Binary and Hexadecimal Notation

Why Binary and Hexadecimal?

- Binary and hexadecimal are important to us in computing because of their connection to bits and bytes.
- A bit is a single digit, representing two states: on (1) and off (0)
- A byte is a sequence of 8 bits. If we think about this sequence in decimal like we are most used to, it can represent 2^8 or 256 different values (0 to 255).
- The 8 digits of a byte can be broken up into two groups of four. Four digits means we have 16 possible values in binary, which is where hexadecimal comes in: We can represent a byte with two digits of a hexadecimal number (16^2 = 256).

Converting Bases

- It's often easiest for us to think about numbers in bases other than 10 (decimal) by converting them back to decimal first.
- When we are representing numbers, each digit represents the base raised to a different power.
- Converting from other bases into base-10 (decimal) involves us multiplying each digit's base-10 value by the respective base^digit's place. For example:
  - When we say 476 in our standard base-10 or decimal notation, we are saying that we have `4*10^2 + 7*10^1 + 6*10^0 = 476`.
  - This same logic applies when we are talking about other bases.
  - In binary or base-2, if we have 1101, we are talking about `1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 9`
  - In hexadecimal or base-16, our digits go from 0-9, A-F, meaning if we have F13, we are talking about `15*16^2 + 1*16^1 + 3*16^0 = 3859`
- Converting decimal into other bases follows a reversed process. Dividing our base-10 number by the base we are converting into will give us a remainder that represents the corresponding digit, continuing until we've captured all digits:
  - To convert 217 into binary:
    - Smallest digit: 217 / 2 = 108 remainder 1 => 1
    - Next digit: 108 / 2 = 54 remainder 0 => 0
    - Next digit: 54 / 2 = 27 remainder 0 => 0
    - Next digit: 27 / 2 = 13 remainder 1 => 1
    - Next digit: 13 / 2 = 6 remainder 1 => 1
    - Next digit: 6 / 2 = 3 remainder 0 => 0
    - Next digit: 3 / 2 = 1 remainder 1 => 1
    - Next digit: 1 / 2 = 0 remainder 1 => 1
    - All together, largest to smallest digit: 11011001
  - To convert 497 to hexadecimal:
    - Smallest digit: 497 / 16 = 31 remainder 1 => 1
    - Next digit: 31 / 16 = 1 remainder 15 => F (our hexadecimal digit)
    - Next digit: 1 / 16 = 0 remainder 1 => 1
    - All together, largest to smallest: 1F1

How is this relevent to networking specifically?

- As with most computing, at the lowest level, networking is just a series of 1s and 0s.
- The packets of information that we send across the internet and are deciphered on the other end are made up of binary digits.
- Hexadecimal is an easy way for us to easily reference sections of these binary streams in a more condensed format (a quarter the size!)
- Specifically, we'll see that the version of IP that we are using is version `0100` for IPv4 and version `0110` for IPv6. We'll also see that IPv6 IP addresses and MAC addresses for physical devices use hexadecimal in order to condense the very large number of possible outcomes as opposed to the long strings that their binary counterparts would require.

Applications to JavaScript

- We can convert a standard decimal number into a different base representation by supplying the base that we are wanting as an argument to the `toString` method:
  - `Number(30).toString(16); // 1e`
  - `Number(30).toString(2); // 11110`
- We can convert these string representations back into a number in base-10 by supplying a second argument to `parseInt` to indicate what base we are parsing from:
  - `parseInt('1e', 16); // 30`
  - `parseInt('11110', 2); // 30`

Main Takeaways

- Recognize and be able to convert between simple base-10 (decimal) numbers and our other common computing bases, base-2 (binary) and base-16 (hexadecimal)

