

**OSI Model**

The Open Systems Interconnection (OSI) model is a 7-layer view of networking that abstracts and encapsulates the functionality of each component of networking.

OSI Layer 1: \_\_\_\_\_

OSI Layer 2: \_\_\_\_\_

OSI Layer 3: \_\_\_\_\_

00	4500 00c6 1e1f 4000 4006 152e ac16 b4a3
10	12dc 95a6 ...

IPv4, Packet Length: 0x00c6 (198 bytes); Source IP: ac.16.b4.a3 (172.22.180.163); Destination IP: 12.dc.95.a6 (18.220.149.166)

OSI Layer 4: \_\_\_\_\_

10	... bafa 0050 0f60 c9b4 356a 523f
20	8018 01f6 079e 0000 0101 080a 8146 30a0
30	31d4 daac ...

Port :0xbafa (47866) connecting to Port :0x0050 (80); Checksum 0x079e; Timestamp: 0x814630a0 (2168860832)

OSI Layer 5, 6, and 7: \_\_\_\_\_

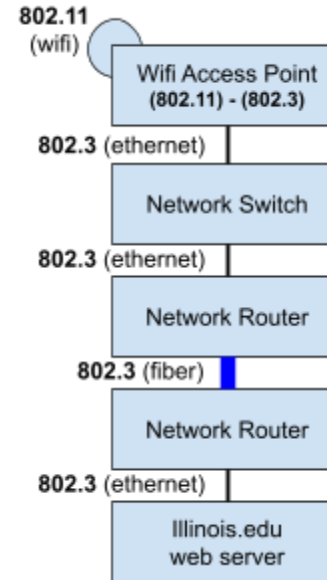
30	... 4745 5420 2f20 4854 5450 2f31	GET / HTTP/1.1\r\n
40	2e31 0d0a 5573 6572 2d41 6765 6e74 3a20	User-Agent: Wget/1.20.3
50	5767 6574 2f31 2e32 302e 3320 286c 696e	(linux-gnu)\r\n
60	7578 2d67 6e75 290d 0a41 6363 6570 743a	Accept: */*\r\n
70	202a 2f2a 0d0a 4163 6365 7074 2d45 6e63	Accept-Encoding:
80	6f64 696e 673a 2069 6465 6e74 6974 790d	identity\r\n
90	0a48 6f73 743a 2077 6166 2e63 732e 696c	Host:
a0	6c69 6e6f 6973 2e65 6475 0d0a 436f 6e6e	waf.cs.illinois.edu\r\n
b0	6563 7469 6f6e 3a20 4b65 6570 2d41 6c69	Connection: Keep-Alive\r\n
c0	7665 0d0a 0d0a	\r\n

**Full Packet Journey**

Consider an HTTP request you are making from your browser to waf.cs.illinois.edu (just as I did using tcpdump). Assuming we start from your laptop on a WiFi network:

Your Laptop:

802.11 Header	IP Header	TCP Header	HTTP Header	HTTP Content
---------------	-----------	------------	-------------	--------------



**Network Layer (Layer 3) Protocol: Internet Protocol (IP)**

- The **network layer** provides \_\_\_\_\_ communication.
- When on the Internet, every host relies on the **IP protocol**:

- IP (IPv4) Address:

- IPv6 Addresses:

## Transport Layer (Layer 4) Protocols:

Two protocols for \_\_\_\_\_ communications:

- 1.
- 2.

## Application Layer Protocols:

When a protocol runs on “TCP/IP”, that information alone tells us the lower-level protocols used. The top-most layers are the Application Layer, and will be application-specific.

## MP4 Protocol

Some protocols are extremely simple, like the MP4 protocol you’re working on right now:

1	>	GET midnights\n
2		0\n
3	>	MOD midnights 5\n
4		5\n
5	>	MOD midnights 10\n
6		15\n
7	>	GET midnights\n
8		15\n

## HTTP Web Services

One of the primary ways that processes will communicate is via “web services” -- applications that communicate using the HTTP protocol.

The **HTTP protocol** has two parts:

**Protocol Part #1:** \_\_\_\_\_

R	1	POST /extract HTTP/1.1\r\n
E	2	Host: localhost:5000\r\n
Q	3	User-Agent: curl/7.68.0\r\n
U	4	Accept: */*\r\n
E	5	Content-Length: 3046796\r\n
S	6	\r\n
T	...	{ 3,046,796 bytes payload }

## Request Packet Organization:

- Line Delineation:
- Start Line (Line 1): HTTP method (verb), target, and version
- Request Headers (Lines 2+):
- Payload (or sometimes just the “Data”):

**Protocol Part #2:** \_\_\_\_\_

R	1	HTTP/1.0 200 OK\r\n
E	2	Content-Length: 3044143\r\n
S	3	Content-Type: image/gif\r\n
P	4	Last-Modified: Mon, 28 Sep 2022 21:16:13\r\n
O	5	Cache-Control: public, max-age=43200\r\n
N	6	Expires: Tue, 22 Mar 2023 09:16:12 GMT\r\n
S	7	ETag: "1601327773.0845277-3044143-32865"\r\n
E	8	Server: Werkzeug/0.16.1 Python/3.8.2\r\n
	9	Date: Thu, 22 Feb 2023 21:16:12 GMT\r\n
	10	\r\n
	...	{ 3,044,143 bytes of content }

In general, the request and response follows the same format with only one major exception:

- Response “Status Line” (Line 1):

## HTTP Status Codes

1xx	2xx	3xx	4xx	5xx
Informational	Success	Redirection	Client Error	Server Error
100: Continue ...	200: OK 201: Created ...	304: Not Modified ...	400: Bad Request 404: File Not Found ...	500: Internal Server Error ...