CSE 127: Introduction to Computer Security

Lecturer: Nadia Heninger
Lecture #10: Introduction to Networking

These lecture notes were scribed by students in CSE 127 during Winter 2021. They have been lightly edited but may still contain errors.

1 The Internet: A Broad Overview

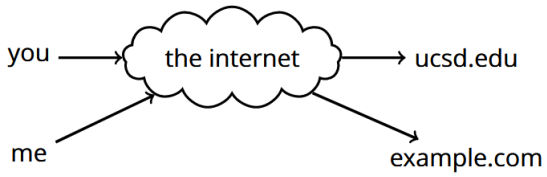

Figure 1: The basic idea of the Internet. (Source: lecture slides.)

At its core, the Internet is a "dumb" network: a simple, abstracted service used to transfer information between endpoints, such as your laptop or `ucsd.edu`. Those endpoints handle the bulk of the complexities involved with sending and receiving information. The end result is a network whose simplicity makes it robust, despite the complicated guarantees involved with the services it provides.

Importantly, the Internet is packet-based rather than circuit-based. Rather than sending information in a continuous stream of data in a fully constructed connection between endpoints, as a circuit-based system would, the Internet transfers information in discrete chunks, or packets. The postal system serves as a useful analogy: just as physical packages and mail are individually transferred from sender to addressee, individual data packets are similarly transferred from sender to receiver.

2 Protocols

In order to effectively transfer data over the Internet, both the sender and receiver must agree on how that data should be structured and encoded. The solution is a **protocol**: a formal agreement on how to communicate.

Protocols dictate both syntax and semantics. **Syntax** is how communication is specified and structured (e.g. format, the order in which messages are sent and received). **Semantics** specify what a communication means for the sender and receiver (e.g. actions that must be taken when transmitting or receiving a packet, or when a timer expires).

The protocols that make up the Internet can be viewed a layered stack, with certain protocols layered above or below others. Each layer provides services to the layer(s) above it, while using the services provided by any layers below. A given layer has no insight into the workings of the layers around it: it doesn't matter what the higher layers do with the service that layer provides, nor does that layer care about how the lower layers' services are implemented.

Put another way, each layer defines an abstraction boundary. At that boundary, the layers above and below a given layer are opaque. The only interaction between layers occurs at the interface through which a layer uses another layer's services or provides its own.

3 OSI: Open Systems Interconnection

The Open Systems Interconnection (OSI) model represents the Internet in a seven-layer model. Each OSI layer includes one or more protocols and captures a single aspect of the performance and services that the Internet provides.

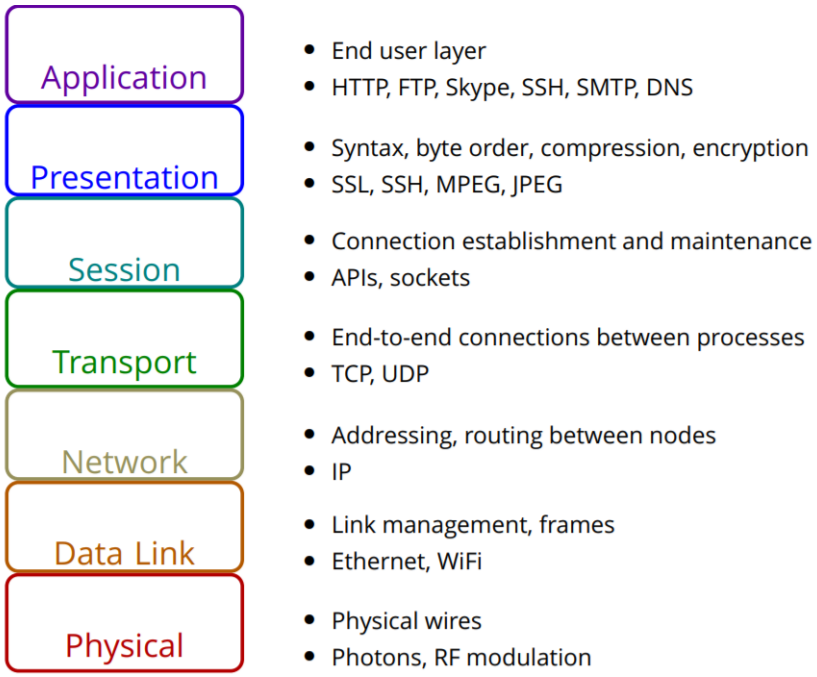

Figure 2: Visual summary of the seven OSI layers. (Source: lecture slides.)

From bottom to top, the OSI layers are as follows:

- The **physical layer** is at the bottom of the stack. This layer encompasses electrical and physical objects such as physical wires, photons, and RF modulation. In this layer, bits are transmitted from the source to its destination through some physical object (e.g. a physical wire).