TCP/IP History

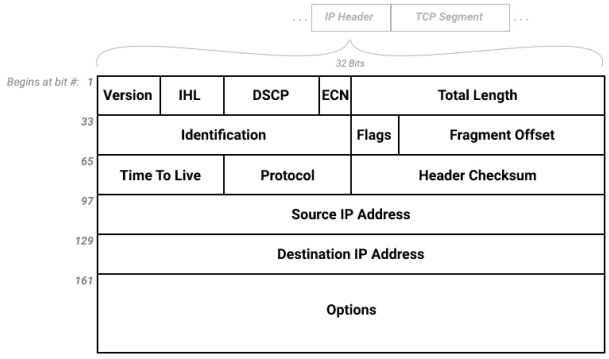
- The Transmission Control Protocol and Internet Protocol were broken apart from the original Transmission Control Program.
- TCP is responsible for fault-tolerance between networks, allowing for data to be re-sent if it fails to reach its destination
- IP is responsible for the end-to-end nature of networks, allowing data to be sent and received from each host, eliminating a single central system that could take down a whole network

Packet-Switching

- A packet is how data is broken up to be sent across a network
- Each packet has metadata stored in headers as well as a body with the content that is being sent

IPv4

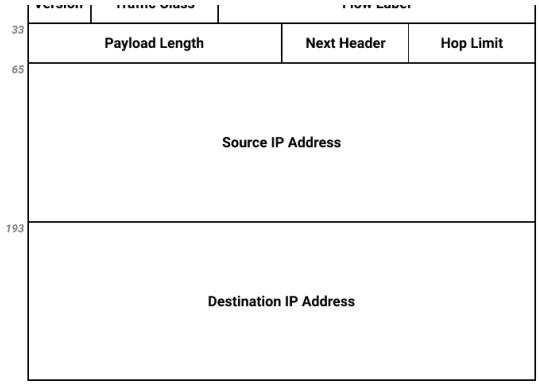
- Best known version of IP, finalized back in 1983
- Consists of a series of at least 13 fields, each with a specified length
- The first field is the version, which is the four bits that represent version 4: `0100`
- The next 10 fields after the version specify metadata about the packet, such as header length and protocol (what kind of data is being transferred).
- The last two required fields are the source IP address and the destination IP address, which guide the packet along the internet to its destination.
- We also have a final `Options` field, which is very rarely used. Typically any metadata needed to be transferred would be caught by one of the previous fields. (Unnecessary example: a Router Alert, which tells every router to evaluate this packet, even if it's not the final destination)



- Overall, if we do not have any extra options, our packet header is 160 bits, or 20 bytes long.
- An IPv4 address is a series of 4 octets, meaning 4 8-bit binary numbers, which as we saw is one byte each.
  - 192.18.1.1 -> 11000000.00010010.00000001.00000001
  - This format allows for approximately 4 billion unique addresses

IPv6

- Finalized in 2017, the main goal was to expand the number of addresses available, but also restructures and consolidates our packet headers.
- We now only use 8 fields:
  1. Version: Just like IPv4, the first four bits of our packet header represent our version number, which is `0110`, representing our decimal 6.
  2. Traffic Class: Identifies the type of packet data
  3. Flow Label: Adds packet sequencing to IP, but is experimental
  4. Payload Length: Specifies the size of this packet
  5. Next Header: Typically identifies the Transport Layer protocol (TCP, UDP, etc.). It can also indicate an extension header is present, where extra options for the packet are specified (similar concept to the Options field of IPv4, but a different, chainable implementation)
  6. Hop Limit: Decremented by one each time this packet passes through an intermediary. Prevents the packet from being passed around forever.
  7. Source IP Address



8. Destination IP Address

- Overall, our packet header is 320 bits, or 40 bytes long
- IPv6 addressess take up the majority of a packet header. They are 128 bits long each vs IPv4's 32 bit addresses. They are typically represented by "eight colon-ed hexadecimal"
  - 2600:6c5e:157f:d48c:138f:e0ba:6fa7:d859 -> 0010011000000000:0110110001011110:0001010101111111:1101010010001100:0001001110001111:1110000010111010:0110111110100111:1101100001011001
  - The expansion of the address length now allows for approximately 350 undecillion addresses (3.5×10^38) instead of IPv4's 4 billion.

