

CMPS 3390 Homework 6

Spring 2026

Internet Overview

- [TECHNICAL DETAILS AND DEFINITIONS](#)
- Video: [What is the internet?](#)
- Video: [How the internet works in 5 Minutes.](#)
- Video: [Connections, latency, bandwidth](#)
- Video: [Ports Explained](#)

Web Protocols For App Dev

- **HTTP (HyperText Transfer Protocol)**
HTTP is the foundation of data exchange on the web. It's what makes browsing websites possible, and allows applications to request information from a server. It's stateless, meaning it doesn't remember anything between different requests.
- **HTTPS (Hypertext Transfer Protocol Secure)**
HTTPS is the secure version of HTTP. It encrypts the data sent between your browser and the website using SSL/TLS encryption, protecting it from eavesdroppers. You'll notice it when there's a padlock icon in the URL bar. It's crucial for online shopping, banking, and any other activities where sensitive data is shared.
- **WSS (WebSocket Secure)**
WSS is a protocol for real-time, bidirectional communication between a web browser and server. It's an extension of WebSockets, adding encryption. It's used in applications like online games, live chat, and financial trading platforms where updates need to happen instantly.
- **FTP (File Transfer Protocol)**
FTP is used to transfer files between computers over a network. It's useful when you want to upload files to a web server or download large files from one. It's older and not inherently secure, so it's often replaced by secure alternatives like SFTP (SSH File Transfer Protocol).
- **SFTP (SSH File Transfer Protocol)**
SFTP is a secure version of FTP, which uses SSH (Secure Shell) to encrypt file transfers. It's widely used for securely managing files on remote servers, especially for web development and system administration tasks.

Additional Protocol Info

- **SMTP (Simple Mail Transfer Protocol)**

SMTP is the protocol used to send emails. When you hit "send" on an email, your email client uses SMTP to push that email to the recipient's mail server. It's mainly for sending, while other protocols like IMAP and POP handle receiving emails.

- **IMAP (Internet Message Access Protocol)**

IMAP is used for retrieving emails from a server, allowing users to view and manage emails without downloading them. This means you can access your email from multiple devices and keep them in sync. It's especially helpful for users who need to check email on both laptops and phones.

- **POP3 (Post Office Protocol v3)**

POP3 is another email retrieval protocol, but unlike IMAP, it downloads emails to your device and often deletes them from the server afterward. This makes it less useful if you check emails from multiple devices, though it's still in use in some systems.

- **DNS (Domain Name System)**

DNS is like the phonebook of the internet. It translates domain names into IP addresses (the actual location of a server). Without DNS, you'd have to remember long strings of numbers to visit websites.

Types Of APIs (Application Public/Programming Interfaces)

- **Local/System API**

Local/System APIs are operating system libraries or third party services offered to expand the functionality of an application/program. For instance, Microsoft's .NET APIs, Steamworks, and OpenGL (Vulcan).

- **Web API**

Web API is an Interface that can be easily accessed using the HTTP Protocol, generally called an API over the web. It leverages a large number of client entities, like Smartphones, Tablets, or Laptops. A Web API can be developed using various technologies like Java, Node.js, Golang, PHP, etc. providing superior performance and faster service development. However, as Web APIs are designed for distributed services, they are lightweight and have limitations in bandwidth.

- **Program API**

Program APIs are based on Remote Procedure Call (RPC) technology that makes a remote program component appear local to the rest of the software. Examples include Windows Web Services API, the client/service interface generated by Apache Thrift, or gRPC (using proto buffers)

Web API Protocols & Architecture

- [XML-RPC](#)

The XML-RPC protocol was created by Dave Winer to exchange information between two or more networks. The client performs RPC by using XML to encode its calls and HTTP requests for data transfer.
- [JSON-RPC \(OpenRPC\)](#)

The JSON-RPC is a lightweight RPC encoded in JSON, similar to XML-RPC, which allows notifications and multiple calls to the server, which may be asynchronously answered.
- [SOAP \(Simple Object Access Protocol\)](#)

SOAP is an established Web API protocol for exchanging structured information. It uses XML to Authenticate, Authorize, and Communicate processes running on operating systems. Since web protocols like HTTP run on most operating systems, SOAP allows clients to invoke web services and receive responses irrespective of language and platform.
- [REST \(OpenAPI/Swagger\)](#)

REST is an architectural style to provide standards between systems on the web. REST is neither a protocol, nor library, nor a tool, so communication between systems becomes easy. REST architecture makes the implementation of Client and Server independent without affecting the operation of the other.
- [GraphQL](#)

GraphQL is a query language for APIs that lets clients request exactly the data they need, nothing more and nothing less. Unlike REST, which exposes multiple endpoints, GraphQL uses a single endpoint (similar to JRPC) where clients specify the shape of the response. This makes it efficient, flexible, and especially useful for applications with more specific, complex data needs.

Binary API/RPC code generators

- [gRPC](#)

RPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services across data centers with pluggable support for load balancing, tracing, health checking and authentication. Data is modeled using an IDL and [protocol buffers](#) to facilitate implementation in almost any language.
- [Apache Thrift](#)

The Apache Thrift software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi and other languages.

Additional Reading

- [REST, GraphQL, or RPC](#)

A detailed analysis on the implementation, differences, pros, and cons of various HTTP API options. It is a long article, but there's a ton of good information in there.