

Lecture 2:

Links and Signaling

CSE 123: Computer Networks

Aaron Schulman



DISCUSSION @2pm Friday



UCSD CSE



Our Problem

- Communications is complicated
 - ◆ Modulation and encoding bits
 - ◆ Splitting sequences of bits into packets
 - ◆ Fixing errors
 - ◆ Controlling access to network
 - ◆ Routing
 - ◆ Recovering from lost messages
 - ◆ Etc....
- Really hard to think about all of this and get it right
- Not all applications need all of it
- How to achieve interoperability?



Protocol Standardization

- Communicating hosts speaking the same protocol
 - ◆ Standardization to enable multiple implementations
 - ◆ Or, the same folks have to write all the software
- Internet Engineering Task Force
 - ◆ Based on working groups that focus on specific issues
 - ◆ Produces “Request For Comments” (RFCs)
 - » Rough consensus and running code
 - » After enough time passes, promoted to Internet Standards
- Other standards bodies exist
 - ◆ ISO, ITU, IEEE, etc.

Layering: A Modular Approach



- Sub-divide the problem
 - ◆ Each layer relies on services from layer below
 - ◆ Each layer exports services to layer above
- Interface between layers defines interaction
 - ◆ Hides implementation details (encapsulation)
 - ◆ Layers can change without disturbing other layers (modularity)
- Interface among peers in a layer is a **protocol**
 - ◆ If peers speak same protocol, they can interoperate

UCSD students wins prize from IETF!



News 2021

Applied Networking Research Prize

January 29, 2021 cnsmember

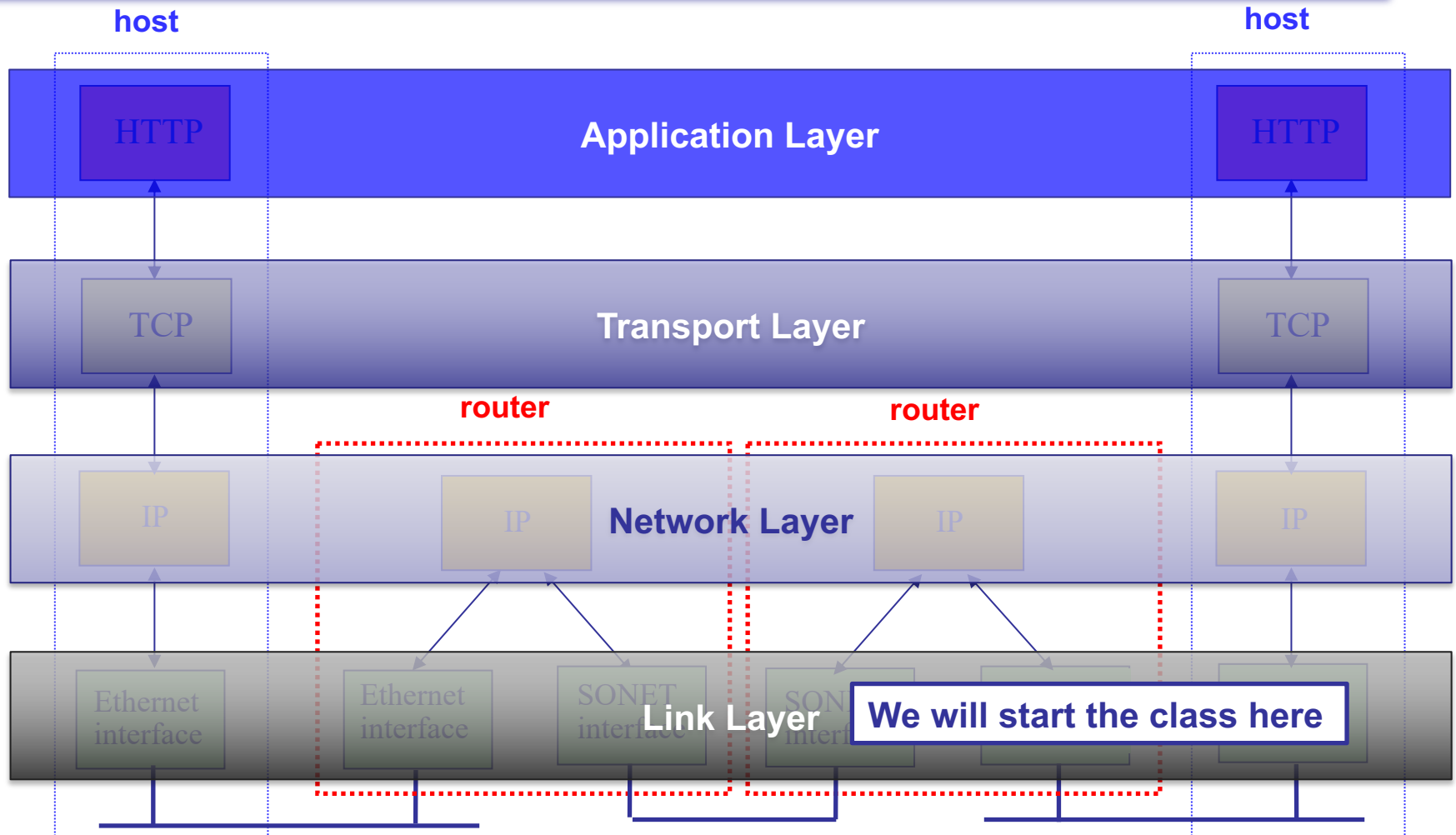
CSE/CNS graduate student [Audrey Randall](#) was recently awarded the [Applied Networking Research Prize](#) (ANRP) for her work on DNS caching and privacy. The ANRP winners for 2021 were selected from 76 nominations. They were reviewed according to a diverse set of criteria, including scientific merit, relevance to [IETF](#) and/or IRTF activities, and the nominee's potential to impact the community.