Localhost Address

- A special address that references your current machine.
  - IPv4: 127.0.0.1
  - IPv6: ::1
- Also referred to as the loopback address (the machine is referring to itself)

All interfaces Address

- An address that accepts all interfaces on a network
  - IPv4: 0.0.0.0
  - IPv6: ::
- We wouldn't send a packet to this address, it just indicates that if we are listening on this address we are looking at all packets that pass through on the network.

Main Takeaways

1. TCP vs IP

- TCP is responsible for fault-tolerance between networks, allowing for data to be re-sent if it fails to reach its destination
- IP is responsible for the end-to-end nature of networks, allowing data to be sent and received from each host, eliminating a single central system that could take down a whole network

2. Packet Switching

- Data is sent in packets, with headers that have metadata about the packet

3. Packet Structure

- The packet header always starts with the version number, followed by other metadata fields, then the source and destination addresses
- The TCP segment (next section!) then follows the IP headers

4. Version Field

- IPv4: `0100`, which is 4 in decimal notation
- IPv6: `0110`, which is 6 in decimal notation

5. Special Addresses

- Localhost: References the current machine
  - IPv4: 127.0.0.1
  - IPv6: ::1
- All interfaces: Receives all incoming packets on a network
  - IPv4: 0.0.0.0
  - IPv6: ::

How do Transport Protocols fit in?

- Packages up units of data with instructions for how to put it back together (order of segments), providing reliability in data transfer
- Acts as the middleman between HTTP (application-to-application communication) and IP (machine-to-machine communication)

Ports

- Virtual interfaces used by transport protocols in order to determine what application or service the data is intended for at the IP address.
- Indicated by a number:
  - 127.0.0.1:3000 indicates port 3000 at IP address 127.0.0.1 (localhost:3000)
  - Written together like this is called a socket

TCP

- Transmission Control Protocol
- Connection oriented protocol, connecting two sockets to transmit data.
- Segments of data have an order, allowing the data to be built back up on the other side.
- If segments are received out of order or lost, we are able to request the segment to be resent.
- Takes a relatively long amount of time because of the checks in place to make the transmission reliable.
- Examples: HTTP, file transfers, media streaming (YouTube)
- The optional TCP Connections reading gets into some more specifics of what these segments look like and how the reliability and consistency is maintained.

UDP

- User Datagram Protocol
- Connection-less, meaning there is no verification that data was received.
- Services that prioritize speed rely on UDP
- Examples: Live video (Zoom!), VoIP, DNS (coming up soon!)

Main Takeaways

1. Transport protocols define the middlemen between our Application Layer (HTTP) and our Internet Layer (IP). Converts the data that the applications are communicating with into segments that can be transferred by our IP packets.
2. Ports define which application/service our segments are intended for at a destination IP address.
3. TCP is a reliable transport protocol that protects against loss of data.

- Good use case: HTTP, file transfers, media streaming (YouTube, etc.)

4. UDP is an unreliable transport protocol that prioritizes speed.

- Good use case: real-time communication like live video and VoIP, DNS (prioritizing speed here)

What is DNS?

- Domain Name System
- Provides names for devices on a network instead of having to use IP addresses.
- Simple and distributed (specifically formatted text files that are redundant across many servers)

Domains and Subdomains

- Domains are the "friendly" name for a website's host (the server that is responding to requests with content)
- Top-level domain (TLD): the last part of the URL before application routes. Common examples would be .com, .org, .gov, etc.

    - TLDs are managed by domain registries
- Second-level domain: often combined with the TLD to be referred to as a generic "domain", this is the main name associated with the website. For example, google is the second-level domain, which we may refer to as the domain google.com. appacademy is the second-level domain for what we may refer to as the domain appacademy.io.
    - Second-level domains are purchased from domain registrars.
- Subdomains can be expanded even farther to the left (but after the `https://` protocol) into third-level, fourth-level, etc. `www` would be considered a subdomain, or `open` from `open.appacademy.io`.

