Review of concepts

Domain Name System
Human readable names → IP addresses

Hierarchical, distributed database
Root server
TLD server
Authoritative name server

Recursive queries
Iterative queries

<table>
<thead>
<tr>
<th>identification</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of questions</td>
<td>number of answer RRs</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>number of additional RRs</td>
</tr>
</tbody>
</table>

QR OPCODE

12 bytes
DNS Resource Records
DNS is a distributed database

- DNS stores resource records (RRs)

- (Incomplete) message format for each resource record (RR):
  - Class, type, name, value, TTL

- You can read all the gory details of the message format at https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml
DNS records

Type=A
- **name** is hostname
- **value** is IPv4 address

Type=AAAA
- **name** is hostname
- **value** is IPv6 address

- **Type=NS**
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
  - Sometimes, you’ll see SOA record

Type=CNAME
- **name** is alias name for some “canonical” (the real) name
  e.g., www.ibm.com is really www.ibm.com.cs186.net
- **value** is canonical name

Type=MX
- **value** is name of mailserver associated with **name**

More complete info at https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml
DNS record types

- `dig -t <type> <domain-name>`
**DNS record example**

**RRs in response to query**

<table>
<thead>
<tr>
<th>NAME</th>
<th>Design.cs.rutgers.edu</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>A</td>
</tr>
<tr>
<td>CLASS</td>
<td>IN</td>
</tr>
<tr>
<td>TTL</td>
<td>1 day (86400)</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>192.26.92.30</td>
</tr>
</tbody>
</table>

**records for authoritative servers**

**Information about nameserver**

<table>
<thead>
<tr>
<th>NAME</th>
<th>Cs.rutgers.edu</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>NS</td>
</tr>
<tr>
<td>CLASS</td>
<td>IN</td>
</tr>
<tr>
<td>TTL</td>
<td>1 day (86400)</td>
</tr>
<tr>
<td>NSDNAME</td>
<td>Ns-lcsr.rutgers.edu</td>
</tr>
</tbody>
</table>

**DNS serves as a general repository of information for the Internet**
Summary of DNS

• Hostname to IP address translation via a global network of servers
• Embodies several scaling principles
  • Partition through a hierarchy to silo query load
  • Replication to scale out at each level of hierarchy
  • Caching to reduce query load
• Once you have a reliable DB, can implement many useful things on top!
• Example 1: Scaling large web services, e.g., google search, by redirecting different clients to different servers (IP addresses)
  • Reliability, load balancing, performance optimization
• Example 2: Associating certificates, keys (security info) with domain names
  • https://www.rfc-editor.org/rfc/rfc8162.html
  • https://datatracker.ietf.org/doc/draft-ietf-dnsop-svcb-https/00/
The Web
The Web: Humble origins

Tim Berners-Lee: a way to manage and access documents at CERN research lab

Info containing links to other info, accessible remotely, through a standardized mechanism.

“Hypertext”
Web and HTTP: Terms

• HTTP stands for “HyperText Transfer Protocol”
• A web page consists of many objects
• Object can be HTML file, JPEG image, video stream chunk, audio file,…
• Web page consists of base HTML-file which includes several referenced objects.
• Each object is addressable by a uniform resource locator (URL)
  • sometimes also referred to as uniform resource identifier (URI)
• Example URL:
  
  www.cs.rutgers.edu/~sn624/index.html

  domain/host name                       path name
HTTP Protocol
I want to browse cs.rutgers.edu

<table>
<thead>
<tr>
<th>Hostname</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cs.Rutgers.edu</td>
<td>10.0.1.2</td>
</tr>
</tbody>
</table>

HTTP application typically associated with port 80
HTTP Request: Message Format

- Type of request
- Object/process requested
- Protocol version
- Parameters influencing the request
- Data needed to fulfill request

Entity Body

http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14
HTTP messages: request message

- ASCII (human-readable format)

request line
(GET, POST, HEAD commands)

Header lines

Carriage return, line feed indicates end of header

GET /352/syllabus.html HTTP/1.1
Host: www.cs.rutgers.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: en
(extra carriage return, line feed)
The URL

• Universal Resource Locator: a way to name objects on server
• But can also name an application process on the server!
• Examples:
  • Data storage from data entered in web forms
  • Login pages
  • Web carts
• Providing almost any service requires data handling by running code at the server
  • Not just rendering “static” resources
HTTP method types

• **GET**
  • Get the resource specified in the requested URL (could be a process)

• **POST**
  • Send entities (specified in the entity body) to a data-handling process at the requested URL

• **HEAD**
  • Asks server to leave requested object out of response, but send the rest of the response
  • Useful for debugging

• **PUT**
  • Update a resource at the requested URL with the new entity specified in the entity body

• **DELETE**
  • Deletes file specified in the URL

• and other methods
Uploading form input: GET and POST

POST method:
• Web page often includes form input
• Input is uploaded to server in entity body
• Posted content not visible in the URL
  • Free form content (ex: images) can be posted since entity body interpreted as data bytes

GET method:
• Entity body is empty
• Input is uploaded in URL field of request line
• URL must contain a restricted set of characters
• Example:
  • http://site.com/form?first=jane&last=austen
Difference between POST and PUT

• POST: the URL of the request identifies the resource that processes the entity body

• PUT: the URL of the request identifies the resource that is contained in the entity body

Difference between HEAD and GET

- GET: return the requested resource in the entity body of the response along with response headers (we’ll see these shortly)

- HEAD: return all the response headers in the GET response, but **without the resource** in the entity body

HTTP Response: General format

Unlike HTTP request, 
No method name

HTTP protocol version 
used by server

Was request successful? 
(or error condition)

Returned object data

Entity Body
HTTP message: response message

- status line (protocol, status code, status phrase)
  - HTTP/1.1 200 OK
  - Connection: close
  - Date: Thu, 06 Aug 1998 12:00:15 GMT
  - Server: Apache/1.3.0 (Unix)
  - Last-Modified: Mon, 22 Jun 1998 

- response header lines
  - Content-Length: 6821
  - Content-Type: text/html

- data, e.g., requested HTML file in entity body
  - data data data data data data data ...

23
HTTP response status codes

In first line in server->client response message.
A few sample codes:

200 OK
• request succeeded, requested object later in this message

301 Moved Permanently
• requested object moved, new location specified later in this message (Location:)

403 Forbidden
• Insufficient permissions to access the resource

404 Not Found
• requested document not found on this server

505 HTTP Version Not Supported
Observing HTTP behaviors

- `wget google.com` (or) `curl google.com`

- `telnet example.com 80`
  - `GET / HTTP/1.1`
  - `Host: example.com`

(followed by two enter’s)

- Exercise: try
  - `telnet google.com 80`
  - `telnet web.mit.edu 80`