Computer Networks

RUTGERS
UNIVERSITY | NEW BRUNSWICK

A collection of computers exchanging information

The Internet

prime

ROBLOX

UBER

aws

BitTorrent™

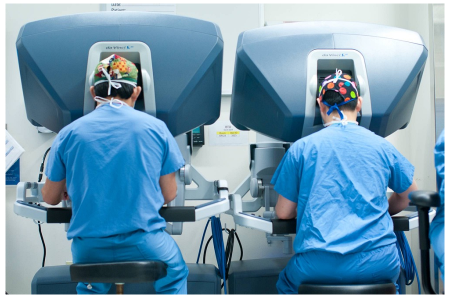
Instagram

The New York Times

WhatsApp

NETFLIX

Spotify®

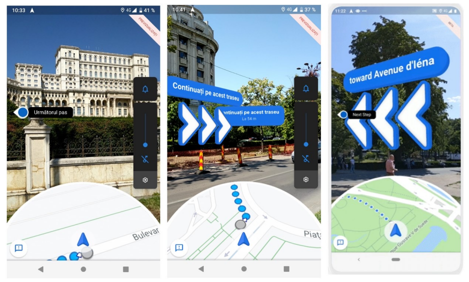


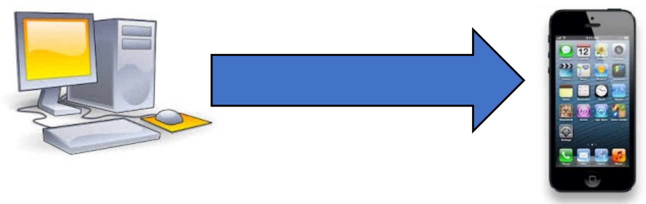
A collection of computers
exchanging information

Why is that useful?

- Communicating with other human beings
- Processing vast quantities of data
- Learning what’s going on in the world
- Education
- Transacting and doing business
- Entertaining ourselves
- Living sanely in an isolated world
- Moving and operating in the physical world

Evolution of Internet applications





1992
ftp
Web
email

1996
chat
Games
IM
Yahoo!

2000
news
Blog
Search
amazon

2004
Music
itunes
Games
search
Google

2008
Wikipedia
Craiglist
Youtube

2010-2020
UBER
airbnb
NETFLIX
Spotify

2020--
NVIDIA
OMNIVERSE
FORTNITE
MINECRAFT

Gather

Text-heavy

Multimodal media

Augment physical world

Replace phy world

There’s a science to communication

- What language should machines speak?
- How to partition functionality? “Who” should do “what”?
- How to make communication effective?
    - Achieve better scale, performance, resource efficiency
    - Evolve to address new needs over time
- How to grow organically? include more communicating parties
    - Make it easy for humans to build and manage?
- How to make communication worthy of societal trust?

Computer Networks

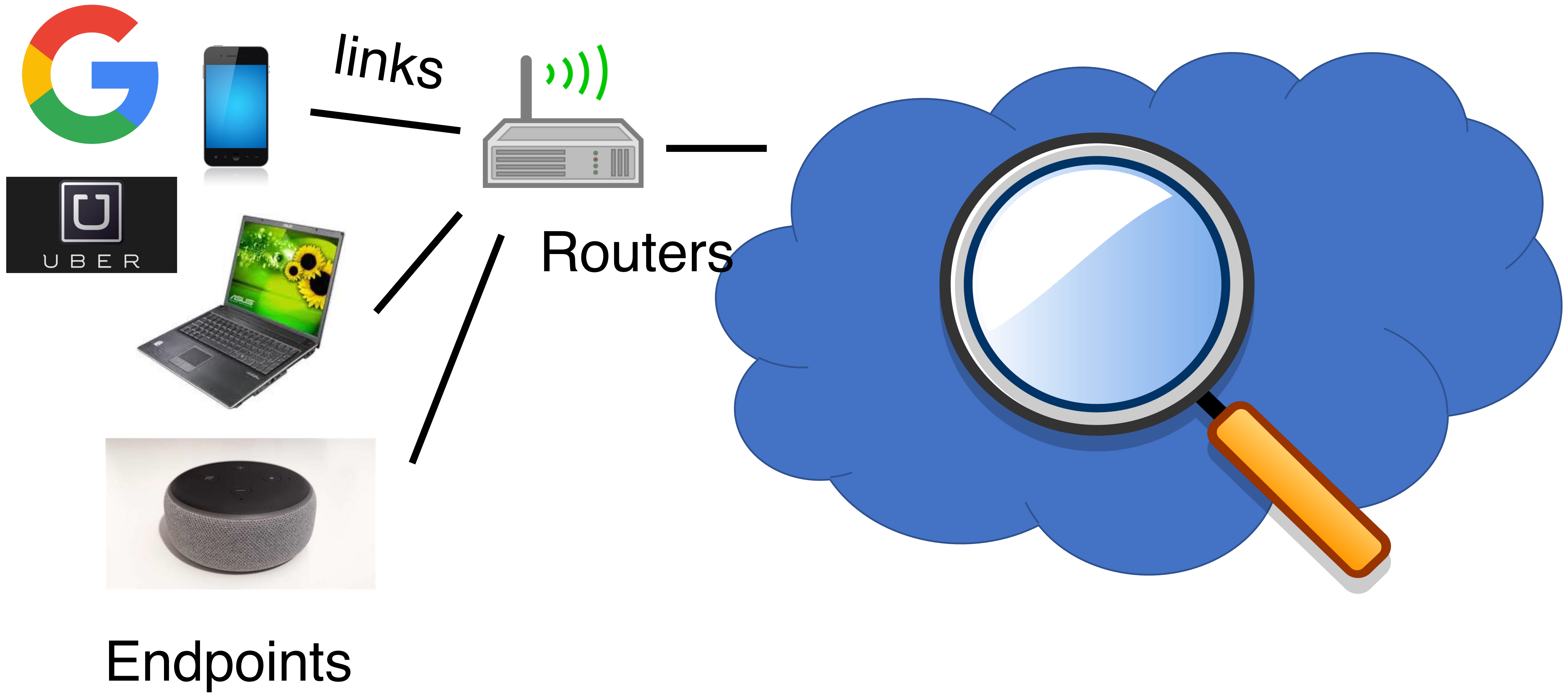
Principles and algorithms by which communication software & hardware are organized

