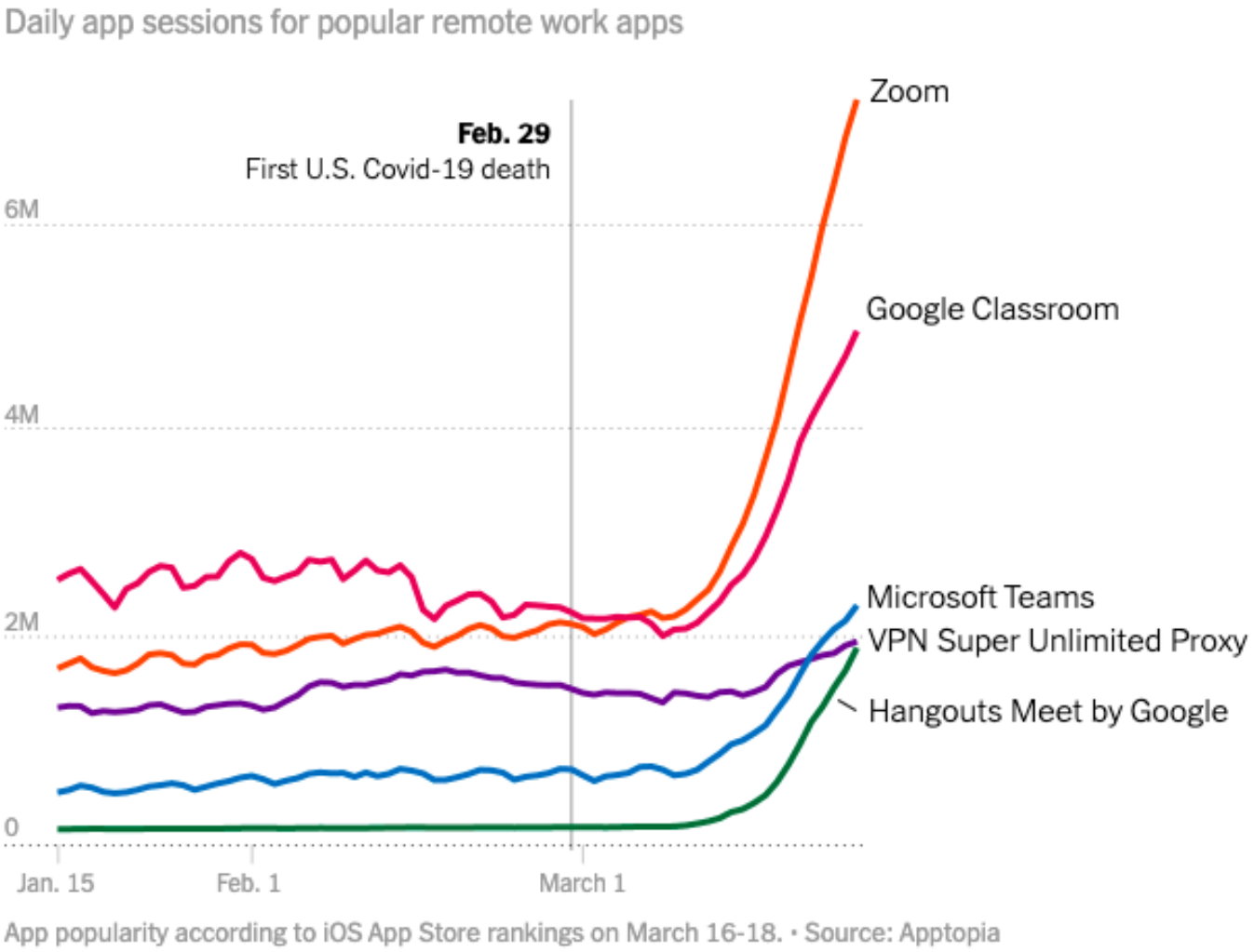
Text-heavy

Multimodal media
User-generated content

Augment physical world

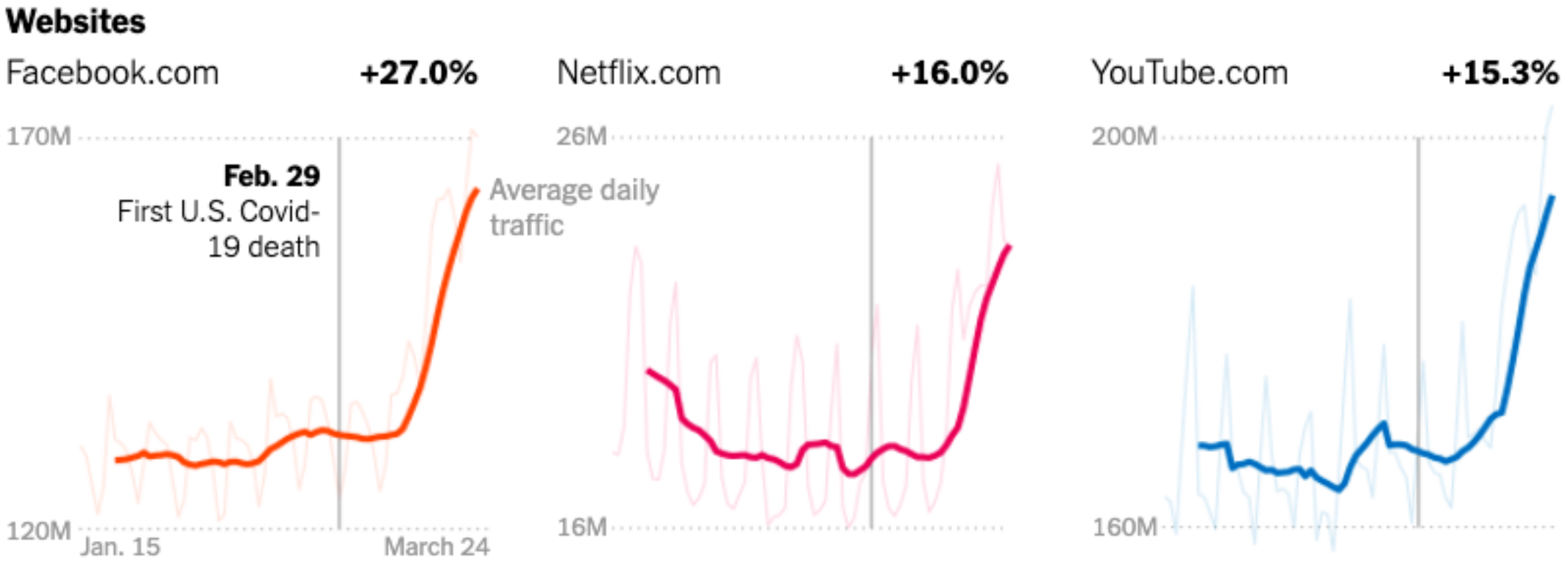
Replace phy world

Pandemic shifts: how we worked



Data shows number of daily sessions in the US over a period in 2020. Source: nytimes

