CSE 127: Introduction to Security

Lecture 9: Intro to Networking

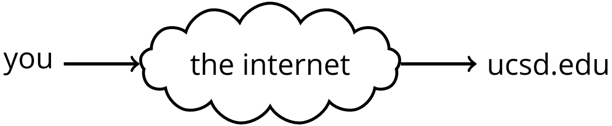
Deian Stefan

UCSD

Winter 2020

Some material from Nadia Heninger, Zakir Durumeric, David Wagner

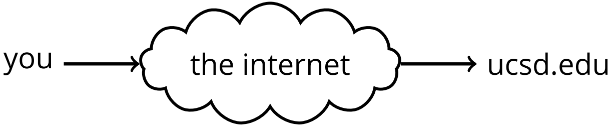
The Internet



Original Idea:

- Network is dumb
- Simple, robust service
- Shift complexity to endpoints

The Internet



Original Idea:

- Network is dumb
- Simple, robust service
- Shift complexity to endpoints
- Acts like postal system (packet-based) rather than traditional phone system (circuit-based)
- Need protocols to actually communicate

Network protocol

A protocol is an agreement on how to communicate.

Includes syntax and semantics.

- **Syntax:** How a communication is specified and structured.
    - Format, order messages are sent and received.

Network protocol

A protocol is an agreement on how to communicate.

Includes syntax and semantics.

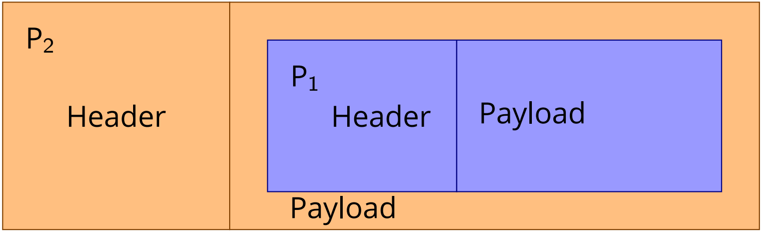
- **Syntax:** How a communication is specified and structured.
  - Format, order messages are sent and received.
- **Semantics:** What a communication means
  - Actions taken when transmitting, receiving, or timer expires.

Protocols are layerd

- Networks use a stack of layers
- Lower layers provide services to layers above
    - Don’t care what higher layers do
- Higher layers use services of layers below
    - Don’t care how lower layers implement services
- Layers define abstraction boundaries
    - At a given layer, all layers above and below are opaque

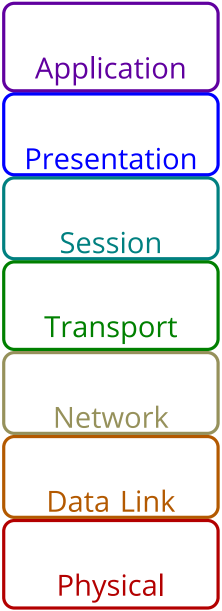
Packet abstraction/encapsulation

- Protocol N_1 can use services of lower layer protocol N_2
- A packet P_1 of N_1 is encapsulated into a packet P_2 of N_2
- The payload of P_2 is P_1
- The control information of P_2 is derived from that of P_1



OSI Layers

(Open Systems Interconnection)



