Web Architecture & Binary Data

Protocol

An agreed upon set of rules two sides use when communicating.

A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as actions taken on the transmission and/or receipt of a message or other event.

Five Layer Internet Protocol Stack

Encoding & Decoding

Character Sets

Each character in a set is assigned a number.

The primary concern of a character set is identification of a character via a numerical representation.

Character Encoding

Character encoding converts the number assigned to a character via the character set into a binary representation.

The binary number representing the character is not simply the binary version of the number that represents that character in the character set.

Concerned with how many bits are used to store the number in memory.

Strings

There is no such thing as "plain text". If you have a string, it has an encoding.

There are over a hundred different types of encodings, and everything above 127 has no guarantees.

ASCII

American **S**tandard **C**ode for **I**nformation **I**nterchange - a character encoding for the English alphabet developed in Bell labs in the 60s.

Could represent all these characters using numbers between 32 and 127 using only 7 bits.

128 through 255 weren't specified, and a lot of developers used them for custom characters.

Unicode

An attempt at a character set that includes every existing writing system as well as a few fictional ones.

A letter maps to a code point that looks like U+0639.

Unicode provides a unique number for every character, no matter which platform, program or language.

UTF-8

Unicode **T**ransformation **F**ormat - A system for storing the string of Unicode code points in memory using 8 bit bytes in order to maximize compatibility with ASCII.

Every code point from 0 to 127 is stored in a single byte. Above that, the byte usage increases to somewhere from 2 to 6 bytes.

TCP/IP

History

Originally proposed in 1974 by a paper, "A Protocol for Packet Network Intercommunication." published by the IEEE by authors Vint Cerf and Bob Kahn.

The central component of their proposed model was divided into two parts, the TCP and the IP.

Transmission Control Protocol

Manages the assembling of information into packets that are to be transmitted over the Internet as well as reassembling received packets from the internet into the original form.

The higher layer of the two protocols making up TCP/IP.

TCP is the reliable, in order method.

Internet Protocol

Handles addressing the packets created by the TCP layer.

The lower layer of the two protocols making up TCP/IP.

Client/Server Model

TCP/IP is considered stateless because each request a client makes of the server is considered a new connection.

Being stateless means that network paths can be used continuously by everyone.

TCP/IP and Web Dev

TCP/IP is responsible for the communication between the clients and servers we are concerned with as backend web developers.

We don't work directly in these layers, but it helps us be better at our jobs when we know what's going on.

HTTP

History

Originally developed by Tim Berners-Lee at CERN in 1989, along with HTML.

Hypertext was coined by Ted Nelson in 1965 while working on document publication and editing software.

Started out as a way to host and transfer files for academic research.

The first implementation had only one method, GET.

Hypertext Transfer Protocol

A network protocol used to deliver files and data on the World Wide Web.

Uses a client/server model, with clients such as browsers or applications requesting services from servers which provide resources such as documents and data.

Like other protocols, HTTP is stateless. The server maintains no information about past requests from clients.

HTTP and TCP

Client initiates a TCP connection to the server on port 80.

Server accepts the TCP connection from the client.

HTTP messages are exchanged between the client and the server.

The TCP connection is closed.

HTTP Connections

- Nonpersistent HTTP
  - At most, one object is sent over the TCP connection.
- Persistent HTTP
  - Multiple objects can be sent over the same TCP connection.

HTTP Messages

- Request
  - The client asks for something from the server.
- Response
  - The server processes the client's request, then returns the results.

HTTP Request

A typical HTTP request has a type along with some additional information about the request.

```
GET /_resources/images/current-students.jpg HTTP/1.1
Host: http://www.uark.edu
Connection: keep-alive
User-agent: Mozilla/5.0
Accept-language: en-US,en
```

HTTP Verbs

- **GET**
- HEAD
- **POST**
- **PUT**
- **DELETE**
- CONNECT
- OPTIONS
- TRACE
- PATCH

Frequently Used HTTP Verbs

- GET: a request that is used to exclusively retrieve data.
- POST: a request that is used to submit an entity to the specified resource.
- PUT: a request that replaces all current representations of the target resource with the request payload.
- DELETE: a request that deletes the specified resource.