… and played



Data shows number of daily sessions in the US over a period in 2020. Source: nytimes

Internet Services

Understand how modern Internet services designed

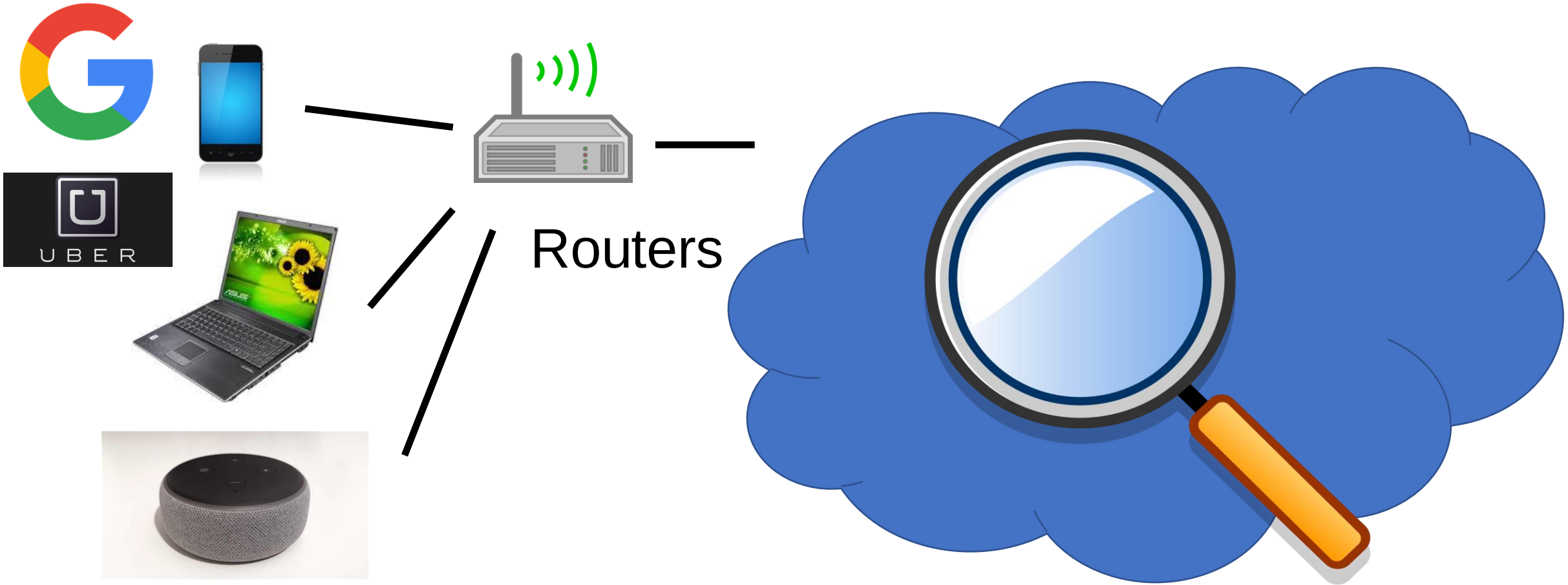
Learn fundamental concepts in application, system, and network design that make them possible

Practice your knowledge

What is an Internet service made of?

Applications, systems, interconnecting network, abstractions.