Application

- End user layer
- HTTP, FTP, Skype, SSH, SMTP, DNS

Presentation

- Syntax, byte order, compression, encryption
- SSL, SSH, MPEG, JPEG

Session

- Connection establishment and maintenance
- APIs, sockets

Transport

- End-to-end connections between processes
- TCP, UDP

Network

- Addressing, routing between nodes
- IP

Data Link

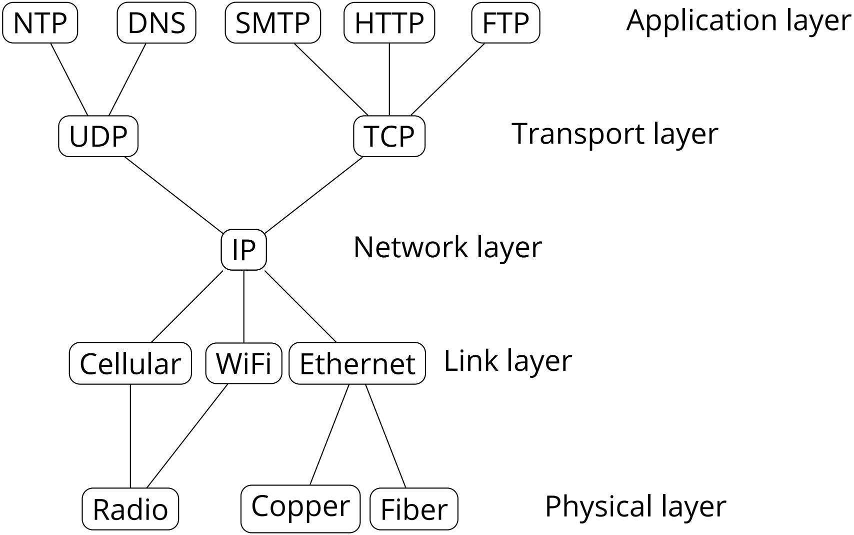
- Link management, frames
- Ethernet, WiFi

Physical

- Physical wires
- Photons, RF modulation

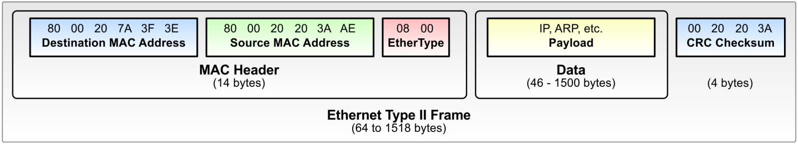
Basic Internet Archictecture “Hourglass”

Narrow waist = interoperability



Link layer: Connecting hosts to local network

Most common link layer protocol: **Ethernet**



80 00 20 7A 3F 3E
Destination MAC Address

80 00 20 20 3A AE
Source MAC Address

08 00
EtherType

MAC Header
(14 bytes)

IP, ARP, etc.
Payload

Data
(46 - 1500 bytes)

00 20 20 3A
CRC Checksum

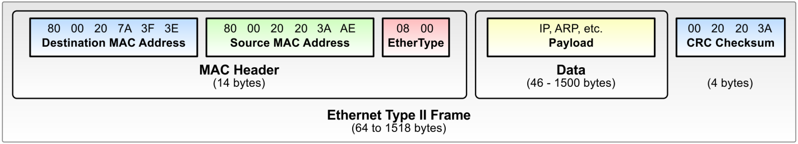
(4 bytes)

Ethernet Type II Frame
(64 to 1518 bytes)

- Messages organized into *frames*
- Every node has a globally unique 6-byte MAC (Media Access Control) address

Link layer: Connecting hosts to local network

Most common link layer protocol: **Ethernet**



80 00 20 7A 3F 3E
Destination MAC Address

80 00 20 20 3A AE
Source MAC Address

08 00
EtherType

MAC Header
(14 bytes)

IP, ARP, etc.
Payload

Data
(46 - 1500 bytes)

00 20 20 3A
CRC Checksum

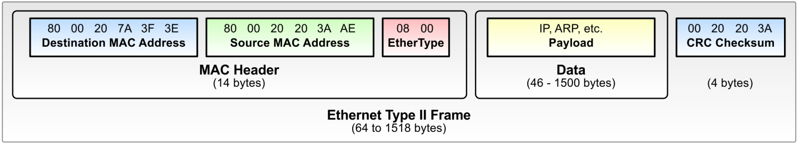
(4 bytes)

Ethernet Type II Frame
(64 to 1518 bytes)

- Messages organized into *frames*
- Every node has a globally unique 6-byte MAC (Media Access Control) address
- Originally a broadcast protocol: every node on network received every packet
- Now switched: switch learns the physical port for each MAC address and sends packets to correct port if known

Link layer: Connecting hosts to local network

Most common link layer protocol: **Ethernet**



80 00 20 7A 3F 3E
Destination MAC Address

80 00 20 20 3A AE
Source MAC Address

08 00
EtherType

MAC Header
(14 bytes)