HTTP Response

A typical HTTP response has a status code along with some additional information about the response.

```
HTTP/1.1 200 OK
Date: Wed, 18 Jan 2017 03:39:28 GMT
Server: Apache/2.2.15 (Red Hat)
Connection: close
Content-Length: 60709
Content-Type: image/jpeg
Last-Modified: Wed, 21 Aug 2013 13:16:22 GMT
```

HTTP Status Codes

- 100: Information Responses
- 200: Success Responses
- 300: Redirect Responses
- 400: Client Error Responses
- 500: Server Error Responses

Common HTTP Status Codes

- 200 OK
- 304 Not Modified
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error
- 502 Bad Gateway
- 503 Service Unavailable

MIME Type

Stands for **M**ultipurpose **I**nternet **M**ail **E**xtensions. Originally used for email, now used for HTTP.

A standard for specifying the type of data being sent over the internet. MIME types are used because file extensions are meaningless on the web.

```
type/subtype
```

MIME Type Examples

```
text/plain
text/css
text/html
image/gif
image/jpeg
image/png
image/svg+xml
audio/wav
audio/mpeg
video/mp4
application/octet-stream
application/xml
application/pdf
multipart/form-data
```

HTTPS

History

HTTPS was created in 1994 by Netscape for use in its Navigator web browser.

Originally it was designed to work with the SSL (**S**ecure **S**ockets **L**ayer) protocol, which later evolved into Transport Layer Security.

Use of HTTPS

HTTPS is a protocol for secure communication over a network by providing an encrypted connection.

Often used for authentication as well as protecting data integrity.

Looks mostly identical to HTTP, with the added benefit of signaling to the browser to use SSL/TLS to protect the traffic.

How HTTPS Works

The SSL/TLS encryption uses RSA and public-key cryptography to encrypt and decrypt outgoing and incoming requests and responses.

In order to achieve this, you need a functional digital certificate granted by a recognized certificate authority hosted somewhere on your server.

RSA

An asymmetric encryption algorithm utilizing the fact that multiplying two very large prime numbers is easy, but factoring the result of multiplication back into the two primes is very, very difficult.

The result of the two very large prime numbers multiplied together is the public key that is distributed and can be used to encrypt messages intended for the holder of the public key's paired private key.

The corresponding private key is generated by performing additional complex mathematical operations on the two primes that formed the public key.

Encrypt/Decrypt Process

Party A wants to send encrypted message M to Party B.

1. Party A requests Party B's public key.
2. Party B transmits their public key to Party A.
3. Party A converts message M into an integer MPRIME using an agreed upon reversible protocol padding scheme.
4. Party A computes the encrypted version of MPRIME into ciphertext using Party B's public key.
5. Party A transmits MPRIME ciphertext to Party B.
6. Party B uses their private key to decrypt the ciphertext back into MPRIME.
7. Party B reverses the padding scheme and turns MPRIME back into M.

Video Break

SOAP

SOAP Web Services

Simple **O**bject **A**ccess **P**rotocol - a web service specification for exchanging structured information.

SOAP was designed in 1998 for Microsoft, and made publicly available in 1999.

SOAP uses XML for its message format and relies on HTTP and SMTP for message transmission.

SOAP Message

Envelope: Defines the start and end of the message.

Header: An element containing metadata attributes.

Body: The XML data comprising the message being sent.

Fault: An element that contains information about any errors that occurred while processing.

SOAP Message Structure

```
POST /OrderEntry HTTP/1.1
Host: www.example.com
Content-Type: application/soap;  charset="utf-8"
Content-Length: 100
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelo

   <SOAP-ENV:Header>
      ...
      ...
   </SOAP-ENV:Header>

   <SOAP-ENV:Body>
      ...
      ...
      <SOAP-ENV:Fault>
         ...
         ...
      </SOAP-ENV:Fault>
      ...
   </SOAP-ENV:Body>

</SOAP_ENV:Envelope>
```

Benefits of SOAP

Provided a solid way in which to define services using **W**eb **S**ervices **D**escription **L**anguage.

Being XML based made it language agnostic and platform/transport independent.

Supports stateful operations.

Drawbacks of SOAP

Less scalable and worse performance than REST interfaces because XML payloads are comparatively large.

Standard programming languages now provide built in tooling to interacting with REST interfaces, while SOAP libraries are usually third-party.

SOAP is more restrictive than REST.

REST

REST Web Services

Representational **S**tate **T**ransfer - web services that allow for requesting systems to access text representations of resources using a predefined set of stateless operations.

Defined by Roy Thomas Fielding in 2000 in his PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures" at UC Irvine.

Has nothing to do with naps. :(

Benefits of REST

Performance

Scalability

Simplicity

Modifiability

REST and HTTP

REST and HTTP are highly coupled, and REST uses HTTP verbs to define the behaviors of a **C**reate **R**ead **U**pdate **D**elete (CRUD) interface.

REST Use Exercises

REST Implementation Exercise