DNS and HTTP

CS 168
Domain Name Service

• Host addresses: e.g., 169.229.131.109
  – a number used by protocols
  – conforms to network structure (the “where”)

• Host names: e.g., instr.eecs.berkeley.edu
  – mnemonic name usable by humans
  – conforms to organizational structure (the “who”)

• The Domain Name System (DNS) is how we map from one to the other
  – a directory service for hosts on the Internet
Hierarchical Namespace

- “Top Level Domains” are at the top
- Domains are subtrees
  - e.g.: .edu, berkeley.edu, eecs.berkeley.edu
- Name is leaf-to-root path
  - instr.eecs.berkeley.edu
- Name collisions trivially avoided!
  - each domain’s responsibility
Recursive DNS Query

Where is www.google.com?

- Ask local DNS server to get the response for you
- “Let me find out where it is for you”
Iterative DNS query

Where is www.google.com?

• Ask Server who to ask next
• “I don’t know this name, but this other server might”
DNS Records

• DNS info. stored as resource records (RRs)
  – RR is (name, value, type, TTL)

• Type = A: (-> Address)
  – name = hostname
  – value = IP address

• Type = NS: (-> Name Server)
  – name = domain
  – value = name of dns server for domain
DNS Records (contd.)

- Type = CNAME: (-> Canonical NAME)
  - name = hostname
  - value = canonical name

- Type = MX: (-> Mail eXchanger)
  - name = domain in email address
  - value = canonical name(s) of mail server(s)
Fun with dig!
Hyper Text Transfer Protocol (HTTP)

- **Client-server architecture**
  - server is “always on” and “well known”
  - clients initiate contact to server

- **Synchronous request/reply protocol**
  - Runs over TCP, Port 80

- **Stateless**

- **ASCII format**
Client/Server communication

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

HTTP Request
(Client to Server)

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 2006 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 2006 ...
Content-Length: 6821
Content-Type: text/html
```

HTTP Response
(Server to Client)

```
data data data data data ...
```
HTTP’s stateless-ness

• Pros?
  – Scalable
  – Easier to handle failures
  – Order of requests is immaterial

• Cons?
  – Can’t keep state! (shopping cart, user profiles…)

• Solution?
  – Client-side state
    – Cookies!
HTTP Performance: Non-persistent TCP Connection

1 RTT

1 RTT + transmission

TCP SYN

TCP SYN-ACK

TCP ACK + HTTP REQUEST

TCP ACK + HTTP RESPONSE

TCP ACK

TCP ACK

TCP FIN

TCP FIN-ACK

TCP ACK
Other options?

• Concurrent Requests and responses
  – Use multiple connections *in parallel*

• Persistent Connections
  – Maintain TCP connection across multiple requests

• Pipelined Requests and Responses
  – Batch requests and responses to reduce the number of packets
Easy ways to order!

1. Go to store
2. Order burger
3. Go to store
4. Order drink
5. Go to store
6. Order fries

1. Go to store
2. Order burger
3. Order drink
4. Order fries

1. Go to store with two friends
2. Each person orders one item (in parallel)

1. Go to store
2. Order burger, drink and fries

Cheeseburger, French Fries, and Medium Drink
<table>
<thead>
<tr>
<th>c</th>
<th>Page</th>
<th>Media 1</th>
<th>Media 2</th>
<th>Media 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential requests with non-persistent TCP connections</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>8 RTTs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 RTT (HTTP)</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>+ P/T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P/T</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>+ 3M/T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M/T</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>Page</td>
<td>Media 1</td>
<td>Media 2</td>
<td>Media 3</td>
<td>Total</td>
</tr>
<tr>
<td>------------------------------------------------------------------</td>
<td>--------------------------</td>
<td>--------------------------------------</td>
<td>--------------------------------------</td>
<td>--------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Concurrent with non-persistent TCP connections</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>4 RTTs</td>
</tr>
<tr>
<td></td>
<td>1 RTT (HTTP)</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (TCP)</td>
<td>+ P/T</td>
</tr>
<tr>
<td></td>
<td>P/T</td>
<td>M/(T/3)</td>
<td>M/(T/3)</td>
<td>M/(T/3)</td>
<td>+ 3M/T</td>
</tr>
<tr>
<td>c</td>
<td>Page</td>
<td>Media 1</td>
<td>Media 2</td>
<td>Media 3</td>
<td>Total</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>Sequential with a single persistent TCP connection</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (HTTP)</td>
<td>P/T</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>M/T</td>
<td>1 RTT (HTTP)</td>
<td>M/T</td>
<td>1 RTT (HTTP)</td>
</tr>
<tr>
<td>Pipelined within a single persistent TCP connection</td>
<td>1 RTT (TCP)</td>
<td>1 RTT (HTTP)</td>
<td>P/T</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>M/T</td>
<td>1 RTT (HTTP)</td>
<td>M/T</td>
<td></td>
</tr>
</tbody>
</table>