- The **data link** or **link layer** lies immediately above the physical layer. Packets sent by this layer are encoded by the bits transmitted at the physical layer. This layer manages data links on a local network through protocols such as Ethernet and Wi-Fi, as well as introducing the notion of data being organized into frames.
- Above the link layer, the **network layer** relies on the Internet Protocol (IP) and provides many of the guarantees that make the Internet work. This layer handles addressing and routing between nodes, which enables the transfer of data between sources and destinations on different local networks.
- The **transport layer** lies above the network layer and handles end-to-end connections between the processes that send and receive data. This layer deals with the intent of data packets, regulating data transfer and checking packets for errors. The transport layer includes the TCP and UDP protocols.
- Next, the **session layer** controls the interactions between endpoints. This layer enables hosts to connect with each other via APIs and sockets, and to maintain those connections.
- The **presentation layer** handles details of how data is prepared for transfer, including syntax, byte order, compression, and encryption. This layer involves protocols such as SSL, SSH, MPEG, and JPEG.
- Finally, the **application layer** serves as the end user layer, providing an abstraction for all the services implemented by lower layers. This layer includes the high-level protocols that users and application developers most commonly interact with, including HTTP, FTP, SSH, SMTP, and DNS, among others.

Later sections examine some of these layers and their protocols in more detail, including the data link, network, and transport layers.

4 Packet Encapsulation

Each protocol defines its own syntax for data packets, where packets include headers that provide control information, such as the source and destination, as well as a payload containing the actual data. Packets from high-level protocols are encapsulated in the payloads of lower-level packets, with control information for the lower-level layer's packet derived from that of higher-level layers. Figure 3 provides a visualization.

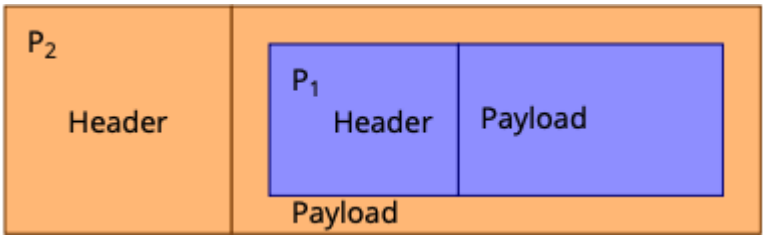

Figure 3: Packet encapsulation, visualized. P1 is the packet of high-level protocol N1, which uses the services of the lower-level protocol N2 with packet P2. P1 serves as the payload for P2, while P2's control information is constructed from P1. (Source: lecture slides.)

For example, a TCP packet from the transfer layer would be encapsulated in the payload of an IP packet, and the control information for that IP packet would be derived from the headers of the TCP packet. The TCP packet would in turn encapsulate a packet from a session-layer protocol, while the IP packet would itself by encapsulated by an Ethernet or Wi-Fi packet at the link layer. Thus the final data packet sent over the Internet consists of a series of nested packets, with the highest-layer packet as the innermost one, and the lowest-layer packet encapsulating all the rest.

5 The Internet Architecture "Hourglass"

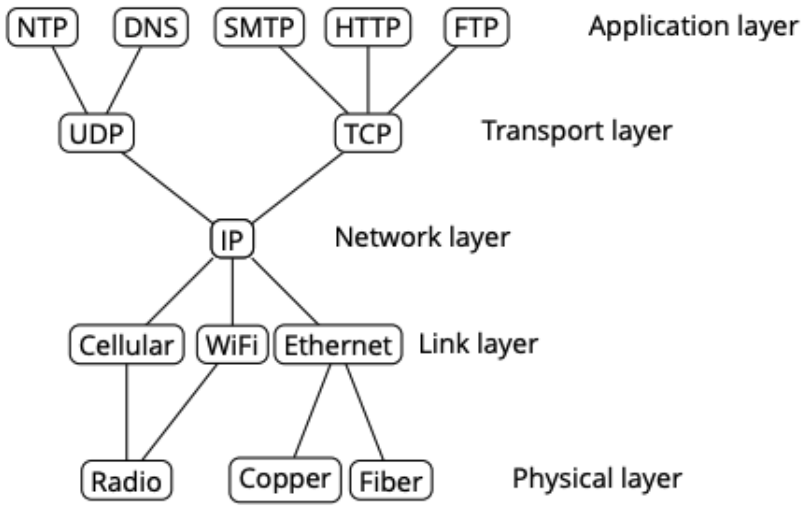

Figure 4: The hourglass structure of Internet architecture. (Source: lecture slides.)

We can visualize the Internet's architecture as following an hourglass shape. IP represents the narrow waist of the hourglass, with higher and lower layers spreading out above and below it. Analogously, everything in the Internet is based around IP. Higher-level packets are ultimately encapsulated in IP, while lower-level packets contain an IP packet within their nested payloads. This universality creates interoperability between protocols, enabling different high-level protocols to be transferred over the same network layer, without regard for the physical and link-layer constructs that IP builds upon.

6 Link Layer

The fundamental job of the link layer is to connect hosts to a local network. Examples of link layer protocols include Wi-Fi, Ethernet, and cellular connections. Here, we focus on Ethernet, as the most common link layer protocol. At the link layer, messages are sent via **frames** rather than packets, with each frame including a header, payload, and (non-cryptographic) checksum. See Figure 5 for a visualization of an Ethernet frame.