IP, ARP, etc.
Payload

Data
(46 - 1500 bytes)

00 20 20 3A
CRC Checksum

(4 bytes)

Ethernet Type II Frame
(64 to 1518 bytes)

- Messages organized into *frames*
- Every node has a globally unique 6-byte MAC (Media Access Control) address
- Originally a broadcast protocol: every node on network received every packet
- Now switched: switch learns the physical port for each MAC address and sends packets to correct port if known
- WiFi similar to Ethernet, but nodes can move

```
$ ip link
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 4c:cc:6a:64:1d:b5 brd ff:ff:ff:ff:ff:ff


$ ifconfig
enp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 132.239.15.12  netmask 255.255.255.0  broadcast 132.239.15.255
        inet6 fe80::4ecc:6aff:fe64:1db5  prefixlen 64  scopeid 0x20<link>
        ether 4c:cc:6a:64:1d:b5  txqueuelen 1000  (Ethernet)
        RX packets 139390143  bytes 147499561034 (137.3 GiB)
        RX errors 0  dropped 347298  overruns 0  frame 0
        TX packets 40001343  bytes 17541668347 (16.3 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 18
```

ARP: Address Resolution Protocol

- Problem: How does a host learn what MAC addresses to send packets to?
- ARP lets hosts build table mapping IP addresses to MAC addresses.

ARP: Address Resolution Protocol

- Problem: How does a host learn what MAC addresses to send packets to?
- ARP lets hosts build table mapping IP addresses to MAC addresses.
- ARP request: source MAC, dest MAC, “Who has IP address N?”
- ARP reply: source MAC, dest MAC, “IP address N is at MAC address M.”

IP: Internet Protocol

- Connectionless delivery model
- “Best effort” = no guarantees about delivery
- No attempt to recover from failure
- Packets might be lost, delivered out of order, delivered multiple times
- Packets might be fragmented
- Provides hierarchical addressing scheme

- IPv4
  - 32-bit host addresses
  - Written as 4 bytes in decimal,
  - e.g. 192.168.1.1
- IPv6
  - 128-bit host addresses
  - Written as 16 bytes in hex
  - :: implies zero bytes
  - e.g. 2620:0:e00:b::53 = 2620:0:e00:b:0:0:0:53

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example Internet Datagrarm Header

Note that each tick mark represents one bit position.

Routing: BGP (Border Gateway Protocol)

- Internet organized into ASes (Autonomous Systems) with peer, provider, or customer relationships between them
- Rough tree shape, with a small number of backbone ASes in a cllique at the root

Routing: BGP (Border Gateway Protocol)

- Internet organized into ASes (Autonomous Systems) with peer, provider, or customer relationships between them
- Rough tree shape, with a small number of backbone ASes in a cllique at the root

- BGP allows routers to exchange information about their routing tables
- Routers maintain global table of routes
- Each router announces what it can route to its neighbors
- Routes propagate through network

TCP (Transmission Control Protocol)

- Want abstraction of a stream of bytes delivered reliably and in-order between applications on different hosts
- TCP provides:
  - Reliable in-order byte stream
  - Connection-oriented protocol
  - Explicit setup/teardown
  - End hosts (processes) have multiple concurrent long-lived dialogs
  - Congestion control: adapt to network path capacity, receiver's ability to receive packets

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP Header Format

Ports

- Each application is identified by a port number
- TCP connection established between port A on host address M to port B on host address N. Ports are 16 bits, 1–65535
- Some destination ports are used for particular applications by convention