Components of an Internet Service



Routers

Endpoints

Components of an Internet Service

Routers

Data Center

Endpoints

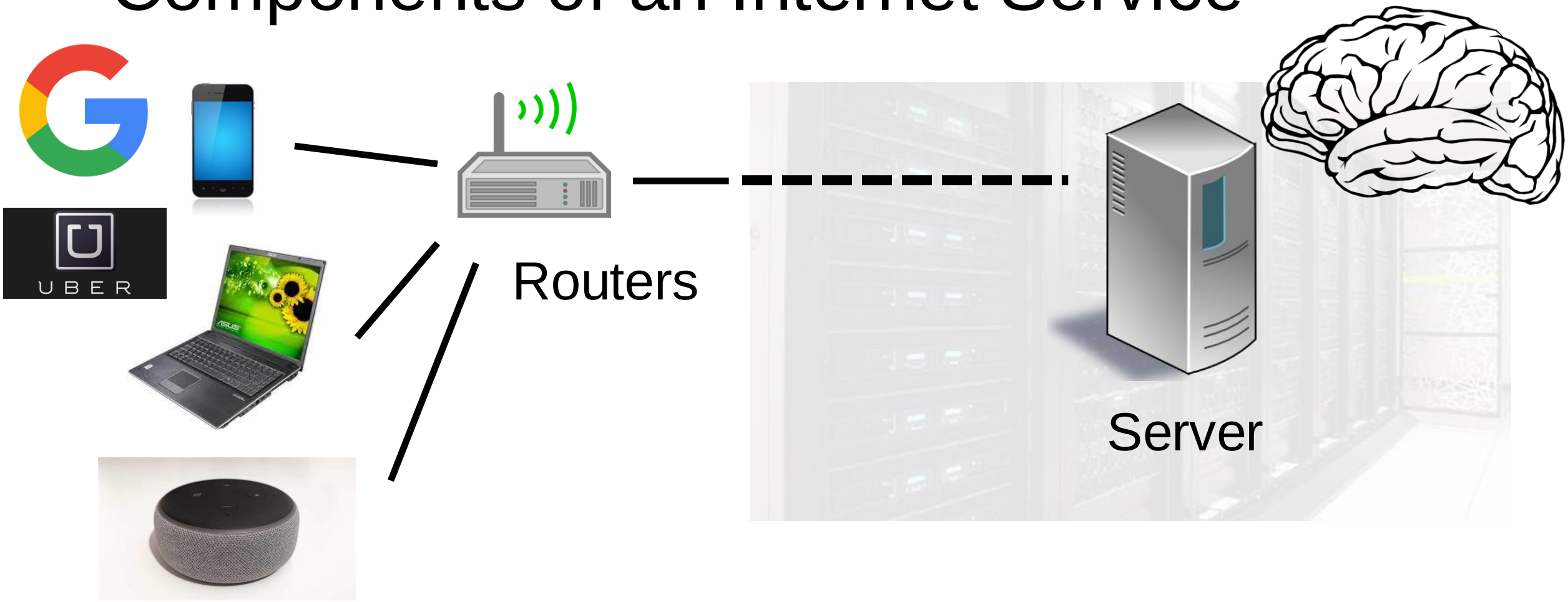
Components of an Internet Service

Routers

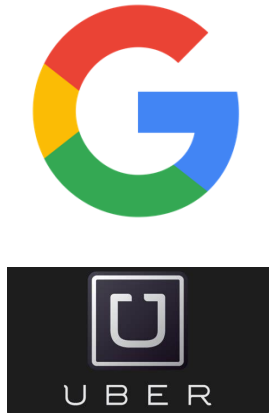
Data Center

Endpoints

Components of an Internet Service



Components of an Internet Service



Routers

App compute and communication patterns

Servers

Modular apps

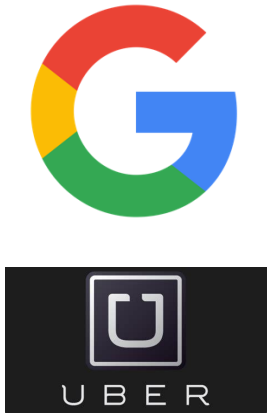
Data Center

Endpoints

Storage

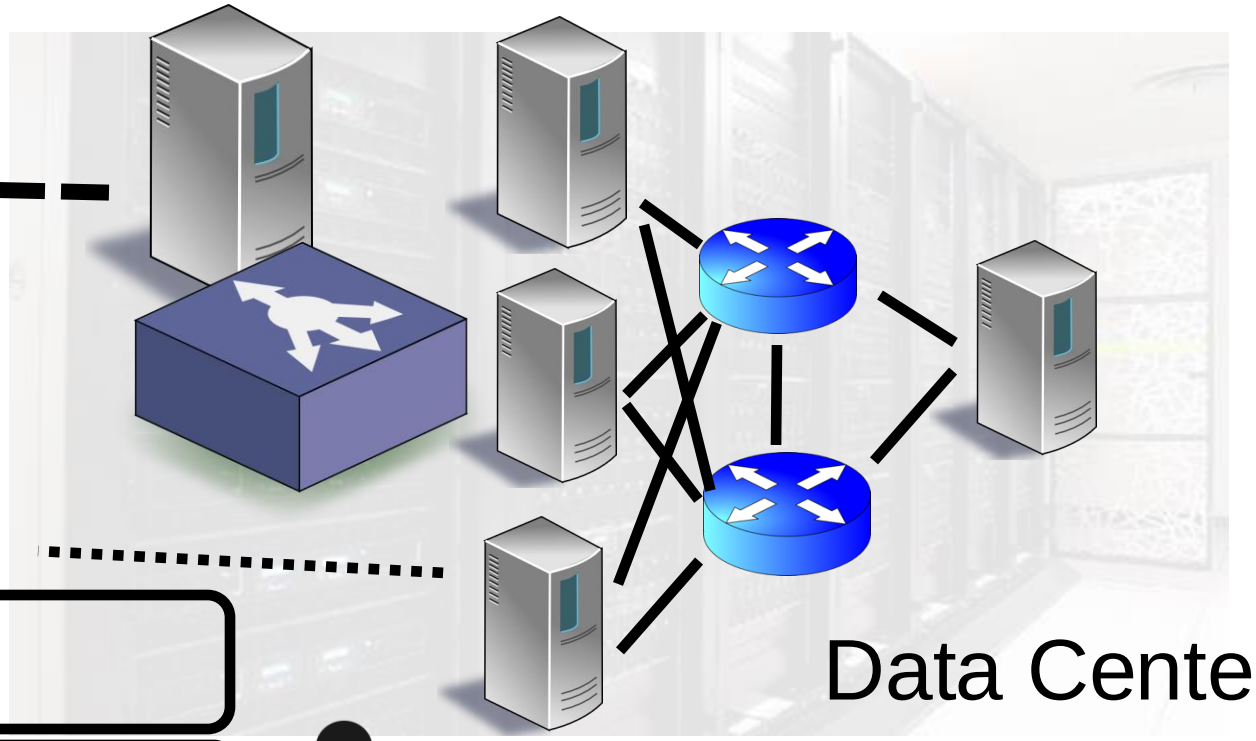
Interconnect: Routers