DNS Resolution

- Generally, we start at the right-most domain and ask the name server if they have the IP address for our desired full domain.
- Name servers that don't have the IP address on file will pass us along to another, more relevant name server until an IP address is found.
- The name server that has the IP address on file is referred to as the authoritative name server for our domain.

Zone Files and DNS Record Types

- A zone file is maintained by name servers that contains host names, IP addresses, and resource types
- The zone file also tracks the Time to Live (TTL) for records. This indicates long we should keep our records cached after a request comes in. Shorter caches mean more frequent updates to the current IP address, but longer caches mean less frequent reading of the zone file, which can be a longer operation.
- There are many comon record types present in these zone files:
    - SOA: Start of Authority. Points to a name server that is the primary authority for the domain. This record is present on every name server.
    - NS: Name Servers. Points to name servers for the zone. There will always be at least two name servers per zone for redundancy.
    - A / AAAA: Map a resource directly to an IP address. These are the ultimate records that our queries are looking for. `A` records are used for IPv4 addresses and `AAAA` records are used for IPv6 addresses.
    - CNAME: Acts as an alias, indicating what resource this domain should also point to. (We often see this with the www subdomain, where a CNAME record for www would exist for google.com, indicating we can request `www.google.com` and get the same response as `google.com`)
    - MX: Mail Exchanger. Used to direct messages to a mail server instead of an IP address (allows us to use `@gmail.com` instead of `@123.45.67.89` - which is obviously not the actual address)

Main Takeaways

1. Domains are the friendly names that we can refer to instead of having to remember IP addresses
2. Top-level domains (TLDs) are the right-most part of the URL before application routes. Moving left we get second-level, third-level, etc., subdomains

- For https://open.appacademy.io/learn/js-py--aug-2020-online/
    - Top-level domain: io
    - Second-level domain: appacademy
    - Third-level domain: open

3. Domains are tracked using DNS records in zone files that are maintained by name servers
4. There are several common DNS record types. Be familiar with each:

- SOA
- NS
- A / AAAA
- CNAME
- MX

Three main types of Network Hardware

- Hubs
- Switches
- Routers

Hub

- Physical layer (OSI layer 1) device
- Takes a single connection and converts it into multiple connections by duplicating the data and broadcasting it to all connected devices.
- No sort of filtering or directed forwarding. Everything that passes into a hub with be duplicated and sent to everything that is connected to it.
- Very little reason to use them in a modern network.

Switch

- Data Link layer (OSI layer 2) device
- Extend physical connection as well as connection management based on frames and MAC addressess
- Can be thought of as "intelligent" hubs. They know which devices are connected to them (through a MAC address table) and can direct network traffic to them specifically.
- A switch can perform three actions with data that it receives:
    - Flood: If it doesn't have the MAC address that was intended in its table, it will send the data out to every device (except where it received it from). If the intended device was found and responds, the switch updates its table so it know specifically where to send the data in the future.
    - Forward: If the MAC address is already in its table, it will send the data specifically to that device.
    - Filter: If the intended device for the data is the same as the source, the switch will not do anything with it and drop the data. This is beneficial because no other device needs to know about it, so we stop the propagation.

Router

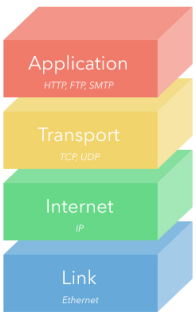
- Network/Internet layer (OSI layer 3) device
- Manage packets at the network level and handle routing between remote networks
- Connect separate networks with each other.
- Maintains a routing table to pass data to hardware connected to the local network.
- A router uses Network Address Translation (NAT), which assigns the router one IP address for all external communication. The router is able to translate ports associated with that data to the IP address used on the internal network for different devices.

Main Takeaways

1. Three main categories of hardware devices are hubs, switches, and routers
2. A hub will broadcast to everything that it's connected to.
3. A switch is able to pass data to a specific device on a network if it already has a reference (forward), or it can broadcast to all devices (flood) if its MAC address is not in its table yet.
4. A router connects a local network to other networks. A single IP address is used for external communication, then the router can forward to the IP addresses used on the internal network.

