Programming on the Web

© by Xia, Tingfeng

Sunday 12th January, 2020

This work is licensed under a Creative Commons "Attribution-NonCommercial-ShareAlike 4.0 International" license.

Contents

1 Communication and Protocols

1.1 The Network 4-Layer Model

1.1.1 Application Layer

- The application layer provides with standardized protocols to exchange data. For example, web browsers need a protocol to get and send data.
- Protocols include, but is not limited to, HTTP, FTP, SSH, SMTP, POP3...

1.1.2 Transport Layer

- Provides ***host-to-host*** communication service
  - It is "Connection Oriented"
  - Sends segments of data from the application layer (packets).
- Transport protocol for the common TCP/IP is TCP , and other transport protocols such as UDP exists.

1.1.3 Internet Layer

- Provides protocols for sending packets across a network or through multiple networks.
- The *Internet Protocol (IP)* handles this in TCP/IP. It routes data across networks using IP addresses.

1.1.4 Link Layer

- Protocols of the physical link between the nodes of the network, for example ethernet, WiFi, DSL, etc.
- Link Layer resides as the lowest layer in the 4 layer model, since TCP/IP can sit on top of any Link Layer.

1.2 TCP

(Definition) - Connection Oriented

- Needs to have a pre-arranged connection before sending data
- Should be ***bi-directional***
- Both client and server should acknowledge when they get data

Acknowledgements Are important part of TCO because

- Can check packet is from correct host, and
- Losing packets is a real problem

If no acknowledgement that packet was received then the packet is sent again.

TCP is Reliable but reacts to losing packets by slowing connection. Whereas UDP is not reliable but doesn't react to packet loss. There is usually a trade-off for different use case scenario in choosing TCP over UDP or otherwise.

1.2.1 3-way Handshake - Starting a Connection

The procedure of 3-way handshake is as follows

1. *Client* $\xrightarrow{\texttt{SYN}}$ *Server*; "Hi! Let's talk."; Signal `(SYN)chronize`
2. *Client* $\xleftarrow{\texttt{SYN-ACK}}$ *Server*; "Ok, let's talk."; Signal `(SYN)chronize - ACK(nowledgement)`
3. *Client* $\xleftarrow{\texttt{ACK}}$ *Server*; "Ok." Signal `ACK(nowledgement)`

The client and server can now send each other data, and must acknowledge to each other when they receive something.

1.3 IP Protocol and IP Packets

- IP is a "connection less" protocol: No pre-arranged connection required to send data, and
- IP ***just*** sends packets over networks. There is absolutely no assurance that they will be delivered, and no way to find out if they were delivered. There is also *no way* to let the destination know to expect a packet.
- It is easy to 'spoof' packets
  - Connection-less protocol means you can send around packets that pretend they came from a specific IP address.
  - Defence against this can come from higher network layers and network monitoring.

2 The World Wide Web and HTTP - Application Layer

Note The World Wide Web ***is not*** the internet, but it uses the internet to do its work.

> ***(Important!)*** The world Wide Web works over the Internet.

2.1 Defining the Web

In general, the web could be defined as follows:

A ***global collection of resources*** which are ***identifiable*** and ***linked*** together.

we will now discuss the three bolded italic keywords in detail.

2.1.1 Global Collection

- A ***web resource*** can be any data we can send through the internet, for example text, images, video, auto, etc.
- ***Global*** - want to access these resources no matter where they are in the world.
- Stored in "web" servers, that is computers with resources that are accessible.

2.1.2 Identifiable

We need a way to get these resources from their respective web servers. To be able to achieve this, it is necessary that we can

1. locate where they are in the entire web
2. use a consistent way to identify and access each one of these resources.

To achieve this, we need Uniform Resource Locators (URLs).

2.1.3 Linked Together

- It is a *web* to begin with and thus we have resources linked each other which allows us to easily discover the web.
- Generally, resources that are similar tend to link to each other.

2.2 HyperText Transfer Protocol (HTTP)

HTTP is the protocol of the web and gives the client and serve a mutual language at the application layer.

Global Collection of Resources *All machines* can use HTTP through applications - global reach.

Identified through URLs Let's take the following example

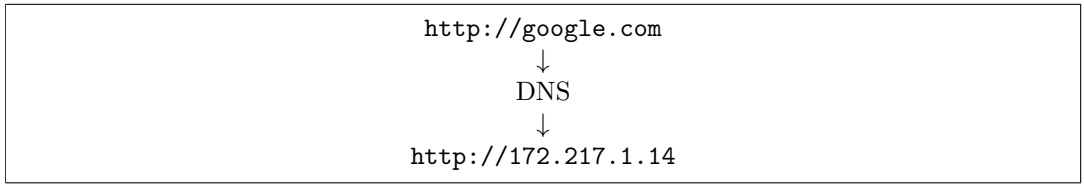
`http://google.com`

in the URL, the green part is called the ***protocol*** while the rest is the ***Hostname*** aka ***Domain name***. **#!Note:** URLs are not unique to HTTP; they are used in other protocols as well.

Linking together everything in HTTP We have resources (usually text in this case) with ***hyperlinks*** which gives us a feeling of connected web.

HTTP URLs

- Hostnames translated to IP addresses by the Domain Name System (DNS).
- IP address can change, name can stay the name.
- A general picture



Why HTTP? Accessing resources through URLs doesn't always mean downloading something. From day-to-day use of web, we generally use the web in the following 4 ways,

- Download things
- Upload things
- Change things
- Delete things

and they are handled in the HTTP!

2.2.1 Request - Response of HTTP

We have requests form clients and responses from the server. *Request and response originate from the Application Layer on both sides.* A HTTP request includes

- ***URL***: to get to the resource on the server we want.
- ***HTTP Method***: to tell the server what we want to do with that resource.
- ***Request Headers and Body***: Give the server additional information about out request.

HTTP Methods HTTP methods are ***verbs*** hat are used to label the actions we expect a server to take. Servers doesn't have 100% obligation to do these expected actions, but they are pretty well followed standards.

Verb	Expected Server Action
`GET`	Retrieve a resource
`POST`	Create a resource
`PATCH`	Update a resource
`DELETE`	Delete a resource

Response The response from a server has

- ***Response code***: Gives us standard indicator of the overall status of the response.
- ***Headers***: Give information about the response.
- ***Body***: The content of the resource, if available and/or applicable.

3 Ports

Why ports? A web server listens for a request, meaning that there needs to be a process on the server that is listening. There could be multiple processes that want to listen for connections.

3.1 Introducing Ports

- Every process on a computer that uses the internet is assigned a port (TCP or UDP port).
- Server process that listens for HTTP requests usually uses ***port 80***.

3.2 Example of TCP working with ports

Assume that the server is listening on usual HTTP port of 80, and client process taking through port 1753 (some randomly assigned port).

1. *Client* $\longrightarrow$ *Server*; "Give me `index.html`"
2. *Client* $\longleftarrow$ *Server*; "Ok, here is `index.html`"
3. *Client* $\longleftarrow$ *Server*; "Ok, I got it. "

3.3 HTTP: "Stateless' Protocol

- Each request independent, i.e.
  - Server doesn't need to keep track of previous requests.
  - Doesn't care how many are sent at once.
- This simplifies the protocol.
- ***#! Important:*** Illusions of state (e.g. knowing which pages the user browsed) can still occur, but is not part of the protocol.