Every node on a local network (including computers, cell phones, etc.) has a globally unique 6-byte **Media Access Control (MAC)** address. These MAC addresses enable addressing, with source and destination addresses encoded into the frame header. In theory, MAC addresses are assigned by fixing some number of bytes in the address for each product vendor, with the vendor

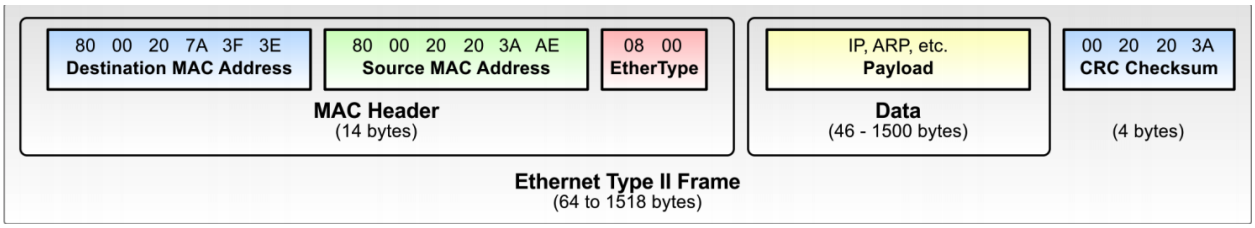

Figure 5: The structure of an Ethernet message frame. (Source: lecture slides.)

handling the distribution of unique addresses for the remaining bytes. In practice, however, some vendors cut corners, either through sloppy assignment or by not requesting an official allocation for the fixed portion of MAC addresses. The global uniqueness of MAC addresses is therefore an assumption that can be violated in certain situations.

Originally, Ethernet used a broadcast protocol where every node on a local network received every packet. However, networks capable of handling a broadcast protocol are limited in size, leading current Ethernet networks to use a switched implementation instead. Under this system, the network switch saves the physical ports for each MAC address and sends incoming packets to the intended physical port, if known.

Wi-Fi functions similarly to Ethernet, with the exception that data is sent over radio instead of through physical wires. The ability of nodes to move in relation to the radio can cause extra complications over the Ethernet model, as nodes may move in and out of communication range relative to each other.

7 IP: Internet Protocol

Internet Protocol (IP) constitutes the core of the network layer and provides the connectionless delivery model used by higher-level protocols. IP sends packets on a "best effort" basis: there are no guarantees that a packet will be delivered correctly, and there are no attempts to fix or recover from failed deliveries. Packets are frequently lost, sent to the wrong place, sent in the wrong order, sent multiple times, or even fragmented by lower layers.

IP does, however, provide a hierarchical addressing scheme, with networks given assigned prefixes that make it easier to find the intended destination of packet. IP addresses exist in two versions:

- IPv4 host addresses use 32 bits. They are written as four bytes expressed in decimal, with each byte separated by a dot.

  Example IPv4 address: `192.168.1.1`

- IPv6 provides 128-bit host addresses and has become increasingly common since unique IPv4 addresses ran out some years ago. IPv6 addresses are written as sixteen bytes in hexadecimal, separated by colons. Double colons (::) imply zero bytes filling that portion of the address.

  Example IPv6 address: `2620:0:e00:b::53` = `2620:0:e00:b:0:0:0:53`

Figure 6 visualizes the header of an IP packet that uses IPv4 source and destination addresses. The `Total Length`, `Time to Live`, `Fragment Offset`, `Header Checksum`, `Source Address`, and `Destination Address` fields are particularly relevant:

- `Total Length`: the length of the IP packet.
- `Time to Live`: a value that is decremented as the packet moves through the network.
- `Fragment Offset`: indicates whether the packet has been fragmented.
- `Header Checksum`: a non-cryptographic checksum.
- `Source/Destination Address`: the source/destination IP address.

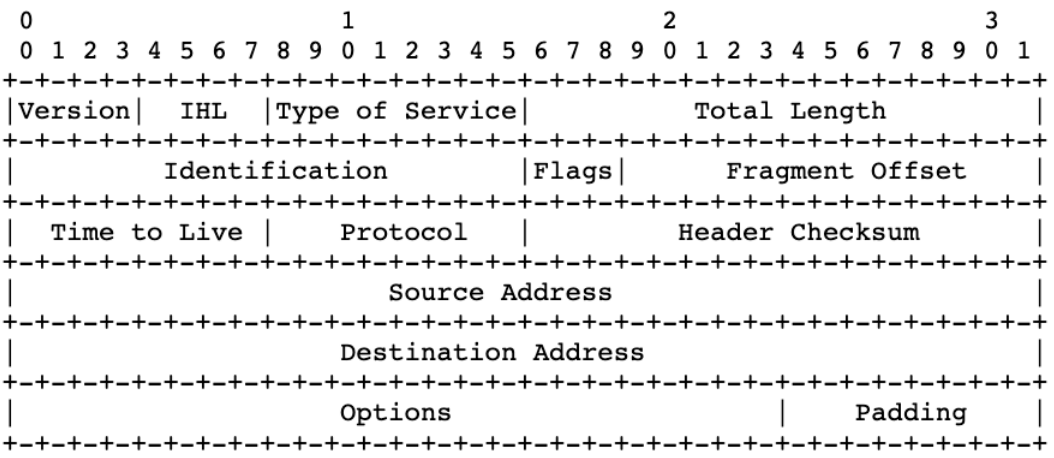

Figure 6: Example Internet Datagram Header - the structure of an IP packet header. *Note: each tick mark represents one bit position.* (Source: lecture slides, [2])

8 [Aside] DHCP: Dynamic Host Resolution Protocol

Note: DHCP was not covered in detail in the original lecture. This section is included as a useful aside, with supplemental information taken from [3].

