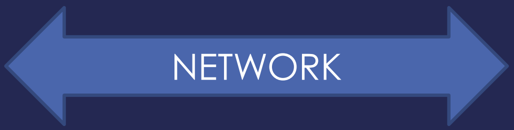
Intro to Networks

UT LAW 379M

Spring 2021

Lecture Notes

Computing 1960-1980 (ish)



NETWORK

This Photo by Unknown Author is licensed under CC BY-SA-NC

"DUMB" TERMINAL

Basically, just a keyboard and monitor

MAINFRAME

All the processing (computer brains) happens h

Computing 1980-2000 (ish)

For the most part, NO NETWORKING

This Photo by Unknown Author is licensed under CC BY-NC

Computing 2000 – Present

This Photo by Unknown Author is licensed under CC BY

NETWORK

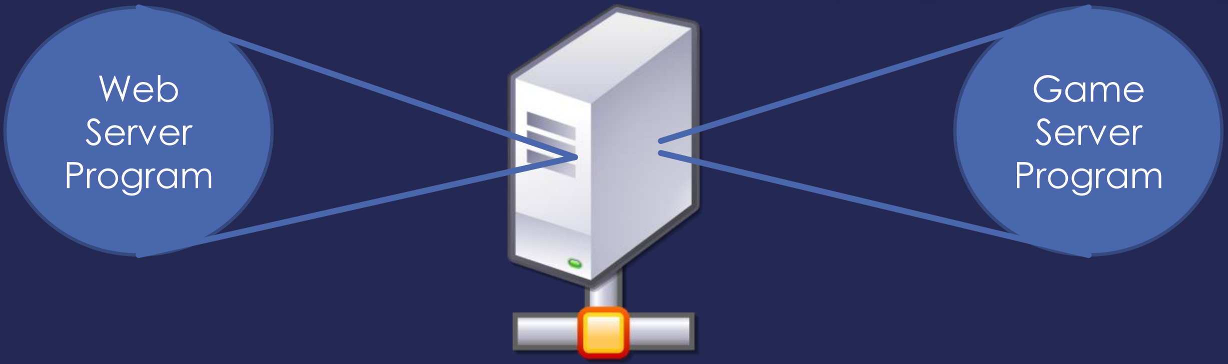
This Photo by Unknown Author is licensed under CC BY-NC-ND

General Ideas Behind Client-server

- Put a bunch of resources in a high-performance, centralized machine (SERVER)
- Clients can be much “dumber” *by comparison*
- Much more efficient
  - Sharing data between devices, applications, and people (and marketing)
  - Access from multiple locations (including hackers!)
  - Time-sharing a central machine is more scalable and cost-effective

Confusing Meaning of “Server”

- Server sometimes refers to the actual physical machine
- Server sometimes refers to the computer program that provides service
- A machine is a “server” if it has 1+ server programs running



Web Server Program

Game Server Program

This Photo by Unknown Author is licensed under CC BY-SA

Server Abstraction

I'm Lonely. I wish someone would talk to me!

This Photo by Unknown Author is licensed under CC BY-SA

SERVER (Program) ***LISTENS*** FOR INCOMING REQUESTS

Addresses Needed

- The server can't receive a request without an "address"
- Typically identify ***machines*** with an Internet Protocol (IP) address
- Typically identify ***programs on the machine*** with a "port"
  - TCP ports are for guaranteed delivery, like for file transfer (most common)
  - UDP ports are for best-effort delivery, like streaming music or games

Addresses and Ports

- Address is kind of like the address of a building
- Port is kind of like the apartment number

IP (Version 4) Address

- Although IP Version 6 addresses are in use…
- IP Version 4 addresses are still pretty common

172.217.1.142

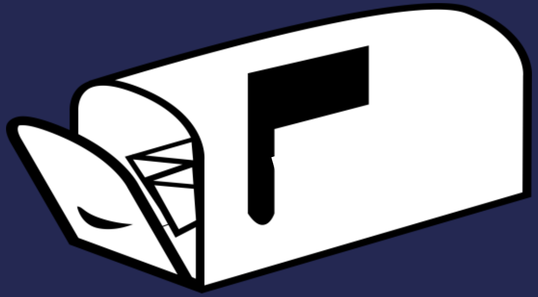
Type this in your browser!

Four numbers between 0 and 255
(separated by ".")