TCP/IP Model

- Transmission Control Protocol / Internet Protocol reference model
- More practical model of the two, focusing on getting data from one place to another



tcp-ip model

Layers of the TCP/IP Model

Application - User-facing data - HTTP and FTP - Anything that is transmitted from the Tansport Layer is considered Application Layer data

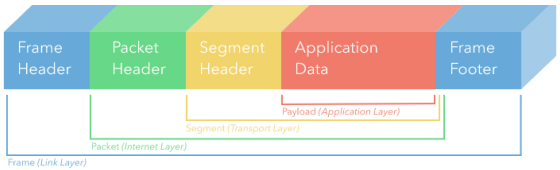
Transport - TCP and UDP - Connectivity between and clients and servers - Relies on lower layers to establish network connectivity

Internet - IP - Data is processed in packets - Routing is handled with IP addresses - Connects separate networks together

Link - Low-level communication standards - Aren't concerned with type of data, focus on getting data from one local network resource to another - When we are concerned with other networks, we go to the internet layer

(*)Physical - Not an official layer! Included here because some people will break this out unofficially - Electrical concepts like transmission across wires, conversion of electricity into bits

Encapsulation



- The layers of the TCP/IP model represent an encapsulation of data - The Link Layer frame includes the whole stack, for example, while a Transport Layer segment includes the segment header and the Application Layer's payload.

Main Takeaways

1. Be able to recognize a diagram as the TCP/IP model, as opposed to the OSI model
2. Give a brief description of each layer (What is its major concern and an example)

- Application: User-facing data, such as HTTP or FTP (file transfer)
- Transport: Connectivity between clients and servers, such as TCP or UDP
- Internet: Routing between separate networks, such as IP
- Link: Low-level communication between local resources on a network, such as Ethernet

3. Each layer of the model encapsulates the previous.

- For example, the application layer is the data that is being transferred, but the transport layer surrounds this data with transport protocols (such as TCP headers), which are surrounded by IP packets defined by protocols of the internet layer, which are sent along an ethernet cable with standards defined in the link layer.

OSI Model

- Open Systems Interconnection reference model
- Conceptual model, limited practical use
- Good for understanding networking concepts

osi-model

Layers of the OSI Model

7. Application - Example: HTTP - Information used by client-side software - Data will interact directly with applications and can be displayed to the user with limited translation

6. Presentation - Example: JPEG, GIF - Data gets translated into a presentable format - Often called the syntax layer since it translates machine-readable syntax into human-readable syntax - Data compression, encryption, character encoding, image formats are examples

5. Session - Example: RPC (Remote Procedure Call) - Authentication and data continuity - Authorize actions, reestablish dropped connections

4. Transport - Example: TCP, UDP - Mirrors TCP/IP's Transport Layer - Focused on data integrity and connectivity

3. Network - Example: IP - Mirrors TCP/IP's Internet Layer - Manages connections between different remote networks - Transfers packets across intermediary devices

2. Data Link - Example: Ethernet - Connections between one network interface to another - Primarily used by machines in a local network (ie targeting different MAC addresses)

1. Physical - DSL, 802.11 (Wi-Fi) - Translating from electrical signals to bits of data

Main Takeaways

1. Be able to recognize a diagram as the OSI model, as opposed to the TCP/IP model
2. Give a brief description of each layer (see short examples above)

Recap

Binary and Hexidecimal

1. Recognize and be able to convert between simple base-10 (decimal) numbers and our other common computing bases, base-2 (binary) and base-16 (hexidecimal)

Internet Protocol

1. TCP vs IP

- TCP is responsible for fault-tolerance between networks, allowing for data to be re-sent if it fails to reach its destination
- IP is responsible for the end-to-end nature of networks, allowing data to be sent and received from each host, eliminating a single central system that could take down a whole network