80	HTTP (web)
443	HTTPS (web)
25	SMTP (mail)
67	DHCP (host configuration)
22	SSH (secure shell)
23	telnet

TCP Sequence Numbers

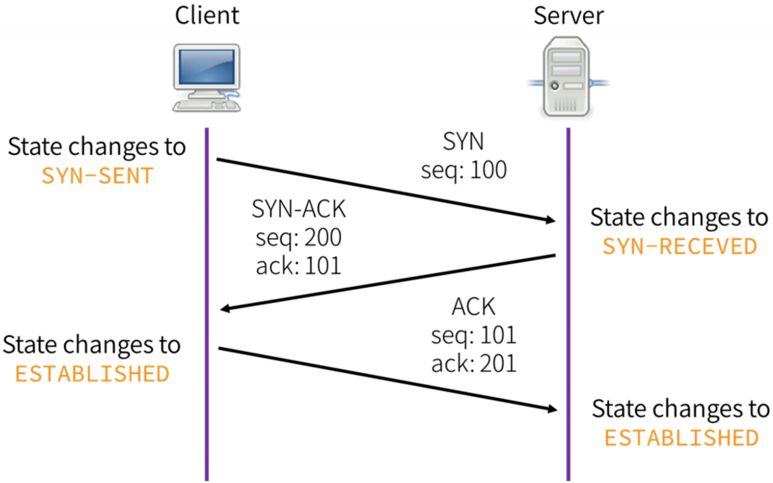
- Bytes in application data stream numbered with 32-bit sequence number
- Data sent in segments: sequences of contiguous bytes sent in a single IP datagram
- Sequence number indicates where data belongs in byte sequence
- Sequence number in packet header is the sequence number of the first byte in the payload

TCP Sequence numbers and Acknowledgement

- Two logical data streams in a TCP connection: one in each direction
- Receiver acknowledges received data: acknowledgement number is sequence number of next expected byte of stream in opposite direction
- ACK flag set to acknowledge data
- Sender retransmits lost data
- Congestion control: sender adapts retransmission according to timeouts

TCP 3-Way Handshake

Starting a TCP connection



FIN/RST: Closing TCP connections

- FIN initiates a clean close of a TCP connection, waits for ACK from receiver

FIN/RST: Closing TCP connections

- FIN initiates a clean close of a TCP connection, waits for ACK from receiver

- If a host receives a TCP packet with RST flag, it tears down the connection
- Designed to handle spurious TCP packets from previous connections

UDP (User Datagram Protocol)

- UDP offers no service quality guarantee
- Essentially a transport layer protocol that is a wrapper around IP
- Adds ports to let applications demultiplex traffic
- Useful for applications that only need best-effort guarantee
- e.g. DNS, NTP

RFC 768

J. Postel
ISI

28 August 1980

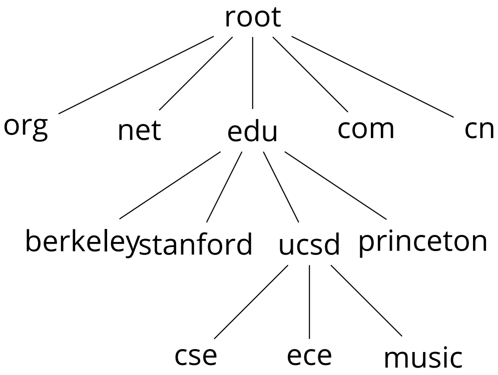
User Datagram Protocol

```
                  0      7 8     15 16    23 24    31
                 +--------+--------+--------+--------+
                 |     Source      |   Destination   |
                 |      Port       |      Port       |
                 +--------+--------+--------+--------+
                 |                 |                 |
                 |     Length      |    Checksum     |
                 +--------+--------+--------+--------+
                 |
                 |          data octets ...
                 +---------------- ...
```

User Datagram Header Format

DNS (Domain Name Service)

- Handle mapping between host names (e.g. ucsd.edu) and IP addresses (e.g. 132.239.180.101)
- DNS is a delegatable, hierarchical name space



DNS Records

```
nadiah$ nadiah$ dig cseweb.ucsd.edu

; <<>> DiG 9.10.6 <<>> cseweb.ucsd.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3727
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;cseweb.ucsd.edu. IN A

;; ANSWER SECTION:
cseweb.ucsd.edu. 3140 IN CNAME roweb.eng.ucsd.edu.
roweb.eng.ucsd.edu. 2855 IN A 132.239.8.30

;; Query time: 57 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Sun Nov 03 20:49:08 PST 2019
;; MSG SIZE  rcvd: 84
```