Ports Allow Multiple Servers

- 80 – Unencrypted web traffic
- 443 – Encrypted web traffic
- 25 – Email data tranfer

Assigning an Address and Port



Now I have an Address/Port! Maybe I'll get Requests!

SERVER HAS AN **IP ADDRESS** AND **TCP PORT**

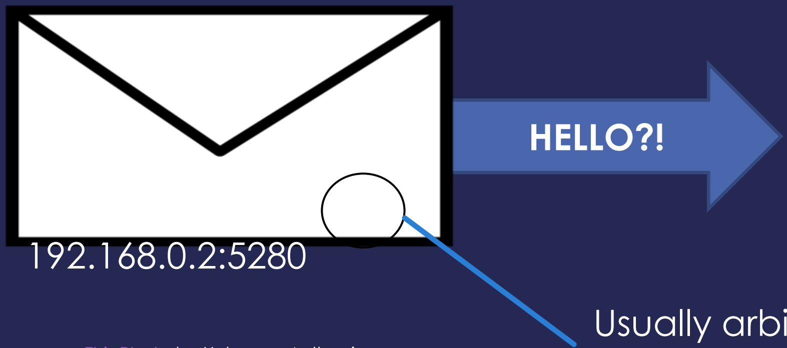
Meanwhile, Client Abstraction

This Photo by Unknown Author is licensed under CC BY-NC

HELLO?!

CLIENT (program) ***CONNECTS*** TO MAKE OUTBOUND REQUESTS

Client (program)
Needs Return Address



Usually arbitrary, picked by operating system

CLIENT (program) ***CONNECTS*** TO MAKE OUTBOUND REQUESTS

Incoming Request

192.168.0.2:5280

I GOT A REQUEST!!!

This Photo by Unknown Author is licensed under CC BY-SA

SERVER RECEIVES REQUEST

Request Response

This Photo by Unknown Author is licensed under CC BY-SA

SERVER **INVERTS TO/FROM** FOR RESPONSE

What is a Protocol?

- A protocol is the set of rules that govern the interaction of two or more parties
- In the context of networking, it defines how two nodes (like client and server) communicate
  - When a party can communicate
  - What a party can communicate, *including message structure*
  - How a party responds to received communications
- ***Certain outcomes or results are guaranteed when the rules are followed***

Overloaded Term

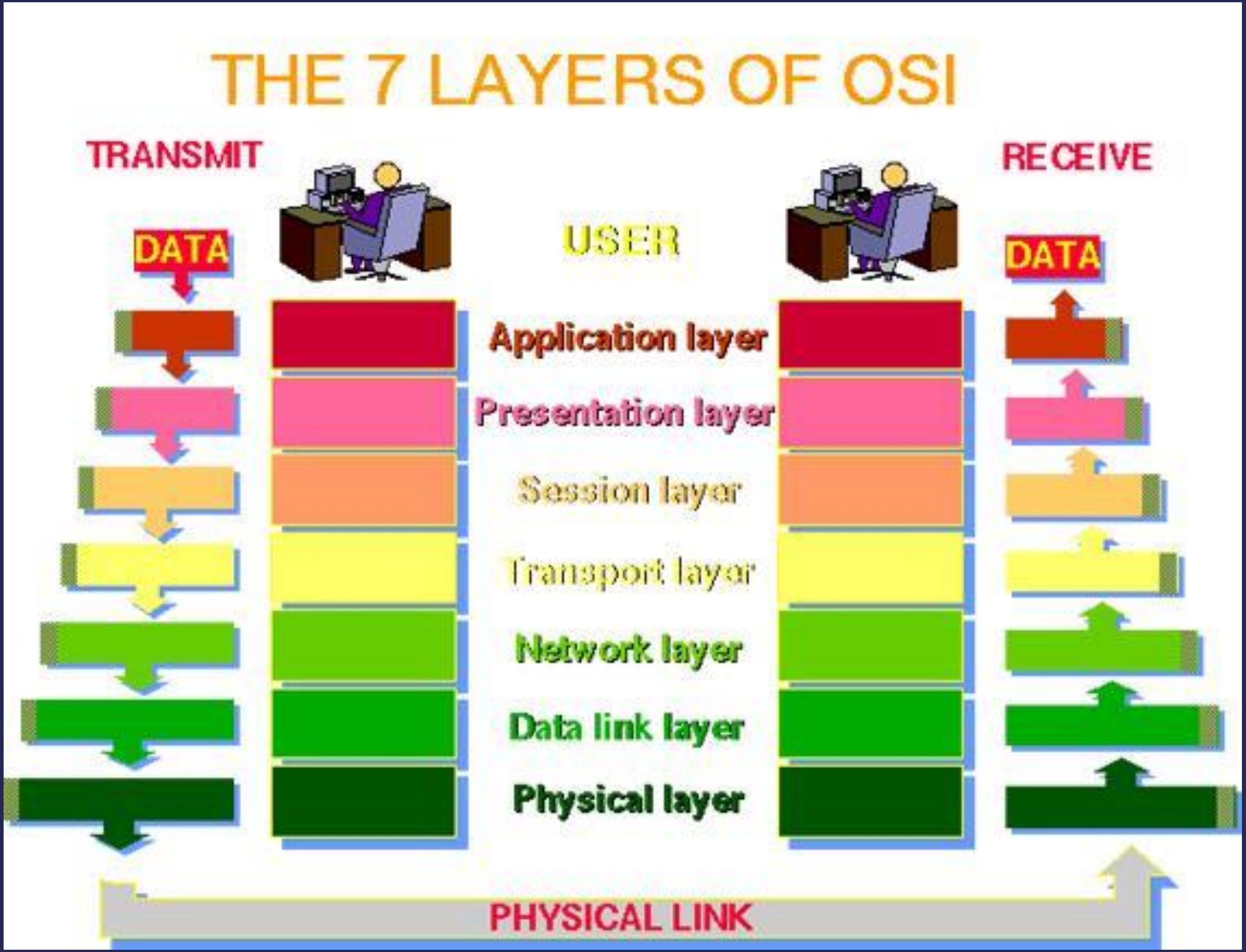
- Actually, a protocol often refers to two separate things
- **FIRST**, the rules/specification referred to on the previous slide
- **SECOND**, the computer module that *implements* the rules

