Computer Networking: A Top-Down Approach

Summary by V.D

June 1, 2021

1 Computer Networks and the Internet

1.1 Introduction

Internet: A computer network that interconnects billions of computing devices throughout the world.

End systems: Devices connected to the Internet, also known as hosts. End systems are connected together by a network of communication links and packet switches.

Packets: A package of information that are sent through the network between hosts.

Packet Switch: Takes an packet arriving on one of its incoming communication links and forwards that packet on one of its outgoing communication links. The two most common ones known today are **routers** and **link-layer switches**. Both of these forward packets to their ultimate destinations. A **router** is used to send packets within a network, while a **link-layer switch** is used to send packets between networks.

Route: A sequence of communication links and packet switches traversed by a packet from the source to destination.

Internet Service Providers (ISPs): ISPs provide Internet access to end systems.

1.2 Protocols

Protocol: A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and or receipt of a message or other event.

Internet Protocol (IP): Specifies the specific format of packets sent and received among routers and end systems.

1.3 The Network Edge

Client: A type of host or end system that tend to be desktops or mobile PCs etc.

Server: A type of host or end system that tends to be a powerful machine capable of storing and distributing web pages. Can also be part of a data centre.

1.4 Packet Switching

Packet Switching: Used to help end-systems exchange messages or **packets** with each other. Data is split into smaller chunks of data known as packets, which are sent and travel through communication links and packet switches to arrive at a destination.

Store and forward transmission: The packet switch must receive the entire packet before it can begin to transmit the first bit of the packet onto the outbound link. Basically has to wait for the entire packet

to be received before it can pass it on to the next destination.

Packet Loss: If an arriving packet needs to be transmitted, but the router is still busy with the transmission of another packet, the arriving packet needs to wait in the output buffer. If there are too many packets in that buffer, then **packet loss** occurs and we have to drop a packet.

Forwarding Table: Each router or link contains a forwarding table where they can use a portion of the destination address to look up the next destination of a packet. Using a 'real-world' example we can ask directions from the US to Australia, then Sydney to Melbourne, then Tullamarine to Sunshine, then to a specific road, then house number. This has varying degrees of scope, which are all narrowed down as we pass through more routers. We don't require a router to be able to tell us how to get from the US to a specific address in Australia.

1.5 Circuit Switching

Circuit Switching: Where packet switching deals with packets as they arrive at a link, circuit switch networks are able to reserve resources along a path for a duration of a communication session between end systems. We don't have to wait in a queue to handle packets, however we do note that resources must be reserved for this to work. Therefore if we 10 connections open, we need to set aside 10 connection resources of the same size. If only one of those connections are active, then we will have 9 connections with allocated resources however they aren't using it.

1.6 Protocol Stack

Application Layer: Where network applications and their application layer protocol reside. This includes HTTP, SMTP and FTP which are involved with web documents and file transfers.

Transport Layer: The Internet's transport layer transports application-layer messages between application endpoints. There are two main protocols, TCP and UDP, both of which can transport application-layer messages.

Network Layer: This layer is responsible for moving network layer packets known as datagrams from one host to another. It provides the service of delivering a packet to the transport layer in the destination host. The main protocol used here is the IP protocol, which defines the fields in the datagram. It also contains routing protocols.

Link Layer: The link layer provides a service to the network layer by receiving a datagram and delivers the datagram to the next node along the route. At this node, the link layer passes the datagram back up to the network layer.

Physical Layer: The link layer moves link layer packets from a network to the adjacent one, but the job of a physical layer is to move individual bits from the link layer packet to the next node.

Encapsulation: At each layer, the data is encapsulated with a layer header. This is similar to wrapping a present and sending it via post, then unwrapping the package to get to the data. Eg. We buy a box of legos, the lego is the data, and the box has instructions relevant to how to use the legos, we wrap it in wrapping paper and attach a card which has information relevant to the recipient, we then bring it to the post office which puts it into a box to be sent, and has information relevant to the delivery and destination. Each layer contains a different set of information, relevant to different parties. As we move down a stack, we add more headers, and as we move up a stack, we can read and remove the header that is relevant.

2 Application Layer

Process Communication: For processes between hosts, they communicate by sending messages to each other over a network.

Client: The process that initiates the communication.

Server: The process that waits to be contacted to begin the session.

Socket: How a process sends and receives messages. A socket is the interface between the application layer and the transport layer within a host. It is also referred to as the Application Programming Interface (API) between the application and network.

2.1 Transport Services

Transport Layer Protocol: Has the responsibility of getting the message to the socket of the receiving process. We can choose which protocol our application wants to use, depending on the services that the protocol can offer.

