Networking Background

CMSC 414

October 30, 2017

General Overview

We are going to take a quick look at

- What a network protocol is
- The abstract design of the network
- The “7-Layer” network stack

Protocols

We've discussed a few protocols already

A **Protocol** is an agreement on how to communicate

Specifies the *syntax*
- Format of messages
- Order in which they're exchanged

Specifies the *semantics*
- What messages mean
- What to do on message send or receipt, or at specified times

Defines a *language* for communications

Protocol Example

Diffie-Hellman key exchange is a *protocol*

At time T_0, party A selects a random exponent, and sends a message to party B

On receipt of the message from A, B selects a random exponent, computes the shared value, and sends a message to A

On receipt of the message from B, A computes the shared value

To make this a full protocol, we'd have to specify

- How exactly the protocol is initiated
- The message formats (including header fields, identifiers, and sizes for everything)
- How receiving each possible type of message triggers further actions

Internet Protocol

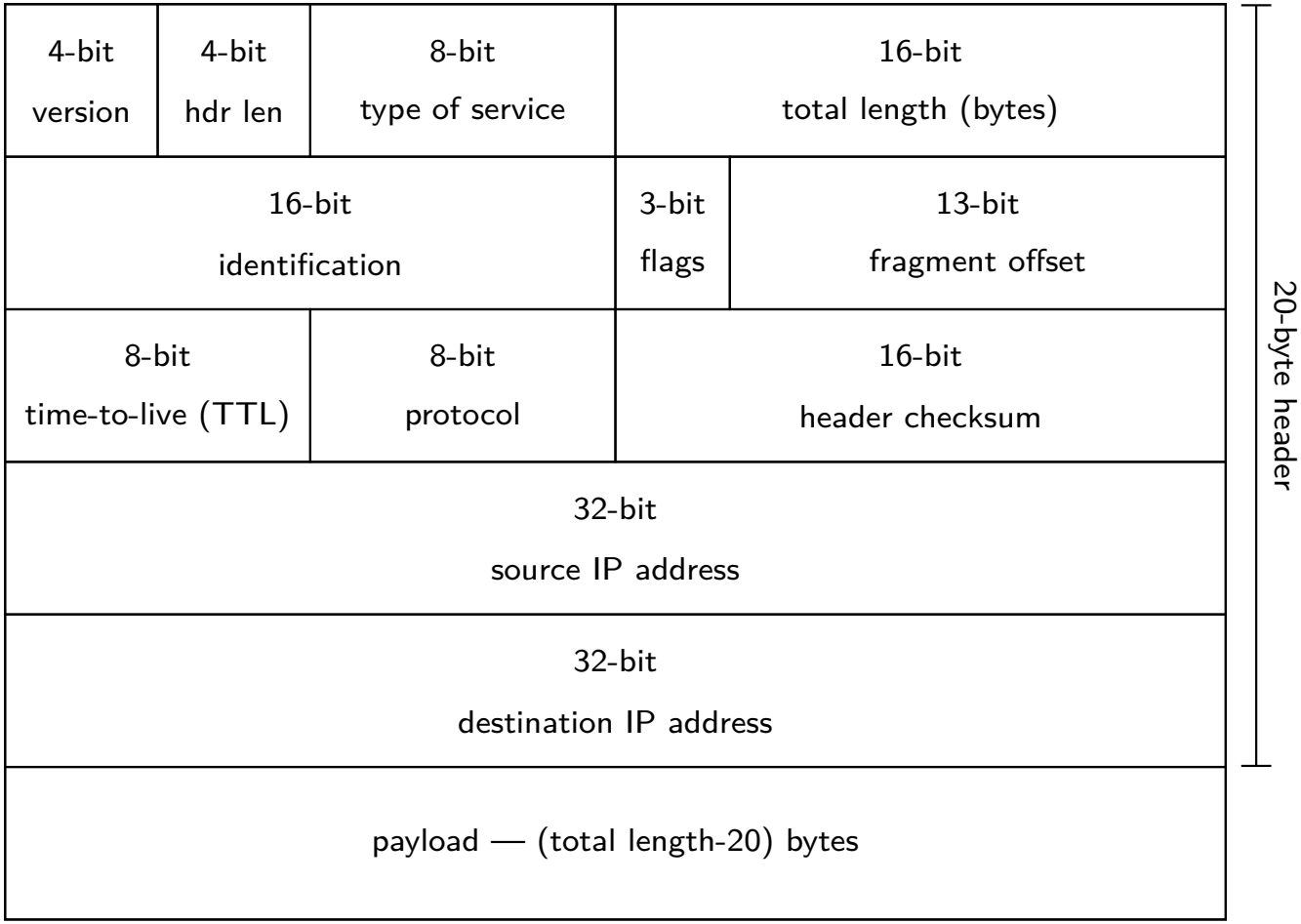
The **Internet Protocol** (IP) is what lets computers communicate around the world (and, to some extent, beyond)