Common Contemporary Protocols

- HTTP – HyperText Transfer Protocol
- IP – Internet Protocol
- SMTP – Simple Mail Transport Protocol

One Protocol is not Enough

- There are too many rules for any one protocol to handle
- Also, behavior/rules need to change for different hardware/goals
- OSI defined a conceptual “stack” of protocols.
  - Each protocol “layer” can push data down to a lower layer, or pop data up to a higher layer
  - The protocol on one machine (e.g., client) is a “peer” with the same protocol on the other machine

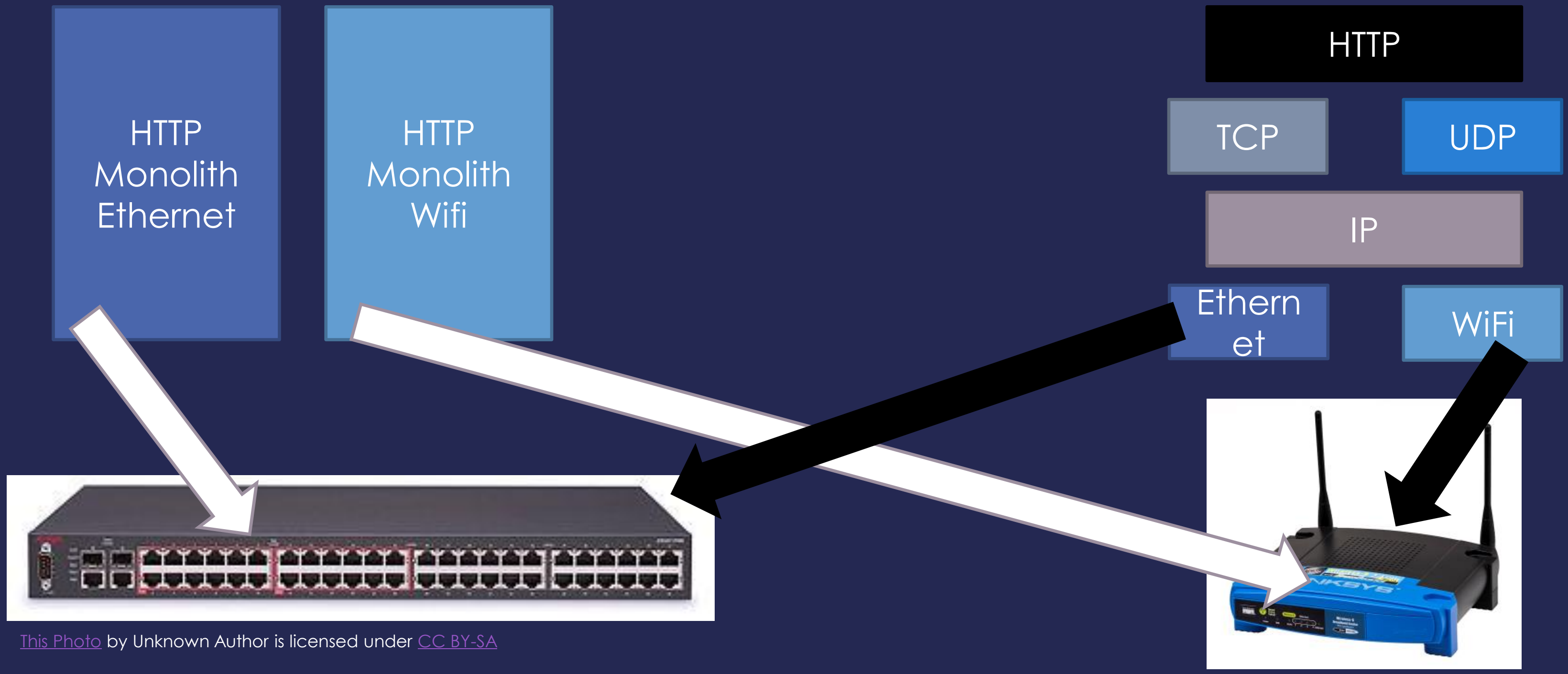


THE 7 LAYERS OF OSI

The OSI Model in Practice

- Very few systems follow the seven-layer “ideal”
- Mostly just care about TCP/IP and the following layers:
  - Application (Layer 7; example: HTTP)
  - Transport (Layer 4; TCP)
  - IP (Layer 3; IP)
  - Data Link (Layer 2; example: Ethernet or Wifi)
- NOTE: It’s common to just refer to a layer by it’s number (e.g., a layer-4 protocol)

