Intro

1. Define and describe the structure and function of each layer in the hourglass model of computer networks.

When we connect to the internet and send our friend a meme, an incredibly interconnected and complex process allows our data to reach its end destination. Most people, including myself prior to taking this course, do not understand or appreciate the nuisance and intricacies of how the layered architecture of the internet, full of technical decisions, constraints, and trade-offs that shape how data moves, dictates how we interact with the world. This interaction was the central topic this course aimed to cover.

Across my projects I saw the services provided by each layer of the hourglass model of the internet which tackled this core topic. Namely, I studied the **Subnetwork, Network, Transport, and Application Layer.** Each layer performs a service and adds its own headers, before passing it on to the layer below.

Subnetwork Layer

Prior to starting the class, my conception mainly centered around the hops that get my data to its end location, which is what the network layer is concerned with. But behind the network layer is actually the subnetwork layer that makes this network of networks we call the internet possible by providing point to point communication. The services provided by the subnetwork to the network layer above it include the following:

- **Framing** is one of the key services that the subnetwork layer uses to make distinct boundaries between streams of bits that allows the network layer to interpret data packets. Different technologies work differently, so **PPP** uses sentinel values and character stuffing. **HDLC** uses bit stuffing with flags.
- **Error detection and correction**: this ties in to the decisions that are made at each layer, as subnetwork can involve either detecting corrupted data so we can resend it, or simply correcting it. Typically, we go with detection, because it's cheaper, we don't have to send as many bits, and the hardware is also simpler. **Checksums** (one's complement) and **Cyclic Redundancy Checks (CRC)** are error detection methods used.
- **Reliable transmission**s are primarily concerned with the Automatic Repeat Request (ARQ) strategy subnetwork uses involving timeouts and acknowledgements. Different algorithms are used.
    - **Stop and wait**: the sender transmits one frame and then stops and waits for the corresponding ACK before sending the next frame, but this can be very slow.

    - **Sliding window**: allows the sender to have multiple frames outstanding defined by a send window size, where the received ACKs the highest sequence received. This ensures both flow control and ordered delivery.

- **Access Control** involves obtaining control of the shared medium to send or receive messages, since the medium can support limited resources. So the subnetwork needs to manage it so only one device transmits at a time to prevent collisions, and involves backing off if someone is sending data, waiting a tad, and then reserving the channel with WAP and if it is free, WAP sends CLS and you can send data. The two protocols, **Ethernet** and Wi-Fi, use **CSMA/CD** and **CSMA/CA** respectively.

Subnetwork drastically informs the interactions we have with the world. Use fiber optic ethernet over wifi for example,ensures information is faster, has more reliability and range.

Network Layer

The hopping of information around before reaching the interest is central to what I believed the internet fundamentally did, and that is what the network layer does. It expects everything the subnetwork layer provides, and then in turn, provides best effort delivery to get data closer to its destination. This involves the following.

Addressing

The **Internet Protocol (IP)** assigns every device a globally unique numerical identifier.
- Managed by IANA
- Historical class-based addressing (A, B, C): rigid and wasteful.
- Modern **CIDR** (Classless Inter-Domain Routing): flexible allocation.
- Static addresses for stable services (e.g., web servers); dynamic for everyday clients.

Routing

Involves selecting paths for packets to travel across the internet. This involves intra-domain (within Autonomous systems or ASes) and inter-domain (between ASes) routing.

a. **Intra-domain: Link State (e.g., OSPF):** Each router floods its link costs Every router computes shortest paths (Dijkstra). Results in fast convergence.
   Distance Vector (e.g., RIP): Routers exchange distance information with neighbors. Suffers from the count-to-infinity problem.

b. **Inter-domain routing: BGP**, a path-vector protocol dictates this. Policies (business, and economic) shape route decisions and support the “network of networks” architecture. My network routing project analyzed BGP routes (discussed later).

> **Note:** IP also provides Best-Effort Delivery with no reliability or delivery guarantee, as the layers above it manage that. Routers also break large packets into fragments; end hosts reassemble.