Reliable Data Transfer: Provides a guarantee that data sent by one end of the application is delivered correctly and completely to the other end of the application. We can assume, that once data is passed through the socket, it will arrive at its destination without errors.

Throughput: The rate at which the sending process can deliver bits to the receiving process. A service that can be provided is guaranteed available throughput at a specified rate, which can be useful for applications such as voice calls which require a specific rate as not being able to hear half of the call would render the application useless.

Timing: Can guarantee a timing constraint on packet delivery, for example it will arrive no longer than 100msec after it is sent. This is particularly useful for applications where tight timing constraints are required on data to be effective, for example a video game can't have minute long delays.

Security: Can provide various security services, one being encrypting all data before sending, then decrypting before passing it back to the receiving host. It can also provide data integrity and end point authentication.

2.2 Internet Transport Services

2.2.1 TCP

TCP Services: Provides a connection oriented service, and a reliable data transport service.

Connection-oriented Services: TCP has the client and server exchange transport layer control information with each other before the application level messages are sent. This handshake procedures alerts the client and server and allows them to prepare for communication.

TCP Connection: Established after the handshake protocol in TCP. This is a duplex communication, in that both processes can send messages to each other over the connection at the same time.

TCP Reliable Data Transfer Service: The communicating processes can rely on TCP to deliver all data sent without error and in the correct order.

2.2.2 UDP

UDP Services: Provides minimal service, is connectionless and is also an unreliable data transfer. There is no guarantee that the message will reach the receiving process, or that they will arrive ein order.

2.3 Application Layer Protocols

Application Layer Protocol: Defines how an application's processes, running on different end systems can pass messages to each other. It defines:

- The types of messages exchanged, for example request messages and response messages
- The syntax of the various message types such as the fields in the messages and how the fields are delineated
- The semantics of the field, or the meaning of the information in the fields
- Rules for determining when and how a process sends messages and responds to messages

HTTP: HyperText Transfer Protocol, the Web"s application layer protocol. It allows communication between a client program and a server program. They can communicate with each other by exchanging HTTP messages, which defines the structure of those messages and how the client and server exchange those messages.

Web Page: A document that consists of objects. An object is a file, however if a web page has 5 images and 1 HTML file, then it contains 6 objects.

Web Browsers: Implement the client side of HTTP, such as the display of web pages.

Web Servers: Implement the server side of HTTP, and house web objects. Examples include Apache.

HTTP Process: A client, such as a laptop or mobile phone establishes a TCP connection with the server. The client sends HTTP request messages into the socket interface, and receives HTTP response messages from its socket interface. HTTP is a stateless protocol as it doesn't remember what has been requested in the past. It just responds to the requests it receives.

Non-persistent Connections: Each request uses a different TCP connection to send and receive messages. In terms of HTTP, every object requires a new TCP connection. This results in a longer RTT, as one RTT is used to establish a connection, then another RTT is used to request and receive an object. Therefore for every object we request, the response time is at least 2 RTTs.

Persistent Connections: We can use the same TCP connection to send and receive multiple requests. The server leaves the TCP connection open after sending a response, and subsequent requests and responses between the same client and server can use this connection. An entire web page can be sent over the same request, and we can even request an object without waiting for the previous one to return a respond. This is more efficient, as we can not only reduce the RTT, but we can load the web page before all the data is received.

Cookies: Stores state information about a user's history. A cookie is stored in the browser, then attached as a header to a HTTP request when sending it to the server. The server uses this header to find information stored about a user and return it back with the response.

Web Caching: A network entity that can satisfy HTTP requests on behalf of an origin Web server. A web cache has its own disk storage and keeps recently requested objects in this storage. A browser can be configured so that all the HTTP requests are first directed to this cache, and if it has the object it will respond with what is in the cache. If not it will open a connection to the server and query it. Other factors, like expired objects or modified objects may also result in a a new query being sent.

Simple Mail Transfer Protocol (SMTP): The principal application-layer protocol for Internet electronic mail. It uses the reliable data transfer service of TCP to transfer mail from the sender's email to the recipient's mail server.

SMTP Process:

1. Alice opens up her client to send mail, provide's Bob's email address, composes the message then tells her client to send the message.
2. Alice's user agent (client) then sends the message to her mail server where it is placed in a message queue.
3. The client side of SMTP, running on Alice's mail sever, sees the message queue. It attempts to open a TCP connection to an SMTP server, running on Bob's mail server. If unsuccessful, it will wait for some period of time, and attempt to establish a connection again.
4. If the SMTP server is available, then Alice and Bob can undergo the SMTP handshaking protocol, and after start sending Alice's message into the TCP connection.
5. At Bob's mail server, the server side of SMTP receives the message. Bob's mail server than places the message into Bob's mailbox.
6. Bob can now invoke his user agent to read the message whenever he wants.