Design of foundational artifacts of the Internet and other modern networks

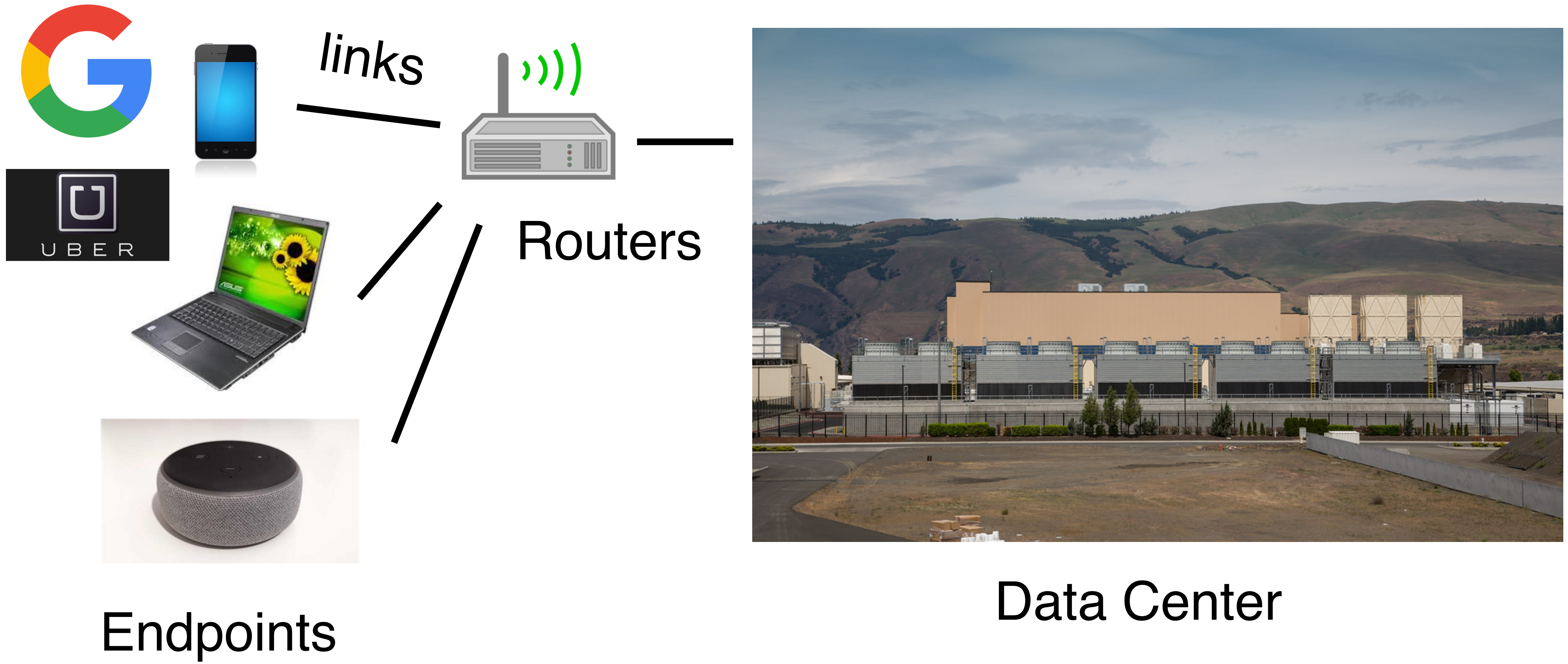
Readings and programming assignments

There are different kinds of computers and networks

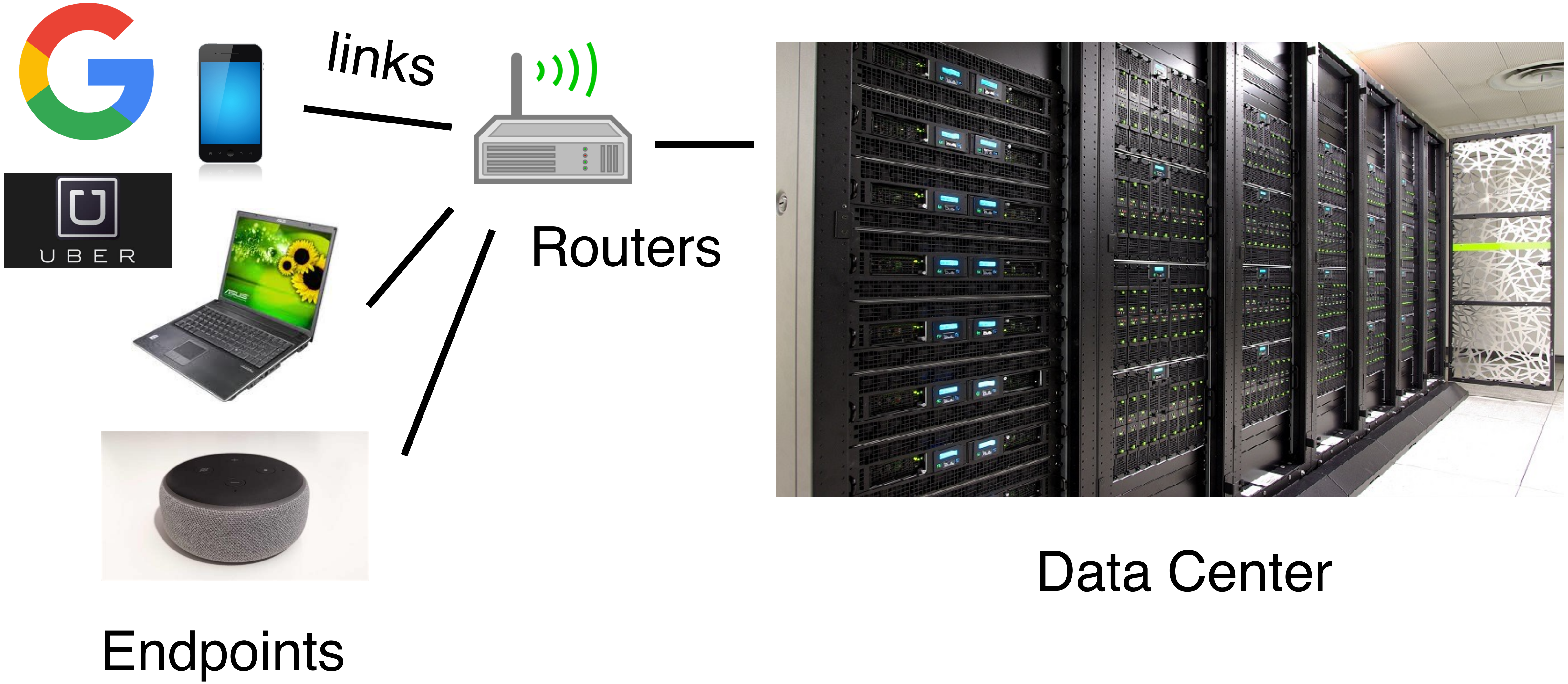
Your experience of the Internet



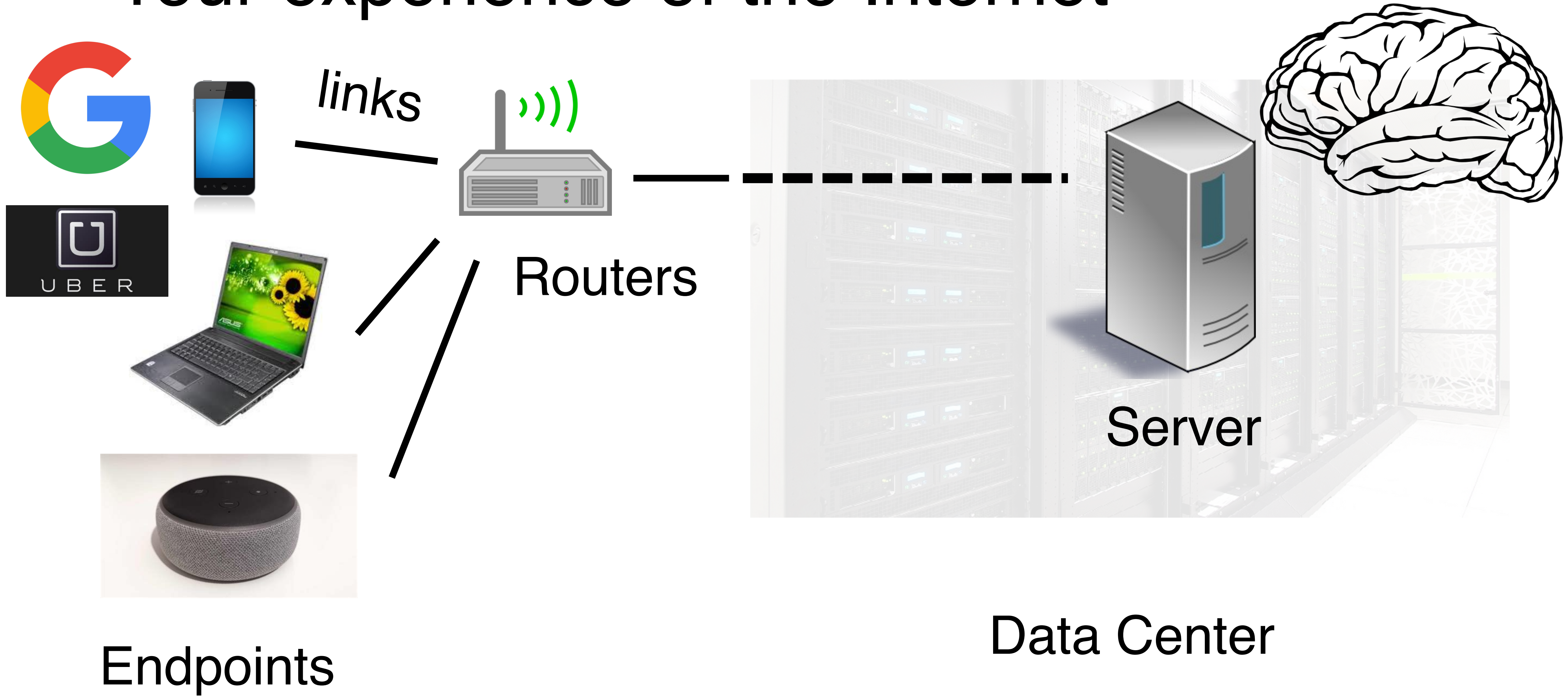
Your experience of the Internet



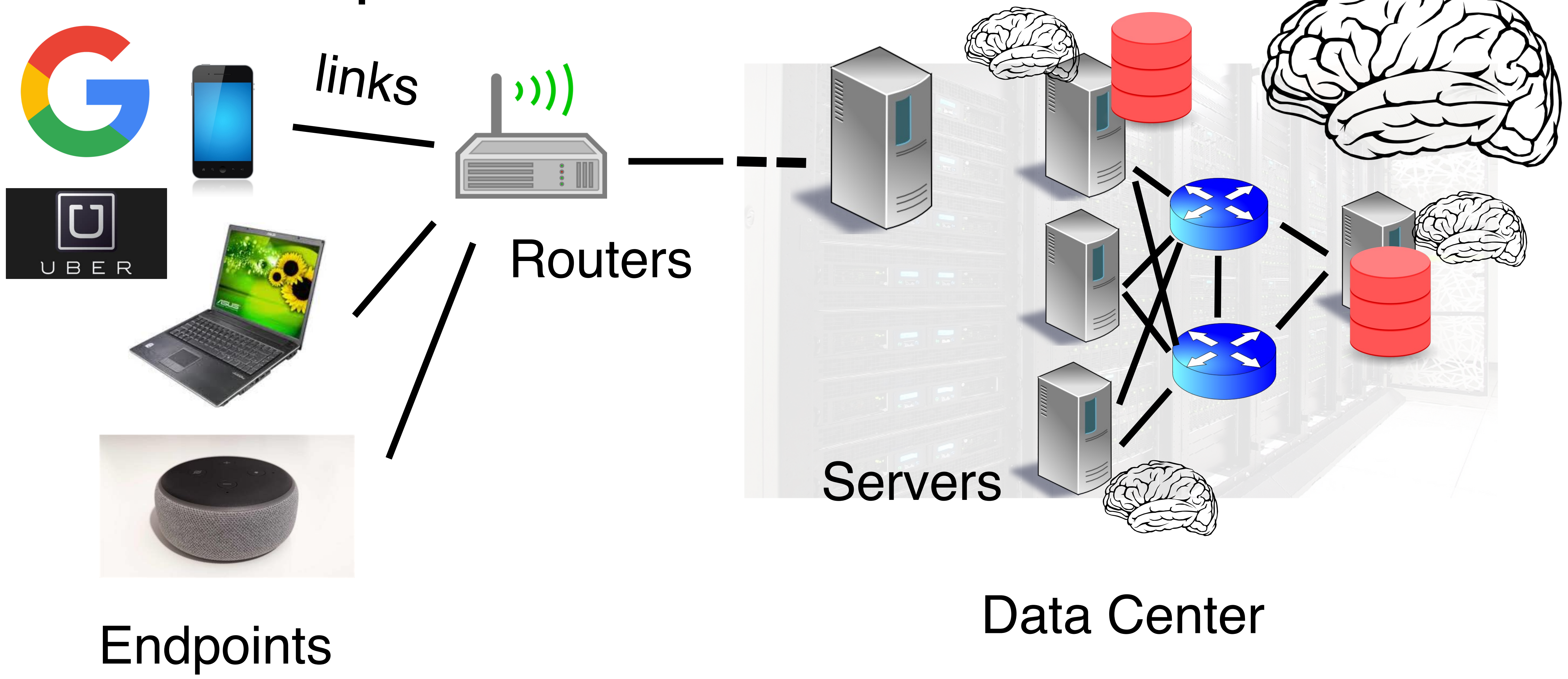
Your experience of the Internet



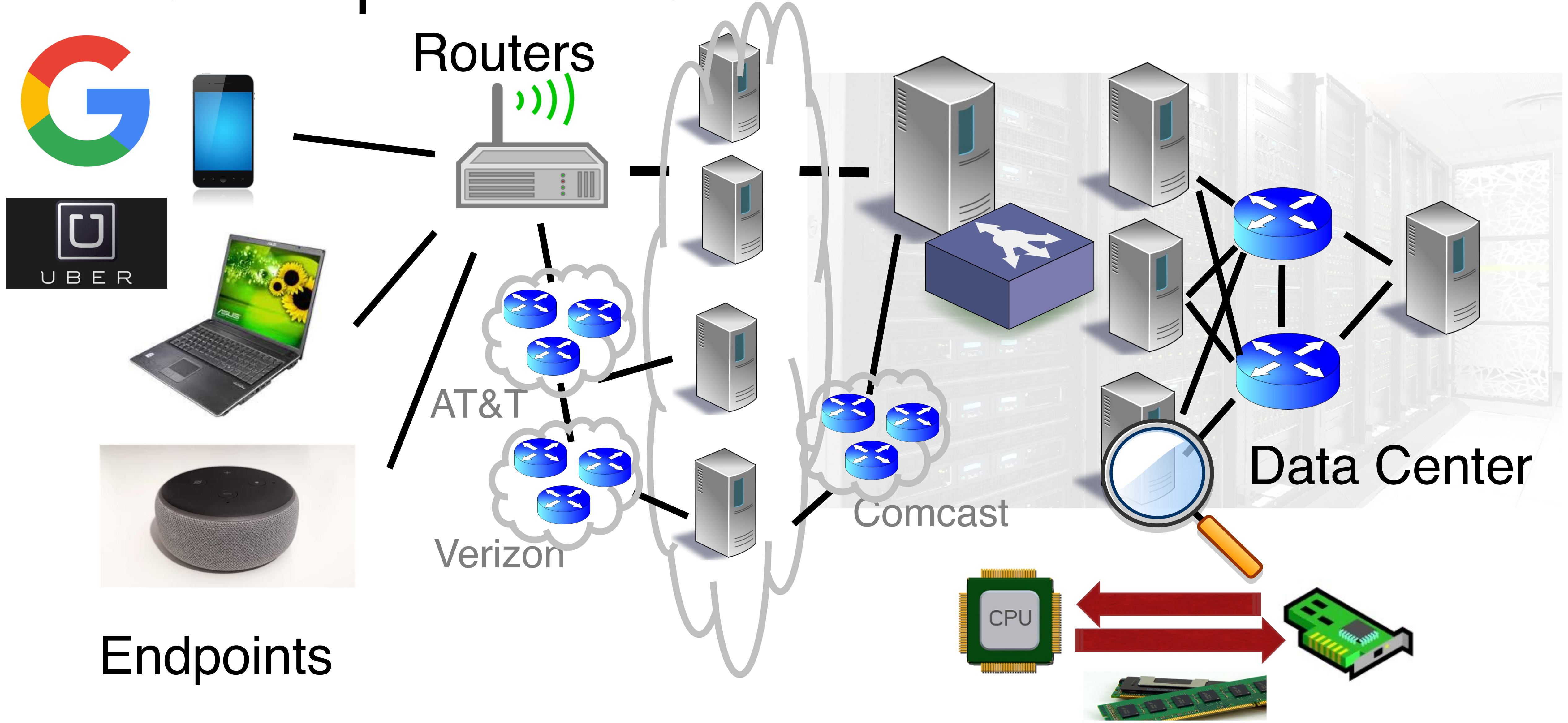
Your experience of the Internet



Your experience of the Internet

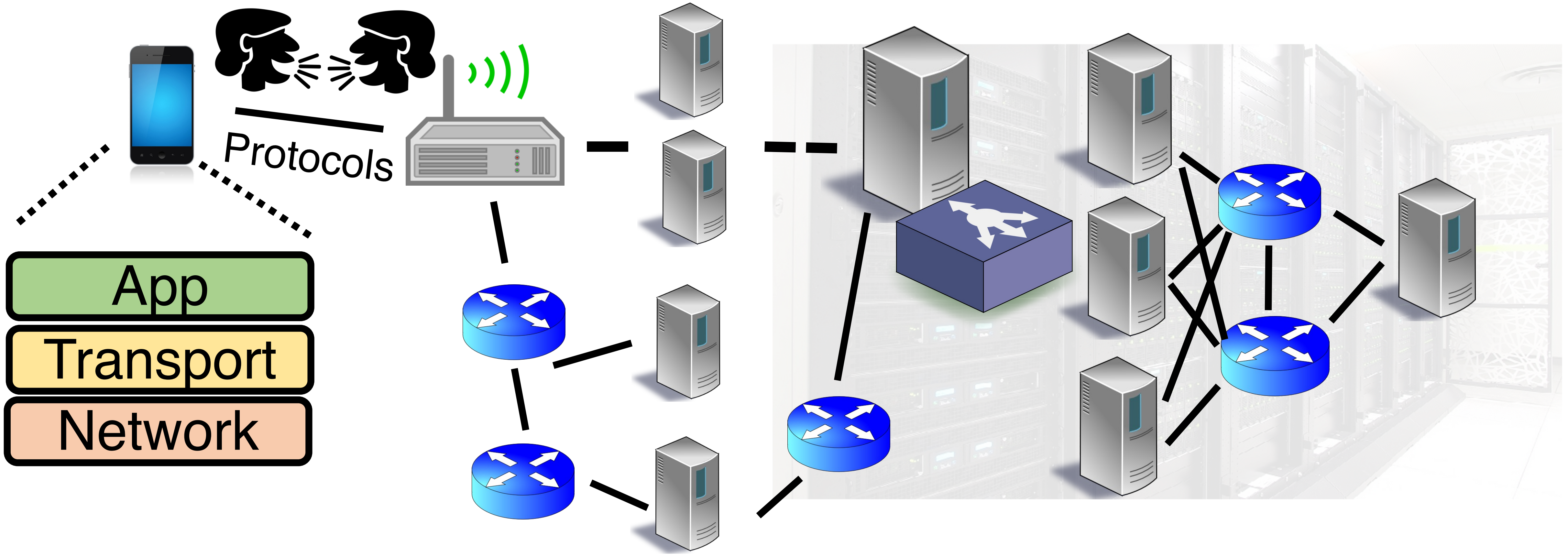


Your experience of the Internet



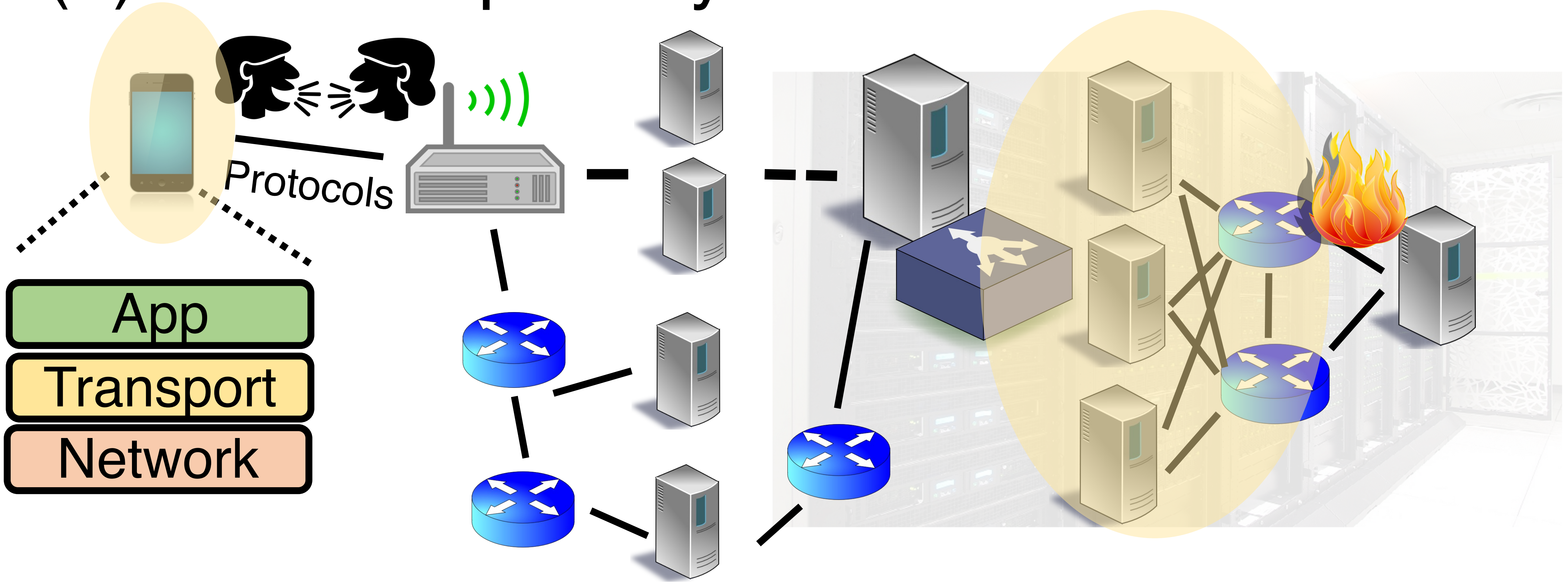
Course content overview

(1) Internet architecture



Decomposition of functionality