Components of an Internet Service



Endpoints

Routers



Data Center

App

Container

Virtual Machine

OS & Net Stack

Policies

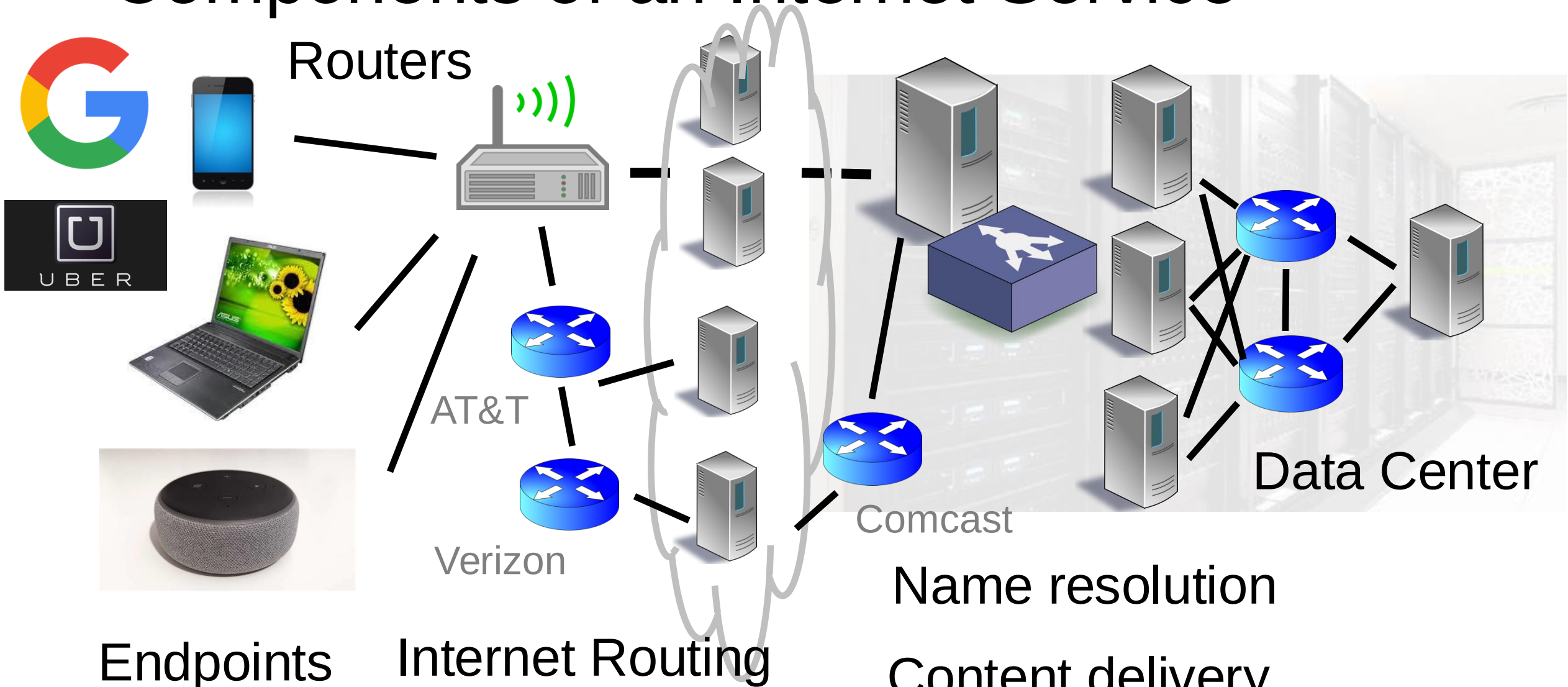
aws

Monitoring

Performance

Availability

Components of an Internet Service



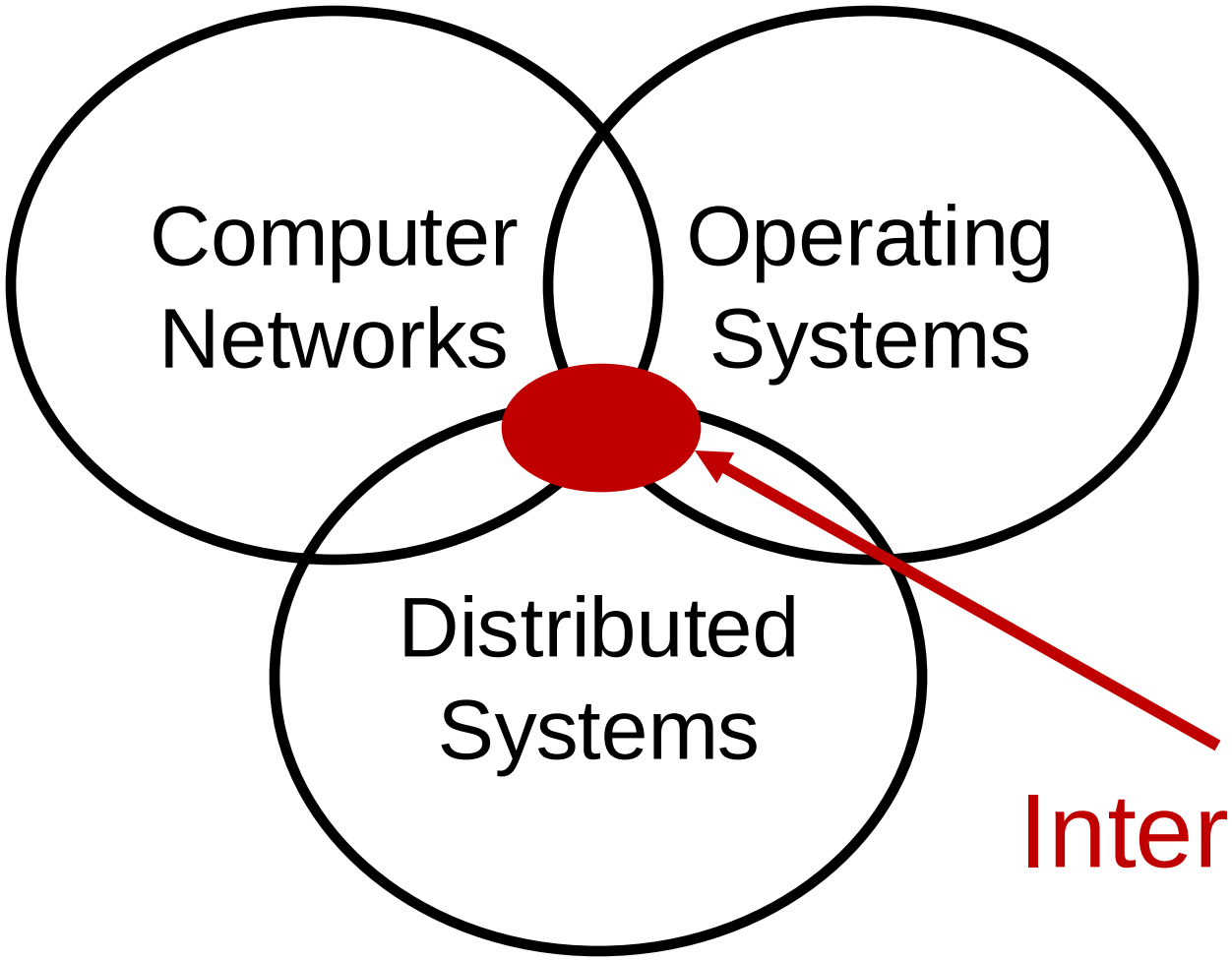
Endpoints

Internet Routing

Name resolution

Content delivery

Core technical disciplines (incomplete)



*app abstractions

*prog languages

*algorithms

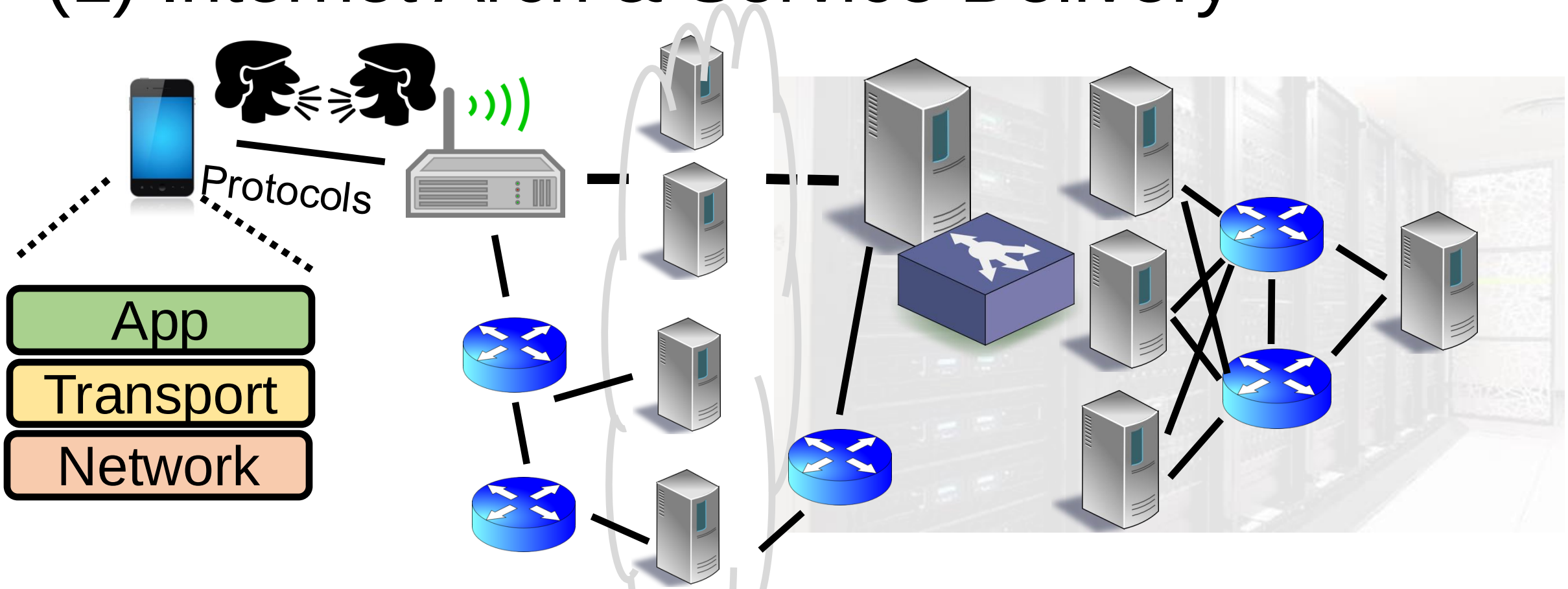
*security

Why should you study Internet services?