Pull Protocol: Files are uploaded to a server, and people primary use a protocol to pull this information as their own convenience. The TCP connection is initiated by the machine who wants to receive the file.

Push Protocol: The sending mail server pushes the file to the receiving mail server. The TCP connection is initiated by the machine that wants to send the file.

2.3.1 Mail Access Protocols:

Purpose: We need mail access protocols to differentiate the server and client responsibilities of sending mails. However this raises an issue with the mail server, which if we combine with a local computer, this computer must always be on to receive messages. As this is pretty impractical, we have to store the mail on Bob's server, then use Bob's mail agent to pull emails from the server onto his own computer.

POP3: A simple mail access protocol. Has three phases, authorization, transaction and update. Authorization authenticates the user, Transaction is used to retrieve messages and Update occurs after the session is closed, and deletes all messages marked for deletion. An issue with POP3 is its `download` and `delete` mode, which can pull a message onto your own computer, however deletes it from the mail server after. This means that the only copy will be on your computer, so you have no backup and cannot access it from another location.

IMAP: A mail access protocol with more features than POP3 but also more complexity in its implementation. Unlike POP3, IMAP is able maintain user state information across sessions, and allows users to modify folder hierarchies and read/delete messages. It also has the ability to obtain parts of a message for example header (subject line and who it was from) which can conserve data in low-bandwidth areas.

2.4 DNS

Domain Name System: Commonly employed by other application-layer protocols, including HTTP and SMTP to translate user supplied hostnames to IP addresses. A client can query a DNS server with a hostname, which will return a response of the related IP. The client can then use this IP to connection to the HTTP server process located at port 80 of that IP address.

Hostname: Mnemonic and mainly for human understanding purposes. Don't actually provide a lot of information on the location of the host in the internet.

IP addresses: Contains more specific information about the network, host and port number of an address located in the network.

Host Aliasing: DNS provides the service of providing a host with one or more alias names. DNS can be used to ibtain the canonical hostname (original name) of an alias hostname, as well as the IP address of the host. Generally alias names are easier to remember or shorter.

Mail Server Aliasing: Providing the canonical hostname for aliased hostnames in an email server. Eg. dengv@unimelb.edu can be the alias for the canonical hostname victoria.deng.12345@unimelb.edu

Load Distribution: DNS can be used to perform load distribution among replicated servers, such as replicated Web servers. For these servers, it may have a set of IP addresses associated with a canonical hostname. The DNS address can respond with the list of these addresses, however rotates the ordering of these addresses to make sure that traffic is distributed among these servers.

Centralised Design: A simple design for DNS that has one central DNS server which contain all the mappings of every host name and IP address on the internet. This raises several issues:

- **A single point of failure:** If the DNS server crashes, then the entire internet is non functional
- **Traffic volume:** Directing every DNS request to a single DNS server would create a lot of traffic for a server. Likely to be long wait times for a response and multiple crashes or overflowing request buffers.
- **Distant Centralized Database:** There is no physical location on the planet that is close to every single user of the internet. Wherever you put this single server, a number of users on the opposite side of the globe will have long delays when trying to access the internet.
- **Maintenance:** For every new host on the internet, we would have to constantly be making updates to this server. Not to mention the mapping list of addresses would be huge.

Distributed Design: Rather than just having a single server, the mappings are distributed across multiple DNS servers. In a broad overview, there are three classes of DNS servers, root, top-level domains (TLD) and authoritative DNS servers. We can start at the root server, which will then direct us to the correct server down the hierarchy until we get out response.

Root DNS Servers: There are multiple DNS servers all around the world, which are managed by different organizations. Root name servers provide the IP addresses of TLD servers.

Top Level Domain (TLD) Servers: For each of the top level domains, for example `com`, `org`, `edu`, `net`, `gov`, `jp`, etc. there is a TLD server of server cluster. Each server has a organization maintaining it. TLD servers provide the IP addresses for authoritative DNS servers.

Authoritative DNS Servers: Every organization with publicly accessible hosts on the Internet must provide publicly accessible DNS records that maps the names of these hosts to IP addresses. An organization can either choose to create their DNS server, or pay a DNS server to hold these records.