(2) The transport layer

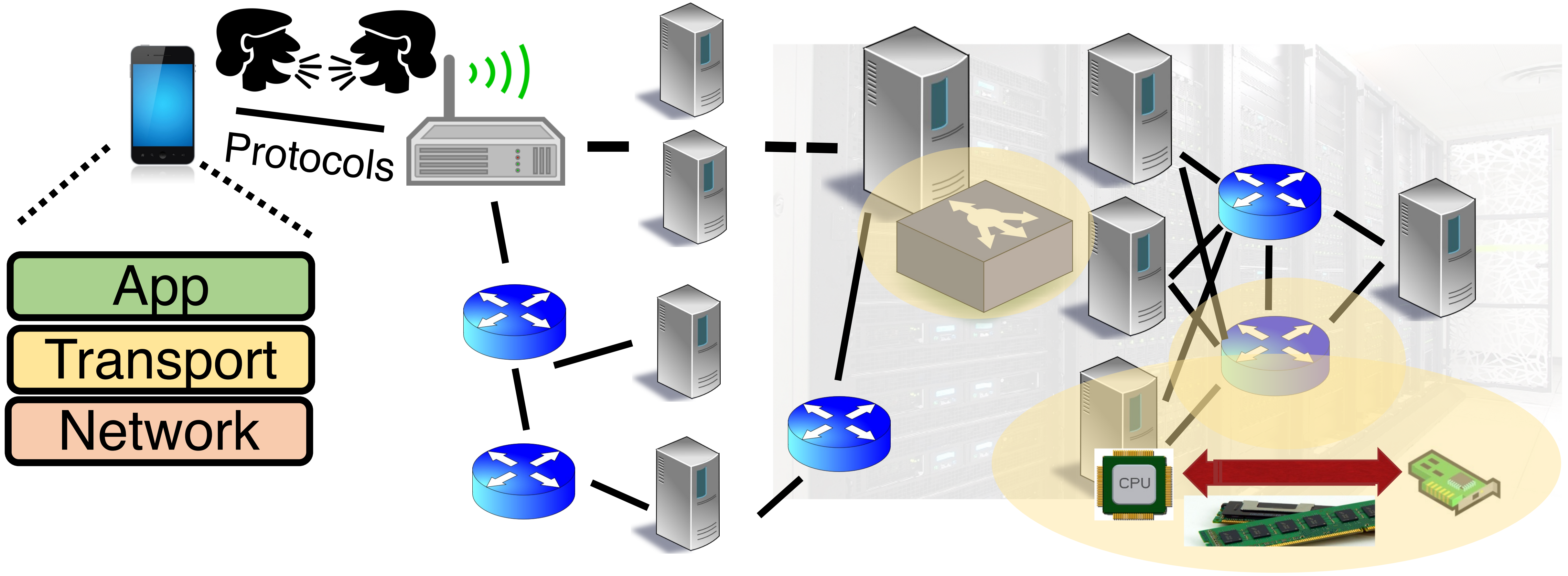


Basic guarantees

Design principles of widely deployed TCP protocols

Packet scheduling

(3) Network data plane

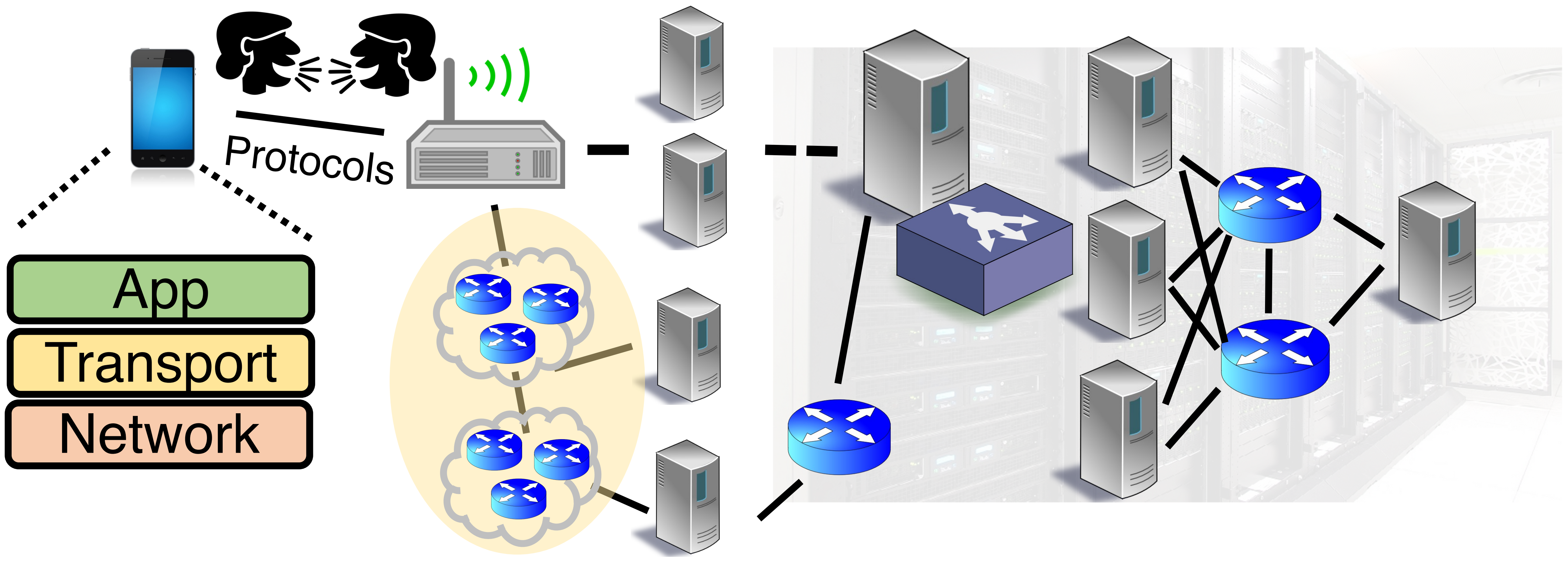


Modern router architectures

High-speed software packet processing

Network functions

(4) Network control plane

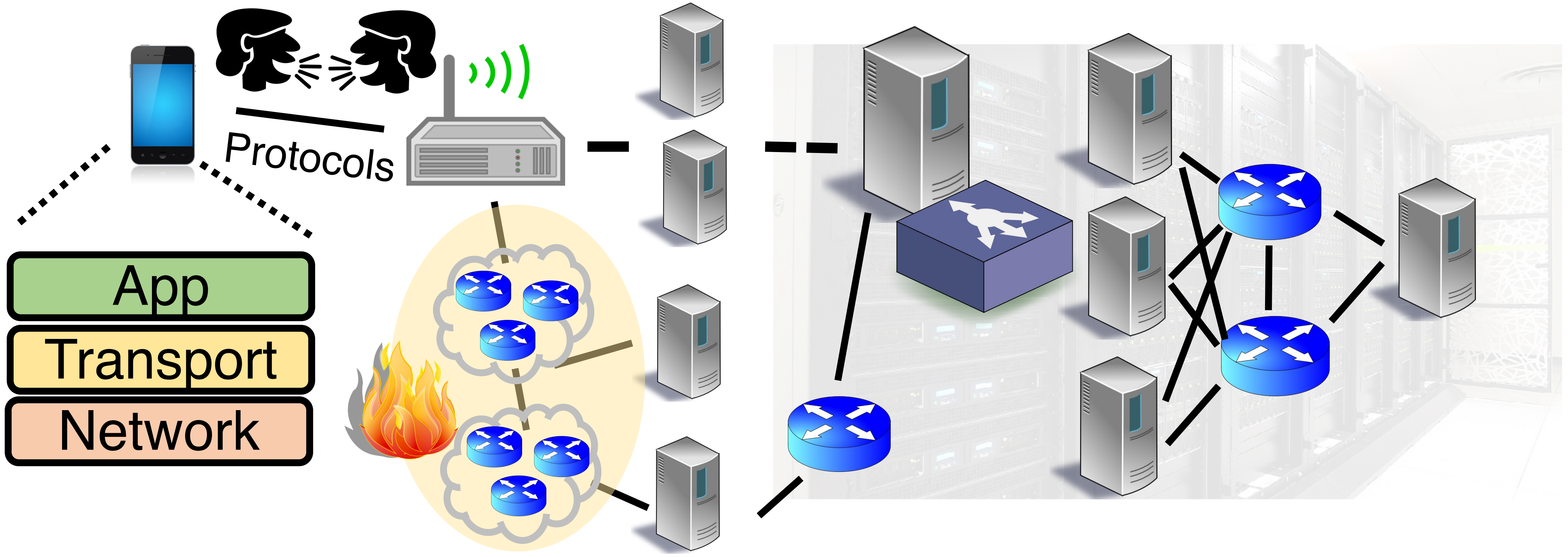


Routing

Software-defined networking

Interconnection (BGP)

(5) Verification



Data plane verification

Control plane verification

Concrete tools and takeaways

- Understanding, configuring, and implementing transport (TCP)
- Writing SDN data and control plane programs (P4)
- Experience with real distributed routing protocols (BGP)
- Writing fast, verified packet-processing code (eBPF)