2. Packet Switching

- Data is sent in packets, with headers that have metadata about the packet

3. Packet Structure

- The packet header always starts with the version number, followed by other metadata fields, then the source and destination addresses
- The TCP segment (next section!) then follows the IP headers

4. Version Field

- IPv4: `0100`, which is 4 in decimal notation
- IPv6: `0110`, which is 6 in decimal notation

5. Special Addresses

- Localhost: References the current machine
  - IPv4: 127.0.0.1
  - IPv6: ::1
- All interfaces: Receives all incoming packets on a network
  - IPv4: 0.0.0.0
  - IPv6: ::

Transport Protocol

1. Transport protocols define the middlemen between our Application Layer (HTTP) and our Internet Layer (IP). Converts the data that the applications are communicating with into segments that can be transferred by our IP packets.
2. Ports define which application/service our segments are intended for at a destination IP address.
3. TCP is a reliable transport protocol that protects against loss of data.

- Good use case: HTTP, file transfers, media streaming (YouTube, etc.)

4. UDP is an unreliable transport protocol that prioritizes speed.

- Good use case: real-time communication like live video and VoIP, DNS (prioritizing speed here)

Domain Name System (DNS)

1. Domains are the friendly names that we can refer to instead of having to remember IP addresses
2. Top-level domains (TLDs) are the right-most part of the URL before application routes. Moving left we get second-level, third-level, etc., subdomains

- For https://open.appacademy.io/learn/js-py--aug-2020-online/
  - Top-level domain: io
  - Second-level domain: appacademy
  - Third-level domain: open

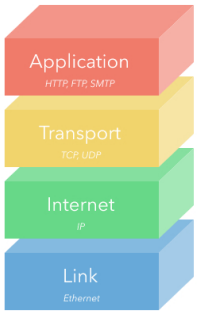
3. Domains are tracked using DNS records in zone files that are maintained by name servers. These records map to IP addresses.
4. There are several common DNS record types. Be familiar with each:

- SOA: Start of Authority. Points to a name server that is the primary authority for the domain. This record is present on every name server.
- NS: Name Servers. Points to name servers for the zone. There will always be at least two name servers per zone for redundancy.
- A / AAAA: Map a resource directly to an IP address. These are the ultimate records that our queries are looking for. `A` records are used for IPv4 addresses and `AAAA` records are used for IPv6 addresses.
- CNAME: Acts as an alias, indicating what resource this domain should also point to. (We often see this with the www subdomain, where a CNAME record for www would exist for google.com, indicating we can request `www.google.com` and get the same response as `google.com`)
- MX: Mail Exchanger. Used to direct messages to a mail server instead of an IP address (allows us to use `@gmail.com` instead of `@123.45.67.89` - which is obviously not the actual address)

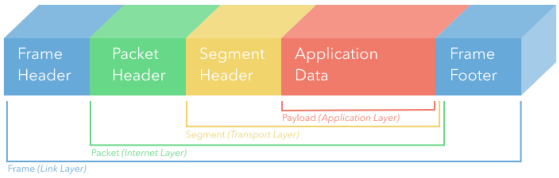
Network Hardware

1. Three main categories of hardware devices are hubs, switches, and routers
2. A hub will broadcast to everything that it's connected to. It is generally considered a Layer 1 (Physical Layer) device.
3. A switch is able to pass data to a specific device on a network if it already has a reference (forward), or it can broadcast to all devices (flood) if its MAC address is not in its table yet. It is generally considered a Layer 2 (Data Link Layer) device.
4. A router connects a local network to other networks. A single IP address is used for external communication, then the router can forward to the IP addresses used on the internal network. It is generally considered a Layer 3 (Network Layer) device.

TCP/IP Model (IP Suite)



1. Be able to recognize a diagram as the TCP/IP model, as opposed to the OSI model 2. Give a brief description of each layer (What is its major concern and an example) - Application: User-facing data, such as HTTP or FTP (file transfer) - Transport: Connectivity between clients and servers, such as TCP or UDP - Internet: Routing between separate networks, such as IP - Link: Low-level communication between local resources on a network, such as Ethernet 3. Each layer of the model encapsulates the previous. - For example, the application layer is the data that is being transferred, but the transport layer surrounds this data with transport protocols (such as TCP headers), which are surrounded by IP packets defined by protocols of the internet layer, which are sent along an ethernet cable with standards defined in the link layer.