Unlike Ethernet MAC Addresses, IP addresses must not only be unique but also reflect the structure of the internet. As mentioned above, IP addresses are hierarchical, meaning hosts in a given network should share the same network-prefix. As a result, IP addresses cannot be configured by the manufacturer, as that would require (a) the host to remain in a single fixed network, and (b) the manufacturer to be aware of this network.

Even with an IP address, a host must gather other configuration information before it can start sending packets. Most notably, the host will need the address of the default gateway for a given network in order to send packets to hosts outside of the local network.

Dynamic Host Configuration Protocol (DHCP) is an automated configuration method used in IP networks to assign IP addresses to connecting hosts and provide any other necessary information. In each administrative network, at least one DHCP server manages and leases a range of available IP addresses. By leasing the IP addresses, the DHCP server can safely recall IP addresses without interacting with a host, and this approach allows a given network to support more hosts than there are available addresses.

Although DHCP automates the process of assigning addresses, having a separate DHCP server in every local network would not be either effective or efficient, since each server would need to be

correctly and consistently managed. Instead, some networks have DHCP relays configured with an IP address of a DHCP server. The DHCP relay listens to DHCP messages on the local network and unicasts those messages to that DHCP server.

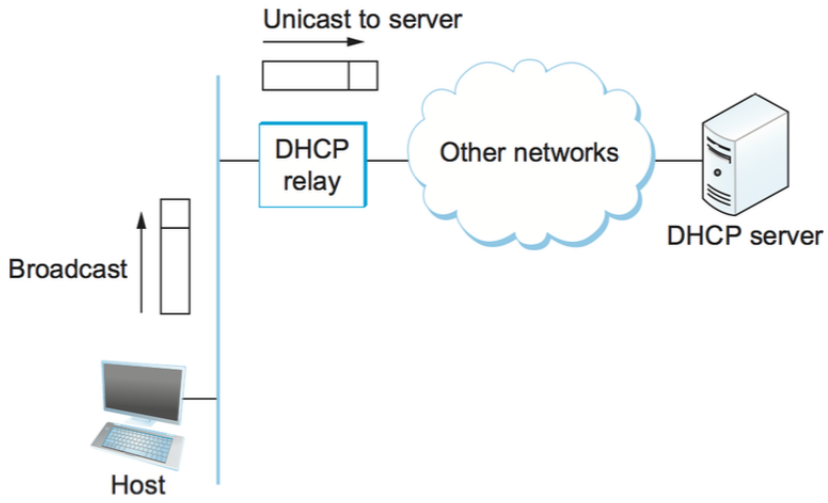

Figure 7: A host using a DHCP relay to connect to a DHCP server. (Source: [3].)

Before a host can lease an IP address, the host must first connect with the DHCP server in a process termed server discovery. To minimize the amount of manual configuration required, a newly connected host broadcasts a DHCPDISCOVER message by sending a packet with the IP broadcast address `255.255.255.255`. After establishing discovery, the server and host would communicate by sending IP offer, IP request, and acknowledgment messages. Eventually, the host is assigned an IP address and can begin sending packets.

9 ARP: Address Resolution Protocol

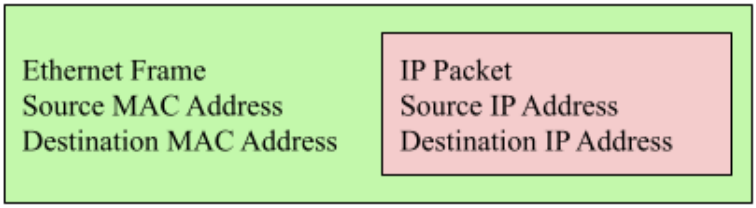

Figure 8: An IP packet encapsulated in an Ethernet frame. (Source: Drawn by one of the scribes.)

We now have two different types of protocols providing two different types of addressing: the globally unique MAC addresses used by Ethernet frames on local networks, and the IP addresses used in IP packets. As IP packets travel through a network, they are encapsulated in Ethernet frames. The IP packet contains the source and destination IP addresses, while the Ethernet frame contains the source and destination MAC addresses. So how does a host, such as your computer, learn what MAC addresses to send packets to?

