What you could do from here

CS 352, Lecture 25

http://www.cs.rutgers.edu/~sn624/352-S19

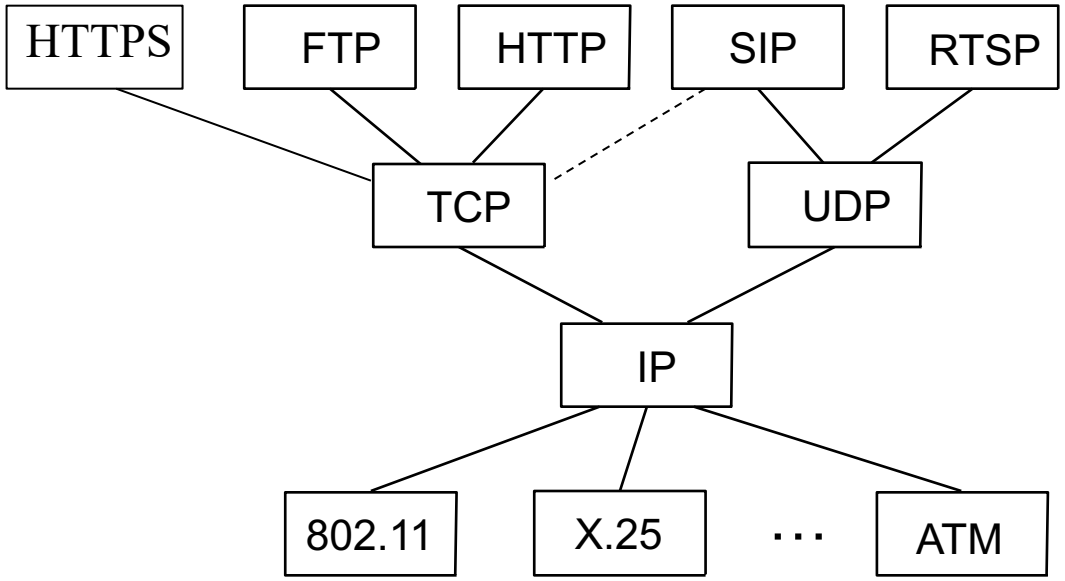
Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

You’ve gone through 24 lectures of 352

The protocols of the Internet

- **Layering** and **Hourglass Design**



Protocol layers: Application layer

- Apps closest to the user: HTTP, SMTP, multimedia
- Helper protocols: DNS

- Deal with human concerns
- Readability of host names to reach certain services
- Often, protocols are in human readable plain text too
  - HTTP, SMTP, DNS
- Optimized for human-perceivable performance
  - The web, VoIP, mail, apps have different “optimizations” built into them

Protocol layers: Transport layer

- Providing guarantees for applications over a best-effort network
  - Transport layer runs on hosts
- Providing connectivity between applications
  - Multiplex data from multiple apps going out of a given machine
  - Demultiplex data coming into a machine to different apps
- UDP: main function is de/multiplexing
  - Also, error detection using checksums
  - Simple and lightweight
- TCP: reliable, in-order delivery
  - Much more heavyweight

Protocol layers: The transport layer

- TCP reliable delivery mechanisms
  - ACKnowledgments
  - Timeouts
  - Retransmission strategies
- TCP ordered delivery mechanisms
  - Sliding window
  - Flow control
- TCP efficiency and fairness
  - Congestion control: slow start, additive increase/multiplicative decrease

Protocol layers: The network layer

- Providing connectivity between machines across the Internet
  - Data plane: Moving data from point to point
  - Control plane: compute how data plane should move data

- Network layer runs on every host and every router

- Main issues: (1) Addressing
  - Machine addresses aren't necessarily human friendly (ex: IPv4/v6)
  - Addresses associated with network interfaces, not hosts

Protocol layers: The network layer

- Main issues: (2) Router design
  - High performance data movement between different network interfaces
  - High-speed destination-based forwarding
  - Longest-prefix-based matching

- Main issues: (3) Routing
  - Link-state, distance-vector, path-vector routing
  - Must try to avoid suboptimal paths and routing loops
  - Try to compute and converge fast

Protocol Layers: The link layer

- Provide connectivity between machines over the physical medium
  - A co-ax cable, optic fiber, the air inside a room
- Main issues: (1) Data encoding
  - Must translate bits 0-1 from software into physical signals
- Main issues: (2) Error detection
  - the media are not without fault!
  - Parity bits

Protocol layers: The link layer

- Main issues: (3) Multiple access
  - Partitioning the medium's resources
  - Random access protocols: exponential back-off
  - Taking turns