Has a well-defined binary representation for transmission over the networks it comprises

Everything is big-endian (aka **Network byte order**)

Any IP-enabled host receiving an **IP packet** knows how to handle it, because of the protocol

IP Packet Format



End-to-End Principles

A and B are **End hosts**

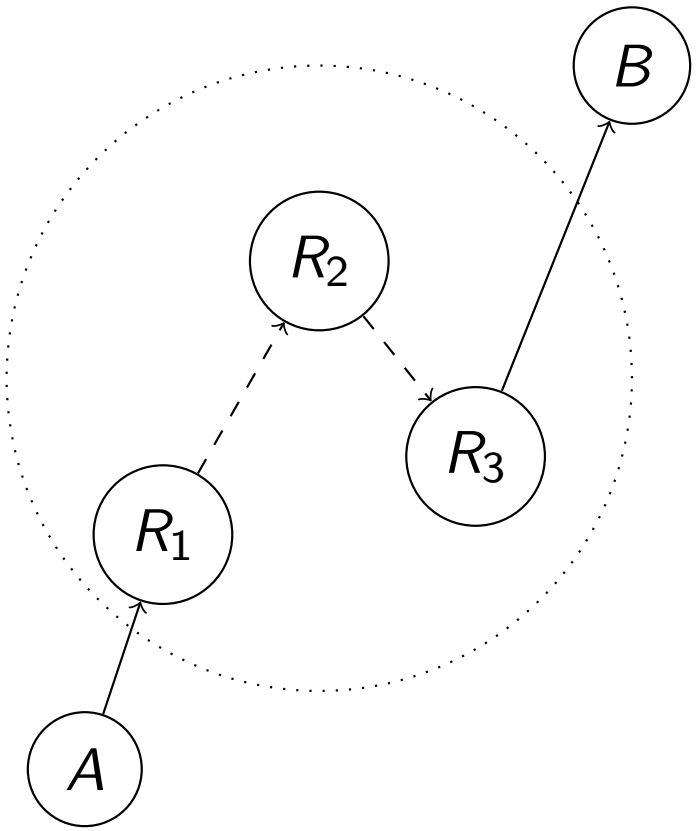
- on the *periphery* of the network
- not physically connected
- communicate through the network

R_i are **Routers**

- in the *interior* of the network
- **Routing** specifies *how to* get to B
- **Forwarding** moves traffic towards B

End-to-end Principles: knowledge/control of connections exclusively at the *periphery*

Interior nodes only forward packets based on local delivery rules



An Analogy

Postal service:

- You write a letter to your friend
- You seal that letter in an envelope
- You write your friend's address on the envelope
- You drop that envelope in the mail
- The USPS picks it up

Internet:

- You click on a link in your web browser
- Your browser constructs a packet with the `GET` request
- Your browser adds the server's address to the packet
- Your computer sends that packet out its network interface
- The gateway router receives that packet

An Analogy

Postal service: Envelope $\rightarrow$ dest with successively smaller hops

- State
- City
- Street
- Address
- Apartment number

$\Rightarrow$ USPS never knows contents of envelope

Internet: Packet $\rightarrow$ dest with successively smaller hops

- ISP
- Region
- Organization's network
- Office's subnet
- Server

$\Rightarrow$ Routers never look at payload of packet

Layers

Abstract network protocol into separate **layers**

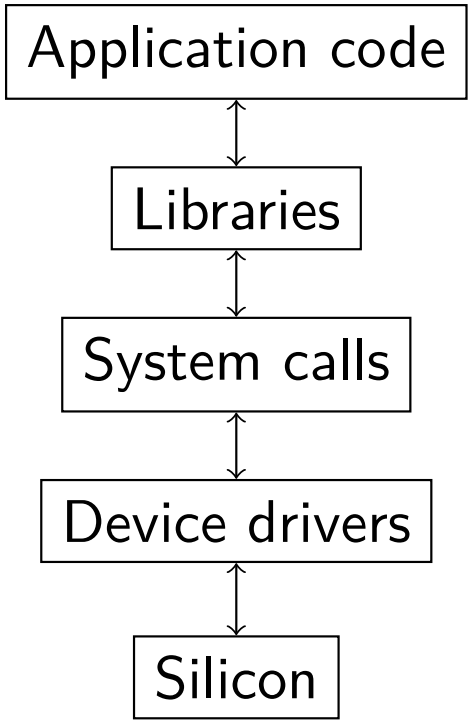
Each layer controls some aspect of communications