The answer is **Address Resolution Protocol (ARP)**. ARP allows hosts to build a table mapping IP addresses to MAC addresses. Hosts begin this process by broadcasting an ARP request to the network: "Who has IP Address N?". If someone knows the answer, the host receives an ARP reply: "IP Address N is at MAC address M." Once the host learns this mapping, it knows to use MAC address M in the Ethernet frame that encapsulates any IP packet sent to IP address N. The host can repeat this process for any IP address to which it needs to send a packet.

10 Routing: Border Gateway Protocol (BGP)

The Internet is organized into **Autonomous Systems (ASes)**, or individual networks that each handle their own internal routing. ASes have peer, provider, and customer relationships among themselves; taken together, these connections form a rough tree, with a small number of backbone ASes forming a clique (a complete graph) at the root. ASes in this clique include backbone Internet providers such as ATT and Level 3.

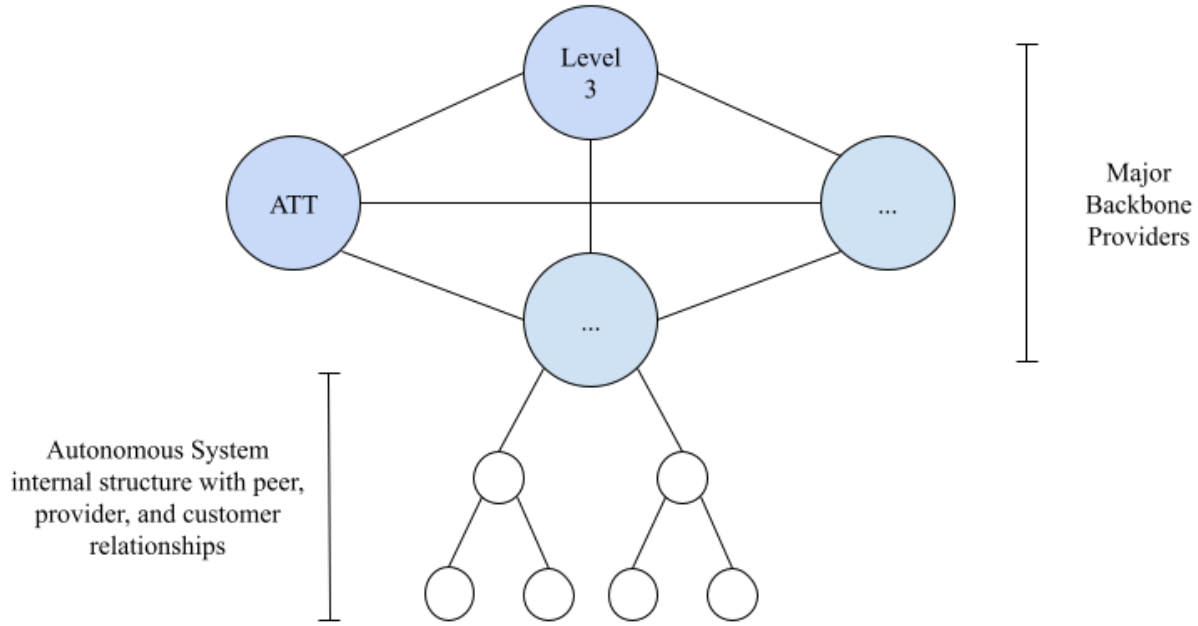

Figure 9: The Internet as a tree of Autonomous Systems. (Source: Drawn by one of the scribes.)

In order to route packets between ASes, routers in connected ASes exchange their routing tables using the **Border Gateway Protocol (BGP)**. Through BGP, each router announces what it can route to its neighboring ASes, which in turn add those routes to their own tables and announce them to their own neighbors. Routes thereby propagate through the AS network, enabling each router to maintain a a global table of routes.

Notably, BGP is not cryptographic, nor is there any verification of messages sent through the protocol. Any router can announce its ability to route to any IP prefix, regardless of whether that router can actually route to that prefix, and that false route will nevertheless propagate through the network.

11 TCP: Transmission Control Protocol

Despite the packet-based structure of the Internet, high-level processes want the abstraction of a stream of bytes that can be reliably delivered between applications on different hosts. The transport-layer **Transmission Control Protocol (TCP)** provides this abstraction, along with several other features:

- *A connection-oriented protocol.* TCP provides the abstraction of a dedicated, persistent connection opened between ports on different hosts, through which a "continuous" stream of bytes can be transmitted between those hosts.
- *Explicit set-up and teardown.* Due to the persistent nature of TCP connections, TCP provides ways for hosts to explicitly indicate to each other when to expect and stop expecting data.
- *Multiple concurrent long-lived dialogues.* TCP allows end hosts to have several connections open at once with several other hosts, including over long periods of time.
- *Congestion control.* TCP adapts the number of packets it sends in a given time period according to the network path's current capacity, to avoid overwhelming the network.