Why might all this be useful to you?

- Intellectually rewarding
  - Principles in designing distributed computer systems
- Pragmatic
  - Modern software requires an understanding of networking
- Real world utility
  - Many programs you use today designed by programmers like you
- Academic impact
  - Still an evolving discipline with many open problems (and awards)

Fun, profit, impact, altruism

Logistics

About us

- Faculty instructor: Srinivas Narayana
    - http://www.cs.rutgers.edu/~sn624
    - sn624@rutgers.edu
- Teaching Assistant: Bala Murali Komanduri
- Office hours TBA
- Canvas and Piazza
- Busch campus SEC 202 on Wed 8:30—11:30

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 office hours regularly to discuss material
  - Use Piazza
- Happy to help you acquire necessary background
  - Ask me for support materials

Grade components

- Programming assignments (60%)
- Quizzes (40%)

Quizzes (40%)

- 4 in-class quizzes, held during lecture
- Exact dates to be announced
  - tentatively, weeks 4, 7, 11, 14
  - 45 minutes to 1 hour each
- Cover lectures and readings
- Open book, but paper-based materials only
  - No sharing, Internet access, mobile use during exam

Readings

- 6—7 technical papers to read deeply over the term

- 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
    - Development: implementing new system 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 13+ years now
    - And I still sometimes spend 5+ hours or even an entire day