A prevalent problem with addressing is our limited range of IP addresses with IPV-4 available, which are being exhausted. The much larger IPV-6 is being transitioned in, but implementation is slow and we currently operate on a combined use of both.

The network layer also has several important performance measures:

- **Bandwidth** (capacity of the connection and in turn, throughput: the number of bits per second the sender can support).
- **Delay-bandwidth product**: how much data can be in transit at any point.
- **Throughput:** effective data rate.
- **Latency:** the time it takes data to travel from point to point and was the key metric used in my Network Topology project where I analyzed the round trip times (RTTs) through traceroutes and pings (**ICMP** packets) to see how long my information reached the following six locations:

    - Lahore University of Management Sciences, Lahore, Pakistan
    - London School of Economics, London, England
    - University of Queensland, Brisbane, Queensland, Australia
    - University of California, Santa Cruz, California, United States
    - University of São Paulo, São Paulo, Brazil
    - Microsoft Corporation, United States

    Starting at Carleton, my results revealed that while the US and Europe are well connected with short RTTs, countries in Asia such as Pakistan and even Australia are far less connected. This was evident from the aforementioned RTT, but also a lack of submarine cables connecting these regions and lack of variety of providers that handle transfer. This fundamentally impacts how people in those regions interact with the internet as it takes them considerably more time to do so.

    > **Note:** I was unable to get pings to my home country of Pakistan which I analyzed for the project due to an internet outage in the region caused by a submarine cable going down and there not being alternatives, so I directly witnessed how network infrastructure influences how we interact with the world.

Transport Layer

My reliable-transmission project gave me an excellent view of how the transport layer, specifically **TCP** ensures reliable transmission. By analyzing a full TCP connection to YouTube in Wireshark (after disabling QUIC so the browser fell back to TCP), I saw how TCP provides four major services to the application layer:

Connection Setup

TCP's three-way handshake (SYN → SYN-ACK → ACK) appeared in my trace. Each side exchanged initial sequence numbers and advertised window sizes, establishing a synchronized and reliable channel for the application. This handshake lets the application treat the connection as a stable, ordered stream rather than raw packets.

Reliable Transmission

TCP ensures that all bytes arrive, without gaps, and in order. My trace showed:

- A retransmission triggered when an earlier segment wasn't acknowledged,
- Selective acknowledgments (SACKs) marking out-of-order data the receiver had already obtained,
- Duplicate ACKs signaling missing segments, and
- The eventual arrival of the missing sequence range, after which the cumulative ACK advanced again.

Flow Control

TCP's sliding window protects receivers from being overwhelmed. Both client and server advertised receive windows initially. I observed the window advancing only when ACKs arrived, showing how TCP controls the sender's rate so the application never has to worry about buffer limits or overload (you can actually see buffering directly when a video stops to load, for example).

Congestion Control

In the stream carrying most of the YouTube data, I saw:

- **Slow Start**, where the congestion window doubled each RTT.
- **Multiplicative Decrease** after a likely timeout, and
- a gradual return to **Congestion Avoidance**.

This AIMD behavior lets TCP probe for available bandwidth while preventing congestion collapse.

Connection Termination

My trace also captured teardown behavior. The client sent a FIN when I closed the tab, but the server continued sending TLS data (a half-close). Because the browser was terminated abruptly, Chrome sent a **RST,** cutting off the connection before the usual FIN/ACK exchange completed. Even this complexity is abstracted away from applications.

This was how TCP provided a reliable, ordered byte stream connection establishment that allows congestion control and manages teardown. These services allow application protocols like **HTTPS** to acquire a dependable communication channel on top of IP's best-effort delivery.

> **Note:** My project also required temporarily disabling **QUIC**, which YouTube normally uses. QUIC implements its own reliability, congestion control, and connection setup over **UDP** (an alternate to TCP with less overhead). Mentioning it highlights how different transport protocols can provide similar services, but it just depends which is more fitting when.

Application Layer

At the top of the hourglass, the application layer is what we interact with the most through browsers such as Chrome and services such as texting. My DNS project, plus the work with implementing Gopher, HTTP, and TLS highlighted some features of this layer. It's important to note that what is required from the layers below depends on the application.