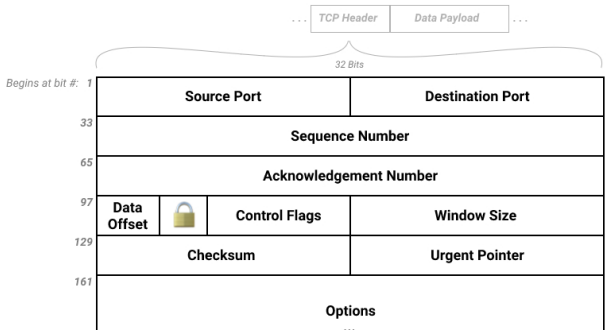
OSI Model

1. Be able to recognize a diagram as the OSI model, as opposed to the TCP/IP model 2. Give a brief description of each layer - Application (Layer 7) - Example: HTTP - Information used by client-side software - Presentation (Layer 6) - Example: JPEG, GIF - Data gets translated into a presentable format - Often called the syntax layer since it translates machine-readable syntax into human-readable syntax - Session (Layer 5) - Example: RPC (Remote Procedure Call) - Authentication and data continuity - Authorize actions, reestablish session from dropped connections - Transport (Layer 4) - Example: TCP, UDP - Mirrors TCP/IP's Transport Layer - Focused on data integrity and connectivity - Network (Layer 3) - Example: IP - Mirrors TCP/IP's Internet Layer - Manages connections between different remote networks - Data Link (Layer 2) - Example: Ethernet - Connections between one network interface to another - Primarily used by machines in a local network (ie targeting different MAC addresses) - Physical (Layer 1) - Example: DSL, 802.11 (Wi-Fi) - Translating from electrical signals to bits of data

TCP Segments

- Application data is broken down into transmittable units
- Each unit has a header attached to it with fields that we can compare in order to make sure that we are receiving all of the intended data.

Segment Header Fields



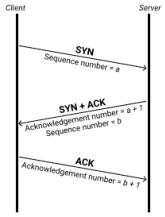
...

- Source Port: The port that the request originated from - Destination Port: The port that the request is intended for (determines which socket to use). Our most recognizable example may be when we think about localhost. If we set up a server on port 3000, the packets being sent need a destination port of 3000 for our server to receive them. - Sequence Number: Extablishes ordering of data. A starting number is created (Initial Sequence Number, or ISN), then each subsequent segment's number is incremented based on the size of the previous segment. - Acknowledgement Number: The number that corresponds to the sequence number we are responding to. The acknowledgement number is the sequence number that we received plus the length of the data that was transferred, plus 1. This allows us to determine if we've received all of the data. If there is a discrepency in numbers, we have to retransmit data. - Data Offset: The length of the segment header, allowing us to determine where our options field ends (or if an options field even exists) - Reserved: 3 bits that are always 0. Available for future flags. - Control Flags: 9 (currently) bits that are used for congestion monitoring and to indicate the connection lifecycle - Window Size: Indicates how much data should be sent, allowing for throttling if a receiver is getting overloaded. - Checksum: Error-checking mechanism for an individual segment - Urgent Pointer: Marks data as urgent, which will be processed right away. Commonly used to terminate a transfer instead of having to wait until it finishes. - Options: Similar to packets' options fields, extra headers that indicate other extra details about the segment.