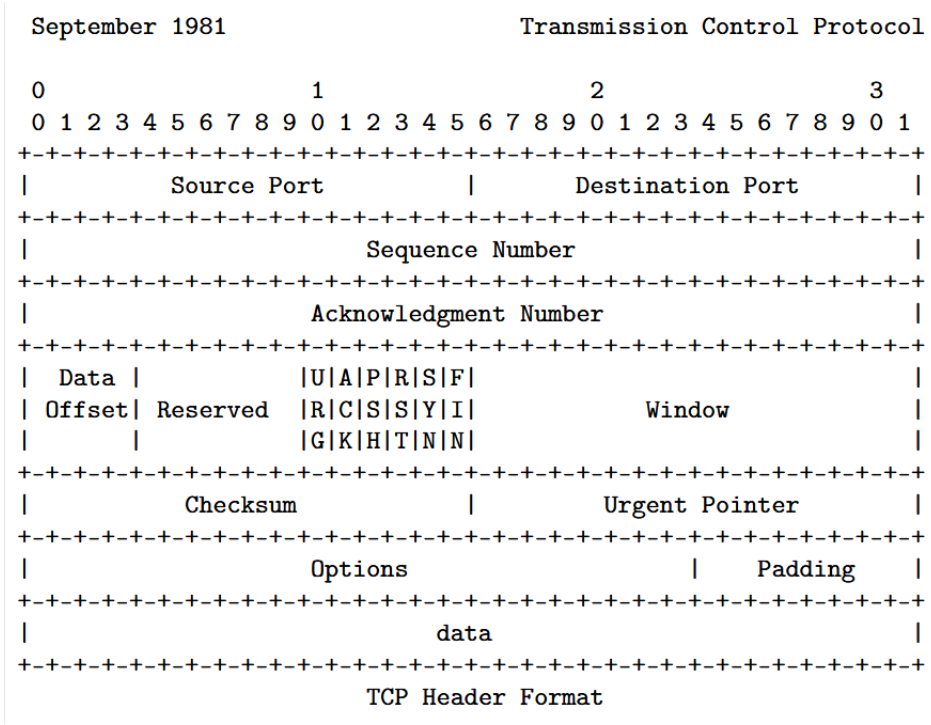

Figure 10: The header of a TCP packet. (Source: lecture slides, [2].)

Figure 10 shows the structure of a TCP packet header. While no addresses are needed at the transport layer, the TCP header does specify source and destination **ports**. TCP connections are established between ports on host machines (for instance, between port A on host M and port B on host N), with each port represented as a 16-bit number. Certain ports are conventionally used for specific applications, such as port 80 for HTTP, 433 for HTTPS, 25 for SMTP (email), 67 for DHCP, 22 for SSH, and 23 for telnet.

Sequence numbers are similarly vital for TCP. Under TCP, every byte in the data stream is assigned its own 32-bit sequence number, with each IP packet containing a segment of continuous bytes from the data stream. The sequence number of the packet is the sequence number of the first (lowest-numbered) byte in the payload and is used to place the payload in the reconstructed data stream. For example, a packet containing bytes numbered 100 to 163 would be assigned the sequence number 100. When the receiver reconstructs the data stream, this packet would be placed after the packet whose payload end at sequence number 99 and before the packet with sequence number 164.

TCP connections provide two logical data streams: one from host A to host B, and another from host B to host A. Each host sends data to the other, and the receiver acknowledges that data in the packets it sends back through the use of **acknowledgement numbers**. To acknowledge a received package, the host sets the ACK flag and includes the next expected sequence number as the acknowledgement number. For example, a host that has received a packet with bytes 100-163 would send an acknowledgement number of 164, indicating that the next sequence of bytes it expects should start with 164. If a given packet is not acknowledged in a reasonable amount of time, its original sender will retransmit that packet, adjusting the frequency with which packets are retransmitted to account for network congestion.

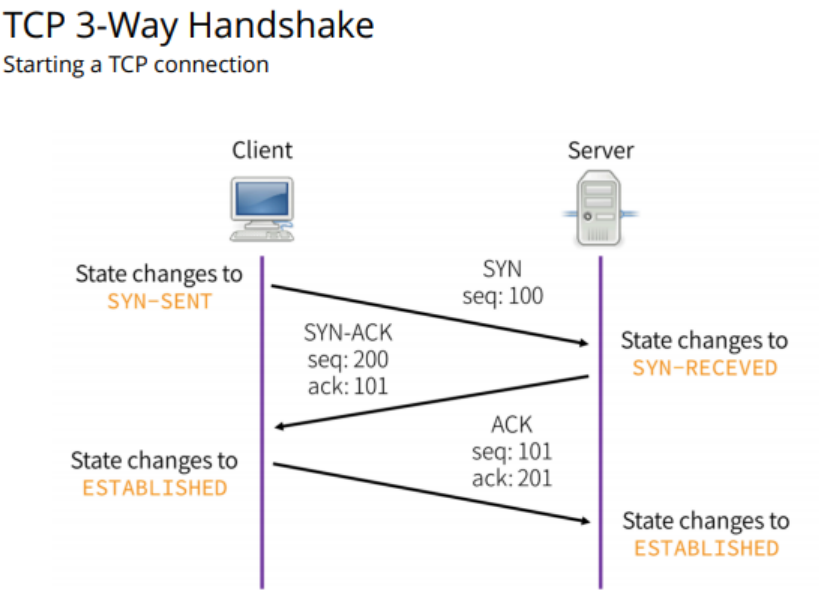

Figure 11: The three-way handshake used to initiate a TCP connection. (Source: lecture slides.)