A layer relies on services provided by the layer *below* it

A layer provides services to the layer *above* it

Similar to how software is designed

Each layer has a *distinct role*



The Network Stack

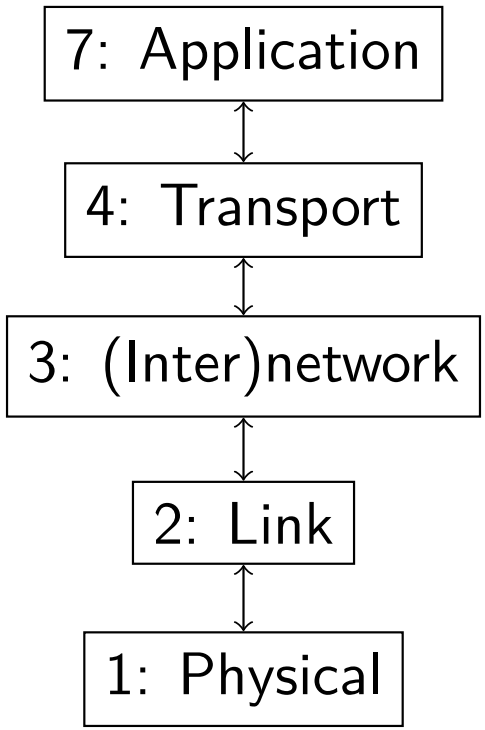
Open Systems Interconnection (**OSI**) Model

Defines *7-Layer Model* for the **network stack**

We will only worry about 5 of these

As with software stack, each layer has a distinct role

These roles are codified/implemented with *protocols*



Group Exercise 1

Your course VM has a program called *Wireshark* installed on it — it's the shark-fin icon on the left-hand side of the screen. You're going to be using this when examining network protocols, so it's worth taking a little time to explore.

You start by *capturing* a *network interface*. Try this, and see what comes up. You may need to resize the packet list in order to see the details. Don't forget to stop the packet capture at some point, or it will keep collecting indefinitely!

Look at the packet details view — what does this suggest about the network layers? How do the details relate to the hex dump of the packet data? What seems to stay the same for all packets, and what changes? What can you say about those changes?

We'll be answering some of these questions in more detail.

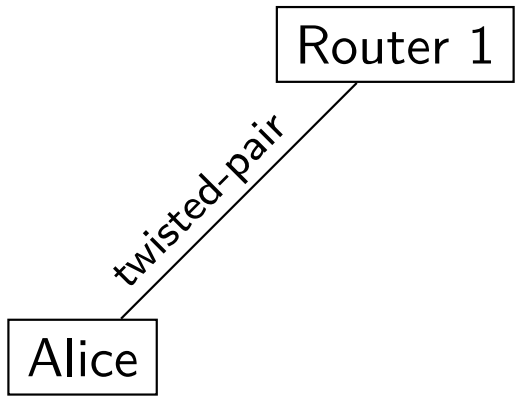
Physical Layer (Layer 1)

How to encode *bits* for a **single physical link**

- Voltage levels
- RF modulation
- Photon wavelengths/intensities

Lots of technologies

- Coaxial cable
- Twisted-pair cable
- RF broadcast
- Fiber-optic cable
- Free-space optical



Link Layer (Layer 2)

Combines bits into **frames**

A frame contains a *single message*

A message might require *multiple frames*

Provides **local addressing** (MAC—Media Access Control)

Supports *point-to-point* and often *broadcast* delivery

Might involve multiple *physical links*

Examples:

- Ethernet
- WiFi

Link Layer (Layer 2)

Link-layer connections grouped into **subnets**

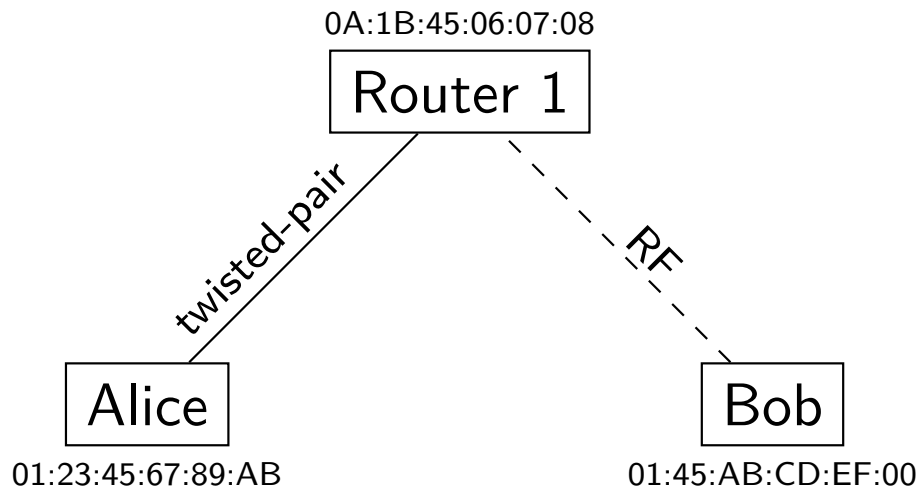
Layer 2 protocol used to transmit messages within this subnet