- Main issues: (4) Handling nuances of wireless media
  - Fading, hidden terminals, half-duplex
  - Link-layer reliability
  - Waiting for fixed periods of time to transmit despite idle medium
  - Explicit reservation (RTS/CTS), resulting in "taking turns"

Protocol layers: The link layer

- Main issues: (5) Supporting mobility of hosts
  - You've got to have "roots" somewhere
  - Indirect and direct routing using home and foreign agents
  - A result of requiring addresses to depend on point of attachment
    - And requiring apps to bind to addresses upon initiating a conversation

Other important related topics

- Securing our communications

- An important (and peculiar) application:
  - Multimedia transfers

Securing communication

- Security properties:
  - Confidentiality, integrity, authenticity, non-repudiation

- Cryptography
  - Obfuscating messages to all except the intended sender and receiver
  - Symmetric key crypto: substitution and permutation
  - Public key crypto: pairs of secrets to get around shared secret distribution

- Building authenticity and integrity
  - Message authentication codes (MACs), digital signatures
  - Transport layer security (TLS): real example of bringing all the tools together

Multimedia transfers

- Streaming (ex: Netflix) and conversational (ex: Skype) media
- Peculiar characteristics:
  - Delay-sensitivity, loss tolerance, varying quality levels for same data
- Application-level adaptations:
  - Client-side buffering
  - Adaptive playout
- System-level adaptations:
  - Relay-based routing
- Network-level adaptations: QoS
  - Resolve contention at router queues
  - Priority queueing, fair queueing, leaky buckets, token buckets

The big picture

- Computer networks are a stack of layers
  - Built that way for modularity
  - Each layer does one set of functions very well
  - Each layer depends on the layers beneath it
  - But modularity can sometimes result in inefficiency

- Many general and useful principles
  - Borrowed from real life (ex: listen before you speak)
  - Borrowed from systems in general (ex: use indirection for flexibility)
  - Applicability goes the other direction as well (ex: how to authenticate a person talking to you over the phone?)

You've gone through 24 lectures of 352…

now what?

A few possibilities

- The course is over. Go about life as usual
- Apply your new-found skills to solve a problem you care about
- Develop a deeper understanding of these technologies
- Consider improving the state of the art

(1) Go about life as usual

- This material will still be useful for a good “CS life”
  - Deeper understanding of the abstractions you use (ex: sockets. How big should socket buffers be?)
  - Why we need certain technologies (ex: HTTPS, digital signatures)
  - A more nuanced understanding of real issues (ex: how are ISPs violating net neutrality using QoS mechanisms?)
  - Enhanced abilities to troubleshoot your own tech problems (ex: why is website X not loading? Is it my Internet connection or the other end?)

(2) Solve a problem you care about

- Most concepts we discussed are supported by real, open-source, freely-available software
  - Many technology and protocol specifications are freely available (RFCs)
  - Linux kernel source code
  - Open source software routers
  - Open source browsers (Mozilla), mail clients (mutt), video clients
  - Most protocols are “open source”
  - Free or cheap infrastructure: EC2 servers, domain names, certificates

(2) Solve a problem you care about

- Improve video chat performance?
- Improve the usability of secure email?
- Improve web transfer performance?
- Make it easier to diagnose home wifi issues?
- <your idea here?>

(3) Deepen your understanding

- Fall 2019: CS 552 “Computer Networks”
  - A deeper take on the fundamentals of Internet design
  - https://www.cs.rutgers.edu/~sn624/552-F18/index.html

- Some questions we’ll talk about:
  - How does Google serve your web traffic so quickly?
  - How do large networks verify that their networks are functioning well?
  - How are high-speed routers built?
  - What transpires inside large data centers run by Amazon & Facebook?
  - How should you optimize your web-app to load faster on browsers?
  - … and more

(3) Deepen your understanding

- 552 requires
  - Paper readings
  - Deep understanding
  - Engaging in lively class discussions

- You will be assessed mainly through a software project
  - On a topic of your choice that *you* are excited about
  - The only requirement is that it must be connected to class material

- Every assessment is “take home”; there will be no exams.

(4) Push the state of the art

- Many of you will embark on CS-related careers
  - Can use your 352 know-how to do cutting-edge work in your org
- Some of you may consider graduate school
  - Networking is a great area to work in
  - Some of the most cited papers in CS, at least 2 Turing awards
  - 552 is a good place to lay a foundation for this path
- If you're interested to work on small research projects during your remaining time @ Rutgers, come talk to me. ☺

Now it’s time for…

- Any questions, general or specific, about the course
- Any topics you’d like me to go over again
- Any feedback you’d like to voice about this course
- Please don’t be shy