- Intellectual merits
  - Interdisciplinary problems, many principles
- Real world utility
  - Low barriers: anyone can do something useful
- Pragmatic
  - Many job opportunities, timely
- Distinct from other coursework at Rutgers
  - OS, distributed systems, databases, networks

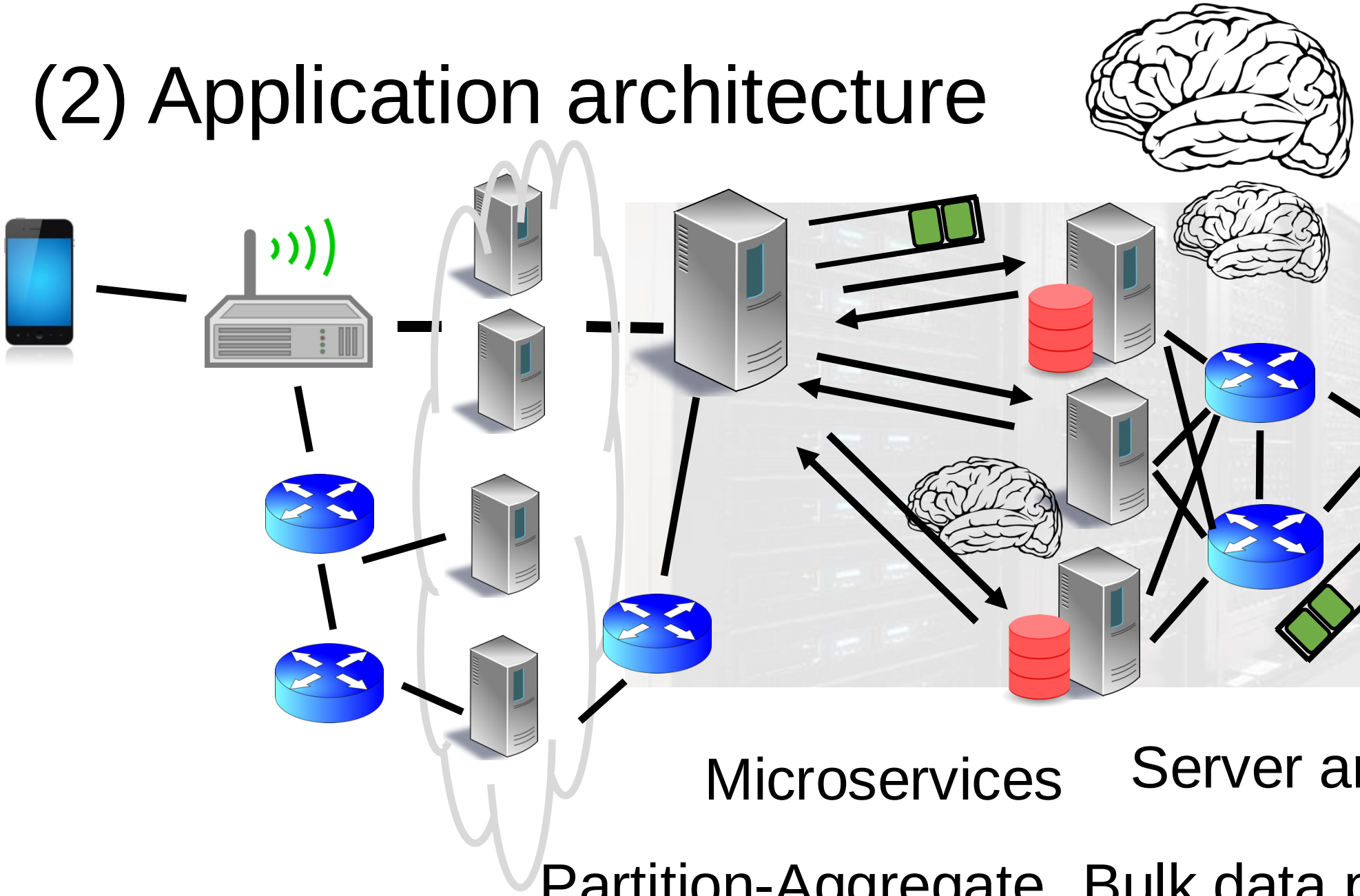
Course Content Overview

(1) Internet Arch & Service Delivery



Content delivery

(2) Application architecture



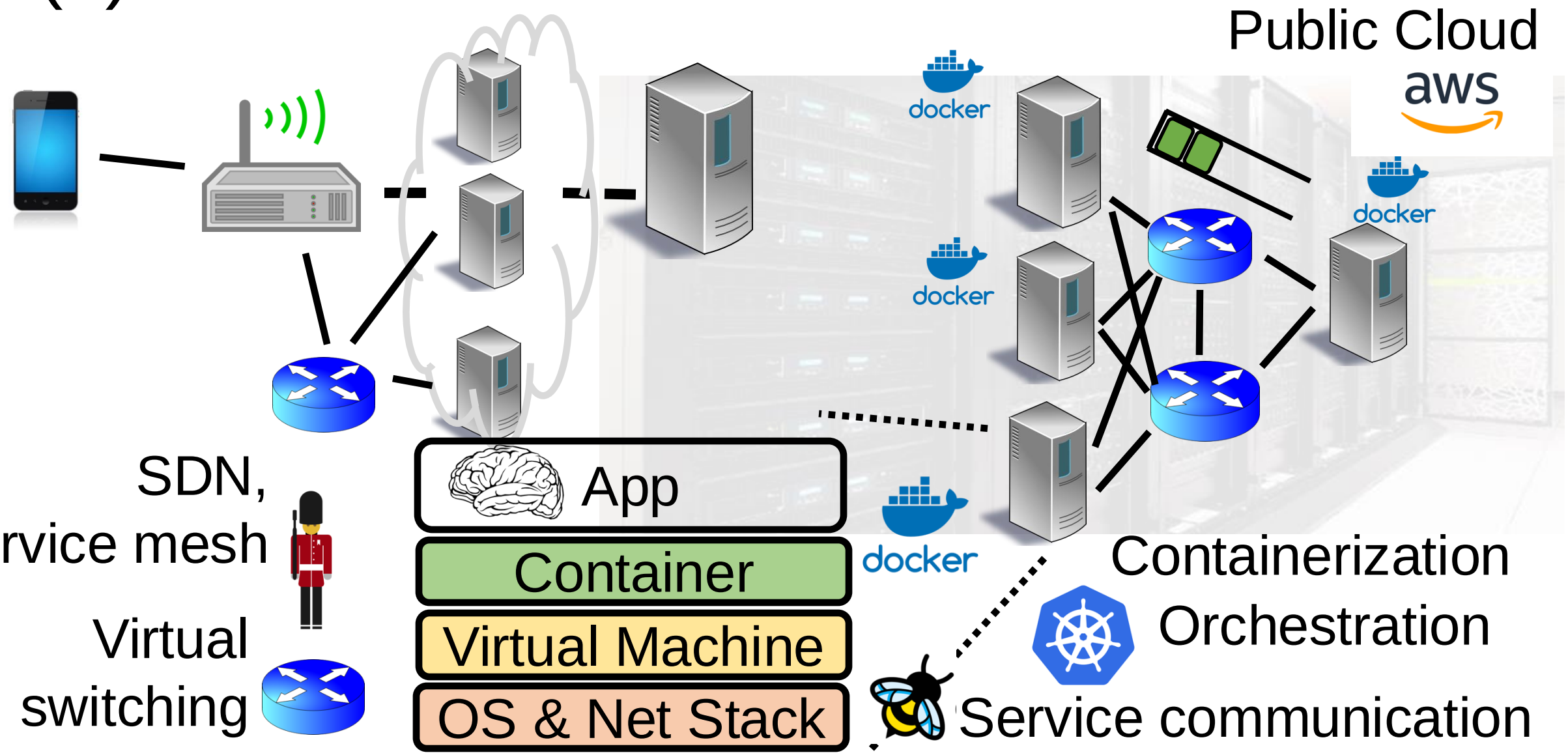
Microservices

Server architecture

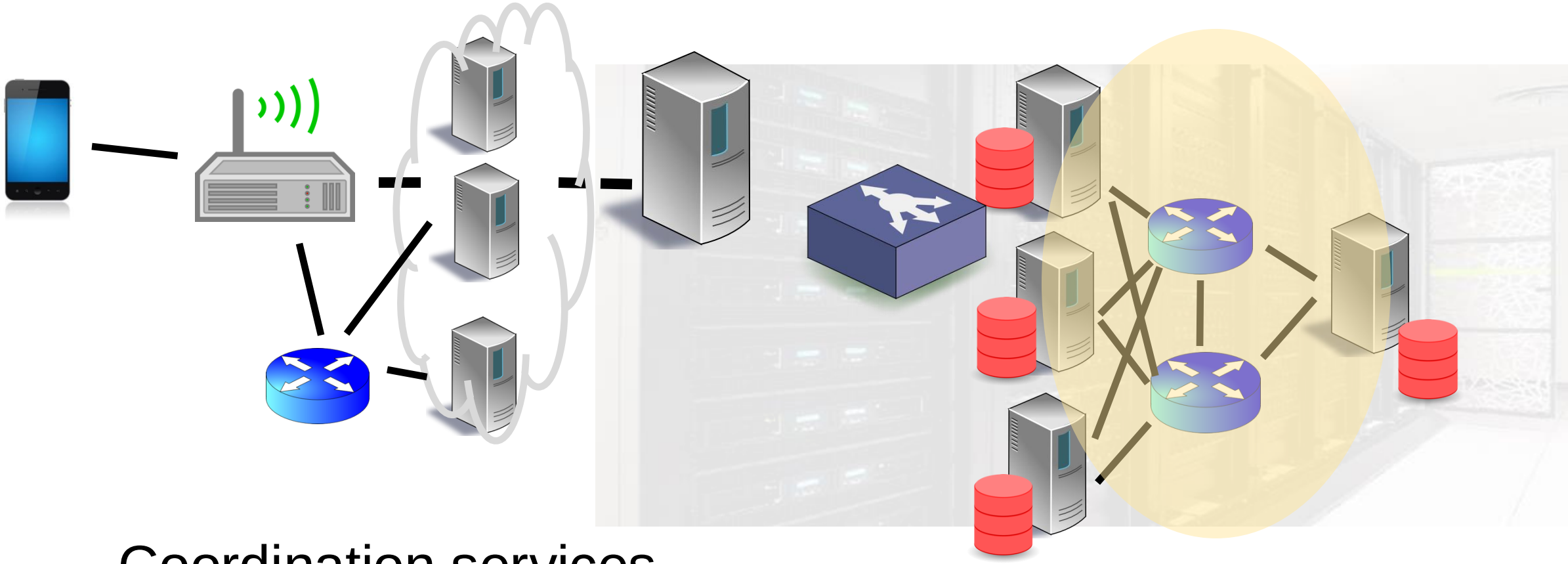
Partition-Aggregate

Bulk data processing

(3) Infrastructure



(4) Distributed System Design



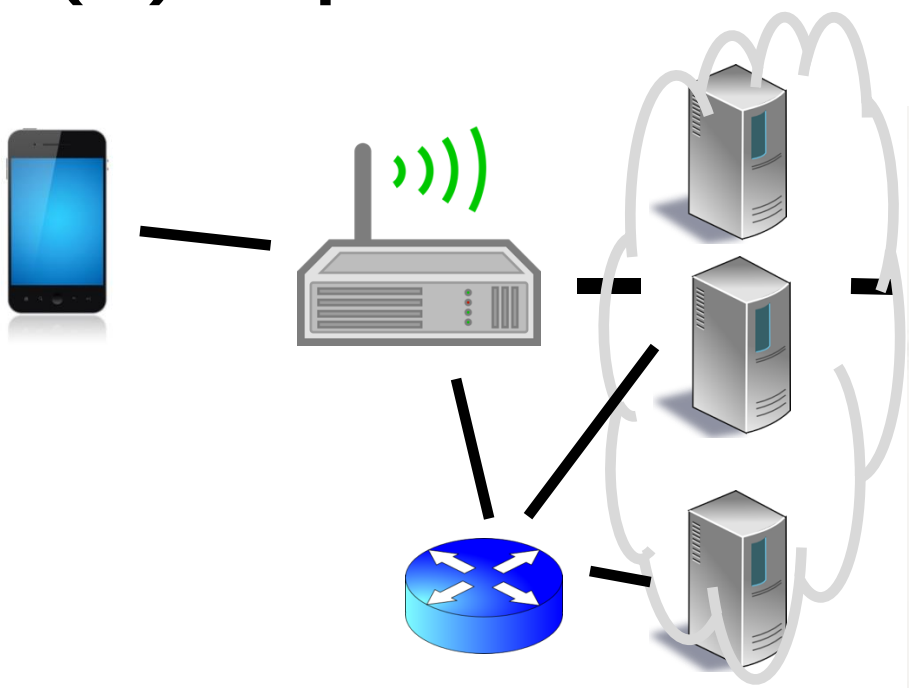
Coordination services

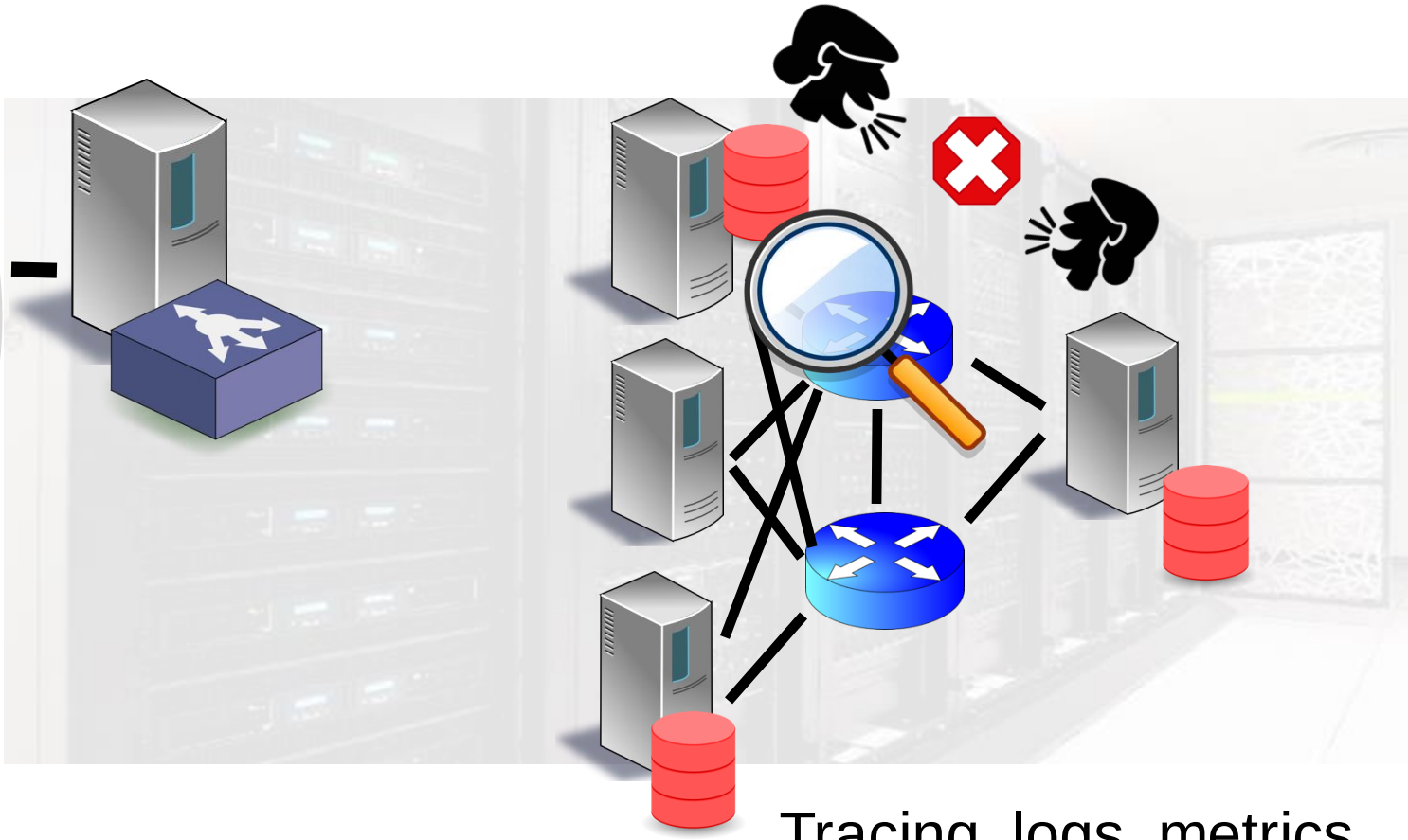
Distributed consensus

Distributed storage

Load management

(5) Operations





Monitoring

Access control

Scaling

Incident management

Tracing, logs, metrics

In-band network telemetry

Network stack monitoring

Course Logistics

About this class

- Faculty Instructor: Srinivas Narayana
  - http://www.cs.rutgers.edu/~sn624
  - sn624@rutgers.edu
  - Office hours: by appointment
  - Lecture Wed 8:30 – 11:30 in Busch HLL 116
- Canvas and Discord