To start a TCP connection, hosts engage in a **three-way handshake** (Figure 11). The client host begins the process by selecting an arbitrary sequence number and sending a SYN packet with that sequence number. In modern systems, this initial sequence number is chosen at random. The server then responds to the client by sending a SYN-ACK packet with its own selected sequence number, which is similarly chosen at random. Finally, the client responds with an ACK packet, thereby opening the TCP connection for the transmission of data.

Once a host no longer wishes to send or receive data, they can close a TCP connection through one of two signals:

1. FIN initiates a clean close of a TCP connection. This method waits on acknowledgement from the receiver before tearing down the connection.

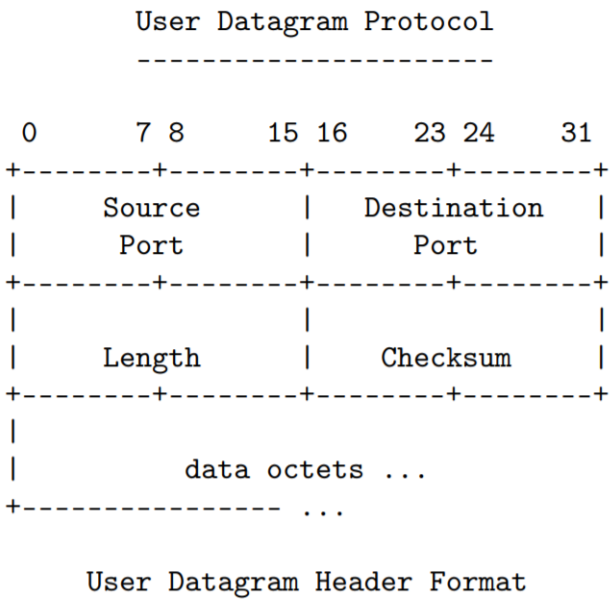

Figure 12: The header of a UDP packet. (Source: lecture slides, [2].)

2. RST, by contrast, causes the receiver to tear down the TCP connection immediately, without any acknowledgment required. This signal is designed to deal with spurious packets from old TCP connections.

12 UDP: User Datagram Protocol

Like TCP, **User Datagram Protocol (UDP)** is a transport-layer protocol that provides port addressing on top of host addresses, ensuring that data sent to a single host reaches the correct application. Unlike TCP, however, UDP provides no service quality guarantees, acting only as a minimal wrapper around IP. As a result, UDP is most useful for applications that need only best-effort guarantees, or that value simplicity and speed over error-checking, such as DNS and NTP.

13 DNS: Domain Name Service

Domain Name Service (DNS) handles mappings between human-readable host names (e.g. `ucsd.edu`) and IP addresses (e.g. `132.239.180.101`). DNS provides a delegatable, hierarchical namespace, in which high-level domains (e.g. `.edu`) control lower-level domains (e.g. `ucsd.edu`), which in turn may control even lower-level domains (e.g. `cse.ucsd.edu`). Figure 13 presents an example DNS structure representing a small sample of high- and low-level domains.

There are thirteen main DNS root servers, which are listed in Table 1. Since DNS requests take some time to evaluate, their responses are cached to make for quicker responses in the future. The DNS authorities are queried progressively according to the domain name hierarchy. This means that if a given DNS authority does not know of a certain domain, it will request the information from the next DNS authority up the namespace hierarchy.

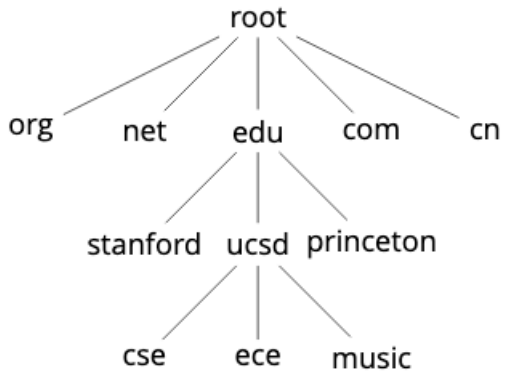

Figure 13: An small portion of the DNS namespace. (Source: lecture slides.)

HOSTNAME	IP ADDRESS	OPERATOR
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	Verisign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	USC, Information Sciences Institute
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	Verisign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

Table 1: List of DNS root servers [1].

To view the DNS record of a particular domain, you use the command `$ dig cseweb.ucsd.edu` (replacing `cseweb.ucsd.edu` with the domain of your choice). Figure 14 displays an example of the output you can expect to see.

For the purposes of this class, the first important section is the `QUESTION SECTION`, which restates the domain being queried. The subsequent section, the `ANSWER SECTION`, provides the answer to that query. In Figure 14, the `ANSWER QUESTION` states that `cseweb.ucsd.edu` has a `CNAME` (Canonical Name record, or alias), which is `roweb.eng.ucsd.edu`, and that the IP address for this alias is `132.239.8.30`.

