The Application Layer: SMTP

CS 352, Lecture 5, Spring 2020
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
Course announcements

• Project 1 will be released on Friday
  • Find a partner if you don’t have one
• Quiz 1 completed yesterday
• Quiz 2 will go up on Friday
  • Due next Tuesday
Review of concepts

- **Application-layer protocols**: DNS, HTTP
- **HyperText Transfer Protocol**:  
  - Client-server model: requests and responses
- **Request methods**: GET, POST, HEAD, PUT, DELETE  
  - And response codes
- Persistent vs. non-persistent HTTP connection
- Remembering HTTP users via **cookies**
- **Common features of DNS, HTTP**:  
  - “Plain” text  
  - Command line tools to directly speak the protocol  
  - Caching
Caching in HTTP
Web caches

Web caches: Machines that remember web responses for a network

Why cache web responses?

• Reduce response time for client requests
• Reduce traffic on an institution’s access link

Caches can be implemented in the form of a proxy server
Web caching using a proxy server

- You can configure a HTTP proxy on your laptop’s network settings.
  - If you do, your browser sends all HTTP requests to the proxy (cache).

- Hit: cache returns object
- Miss:
  - Proxy requests object from original HTTP server (called origin server)
    - Proxy caches object locally
    - Proxy returns object to client

GET foo.html

HTTP Server

Clients

Proxy Server

The Internet

Store foo.html on receiving response
Web Caches: how does it look on HTTP?

- Conditional GET guarantees cache content is up-to-date while still saves traffic and response time whenever possible.

- Date in the cache’s request is the last time the server provided in its response header “last modified”
Content Distribution Networks (CDN)

A global network of web caches
- Provisioned by ISPs and network operators
- Or content providers, like Netflix, Google, …

Uses
- Improve **response time** to user for a service
- Reduce **bandwidth** requirements
  - Both on content provider and on a network (e.g., Rutgers)
- Reduce $$ to provision and maintain origin servers
Without CDN

- Huge bandwidth requirements
- Large propagation delays to reach users
- So, distribute content to geographically distributed cache servers
- Often, use DNS to redirect request to users to copies of content

<table>
<thead>
<tr>
<th>DOMAIN NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>98.138.253.109</td>
</tr>
<tr>
<td>cs.rutgers.edu</td>
<td>128.6.4.2</td>
</tr>
<tr>
<td><a href="http://www.yahoo.com">www.yahoo.com</a></td>
<td>74.125.225.243</td>
</tr>
<tr>
<td><a href="http://www.princeton.edu">www.princeton.edu</a></td>
<td>128.112.132.86</td>
</tr>
</tbody>
</table>

Cluster with Google’s origin servers
CDN terms

- **Origin server**
  - Server that holds the authoritative copy of the content
- **CDN server**
  - A replica server owned by the CDN provider
  - We called this proxy in our earlier example
- **CDN name server**
  - A DNS server used for redirection
- **Client**
With CDN

Scale a service through indirection to CDN name server.

CDN Name Server (124.8.9.8)

<table>
<thead>
<tr>
<th>DOMAIN NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>12.1.2.3</td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>12.1.2.4</td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>12.1.2.5</td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>12.1.2.6</td>
</tr>
</tbody>
</table>

Typically, custom logic to map one domain name to one of many IP addresses.

Policies may depend on location of requesting client, load at the different origin servers, apart from other things.
SMTP
Simple Mail Transfer Protocol
What we’re familiar with
Electronic Mail

Three major components:

1. **User agents**
   - a.k.a. “mail reader”
   - e.g., Applemail, Outlook
   - Web-based user agents (ex: gmail)
2. Mail Servers
   • Mailbox contains incoming messages for user
   • Message queue of outgoing (to be sent) mail messages
   • Sender mail server makes connection to Receiver mail server
     • IP address, port 25

3. SMTP protocol
   • Used to send messages
   • Client: sending user agent or sending mail server
   • server: receiving mail server
Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and “to”
   bob@someschool.edu

2) Alice’s UA sends message to her mail server; message placed in outgoing message queue

3) Client side of SMTP opens TCP connection with Bob’s mail server

4) SMTP client sends Alice’s message over the TCP connection

5) Bob’s mail server places the message in Bob’s incoming mailbox

6) Bob invokes his user agent to read message
Sample SMTP interaction

220 hill.com SMTP service ready
HELO town.com
    250 hill.com Hello town.com, pleased to meet you
MAIL FROM: <jack@town.com>
    250 <jack@town.com>… Sender ok
RCPT TO: <jill@hill.com>
    250 <jill@hill.com>… Recipient ok
DATA
    354 Enter mail, end with “.” on a line by itself
Jill, I’m not feeling up to hiking today. Will you please fetch me a pail of water?
.
    250 message accepted
QUIT
    221 hill.com closing connection
## MAIL command response codes