- Class info: http://www.cs.rutgers.edu/~sn624/553-S25/
- Teaching Assistant: Bhavana Vannarth Shobhana
  - bvs17@cs.rutgers.edu

Class philosophy

- We want you to learn and to be successful
- Significant technical reading & programming
  - Build necessary skills for future tech careers (industry or academic)
- Be proactive: interact, ask, support
  - Attend lectures and discuss class material regularly
  - Use class Discord
- Happy to provide the necessary background materials
  - Ask me or your TA

Course grade components

- Quizzes (32%)
- Programming homework (28%)
- Course project (32%)
- Class participation (8%)
- Absolute grading; no curve

In-class quizzes (32%)

- Every lecture starting 2/12
- Answer questions from the last lecture and assigned reading
  - Requires significant engagement with the technical readings
  - 5—10 hours per reading
- Typical length: 20 minutes
- We only consider the 8 highest grades (drop lowest 3)

In-class quizzes (32%)

- Open book, but paper-based materials only
- Unlimited quantity: research article, textbook, lecture prints, your notes
- No tablet, laptop, smart watch, cell phone use
- No collaboration or Internet use
- No make-up quizzes

You'll spend significant time reading

- This course has an assigned technical reading per week
  - Technical blog posts, accessible survey articles, deep technical papers
- Knowing how to read well technically is worth your time
  - Grad students: reading and writing technical papers
  - Developers: reading RFCs, protocol specifications, other technical specs
  - Implementing any cutting-edge technology requires reading
- Staying broadly technically educated (and employable)
- It's worth reflecting on how to read effectively

There is no magic

- Reading effectively takes a lot of time and effort
- I’ve been reading technical articles for 14+ years now
  - And I still sometimes spend 5+ hours or more
- You will get more effective and better over time
- A few tricks

(1) Three pass approach for papers

- First pass: title, category, context, assumptions, correctness, contribution
- Second pass: get into the technical ideas, figures and graphs. Explain the new technical idea or design to someone else
- Third pass: starting from assumptions and problem statement, reconstruct solution on your own with proof or argument for why it works and why it is the right approach (other approaches?)