For a more detailed report on the DNS information, you can run `$ dig cseweb.ucsd.edu +trace`. This command provides output of the form shown in Figure 15, including the trace of the domain names that were queried before finding the requested domain.

14 Using the Internet: A Worked Example

In this section, we will work through an example of how all of the topics discussed above fit together. Consider the following situation:

```
$ 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
```

Figure 14: Output from running `$ dig cseweb.ucsd.edu`. (Source: lecture slides.)

> You are waiting masked and outdoors for a takeout order from a café, and while you're waiting you perch against the wall at a safe social distance from everyone else, pull your laptop out, connect to the café's WiFi network, and type `ucsd.edu` into your browser's URL bar.

Before the web page for `ucsd.edu` appears on your browser screen, the following series of events will occur, in order:

1. Your laptop uses DHCP to bootstrap itself onto the local network. However, since you have only just connected, your laptop has no information on what routers or IP address it should use. To fix this problem, it broadcasts the signal `DHCPDISCOVER` to `255.255.255.255` with its own MAC address. The DHCP server then responds with some configuration containing a lease on the host IP address (the IP address your laptop should use), the gateway IP address (the router your laptop should talk to), and the DNS server information that your laptop will need to make its request.

2. Your laptop will then make an ARP request to learn the MAC address of the local router. Every connection your laptop makes outside the local network will be encapsulated in a link-layer frame with the local router's MAC address as the destination. Your laptop encapsulates each IP packet in a Wi-Fi Ethernet frame addressed to the local router. Then the local router unwraps these Ethernet frames and forwards the re-encoded information on its fiber connection to its Internet Service Provider (ISP) or another part of the network. Essentially, the Wi-Fi router re-encodes the IP packet with the fiber protocol and thereby communicates to the ISP. Each hop to a different router re-encodes the link layer for its own network.

3. Although you have typed `ucsd.edu` into your URL bar, your browser does not know what IP address corresponds to that domain. Your laptop therefore does a DNS lookup on `ucsd.edu`, using the IP address of a local DNS server learned from DHCP in step 1, or else a server

```
$ 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/MKzOX1UBwu04kzwQNajhg1AflK7j5Jvd9N
;; 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 . BsoO9WI4UphacN5rL0B4f3bCzVPptbmTCKHwcMgb6
;; 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.
9DHS4EP5G85PF9NUFKO6HEKOO48QGK77.edu. 86400 IN NSEC3 1 1 0 - 9V5L4LUB1VNJ9EQQLIHEQCBREACL25OO  NS SOA RRSIG DNSK
9DHS4EP5G85PF9NUFKO6HEKOO48QGK77.edu. 86400 IN RRSIG NSEC3 8 2 86400 20191111043435 20191104032435 47252 edu. M5
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. BK
;; 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
```

Figure 15: Output from running `$ dig cseweb.ucsd.edu +trace`. (Source: lecture slides.)

(like `9.9.9.9`) that your laptop already had hard-coded. Each request your laptop makes is a DNS query encapsulated in one or more UDP packets, encapsulated in one or more IP packets. Each response tells the laptop what next DNS authority to query until it learns the final IP address (`75.2.44.127`) for `ucsd.edu`. This address is cached, along with all of the authorities that were queried in the process. So the next time you query a `.edu` address, your request will not have to go all the way to the root, and can instead start with the `.edu` authority.

4. Now that you have learned the IP address for `ucsd.edu`, your laptop opens a TCP connection using the TCP three-way handshake to `75.2.44.127`. Each packet of the handshake is encoded in an IP packet that is encoded as Ethernet frames, which are repeatedly 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. As the packet passes through the network, the Wi-Fi router in the cafe is going to pass packets addressed to any addresses not on the local network to the upstream provider, which will continue passing it through the network according to the routing information. Your packet thus passes through a series of ASes: from the café network (ATT) to `sbcglobal.net` → `att.net` → `level3.net` → `cenic.net` → `ucsd.edu`.

5. Now that your laptop has opened a TCP connection to `ucsd.edu`, it sends a sequence of packets containing the text of an HTTP GET request. To communicate your request, the HTTP protocol is encapsulated in the TCP packets, which are in turn encapsulated in the IP packets, which are finally encapsulated in the Ethernet packets.

6. Finally, your laptop receives an HTTP response through the TCP connection. Based on the

HTTP response, the laptop performs a new DNS lookup, TCP handshake, and HTTP GET request for every resource in the HTML as it renders. For example, if the HTTP response to your initial GET request includes some HTML with additional images, your laptop may need to perform a DNS lookup on the host name of the image, then perform a new TCP handshake with the new IP address to retrieve the image, and finally send an HTTP GET request for that resource. Once all the resources are thus loaded, your browser will at last render the HTML of `ucsd.edu`.

References

[1] Internet Assigned Numbers Authority. `https://www.iana.org/domains/root/servers`.

[2] Information Sciences Institute University of Southern California. `https://tools.ietf.org/html/rfc791`.

[3] Larry Peterson and Bruce Davie. *Computer Networks: A Systems Approach*.