Monolithic vs Modular



HTTP Monolith Ethernet

HTTP Monolith Wifi

HTTP

TCP

UDP

IP

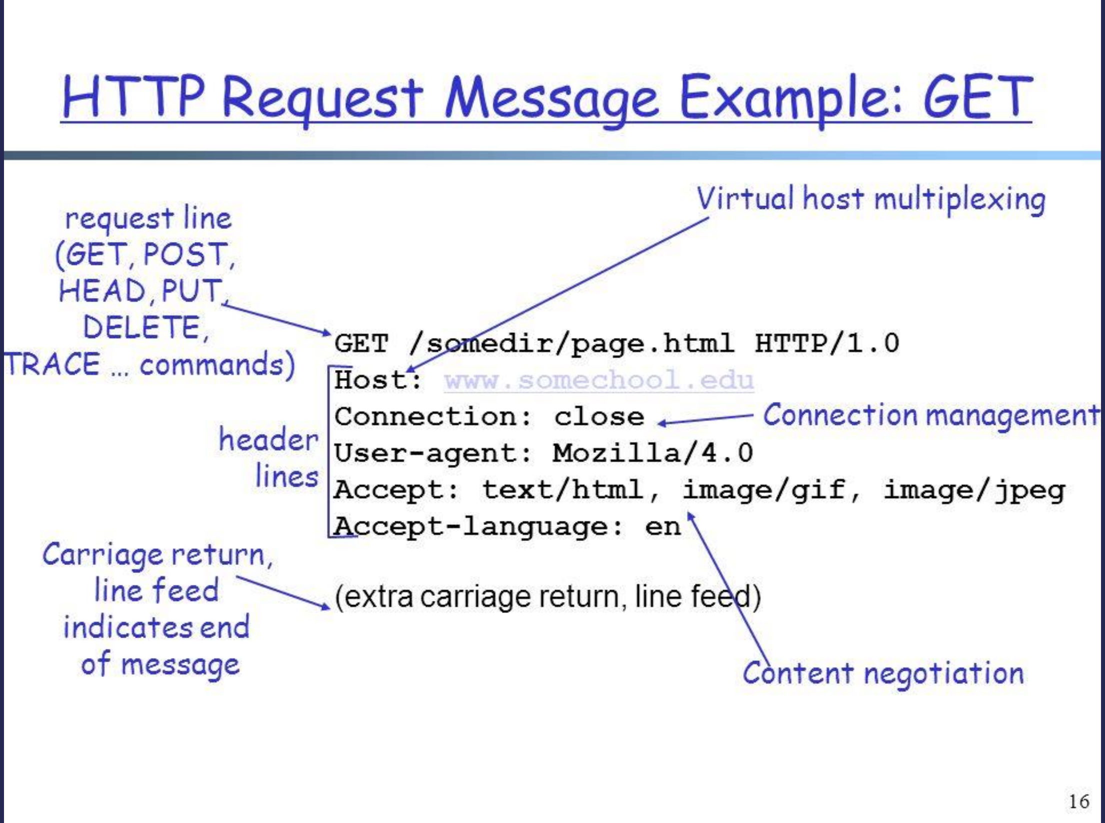
Ethernet

WiFi

This Photo by Unknown Author is licensed under CC BY-SA

This Photo by Unknown Author is licensed under CC BY-SA

HTTP Request



HTTP Request Message Example: GET

request line (GET, POST, HEAD, PUT, DELETE, TRACE ... commands)

Virtual host multiplexing

```
GET /somedir/page.html HTTP/1.0
Host: www.somechool.edu
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: en
```

(extra carriage return, line feed)

header lines

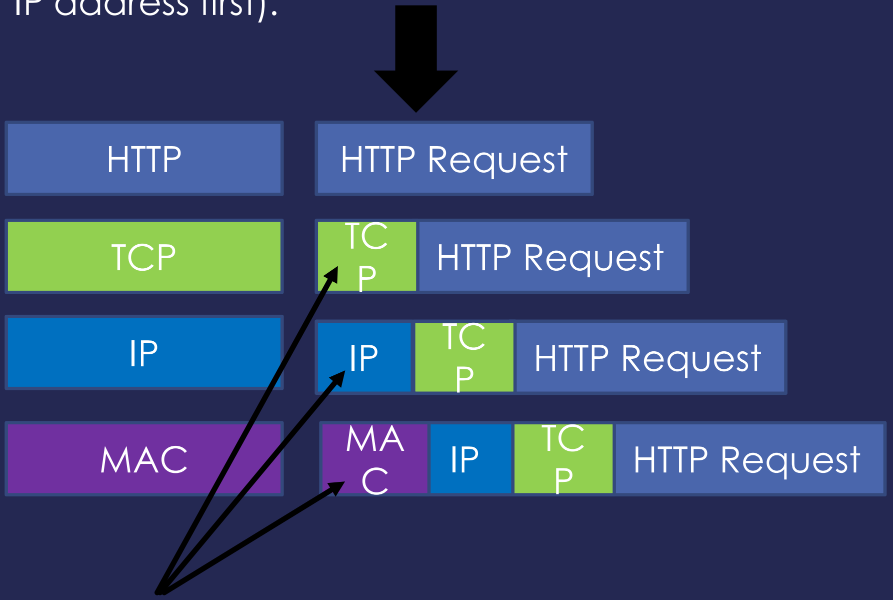
Connection management