### Table 23.2 Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Positive Completion Reply</strong></td>
</tr>
<tr>
<td>211</td>
<td>System status or help reply</td>
</tr>
<tr>
<td>214</td>
<td>Help message</td>
</tr>
<tr>
<td>220</td>
<td>Service ready</td>
</tr>
<tr>
<td>221</td>
<td>Service closing transmission channel</td>
</tr>
<tr>
<td>250</td>
<td>Request command completed</td>
</tr>
<tr>
<td>251</td>
<td>User not local; the message will be forwarded</td>
</tr>
<tr>
<td></td>
<td><strong>Positive Intermediate Reply</strong></td>
</tr>
<tr>
<td>354</td>
<td>Start mail input</td>
</tr>
<tr>
<td></td>
<td><strong>Transient Negative Completion Reply</strong></td>
</tr>
<tr>
<td>421</td>
<td>Service not available</td>
</tr>
<tr>
<td>450</td>
<td>Mailbox not available</td>
</tr>
<tr>
<td>451</td>
<td>Command aborted: local error</td>
</tr>
<tr>
<td>452</td>
<td>Command aborted; insufficient storage</td>
</tr>
<tr>
<td></td>
<td><strong>Permanent Negative Completion Reply</strong></td>
</tr>
<tr>
<td>500</td>
<td>Syntax error; unrecognized command</td>
</tr>
<tr>
<td>501</td>
<td>Syntax error in parameters or arguments</td>
</tr>
<tr>
<td>502</td>
<td>Command not implemented</td>
</tr>
<tr>
<td>503</td>
<td>Bad sequence of commands</td>
</tr>
<tr>
<td>504</td>
<td>Command temporarily not implemented</td>
</tr>
<tr>
<td>550</td>
<td>Command is not executed; mailbox unavailable</td>
</tr>
<tr>
<td>551</td>
<td>User not local</td>
</tr>
<tr>
<td>552</td>
<td>Requested action aborted; exceeded storage location</td>
</tr>
<tr>
<td>553</td>
<td>Requested action not taken; mailbox name not allowed</td>
</tr>
<tr>
<td>554</td>
<td>Transaction failed</td>
</tr>
</tbody>
</table>
Mail message (stored on server) format

SMTP: protocol for exchanging email msgs
RFC 822: standard for text message format:

• header lines, e.g.,
  • To:
  • From:
  • Subject: different from SMTP commands!
  (these would still be under “DATA”)

• body
  • the “message”, ASCII characters only
Message format: multimedia extensions

• MIME: multimedia mail extension, RFC 2045, 2056
• additional lines in msg header declare MIME content type

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.

MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data ..............
                        ....base64 encoded data
Mail access protocols

- **SMTP**: delivery/storage to receiver’s server
- Mail access protocol: retrieval from server
  - **POP**: Post Office Protocol [RFC 1939]
    - Client connects to POP3 server on TCP port 110
  - **IMAP**: Internet Mail Access Protocol [RFC 1730]
    - Client connects to TCP port 143
  - **HTTP**: gmail, Yahoo! Mail, etc.

Why not use SMTP here?
Why do we need a sender side mail server?
POP vs IMAP

• POP3
  • Stateless server
  • UA-heavy processing
  • UA retrieves email from server, then typically deleted from server
  • Latest changes are at the UA
  • Simple protocol (list, retr, del within a POP session)

• IMAP4
  • Stateful server
  • UA and server processing
  • Server sees folders, etc. which are visible to UAs
  • Changes visible at the server
  • Complex protocol
What about web-based email?

• Connect to mail servers via web browser
  • Ex: gmail, outlook, etc.

• Browsers speak HTTP
• Email servers speak SMTP
• Need a bridge to retrieve email using HTTP
Web based email

HTTP server

HTTP

SMTP Client

Internet

HTTP server

SMTP server
Comparing SMTP with HTTP

- HTTP: pull
- SMTP: push

- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

- HTTP: can put non-ASCII data directly in response
- SMTP: need ASCII-based encoding
Try an SMTP interaction
nslookup
> set type=MX
> rutgers.edu
Server: 127.0.0.53
Address: 127.0.0.53#53

Non-authoritative answer:
rutgers.edu mail exchanger = 10 mx.rutgers.edu.

Authoritative answers can be found from:
>
nsrinivas@ubuntu18-vbox:~$ telnet mx.rutgers.edu 25
Trying 128.6.68.142...
Connected to mx.rutgers.edu.
Escape character is '^]'.
220 mx.rutgers.edu ESMTP
HELO cs.rutgers.edu
250 annwn11.rutgers.edu
MAIL FROM: <sn624@cs.rutgers.edu>
250 2.1.0 Ok
RCPT TO: <srlnivas.narayana@rutgers.edu>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Hello, world!
Goodbye, cruel world.
.
250 2.0.0 Ok: queued as 2B2E5460035
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
[flow:~]$ telnet mx.rutgers.edu 25
trying 128.6.68.142...
connected to mx.rutgers.edu.
escape character is '^[].
220 mx.rutgers.edu esmtp
250 cs.rutgers.edu
250 annwn12.rutgers.edu
mail from: <sn624@cs.rutgers.edu>
250 2.1.0 ok
rcpt to: <srinivas.narayana@rutgers.edu>
250 2.1.5 ok
Data
354 End data with <cr><lf>.<cr><lf>
from: sn624@cs.rutgers.edu
To: srinivas.narayana@rutgers.edu
Subject: A test message

Hello. Bleh bleh bleh.
.
250 2.0.0 ok: queued as 904aa634015
QUIT
221 2.0.0 bye
Connection closed by foreign host.
More themes from app-layer protocols

- **Separation of concerns.** Examples:
  - Content rendering for users (browser, UA) separate from protocol operations (mail server)
  - Reliable mail sending and receiving: mail UA doesn’t need to be “always on” to send or receive email reliably

- **In-band vs. out-of-band control:**
  - In-band: headers determine the actions of all the parties of the protocol
  - There are protocols with out-of-band control, e.g., FTP

- **Keep it simple until you really need complexity**
  - ASCII-based design; stateless servers. Then introduce:
    - Cookies for HTTP state
    - IMAP for email organization
    - Security extensions
    - Different methods to set up and use underlying connections, etc.