Naming Resolution and DNS

DNS is an application protocol used by different applications at this layer when a user enters the address of host, which DNS then resolves. So DNS is the closest most people get to working with the world through the hourglass model. `ipconfig` showed that my laptop uses Carleton's DNS servers (`137.22.1.7, 137.ol2.1.6`) which perform the lookup to find the IP address I was trying to reach (it was Youtube's in this case). The resolution proceeds as follows:

- **Local resolver**: My laptop sends a UDP query (low latency matters more than reliability), and Carleton's resolver checks its cache. If the answer is not cached, it performs an iterative lookup.
- **Root servers**: The resolver asks a root server where to find the nameservers for the .com top-level domain. The root doesn't give the final answer just a referral.
- **TLD (.com) servers:** The resolver then queries a .com nameserver, which returns another referral to the authoritative nameservers for google.com.
- **Authoritative servers (ns1 ns4.google.com in my case):** These respond with the final A/AAAA records for youtube.com.

Content and Interaction: Gopher and HTTP

The **Gopher** protocol is a menu-driven, client-server system for distributing and retrieving documents over the internet, popular in the early 1990s before HTTP. Each query returned one response, and it did not involve hyperlinking at all. My team got hands-on experience implementing Gopher for a project, and it highlighted its limitations. We implemented the protocol design decisions, and realized its fragility in doing so as small deviations from what we expected caused numerous problems.

HTTP followed Gopher and implemented hyperlinking which allows for rich forms of media and linking of web pages to provide users content. However, it also evolved and addressed problems throughout the years.

- **HTTP/1.0**: one TCP connection per object, lots of handshakes + slow start.
- **HTTP/1.1**: persistent connections + limited pipelining resulted in better reuse, but head-of-line blocking.
- **HTTP/2**: streams and frames over one TCP connection resulted in multiplexing.
- **HTTP/3 / QUIC**: runs over QUIC (UDP-based), with stream multiplexing in the transport and integrated encryption, so a lost packet blocks only one stream and fewer round trips are needed.

So while Gopher emphasized single resource implementation, HTTP represented linked pages that of course are fundamental to the internet. Gopher was designed to be simple and easy to implement, but shifting conditions and needs of the internet (and the creator's attempts to monetize it) allowed HTTP to prevail and further emphasize how the internet is not a static and perfect creation: it evolves according to changing needs.

Security: TLS and HTTPS

As the Web grew, new applications required different services from the layer below, such as security (e.g., when sending credit card details, which we never initially imagined doing). **TLS (Transport Layer Security)** was designed to solve this problem by sitting between the application and TCP:

When a client connects, TLS allows negotiation for what encryption algorithm to use to protect the session. When the session is established, TLS then provides secure data transfer after its been handed to TCP.

TLS also supports recession resumption to avoid unnecessary handshakes that are expensive with short-lived TCP connections.

Overall, both the Gopher and DNS projects showed me that how we interact with the world online is heavily shaped by application-layer choices:

- DNS and URLs are the most direct way in which we interact with the internet.
- TLS and QUIC determine how secure interactions are, and directly contribute to when we buy something online with our credit card, for example.

2. Understand the Internet as a “network of networks”, and the implications of the “network of networks” model.

As mentioned, I was aware of how the internet was made up of other networks, and how this was central to my understanding of how we used the internet to interact with the world. However, most importantly through my BGP routing project, I realized that these networks are operated by organizations, Carleton, Google, Microsoft, as parts of their Autonomous Systems (AS), each with its own policies, internal routing decisions, and business priorities. This in turn highlights who gets fast, resilient connectivity and who experiences fragile, high-latency paths.

Autonomous Systems and BGP

For my project, I analyzed how my data travels from Carleton to the six aforementioned destinations in the Topology project, and how this involves it travelling through a chain of Autonomous Systems (ASes), each identified by an ASN and run by an ISP, university, cloud provider, or regional carrier.