DNS Details

- 13 main DNS root servers
- DNS responses are cached for quicker responses
- DNS authorities queried progressively according to domain name hierarchy

```
nadiah$ nadiah$ dig cseweb.ucsd.edu +trace

; <<>> DiG 9.10.6 <<>> cseweb.ucsd.edu +trace
;; global options: +cmd
. 105604 IN NS d.root-servers.net.
. 105604 IN NS h.root-servers.net.
. 105604 IN NS c.root-servers.net.
. 105604 IN NS j.root-servers.net.
                          ...
. 105604 IN NS l.root-servers.net.
. 105604 IN NS i.root-servers.net.
. 105604 IN RRSIG NS 8 0 518400 20191115050000 20191102040000 22545 . Z14B+vD/MKzOX1UBwu04kzwQNajhg1AflK7j5Jvd9NZ
;; Received 525 bytes from 192.168.1.254#53(192.168.1.254) in 44 ms

edu. 172800 IN NS b.edu-servers.net.
edu. 172800 IN NS f.edu-servers.net.
edu. 172800 IN NS i.edu-servers.net.
                          ...
edu. 172800 IN NS c.edu-servers.net.
edu. 172800 IN NS e.edu-servers.net.
edu. 172800 IN NS d.edu-servers.net.
edu. 86400 IN DS 28065 8 2 4172496CDE85534E51129040355BD04B1FCFEBAE996DFDDE652006F6 F8B2CE76
edu. 86400 IN RRSIG DS 8 1 86400 20191116170000 20191103160000 22545 . BsoO9WI4UphacN5rL0B4f3bCzVPptbmTCKHwcMgb6e
;; Received 1174 bytes from 192.58.128.30#53(j.root-servers.net) in 20 ms

ucsd.edu. 172800 IN NS ns-auth2.ucsd.edu.
ucsd.edu. 172800 IN NS ns-auth3.ucsd.edu.
9DHS4EP5G85PF9NUFK06HEK0048QGK77.edu. 86400 IN NSEC3 1 1 0 - 9V5L4LUB1VNJ9EQQLIHEQCBREACL2500  NS SOA RRSIG DNSKE
9DHS4EP5G85PF9NUFK06HEK0048QGK77.edu. 86400 IN RRSIG NSEC3 8 2 86400 20191111043435 20191104032435 47252 edu. M5V
3FTB9RSLROQJUOPDNLJJE2I31U25M4MG.edu. 86400 IN NSEC3 1 1 0 - 4586U2HHMPSEAQHJD6R9INNA38POF8KL  NS DS RRSIG
3FTB9RSLROQJUOPDNLJJE2I31U25M4MG.edu. 86400 IN RRSIG NSEC3 8 2 86400 20191111041950 20191104030950 47252 edu. BKv
;; Received 671 bytes from 192.41.162.30#53(l.edu-servers.net) in 9 ms

cseweb.ucsd.edu. 3600 IN CNAME roweb.eng.ucsd.edu.
roweb.eng.ucsd.edu. 3600 IN A 132.239.8.30
;; Received 84 bytes from 132.239.252.186#53(ns-auth3.ucsd.edu) in 14 ms
```

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser's URL bar. What happens?

1. Your laptop uses DHCP (Dynamic Host Configuration Protocol) to bootstrap itself on the local network.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

1. Your laptop uses DHCP (Dynamic Host Configuration Protocol) to bootstrap itself on the local network.
    - New host has no IP address, doesn’t know who to ask
    - Broadcasts DHCPDISCOVER to 255.255.255.255 with its MAC address

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

1. Your laptop uses DHCP (Dynamic Host Configuration Protocol) to bootstrap itself on the local network.
    - New host has no IP address, doesn’t know who to ask
    - Broadcasts DHCPDISCOVER to 255.255.255.255 with its MAC address
    - DHCP server responds with config: lease on host IP address, gateway IP address, DNS server information

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser's URL bar. What happens?

