Introduction to the Internet

What is the Internet?

The Internet is ubiquitous as a tool for transferring data between devices around the world. In this class, we'll be focusing on the infrastructure (both hardware and software) that supports this.

The Internet and the World Wide Web are not the same thing. You can think of the web as applications built on top of the Internet (e.g. Facebook, Twitter) that you can access through a web browser (e.g. Firefox, Chrome). Other applications besides the web can also use Internet infrastructure, too. Examples of non-web applications are Zoom or online games, or even Internet-of-things (IoT) devices like a sensor in your refrigerator or car.

Why is the Internet Interesting?

The Internet is not a new type of network technology (e.g. electrical wires already existed), but is instead about a completely new problem of tying together different, existing networks. Solving this problem required a new design paradigm that influenced other computer science fields.

Networking is a relatively new addition to the field of computer science. The Internet introduced many new challenges that are different from many traditional computer science fields. For example, unlike theory fields, we don't have a formal model of the Internet. Unlike hardware fields, we don't have a measurable benchmark for performance.

Unlike previous classes you might have taken, it's no longer enough to write code that simply works. The code you write has to scale to billions of users. The code you write also has to align with the business relationships of different operators (otherwise, they might not agree to run your code).

Code that works for a lightweight application (e.g. your home computer) might not work for a heavy-duty server. Code that works today might not work tomorrow, when

different computers join and leave the network.

The design of the Internet has influenced the way in which we architect modern systems (e.g. reasoning about goals, constraints, and trade-offs in the design). Network architecture is more about thinking about designs, and less about proving theorems or writing code. It's more about considering tradeoffs, and less about meeting specific benchmarks. It's more about designing systems that are practical, and less about finding the optimal design. The Internet is not optimal, but has successfully balanced a wide range of goals.

The Internet is Federated

The Internet is a federated system, and requires interoperability between operators. In other words, each operator (ISP) acts independently, but every operator has to cooperate in order to connect the entire world. In other words, all the ISPs in the world need to agree on some common protocol(s) in order to achieve global connectivity.

Federation introduces several challenges. Competing entities (e.g. rival companies) are forced to cooperate, even though competitors might not want to share confidential information with each other. When designing protocols, we have to consider real-life business incentives in addition to technical considerations.

Federation also complicates innovation. In other fields, companies can innovate by developing a new feature that no one else has. But on the Internet, if you have a feature that nobody else has, you can't use it. Everybody has to speak a common language (protocol), so any upgrades to the Internet have to be made with interoperability in mind.

The Internet is Scalable

Federation enables the tremendous scale of the Internet. Instead of a single operator managing billions of users and trillions of services, we only need to focus on interconnecting all the different operators. Federation also allows us to build the Internet out of a huge diversity of technologies (e.g. wireless, optical), with a huge range of capabilities (e.g. home links with tiny capacity, or undersea cables with huge capacity). These technologies are also constantly evolving, which means we can't aim for a fixed target (e.g. capacity and demand is constantly increasing by orders of magnitude).

The massive scale of the Internet also means that any system we design has to support

the massive range of users and applications on the Internet (e.g. some need more capacity than others, some may be malicious).

The worldwide scale of the Internet means that our systems and protocols need to operate asynchronously. Data can't move faster than the speed of light (and often moves much slower than that). Suppose you send a message to a server on the other side of the world. By the time your message arrives, your CPU might have executed millions of additional instructions, and the message you sent might already be outdated.

The scale of the Internet means that even sending a single message can require interacting with many components (e.g. software, switches, links). Any of the components could fail, and we might not even know if they fail. If something does fail, it could take a long time to hear the bad news. The Internet was the first system that had to be designed for failure at scale. Many of these ideas have since been adopted in other fields.

Protocols

In this class, much of our focus will be on **protocols** that specify how entities exchange in communication. What is the format of the messages they exchange, and how do they respond to those messages?

For example, imagine you're writing an application that needs to send and receive data over the Internet. The code at the sender machine and the code at the recipient machine need to both agree on how the data is formatted, and what they should do in response to different messages.

Here's an example of a protocol. Alice and Bob both say hello, then Alice requests a file, and Bob replies with the file. To define this protocol, we need to define syntax (e.g. how to write "give me this file" in 1s and 0s), and semantics (e.g. Alice must receive a hello from Bob before requesting a file).

Different protocols are designed for different needs. For example, if Alice needs to get the file as quickly as possible, we might design a protocol without the initial hello messages. Designing a good protocol can be harder than it seems! We might also need to account for edge cases, bugs, and malicious behavior. For example, what if Alice requests a file, and Bob replies with hello? How should Alice respond?

Throughout this class, we'll see many protocols that have been standardized across the Internet. You'll sometimes see the acronym RFC (Request For Comments) when we

mention a protocol. Many standards are published as RFC documents that are eventually widely accepted, though not all RFCs end up adopted. RFC documents are numbered, and sometimes protocols are referred to by their RFC number. For example, “RFC 1918 addresses” refers to addresses defined by that particular document.

There are different standards bodies responsible for standardizing protocols. The IEEE focuses on the lower-layer electrical engineering side. The IETF focuses on the Internet and is responsible for RFCs.