Local DNS Server: Doesn't belong to the hierarchy of servers, but is used to connect an ISP to a local DNS server which is close to a host. When a query is made form a host, it is sent to this local DNS server, which then forwards it to the DNS hierarchy. **Recursive Queries:** Rather than a query going back and forth between the local DNS server and the hierarchies, we can get a response by traversing the hierarchy then return the result back up to the root server which sends it back to the local server.

Iterative Queries: Queries go from local server to root, then local server to TLD, then local to Authoritative then is sent back to client.

Caching: A DNS server can cache a DNS reply in its local memory. If another request for the same address comes, rather than forwarding it to the server, it can just return the reply (assuming its not expired or modified since retrieval).

Resource Records: A four tuple that contains the following fields: `(Name, Value, Type, TTL)`.

2.5 Streaming

HTTP Streaming: A client makes a single `GET` request for a video, the server then sends over the video file within a HTTP response message, which comes streaming down according to what the network and traffic will allow. The client receives this in a client application buffer, which then uses this for display.

3 Transport Layer

Transport Layer: Provides for logical communication between application processes running on different hosts. From an application's perspective it seems that hosts running the processes are directly connected. (Physical communication is using the links, logical just means they seem like they are directly connected). The *hosts* are connected not the processes.

TCP: Offers reliable data transfer, and congestion control. It provides congestion control by giving each connection an equal share of the link bandwidth.

UDP: Provides process to process delivery and error checking (very minimal). With UDP, traffic is unregulated and there's no guarantee that a packet will arrive.

Multiplexing: The job of gathering data chunks at the source host from different sockets, encapsulating that data with header information to create segments, then passing the segments onto the network layer.

Demultiplexing: The process of delivering the data in a transport layer segment to the correct socket.

Pipelining: Rather than sending a packet in a 'stop and wait' manner (eg. packet must be ACK'd before sending another), we can send multiple packets without needing to be ACK'd. Additional functionality for buffering packets and packet loss must be implemented.

Go Back N (GBN): The sender is allowed to have no more than N unacknowledged packets in the pipeline. Is there is a loss, then the sender will retransmit all packets from the lost packet (including ones it has already sent) and the receiver will discard all packets that aren't the lost packet. Maybe inefficient, but ensures that the buffer doesn't have to deal with out of order storage and that sending is simplified.

Selective Repeat (SR): GBN can be slow as it does retransmit all packets from the loss, which can slow down applications if loss is common. If an ACK for a packet is not received within a timeout, the packet will be resent. The buffer stores all the other packets and once all the packets are in the receiver buffer, it will forward them to the upper layer.

3.1 TCP

Connection-Oriented: Has to establish a connection with another party through a handshake before we can start sending data.

Point to Point: The connection for TCP is always point to point which means it is between a single sender and a single receiver.

Three Way Handshake: Three segments are needed to start a TCP connection; the client initiating sequence, the server initiating sequence + ACK of client sequence, the ACK of the server sequence.

TCP Segment Structure:

- **ACK:** Denotes that the Acknowledgement Number is valid.
- **SYN:** Used to initialize a connection. On the first packets sent, this bit is set to 1 (initializing seq number)
- **FIN:** When this bit is set to 1, it indicates that the sender wants to close the connection.
- **PSH:** Push bit, that indicates the receiver should send data to the upper layer immediately

DupACK: A duplicate ACK, that reacknowledges a segment for which the sender has already received an prior acknowledgment. It may send this for a variety of reasons, mainly because it received a packet out of order.

Fast Transmit: If the sender receives 3 DupACKs in a row, they will perform a fast transmit and re-sent the segment following the ACK'd sequence.

Flow Control Service: Provides congestion control for applications that use TCP. It is done by having the sender maintain a *receive window*. This window gives the sender an idea of how much space the buffer has on the receiving end, and allows it to adjust its sending accordingly.

Sliding Window: Keeps track of the last byte sent, and the last byte ACK'd which influences the size of the window. While the max size of the window is controlled by the receiver, the window size is dynamic.

Congestion Control: Congestion Window: `cwnd` is the amount of data that TCP can send over the network. It limits the amount of acknowledged data on the sender, and indirectly limits the sender's send rate.

Receive Window: `rwnd` is the amount of data the receiver can accept.

Self-Clocking: TCP uses acknowledgements to trigger its increase in congestion window size. If we are continuously receiving ACK responses, we can assume all is well and that we can slowly increase the window by say 1 packet.