(2) Look for concrete, simple examples

- Good articles often use good examples to explain their ideas
- If the article does not show examples, construct and work through your own
- Constructing a good example itself often clarifies the technical problems and innovations in the solution
- Discussing with your peers can be substantially helpful

(3) Identify reusable principles

- What do you want to take away from the reading?

- System design or algorithmic techniques
- Existence of a new problem space
- A class of technical solution approaches
- New techniques for empirical evaluation or measurement

- Reading something worthy changes how you think about the world from that point

Programming homework (28%)

- 2—3 over the term; tentative submission dates on syllabus
- Work on your own; released and handed in on Canvas
- C/C++ programming language, Python, shell
- Test and run on class VM (access instructions to be provided)

Programming homeworks (28%)

- Please follow all instructions carefully and exactly

- You will lose significant points if:
  - We are unable to run your code with the stated commands
  - We do not receive your submission in a timely fashion

Programming homeworks (28%)

- You can collaborate with others freely, however all submitted code must be your own work
- Do not blindly lift code from stack overflow, GitHub, chatGPT, etc.
- Incorporate learning from other sources and produce your own solutions
- Mandatory to state collaboration & references at the beginning of submitted program

Course project (32%)

- A team of up to 3; work on this through April
- Aligned with the course topics
- “Significant” programming component
  - Measured by complexity and the size of the source code
  - Talk to me to ensure this (next: project specification)

Course project (32%)

- A small open-ended research problem
- Adding new features to an existing open-source codebase
- Reproducing empirical evaluations and benchmarks from a paper you read
- Re-implementing an existing technique on a different system
- Building a tool that makes further technical work or research possible or easier

Course project (32%)

- I will send out some possible ideas
- Welcome & encouraged to work on something *you* find exciting
- Form teams, brainstorm ideas with each other and me early
- I will help you succeed technically: don’t struggle alone
- 3 concrete deliverables: project specification, source code, technical report

Project specification

- 1—2 pages, tentatively due 04/02

- Specific, measurable, realistic technical goal
- Existing prior work and why your goal or approach is novel
- Brief description of technical idea and solution approach
  - What needs to be built? How will you build it?
- Key performance metrics, qualitative and quantitative, you will evaluate your system on
- Technical and other risks

Source code

- Due end of the semester (tentatively 05/05)
- Host source code on a public repository e.g., on GitHub
- Provide clear documentation (README) with commands and requirements to run your system
- Provide scripts and commands to reproduce your empirical evaluation results

Technical report

- 5—10 pages, due end of the semester (tentatively 5/05)
  - Latex template to be provided
  - One PDF per team submitted on Canvas
- Clearly and fully describe all the technical details:
- Implementation of the solution
- How you evaluated the system: metrics, workloads, executions
- Why did you observe the numerical results you found?
- Optional: presentations or demonstrations of the system
- Assessed for clarity, comprehensiveness, technical design, scientific accuracy

Course project (32%)

- Don’t “just implement” something, also measure and explain it
- Highly coveted technical skills:
  - Identifying good performance indicators, representative workloads and configurations
  - Measuring them accurately
  - Explaining them clearly with more detailed measurements
- Rigorously evaluating a system may take as long as implementing it

Course project (32%)

- Do not blindly lift code from other sources
- When you use existing software libraries, state the nature and scope of their use clearly in your project spec and report
- Do not blindly lift text (e.g., for project report) from other sources
- Please cite references for specific statements and be thorough

Course participation (8%)

- You are welcome to discuss and collaborate extensively
  - Get to know each other, and me

- Meaningful class questions and technical discussion
- Insightful Discord/email questions or answers or follow-up discussion
- Discussion with me after the lecture or outside
- Supporting and helping each other grow in any way
  - e.g., sharing useful materials on class Discord

Course participation (8%)

- Class participation is a consistent and meaningful activity
  - Assessed throughout the semester
  - Not a one-time event or a checkbox
  - Intention to learn and support, not just a grade

- I’m happy to get to know you professionally and engage in technical discussions
  - If you require professional support from me later (e.g., recommendation letters), it’s a great way for me to know you better

Course Policies

Collaboration and Integrity policies

- This course welcomes discussion and collaboration
- Do
  - Ask questions on Discord or e-mail
  - Discuss projects and readings with me and with each other
  - Read references (textbooks, papers, Internet posts) widely
  - Acknowledge each other and all the references
- Report collaborations on programming homework & the project
  - Include who you talked to, references (including on the web) you consulted
  - Be as accurate and complete as possible

Policy on AI

- Specific use with specific attribution
- Use it to resolve specific questions and concerns
  - You are not permitted to use it to produce the entire solution for homework or your project
- List all your prompts and the service you used
  - And discuss how you fix the output

GenAI is experimental

- Do not become a victim of hallucination
- AI makes mistakes: you must learn how to identify problems and fix them
  - You still need to know what you’re doing
  - Learn and understand the fundamentals well
- Use AI as an assist, not to drive your entire solution
- Like anything else on the Internet, you need to think critically

Collaboration and Integrity policies

- All your written (coded) work must be your (team's) own
  - Understand the problem deeply and produce your own solutions

- Do not
  - blindly lift or incorporate other student solutions
  - look at other people's code or solutions
  - copy solution code from the web (e.g., other people's GitHub projects)
  - use generative AI to produce a full solution
  - post programming homeworks or quizzes (questions or solutions) on GitHub, Chegg, CourseHero, etc.

Rutgers takes academic dishonesty very seriously.

Violation of academic integrity at the graduate level is especially serious. Consequences include suspension and expulsion.

We will run plagiarism detection tools on all submitted materials.

If you are ever in doubt, ask me first.

Late policy

- Don’t be late
- If you must be late, inform us in advance
- If you cannot inform us in advance (e.g., medical), provide official medical note of absence through the University
- Unexcused late submissions will result in losing significant fraction of points

24/7 Grading Policy

- You may not dispute a grade or request a regrade before 24 hours or after 7 days of receiving it

- Please contact us if you have a legitimate re-grading request:

  - After 24 hours of receiving the grade: Please take the time to review your case before contacting me

  - Before 7 days have elapsed: we don’t want to forget what the quiz/homework was all about.

Help, Accommodations, etc.

- I'll make every effort to accommodate reasonable requests that support your learning better

- sn624@cs.rutgers.edu

- I am committed to help you succeed in this course.

Next steps

- Sign up for class Discord: link on canvas home page
- Warm up on C/C++ programming this week
  - e.g., linked lists, basic TCP and UDP socket programming
- Next two lectures (1/29, 2/05) online: Video lectures to be posted on the syllabus page
- First quiz on 2/12: covering the first three lectures
  - Thereafter every week
- Meet each other and form project teams