Chapter 4: Internet Applications

We will overview what internet applications are, what they do, and how they do it. To start, the internet and an application that uses it will form a pair of protocols. There are a few important questions to ask when designing these protocols:

- What kind of messages will be sent back and forth?
- Which side of the connection initiates the communication, the client or the server?
- What will happen if there's an error in communication?
- How will both applications know when the communication is complete?

There are two types of application-layer protocols. The first is a private communication, and the second is a standardized service.

- A private communication connects the user to the server over a network connection that is meant to be private. One device connects to one spot on the server, and no one is allowed to listen in.
- In a standardized service, many different operating systems will be requesting server access, and it must be able to handle all of them. For example, YouTube can be accessed via many types of browsers, as well as by mobile phones and tablets. It must have an API that can handle all requests from every different type of client.

When establishing a connection, a client-server model must first understand its data representation and transfer. This happens in the application-layer protocol.

- Data representation exchanges the syntax of the messages, the specific forms used during transfer, and the translation of data types, such as integers, characters, and even files as they are transferred between computers.
- Meanwhile, data transfer decides how files transfer between clients and servers. This includes the syntax between computers, the ways to handle errors, and how to decide when interactions are terminated.

When using the web, three very common protocols are typically used.

- HTML: the HyperText Markup Language is used to specify the contents and layout of a webpage. The HTML helps define how to display the webpage on the client, including how to change the webpage's display when the window size is changed, or how to move the background when the user scrolls. It also describes hat resources will be included in the page, such as videos, text, and hyperlinks.
- URL: a Uniform Resource Locator is a simple, convenient way to find information on the webpage. This allows a user to find resources online in a quick and simple manner, by typing an easy-to-remember name into their search bar.
- HTTP: The HyperText Transfer Protocol describes how a browser interacts with a web server to transfer information. The HTTP allows a web server to send HTML; therefore, HTTP is used to send HTML data to a client, and the client then uses the HTTP data to display the webpage to the user.

HTML is actually just a text file. The client uses it to display a webpage of arbitrary complexity, but at its core, HTML is just a set of text commands that tell the client what should be done.
It is interesting to note that HTML is a "declarative" language. This language does not tell the client HOW to display the webpage, only WHAT should be displayed. For example, the HTML could tell the client

"display a video above this text", but the client itself (typically, the browser) is the one that creates the user-visible webpage with the video above the text. The browser also decides what text is displayed and what resolution to display the page at – this allows the server to conform to the client's hardware.
HTML also allows an arbitrary object, such as text, a picture, or even just a box on the webpage, to contain a link to another webpage. This is called a hyperlink. If you've ever clicked blue text with an underline and been transferred to another webpage, or clicked a picture and had another window open up, that's a hyperlink.
When writing HTML, "tags" are used to specify where contents of a webpage start and end. Tags control all display aspects of a webpage. A tag is surrounded by less-than, greater-than symbols, with a frontslash creating an "end" tag. It looks like this:

```
<TAG>
        (The tag's content goes in here. Tags can be nested.)
        <NESTED_TAG>
                (This allows a sub-area of the webpage to have different content, like a URL.)
        </NESTED_TAG>
        (The frontslash ends the tag.)
</TAG>
```

A URL is a Uniform Resource Locator. It is a protocol used by the internet to find files and documents. A URL is always formatted as such:
protocol://computer_name:port/document_name%parameters

- The "protocol" is usually HTTP or HTTPS; both of these are used to transfer web information.
- The "computer_name" can be a domain name, such as www.example.com, or it can be an actual computer's name, such as its IP number.
- The "port" is the spot a client should connect to the server. If not specified, the client will connect to the server's default port. Typically, a client will connect to port 80, the default for a non-secure web server.
- The "document_name" is the specific part of the webpage that the user will connect to. Sometimes, the document_name will not be the document, but merely a directory that holds documents. These can be nested to an arbitrary depth, which results in URLs that look like this: http://www.example.com/folder/data/subfolder/files/specific_files/file_you_want.pdf
- Parameters are a specifically-used part of the webpage that will adjust how it is displayed or what it does. Parameters are like variables that can tell a webpage to do many different things; to cover everything they are capable of is beyond the scope of these notes, and possibly of this class.

HTTP is used to send and receive information. There are four types of HTTP requests.

- GET: requests a document. The server will send a response with status information, followed by a copy of a document.
- HEAD: requests status information. The server will respond by sending status information, but won't send a copy of the document.
- POST: sends data to a server. The server will append the data to a specific list of information, such as a list of messages send by the user.
- PUT: Sends data to a server; the server uses this information to completely replace the specified item, instead of adding it to a list.

Most commonly used of these requests is the “get” request. The client tells the server the highest level of protocol that it understands, and the server responds with the highest level of protocol it understands. The two then find and use the highest protocol that they both understand.
The server sends a header with some human-readable status information, including what server the information is coming from, the last time the information was modified, and the length and type of the information being sent. The server then sends a blank line, and then the requested document.

If you’ve ever heard the term “clear your cache”, you’re at least passingly familiar with what a cache is. A cache is a computer’s storage space, where it stores its most recent version of a webpage. If a webpage has not been updated since the last cache file was created, the computer does not re-download the webpage, and instead uses its old copy of the webpage. This lets the webpage load faster. Note that browsers do not cache small things, such as little pictures or text, since it’s easier to re-download these. Large pictures are cached, since they take longer to download.

When displaying information from a webpage, a user will most commonly use a browser. A browser is a program that understands HTTP, but will also provide support for other protocols, such as FTP.

FTP is a File Transfer Protocol. It determines how files on the internet are transported. Since a huge amount of information is transferred across the internet in the form of files, it’s important to have a standardized format for transfer. FTP has a few important characteristics:

- Arbitrary file contents. It’s not relevant what’s in the file; anything can be sent using FTP.
- Bidirectional transfer. Files can go either way, to the client OR the server.
- Support for authentication and ownership. The senders and receivers of a file should be able to certify that the files are coming from a trustworthy source.
- Ability to browse files. If there’s more than one file available for download, FTP allows users to browse the files they need.
- Textual control messages. A file download can be halted with a control message, the security connection can be verified, etc.
- Accommodates heterogeneity. FTP can transfer files between an arbitrary pair of computers, and is not limited by the hardware of the machines it is communicating between.

FTP forms two types of connection when a pair of computers are ready to transfer files.

- A control connection is established from the client to the server. The client tells the server it wants to see the files available for download, and then closes the connection. The client will browse the files available for download, and will then tell the server, via separate control connection, what file it wants to download.
- A data connection is used to send information. When the server receives a request to see the directories available to it, it sends a data connection with that information. The server then closes the data connection while the client examines the available directories. Once the client sends the next control connection, requesting a specific file, the server opens a data connection and sends that file.

Email is a way for computers to send messages to one another. The interface allows a user to compose and edit outgoing messages, as well as read incoming messages. Email uses three protocols to facilitate information flow: transfer, access, and representation.

- Transfer is used to move a copy of an email message from one computer to another.
- Access if a protocol that allows a user to access their mailbox and to send or view email messages.
- Representation is a protocol that specifies the format of the email message when it is stored on the disk of a computer.