Audrey Randall, Enze Liu, Gautam Akiwate, Ramakrishna Padmanabhan, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman, "[Trufflehunter: Cache Snooping Rare Domains at Large Public DNS Resolvers](#)," Proceedings of ACM IMC 2020.





TCP/IP Protocol Stack



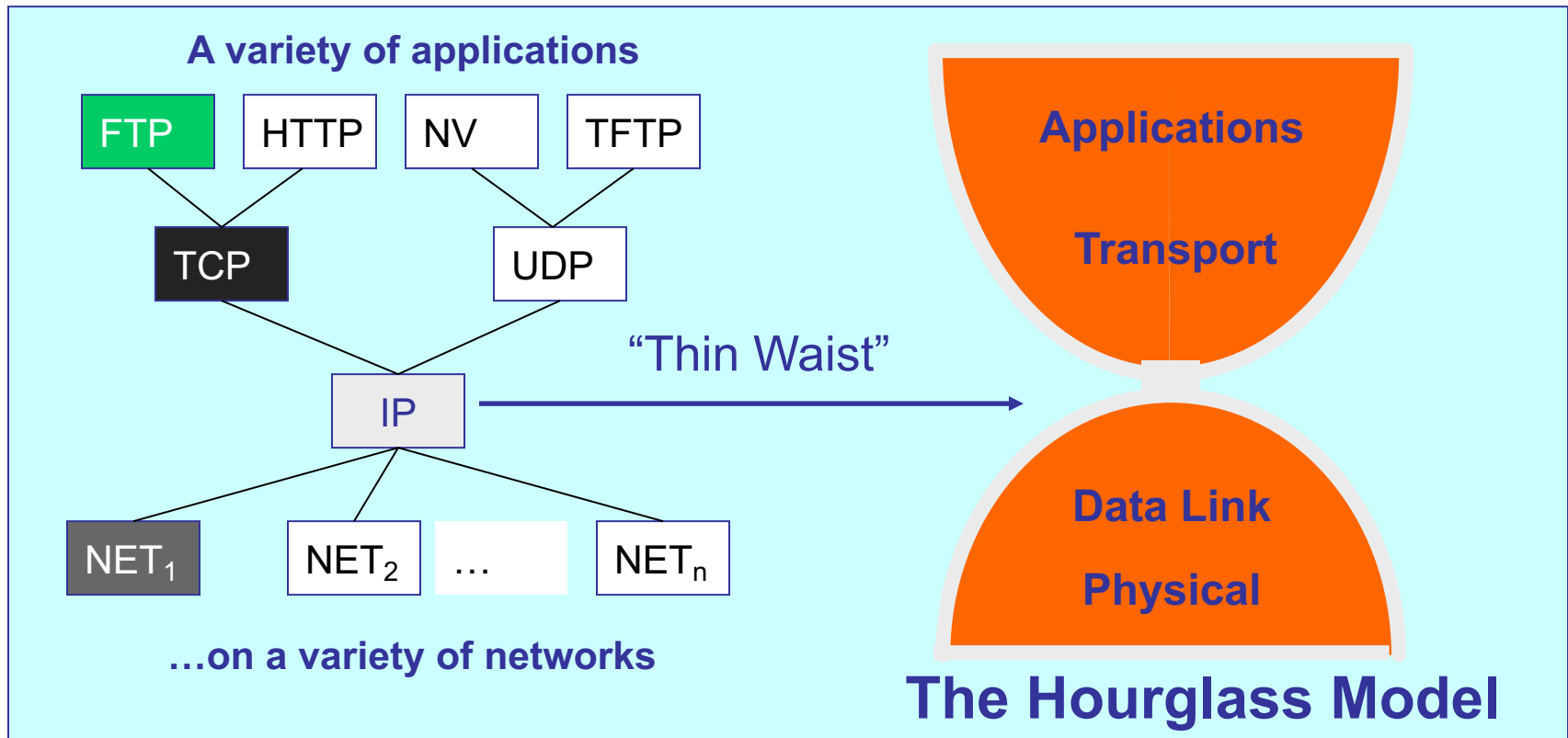


Key Design Decision...

- ...how do you divide functionality across the layers?
- **End-to-end argument [Saltzer84]**
 - ◆ Functionality should be implemented at a lower layer iff it can be **correctly** and **completely** implemented there
 - ◆ Incomplete versions of a function can be used as a performance enhancement, but not for correctness
- Early, and still relevant, example
 - ◆ ARPAnet (link layer) provided reliable link transfers between switches
 - ◆ Was this enough for reliable communication?
 - ◆ No, packets could still get corrupted on host-switch link, or inside of the switches
 - ◆ Hence, still need reliability at higher layers



Internet Protocol Suite

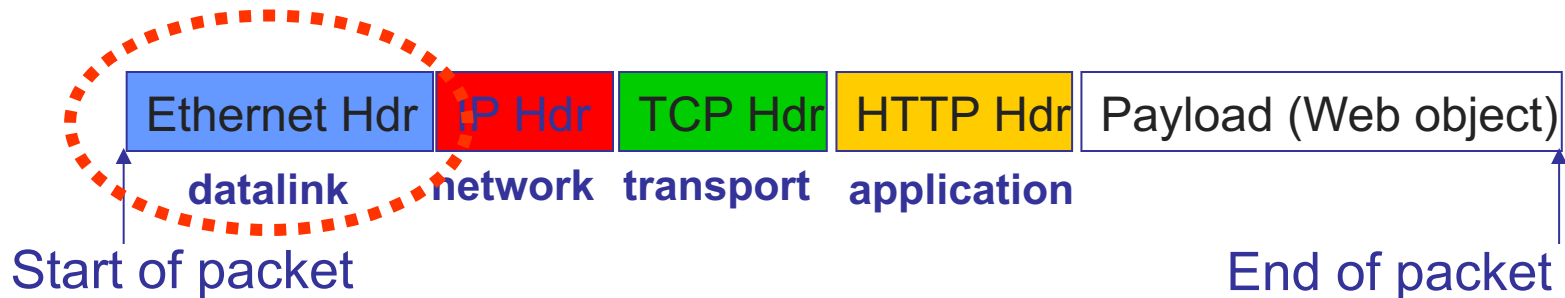


The success of the Internet (and its problems) can largely be attributed to the “thin waist”



Encapsulation via Headers

- Typical packet in Web transfer



- Notice that layers add overhead
 - ◆ Space (headers), effective bandwidth
 - ◆ Time (processing headers, “peeling the onion”), latency

Later: Phy/(MAC)Link layer



- Signal encoding
 - ◆ Encode binary data from source node into signals that physical links carry
 - ◆ Signal is decoded back into binary data at receiving node
 - ◆ Work performed by network adapter at sender and receiver
- Media access
 - ◆ Arbitrate which nodes can send frames at any point in time
 - ◆ Not always necessary; e.g. point-to-point duplex links



For now: (Data) Link Layer

- **Framing**

- ◆ Break stream of bits up into discrete chunks
- ◆ **Multiplexing**
 - » Determine appropriate destination for a given frame
 - » Also not always required; again, point-to-point

- **Error handling**

- ◆ Detect and/or correct errors in received frames



Today's Focus: Framing

Break down a stream of bits into smaller, digestible chunks called **frames**

- Identifies the beginning and end of a piece of information
- Allows the link to be shared (Multiplexing)
 - ◆ Multiple senders and/or receivers can **time multiplex** the link
 - » One frame sent at a time
 - ◆ Each frame can be separately addressed (different src/dest)
- Provides manageable unit for error handling
 - ◆ Easy to determine whether something went wrong
 - ◆ And perhaps even to fix it if desired

Where is the beginning/end?