Slow Start: At the beginning of a TCP connection, the value of `cwnd` is initialized to a small value of 1 MSS, and increases by 1MSS everytime a transmitted segment is first acknowledged. For every RTT, the sending rate is doubled. The send rate starts slow but grows exponentially over time.

Short Start Threshold: `ssthresh` is set after the first packet loss and drop down to slow start, we set it to half the value of `cwnd` at the time of loss. For future iterations, once the `cwnd` reaches this threshold, it will stop increasing exponentially and instead increase linearly.

Fast Recovery: When 3 DupACKs have been received, it can resend the missing packet and enter the fast recovery phase. In the fast recovery phase, instead of starting from short start (1 MSS) it can start from the `ssthresh` and increase linearly.

4 Network Layer

Forwarding: The process of a router moving a packet to the appropriate output link. This is a point to point process.

Routing: The process of determining the route or path taken by packets as they flow from a sender to a receiver. These can be calculated by using a routing algorithm. It is a network wide process and determines the end-to-end paths that packets take.

Forwarding Table: A router forwards a packet by examining the value of one or more fields in the arriving packet's header, then using these header values to index into its forwarding table. The value stored in the outgoing link interface at the router tells out where the packet will be forwarded.

4.1 IPv4

Subnet: An IP network, also known as the network connecting the host interfaces and router.

Subnet mask: The slash '/' after the dotted notation denotes which bits of the address define the subnet address. This can also be denoted in binary.

Prefix: The x most significant bits of an address of the form a.b.c.d/x constitute the network portion of the IP address. As organizations are typically assigned a block of contiguous addresses, this prefix is common within hosts of the organization.

4.2 NAT

Network Address Translation: Converts from a private ip address in a home or private network to an outgoing public ip address for use in the internet.

NAT Translation Table: When a packet from a source wants to make a request to the internet, it will forward it to a NAT box, which records its private IP, port number in the table. It then forwards the request as the networks public IP address and port number to the internet. If it receives a reply, it sends it back to the source, used by finding the correct index in the table.

Per-router Control: Where a routing algorithm runs in each and every router. Each router has a routing component that communicates with the routing components in other routers to compute values for its forwarding table.

Logically Centralized Control: A logically centralised controller computers and distributes the forwarding tables to be used by each and every router.

Routing Algorithms: Main goal is to determine good paths or routes from senders to receivers through a network of routers. Typically, 'good' means shortest path or least cost, however in practice other factors such as policy and deals may factor into the paths calculated.

Static Routing Algorithms: Routes change very slowly over time, generally as a result of human intervention like updating weight costs.

Dynamic Routing Algorithms: Changes the routing paths as the network traffic loads or topology change. These algorithms can update periodically or in response to changes, however they are more susceptible to problems such as routing loops and route oscillation.

Load-sensitive Algorithms: If a specific path or node is congested then the routing algorithm will choose to route around this link. Most current algorithms are load insensitive however.

Link-state Broadcast Algorithm: Where each node broadcasts link-state packets to all other nodes in the network. Each node will have an identical and complete view of the network, and can then run an algorithm which will compute the same set of least-cost paths for every node.

Dijkstra's Algorithm: An iterative algorithm, that visits a node, checks its neighbours for new paths and updates if a lower cost path is found, then closes the node before picking another node to visit.

Autonomous Systems (ASs): A group of routers under the same administrative control. They have the same policies and rules applied to the routers in this group.

Gateway Routers: A router on the edge of an AS that directly connects to one or more routers in other ASs.

Internal Routers: A router that only connects to hosts and routers within its own AS.

Hot Potato Routing: A router gets a packet out of its own AS as soon as possible, without caring about the costs of what happens after it is out. Basically computes shortest path to a gateway router.

ICMP: The Internet Control Message Protocol is used by hosts and routers to communicate network-layer information to each other. The most typical use for this is error reporting. It is considered part of IP, but is similar to TCP/UDP in that it uses IP to send messages as part of an IP payload.

ICMP Use: In traceroute, a UDP segment is sent with a TTL, then the host responds with a ICMP message back to the source. The ICMP message contains information about the name and IP, and can also give an unreachable message to the source so we know when to stop sending packets.

5 Link Layer

MAC Addresses: Also known as a physical address or a LAN address. A MAC address is attached to a device or interface and we assume that it will not change.

Address Resolution Protocol (ARP): Translates network layer addresses (eg. IP addresses) to MAC addresses. It takes any IP address on the same LAN as input, and resturns the corresponding MAC address.

ARP Table: Each host and router in a network has an ARP table in its memory which contains mappings of IP addresses to MAC addresses. They also have a TTL which informs us of when to remove an entry.