How does the structure and implementation of computer networks impact how we interact with the world?

Objective: Structure and Function of the Hourglass Model

This section demonstrates an understanding of the layered architecture of the internet, focusing on the layers in the Hourlglass model of the internet.

Protocols, Connections, and Services

The predominant protocols at each layer were explored through implementation and analysis. At the **Application Layer**, the `gopher` project, with the implementation of `gopherClient.py` and `gopherServer.py`, demonstrated a simple request-response protocol running over TCP. This project showed how the application layer provides a service directly to the user (content retrieval) by using TCP (the service of the layer below). The historical and simple nature shows how the internet has always been a client-server service. A separate analysis in the `DNS` project, explored the DNS protocol, which uses UDP, a seperate transport protocol, for fast, connectionless queries that provide a critical name-to-address lookup service for almost all other applications.

At the **Transport Layer**, the `ReliableTransport` project provided a deep dive into TCP. By analyzing a packet capture (`filtered_trace.pcapng`) in Wireshark, the common elements of TCP were observed (e.g., the three-way handshake, slow start, and termination). The packet capture also showed the flexibility of TCP that handled the distribution of javascript, mapping, and radio elements. My analysis also focused on TCP's connection maintenance throguh sequence and window tracking.

At the **Internet Layer**, the `NetworkTopology` project demonstrated the connectionless nature of IP. The IP protocol showed a best-effort, hop-by-hop delivery service to the transport layer, that was made tangible by tracing the real-world paths to a diverse selection of hosts. The `Routing` project explored BGP, which, while an application-layer protocol itself, governs routing at the Internet Layer. It showed how BGP establishes long-lived TCP connections between peers not for data transport, but for the maintenance and exchange of routing tables.

Design Trade-offs and Alternate Protocols

An understanding of design trade-offs is demonstrated by comparing these protocols. The `gopher` project served as an analysis of a legacy protocol that would be replaced by HTTP, while both used the TCP protocol. Gopher's simple nature of file transfer makes it easy to understand, but its rigidity and lack of extensibility was shown in the project by it having a small, predefined set of single-character codes for content in its generated menu. Gopher's single function of getting the selected files shows it would be impossible to build applications that involve submitting data.

This highlights a key trade-off: **simplicity vs. functionality**. Similarly, the `ReliableTransport` project illuminated the trade-off between TCP's reliability and its overhead. The complexity of handshakes, acknowledgments, and flow control, while ensuring a reliable stream, imposes a performance cost not present in simpler protocols like UDP. At the Internet Layer, the `Routing` project, which produced analyses of best paths files) and all paths, showed the pros of BGP's policy-based routing (flexibility, support for a decentralized model) versus the cons (complexity, potential for misconfiguration, and non-optimal paths). The implementation of the QUIC protocol shows the non-linear adaptation of UDP compared to TCP, where speed is the priority of the internet.

Protocol Implementation

The objective of implementing a protocol that follows the rules of a specific layer is most directly evidenced by the `gopher` project. The creation of `gopherClient.py` and `gopherServer.py` required strict adherence to the protocol specification (RFC 1436). The server correctly interprets requests and maps them to the file system in the `content/` directory, while the client correctly parses the server's response. This project demonstrates a practical application of layer structure, as the implementation relied entirely on the service provided by the transport layer (TCP sockets) to build an application-layer service.

Objective: The Internet as a "Network of Networks"

This section demonstrates an understanding of the internet's decentralized structure and the tools used to analyze it.

Routing Protocols and Decentralization

The `Routing` project, serves to show that internet is an interconnection of independent Autonomous Systems (ASes) that aren't fixed and may change, where each internetwork is controlled by an entity with its own goals. This principle is explictly shown through the the AS-prepending in the route to baidu where artificial ASes are placed. The project focused on BGP, the protocol that enables this model. The various data files (`carleton_bestpath`, `stanford.edu_AP`, etc.) are the direct result of analyzing routing tables to understand how traffic navigates between these different networks, demonstrating the practical operation of BGP.

The pros and cons of this decentralized, "best-effort" model were evident throughout the projects. The `NetworkTopology` raw data shows a clear con: performance can be unpredictable, with latency varying across different network paths. The `Routing` project illustrates a major pro: the model's scalability and resilience, as there is no central point of failure. It also highlights a con: the system's security and stability rely on trust and correct configuration between thousands of independent operators. Both of these projects show how connection speed and internet resources go hand and hand. Understanding this is integral to understanding the main question of the course: How does the structure and implementation of computer networks impact how we interact with the world?

Interpretation of Internet Measurements

Competency in interpreting data from common internet measurement tools is demonstrated across multiple projects.

In the `NetworkTopology` project the speeds that were analysed were not simply a matter of running a command, but of interpreting the results to map network paths, identify routing asymmetries, and understand real-world latency and hop-by-hop progression across the internet.

The `ReliableTransport` project confirmed the operation of the three-way handshake, sequence numbers, acknowledgments, and windowing, showing wireshark's ability to decode data from bits that follows a interpretable stucture, when human analysis is involved.

Objective: Communication of Technical Concepts

This section demonstrates the ability to communicate technical findings to various audiences using appropriate formats.

The Visual and Written Mediums

The ability to choose appropriate visuals and summarize data is evidenced in the `Artifacts` sub-folders of the projects. My revisions of the Network Topology highlighted my hardships when working with a new medium of topology. My final version shows a focus on the correction of the information and presentation of that information. I was also required to brush up on my CSS, python, and excel skills to make sure that my data was communicated effectivly. Through the projects I needed to learn the appropriate usage of tables, graphs, written summaries to communicate my findings clearly. Each medium presented its challenges to how I would be clear enough to my audience such as the Typst typesetting framework that was able to produce clean results with hurdles such as incorpating my speaker notes to onto the slides.

Protocol Interpretation and Implementation

The interpretation of protocol specifications and their implementation in code is a form of technical communication. As mentioned previously, the `gopher` project is the primary evidence for this. The project required collaboration in a team setting to be able to write functional Python implementation that correctly interacted with the Gopher clients and servers of other teams.