Control Flags - The TCP Connection Lifecycle

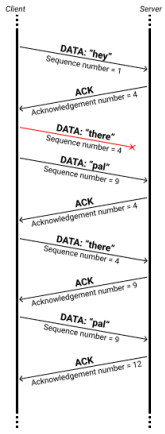
- 12 bits in our segment header
- First 3 are reserved, not in use yet
- Next 3 are used for network congestion monitoring/management
- The last 6 are used to indicate the connection lifecycle of our segments:
  - URG: Urgent. Urgent data is present that should be processsed immediately.
  - ACK: Acknowledgement. A message was received successfully.
  - PSH: Push. Buffered data should now be passed along.
  - RST: Reset. Reset the connection. Typically means we've received unexpected data.
  - SYN: Synchronize. Set on the first segment from each side of a connection, indicating that a connection has been established.
  - FIN: Finished. Sent from each side to indicate the connection is about to close.

Three-way Handshake - Starting a Connection



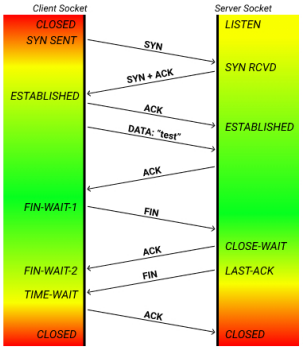
- Our first segment header has a synchronize flag, indicating we are opening a connection. - Our sequence number should be used by the response in their acknowledgement. - The receiver responds with a synchronize flag of their own as well as an acknowledgement flag. - The acknowledgement number sent back is the sequence number we are responding to, plus the length of data sent (0), plus 1. This shows the sender that we received their initial request. - We also send our own sequence number in order to check that the other side of this connection is responding appropriately. - We send back an acknowledgment based on the sequence number sent by the server. - After this acknowledgement, both sides now know that a connection is established.

Dropped Data



- If a segment doesn't reach the server, there will be a discrepency between the sequence number that it receives and what it expects based on the previous segment. - The server resends its most recent acknowledgement in order to indicate that we need to resend the segment that comes next in the series. - This acknowledgement is what makes TCP reliable. If we ever miss data, we know that it has occurred and we can request it to be sent again.

Four-way Handshake - Closing a Connection



- When we want to indicate that we are done with the transmission, we send a finished flag to the server. - The server sends an acknowledgement response, then waits a brief period to make sure there are no more segments are coming in. - The server then sends its own finished flag, indicating that it is closing the connection. - When we receive the finished flag from the server, we also wait a brief period to make sure we've received all segments, then send a final acknowledgment flag. Both sides of the connection are now closed.

Main Takeaways

1. TCP Segments allow for reliable data transfer and consistent connections
2. The segment headers come directly after the IP packet headers and right before the unit of data that is being transferred.
3. Control flags are used to indicate the different parts of the TCP connection lifecycle.
4. Acknowledgements are sent after each segment. The acknowledgment number is based off of the sequence number that was sent and the length of the data in the segment. Inconsistencies in these numbers indicate segments need to be resent (providing reliable data transfer). 147 (1 * 10^2) + (4 * 10^1) + (7 * 10^0)

11 0101 (12^5) + (12^{4})+(02^3) + (12$_{2}$)+(02^1) + (12^0) 32+16+4+1 = 53

// 0101 // (02^3) + (12^{2})+(02^1) + (12$_{0}$) // 32+16+4+1 = 53

0-9, A-F F2 (15*16^1) + (2*16^0) 240+2 = 242

19 to binary Smallest digit: 19 / 2 = 9 + remainder 1 => 1 Next: 9 / 2 = 4 + remainder 1 => 1 Next: 4 / 2 = 2 + remainder 0 => 0 Next: 2 / 2 = 1 + remainder 0 => 0 Next: 1 / 2 = 0 + remainder 1 => 1 Overall representation: 10011 Check: (1 * 2^4) + (12^1) + (12^0) = 16 + 2 + 1 = 19

19 to hexadecimal Smallest digit: 19 / 16 = 1 + remainder 3 => 3 Next: 1 / 16 = 0 + remainder 1 => 1 Overall representation: 13 Check: (1*16^1) + (3*16^0) = 16 + 3 = 19

In computer architecture: most significant bit | 11 0101 | least significant bit

IPv4: 0100 IPv6: 0110