MAC addresses must be **globally unique**

$\Rightarrow$ Device can join *any subnet*

$\Rightarrow$ Address space partitioned by *manufacturers*

A **switch** is a device that implements *up through* layer 2



Network Layer (Layer 3)

Bridges subnets for *end-to-end* connectivity

Provides **global addressing** (IP addresses — 32 bits)

Delivery is **best-effort**
$\Rightarrow$ No retransmissions
$\Rightarrow$ No message integrity

Works across different link technologies

Data is encapsulated in **packets**
$\Rightarrow$ payload for layer-2 *frames*

internet: network of networks using the Internet Protocol
Internet: specific global instance of an internet

Network Layer (Layer 3)

IP addresses are **locally unique**

$\Rightarrow$ assigned *within a subnet* (CIDR notation: `192.168.0.0/16`)

$\Rightarrow$ some hosts *not globally addressable*

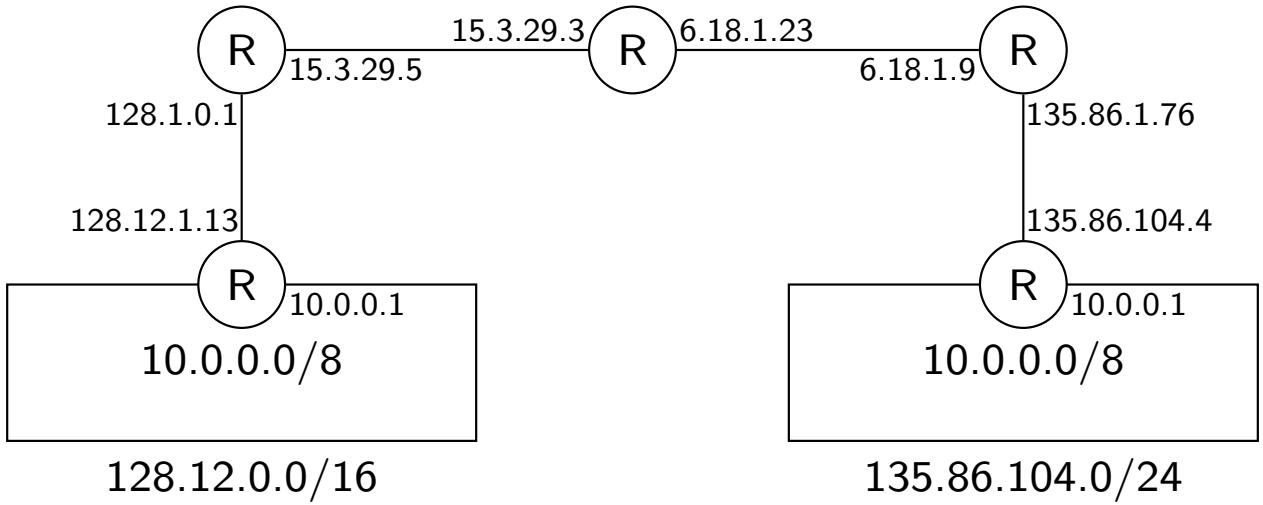
Subnets can be nested, may do **address translation**

$\Rightarrow$ allows us to scale to $> 2^{32}$ hosts

A **router** is a device that implements *up through* layer 3

$\Rightarrow$ connects multiple subnets

$\Rightarrow$ different IP addr on each **interface**



Transport Layer (Layer 4)

End-to-end communication between *processes*

UDP (User Datagram Protocol)

- unreliable, best effort
- **datagram**-based (single-packet messages)

TCP (Transmission Control Protocol)

- reliable, keeps track of data sent/received
- retransmission of lost packets
- **byte**-based (messages/**sessions** span possibly many packets)

Application Layer (Layer 7)

What users/processes interact with

Choice of transport depends on what is needed

- Web browsing $\Rightarrow$ TCP
- Email $\Rightarrow$ TCP
- Voice calls $\Rightarrow$ UDP

Defines its own data formats and protocols, within TCP or UDP

- Web browsing $\Rightarrow$ HTTP (Hypertext Transfer Protocol)
- Email $\Rightarrow$ SMTP (Simple Mail Transfer Protocol)
- Voice calls $\Rightarrow$ RTP (Real-time Transport Protocol)

Group Exercise 2

Now that we've seen a little of the structure of the network stack, let's take another look at our wireshark data. Can you identify the layer 2 and layer 3 addresses? What about layers 4 and 7? What data does layer 4 add to the packet?