Across all six destinations, the average AS path length was about 3.7 hops, with values ranging from 2 (LUMS, UCSC) to 5 (University of Queensland, São Paulo). These short AS-level paths, even for intercontinental routes, highlight the role of large Tier-1 backbones like Hurricane Electric (AS6939), which appeared in every destination's advertised routes. In my analysis, HE acted as a global highway, handing traffic off to regional ISPs like PTCL (Pakistan), AARNet (Australia), and Telefónica (Brazil). The So Many Ways Home part of my writeup showed that each endpoint had 16 to 18 advertised paths, with different path lengths and intermediate ASes, but still strongly centered on a small set of Tier-1 backbones. This redundancy explains how your data usually still gets through, even if one route fails.

Best Effort Benefits and Drawbacks

Evaluating the Decentralized, Best-Effort Internet The Internet’s decentralized, best-effort design became much clearer through my BGP routing project, traceroute and ping measurements, and Wireshark observations, as they showed both why this architecture works at large scale, and why it introduces performance gaps and overreliance concerns.

1. **Why Decentralization Works**: The Internet functions because no single entity controls all of the routing. Each AS manages itself internally (through OSPF/RIP routing as discussed within the network layer). ASes share reachability via BGP, which I observed across the 16/18 advertised paths for each destination in my analysis. The network layer stays simple (best effort), while the transport layer handles reliability, congestion, and flow control (visible in my TCP retransmissions, SACKs, and AIMD behavior in Wireshark). This separation enables rapid innovation at the edges e.g., QUIC

implementing reliability over UDP with built-in encryption, reduced connection establishment time, and multiplexing without head-of-line blocking

2. **The Downsides:** 2. Uneven Performance and slow RTTS were because no one coordinates global routing, performance varies dramatically across regions and providers. Countries performing better economically have better internet architecture and speeds. My traceroute experiments showed short, stable RTTs within the U.S. and Europe. Much higher RTTs and fewer alternate paths to Pakistan, Australia, and Brazil.

   My BGP data also revealed policy-based routing: UCSC and Microsoft traffic sometimes went through Amsterdam because of CDNs and business agreements. Hurricane Electric appeared in almost every route, highlighting how a few Tier-1 ISPs dominate connectivity. These examples show that routing decisions are shaped not only by engineering but by business relationships and pricing.

   > **Note:** this domination of a few big companies controlling the internet also influences the prior problem mentioned about IPV-4 and IPV-6 as switching would require subsidizing the major players to switch, which would then cause smaller companies to switch. In fact, the course shed light on how much of our interactions are controlled by large companies and CDNs, and how issues like Net Neutrality can ultimately negatively influence how us, as the end users, interact with the world.

   Finally, security in a Best-Effort World A decentralized can introduces vulnerabilities, and my Wireshark capture showed why higher layers must compensate: TLS adds encryption, authentication, and integrity.

Internet measurements

The course allowed me to get hands-on experience with common tools for internet measurements.

- BGP route views allowed me to trace what ASes my data went through on the way to its destination for my Network Routing project.
- Ping and traceroutes allowed me to see the different hops taken with my data for my Network Topology project.
- Dig showed the hierarchical lookup process as DNS resolved domain names.
- Wireshark was used extensively throughout the term to study parts of each layer, but became crucial for my Reliable transport project to show the exact services provided by TCP in the transport layer.

Conclusion

By the end of course, I was surprised the internet worked at all. With different layers, each with their complex nuances, services, and different considerations for what is best when coordinating the transfer over the internet, I realized just how much I had taken this enormous network of networks for granted.

The course also highlighted how long switching to newer protocols can sometimes take, such as our current struggle to switch to IPV-6 and the gradual adoption of QUIC. The fact that we have numerous different protocols also means important decisions need to be made for what is most optimal when, such as applications in the application layer dictating what services they require from below.

Most surprisingly, I underestimated how much the internet is controlled by large corporations such as Google, and even though it is decentralized, business oriented routing decisions influence how our data travels. This results in inequality of access for regions with poor infrastructure, and also means large parts of the internet go down because of a single company facing issues (a Cloudflare outage happened while I was taking this class).

But ultimately, I can now appreciate just how complicated and intricate the hourglass model is, and how beautifully it comes together, despite the many, many problems each layer deals with.