01001001001000000111010001101000011010010110111001101011001000000111010001101000011000010111010000100000010010
01001000000111001101101000011000010110110001101100001000000110111001100101011101100110010101110010001000000111
00110110010101100101000101001000001001000000110011101110010011000010111000001101000001000000110110101101111011
1001001100101001000000110110001101110111011001100101011011000111100100100000011101000110100001100001011011100
01000000110000100100000011101000111001001100101011001010010111000010100100000100100000011101000111001001100101
01100101001000000111011101101000011011110111001101100101001000000110001101110010011101010110001101101001011000
01011011000010000001110000011100100110111101110000011001010111001001110100011110010001010010010010111001100100
00001101100011011110110111101110000001011010110011001110010011001010110010100100000011000110110111101101110011
01110011001010110001101110100011010010111011001101001011101000111100100101110000101001000001001000000111010001
11001001100101011001010010000001110100011010000110000101110100001000000110110101110101011100110111010000100000
01100010011001010010000001110011011101010111001001100101001000000111010001101111001000000111001101110000011000
01011011100001010010100110110111100100000011100000110000101100011011010110110010101110100011100110010000001100
0110110000101101110001000000111001001100101011000010110001101101000010000001100101011101100110010101110010011
11001001000000100110001000001010011100010111000010100100011001101001011100100111001101110100001011000010000001
11010001101000011001010010000001110010011011110110111101110100001000000110110101110101011100110111010000100000
01100010011001010010000001110011011001010110110001100101011000110111010001100101011001000010111000010100100001
00111100100100000010010010100010000101100001000000110100101110100001000000110100101110011001000000110010101101
10001100101011000110111010001100101011001000010111000010100100110001100101011000010111001101110100001011010110
00110110111101110011011101000010000001110000011000010111010001101000011100110010000001100110011100100110111101
10110100100000011100100110111101101111011101000010000001100001011100100110010100100000011101000111001001100001
01100011011001010110010000101110000101001001001011011100010000001110100011010000110010100100000011101000111001
00110010101100101001011000010000001110100011010000110010101110011011001010010000001110000011000010111010001101
00001110011001000000110000101110010011001010010000001110000011011000110000101100011011001010110010000101110000
10100100000100100000011011010110010101110011011010000010000001101001011100110010000001101101011000010110010001
10010100100000011000100111100100100000011001100110111101101100011010110111001100100000011011000110100101101011
0110010100100000011011010110010100101100000101001010000110100001100101011011100010000001100010011100100110100
10110010001100111011001010111001100100000011001100110100101110011001000010000001100001001000000111001101110
0000110000101101110011011100110100101101110011001110010000001110100011100100110010100101110



What's a Frame?



- Wraps payload bits up with some additional information
 - ◆ Header usually contains addressing information
 - ◆ Maybe includes a trailer (w/checksum—next lecture)
- Basic unit of reception
 - ◆ Link either delivers entire frame payload, or none of it
 - ◆ Typically some **maximum transmission unit (MTU)**
- Some link layers require absence of frames as well
 - ◆ I.e., minimum gaps between frames



Identifying Frames

- First task is to **delineate** frames
 - ◆ Receiver needs to know when a frame **starts** and **ends**
 - ◆ Otherwise, errors from misinterpretation of data stream
- Several different alternatives
 - ◆ Fixed length (bits) frames
 - ◆ Explicitly delimited frames
 - » Length-based framing
 - » Sentinel-based framing
 - ◆ Fixed duration (seconds) frames



#1 Fixed-Length Frames

- Easy to manage for receiver
 - ◆ Well understood buffering requirements
- Introduces inefficiencies for variable length payloads
 - ◆ May waste space (padding) for small payloads
 - ◆ Larger payloads need to be **fragmented** across many frames
 - ◆ Very common inside switches
- Requires explicit design tradeoff
 - ◆ ATM uses 53-byte frames (cells)
 - ◆ Why 53? $48 + 5$. Why 48?

#2 Length-Based Framing



- To avoid overhead, we'd like variable length frames
 - ◆ Each frame declares how long it is
 - ◆ E.g. DECNet DDCMP
- What's the issue with explicit length field?
 - ◆ Must correctly read the length field (bad if corrupted)
 - » Need to decode *while* receiving
 - ◆ Still need to identify the beginning...



For Next Class

- Read 2.4
- Project 1 out on Friday