Carriage return, line feed indicates end of message

Content negotiation

TCP/IP Send Example

User enters "google.com" into browser (computer converts "google.com" to an IP address first).



HTTP — HTTP Request

TCP — TC P | HTTP Request

IP — IP | TC P | HTTP Request

MAC — MA C | IP | TC P | HTTP Request

Headers. Typically meta data such as "to", "from", etc.

HTTP is the protocol used for sending/receiving data to/from websites.

TCP deals with making sure the data gets to the right server program correctly

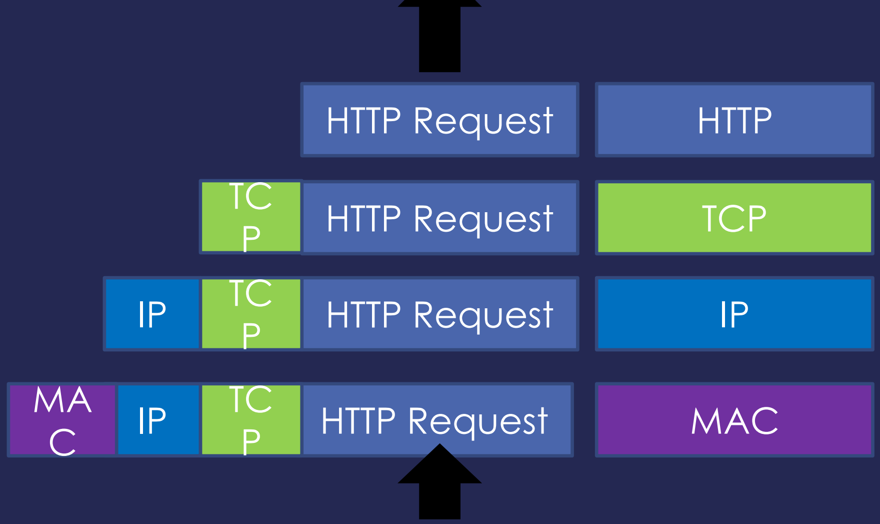
IP is used for getting data over the Internet to the correct machine