2. Your laptop makes an ARP request to learn the MAC address of the local router.
    - Every connection outside the local network will be encapsulated in a link-layer frame with the local router's MAC address as the desination.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

2. Your laptop makes an ARP request to learn the MAC address of the local router.
    - Every connection outside the local network will be encapsulated in a link-layer frame with the local router’s MAC address as the desination.
    - Your laptop encapsulates each IP packet in a WiFi Ethernet frame addressed to the local router.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

2. Your laptop makes an ARP request to learn the MAC address of the local router.
    - Every connection outside the local network will be encapsulated in a link-layer frame with the local router’s MAC address as the desination.
    - Your laptop encapsulates each IP packet in a WiFi Ethernet frame addressed to the local router.
    - The local router decapsulates these Ethernet frames and re-encodes them to forward them on its fiber connection to its upstream ISP, or to another part of the network.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser's URL bar. What happens?

2. Your laptop makes an ARP request to learn the MAC address of the local router.

   - Every connection outside the local network will be encapsulated in a link-layer frame with the local router's MAC address as the desination.
   - Your laptop encapsulates each IP packet in a WiFi Ethernet frame addressed to the local router.
   - The local router decapsulates these Ethernet frames and re-encodes them to forward them on its fiber connection to its upstream ISP, or to another part of the network.
   - Each hop re-encodes the link layer for its own network.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

3. Your laptop does a DNS lookup on ucsd.edu.
    - It learned the IP address of a local DNS server from DHCP, or had a server (like 8.8.8.8) already hard-coded.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

3. Your laptop does a DNS lookup on ucsd.edu.
   - It learned the IP address of a local DNS server from DHCP, or had a server (like 8.8.8.8) already hard-coded.
   - Each request is a DNS query encapsulated in one or more UDP packets encapsulated in one or more IP packets.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser's URL bar. What happens?

3. Your laptop does a DNS lookup on ucsd.edu.
    - It learned the IP address of a local DNS server from DHCP, or had a server (like 8.8.8.8) already hard-coded.
    - Each request is a DNS query encapsulated in one or more UDP packets encapsulated in one or more IP packets.
    - Each response tells the laptop what authority to query, until it learns the final IP address (132.239.180.101) for ucsd.edu

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

3. Your laptop does a DNS lookup on ucsd.edu.
    - It learned the IP address of a local DNS server from DHCP, or had a server (like 8.8.8.8) already hard-coded.
    - Each request is a DNS query encapsulated in one or more UDP packets encapsulated in one or more IP packets.
    - Each response tells the laptop what authority to query, until it learns the final IP address (132.239.180.101) for ucsd.edu
    - This address is cached, along with the authorities for the hierarchy in the hostname.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

4. Your laptop opens a TCP connection to 132.239.180.101.

    - Each packet of the TCP triple handshake is encoded in an IP packet that is encoded as Ethernet frames that are decoded and re-encoded as they pass through the network.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

4. Your laptop opens a TCP connection to 132.239.180.101.
    - Each packet of the TCP triple handshake is encoded in an IP packet that is encoded as Ethernet frames that are decoded and re-encoded as they pass through the network.
    - The local router has a routing table that contains IP prefixes that it matches against the IP address that tells it what address to forward the packets to.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser's URL bar. What happens?

4. Your laptop opens a TCP connection to 132.239.180.101.
   - Each packet of the TCP triple handshake is encoded in an IP packet that is encoded as Ethernet frames that are decoded and re-encoded as they pass through the network.
   - The local router has a routing table that contains IP prefixes that it matches against the IP address that tells it what address to forward the packets to.
   - The packet passes through a series of ASes.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

4. Your laptop opens a TCP connection to 132.239.180.101.
    - Each packet of the TCP triple handshake is encoded in an IP packet that is encoded as Ethernet frames that are decoded and re-encoded as they pass through the network.
    - The local router has a routing table that contains IP prefixes that it matches against the IP address that tells it what address to forward the packets to.
    - The packet passes through a series of ASes.
    - For my home network (ATT), we go through sbcglobal.net -> att.net -> level3.net -> cenic.net ->ucsd.edu.

Using the internet: A worked example

You connect your laptop to a cafe wifi network and type ucsd.edu into your browser’s URL bar. What happens?

5. Your laptop sends a HTTP GET request inside the TCP connection.
6. Based on the HTTP response, the laptop performs a new DNS lookup, TCP handshake, and HTTP GET requests for every resource in the HTML as it renders.