- 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?)

How to read a paper? -- Keshav Srinivasan

(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

(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 assignments (60%)

- 4, released and handed in over Canvas
  - TCP, P4, BGP, eBPF
- Sizable, substantial programming
- C language, and some domain-specific languages (P4, BGP)
- Solo or teams of 2

Programming assignments

- We’ll give you access to a Linux VM with sudo privileges
- Assignments Due roughly every 3 weeks
    - Tentatively, Mondays after lectures 5, 8, 11, 14

Programming assignments

- Please follow all instructions carefully and exactly

- You will lose significant points if:
  - We are unable to compile or run your code
  - We do not receive your submission in a timely fashion

Programming assignments

- You can talk to others, however all submitted code must be your own
- Do not blindly lift code from stack overflow, GitHub, chatGPT, …
- Do not post solutions on GitHub or elsewhere
- Incorporate learning from other sources and produce your own solutions
- Mandatory to state collaboration & references in an attached report

Collaboration and Integrity policies

- This course welcomes discussion and collaboration
- Do
  - Ask questions on Piazza
  - Discuss projects and readings with me and with each other
  - Read references (textbooks, papers, Internet posts) widely
  - Acknowledge each other and all the references
- Use collaboration prompts on programming homeworks; project
  - Include who you talked to, references (including on the web) you consulted
  - Be as accurate and complete as possible

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 solutions
  - look at other people's code or solutions
  - copy code from the web (e.g., other people's GitHub projects)
  - use generative AI (e.g. chatGPT)
  - 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 regrading 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/project 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

- Look out for canvas page, sign up for piazza (link TBD)
- Lecture and recording will be posted online
- Warm up on C programming
- Meet each other to form teams for programming