MAC is used for communicating on Wifi or Ethernet

TCP/IP Receive Example

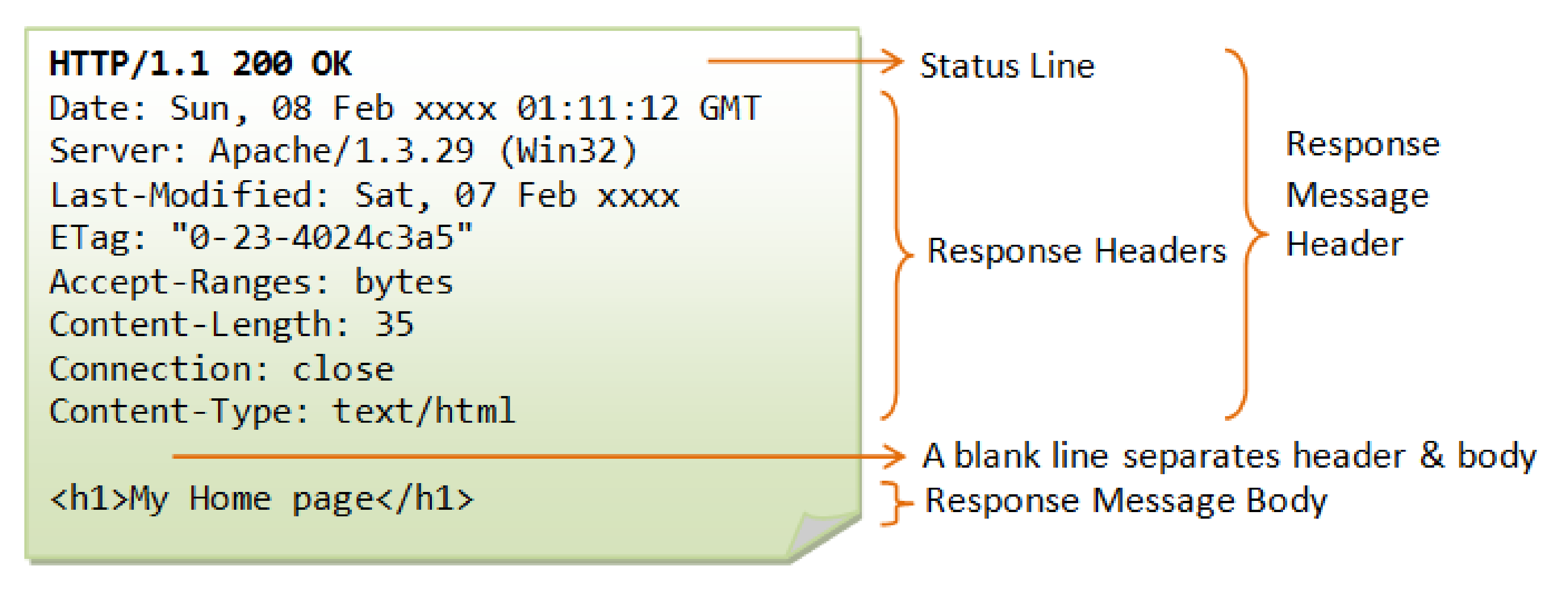
As each layer processes the data, the headers are stripped off. The MAC layer removes the MAC header, the IP layer removes the IP header, etc., before passing the data up. Each layer may or may not need to do some processing based on information in the header.

Web server program begins gather the requested info. When it has it, it will respond by sending a new message down the stack in the reverse direction



Conceptually, data arrives at the bottom of the st
and is processed and then “popped” up to a high

HTTP Response



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

Wireshark Exercises