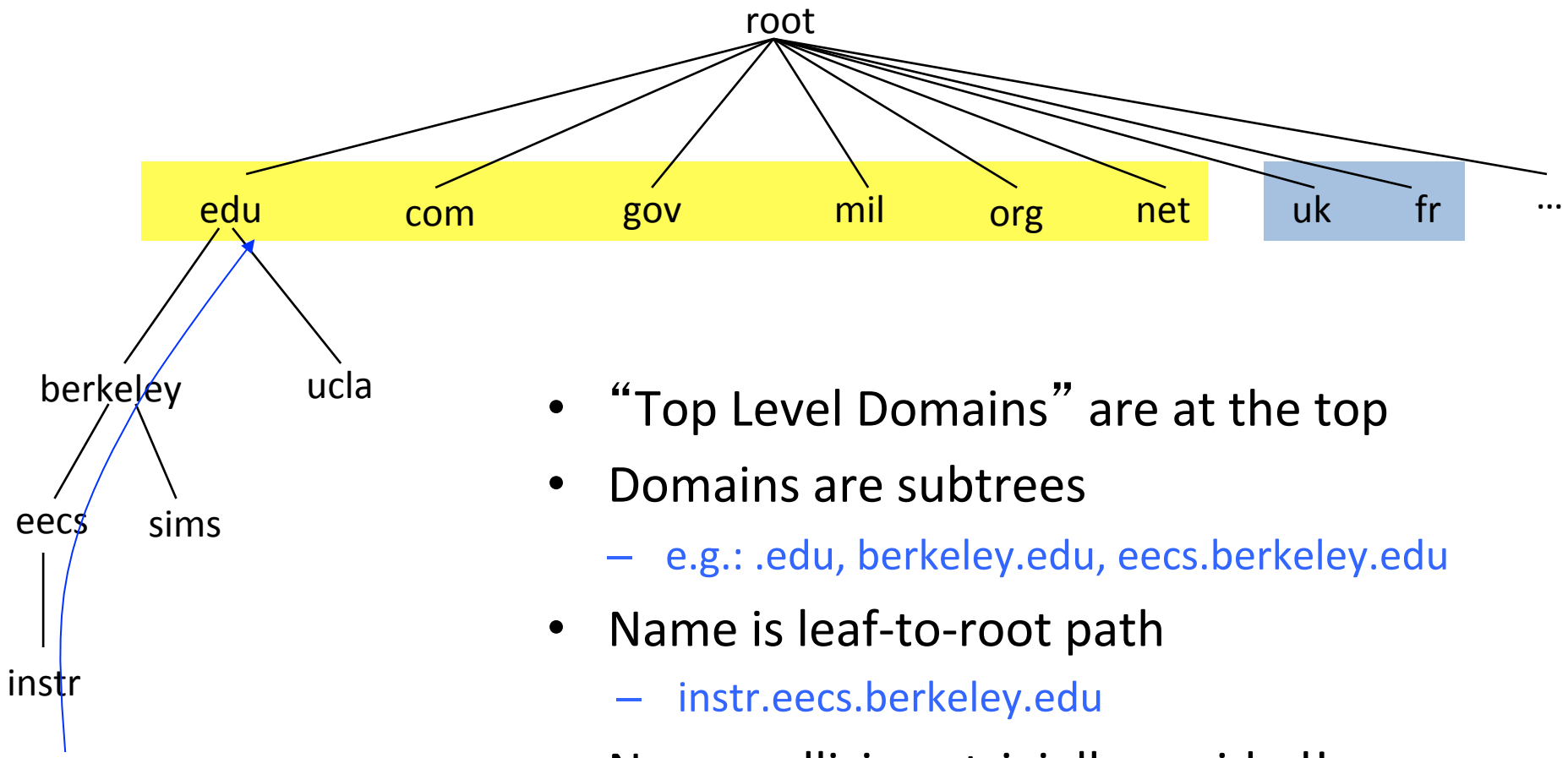
DNS and HTTP

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

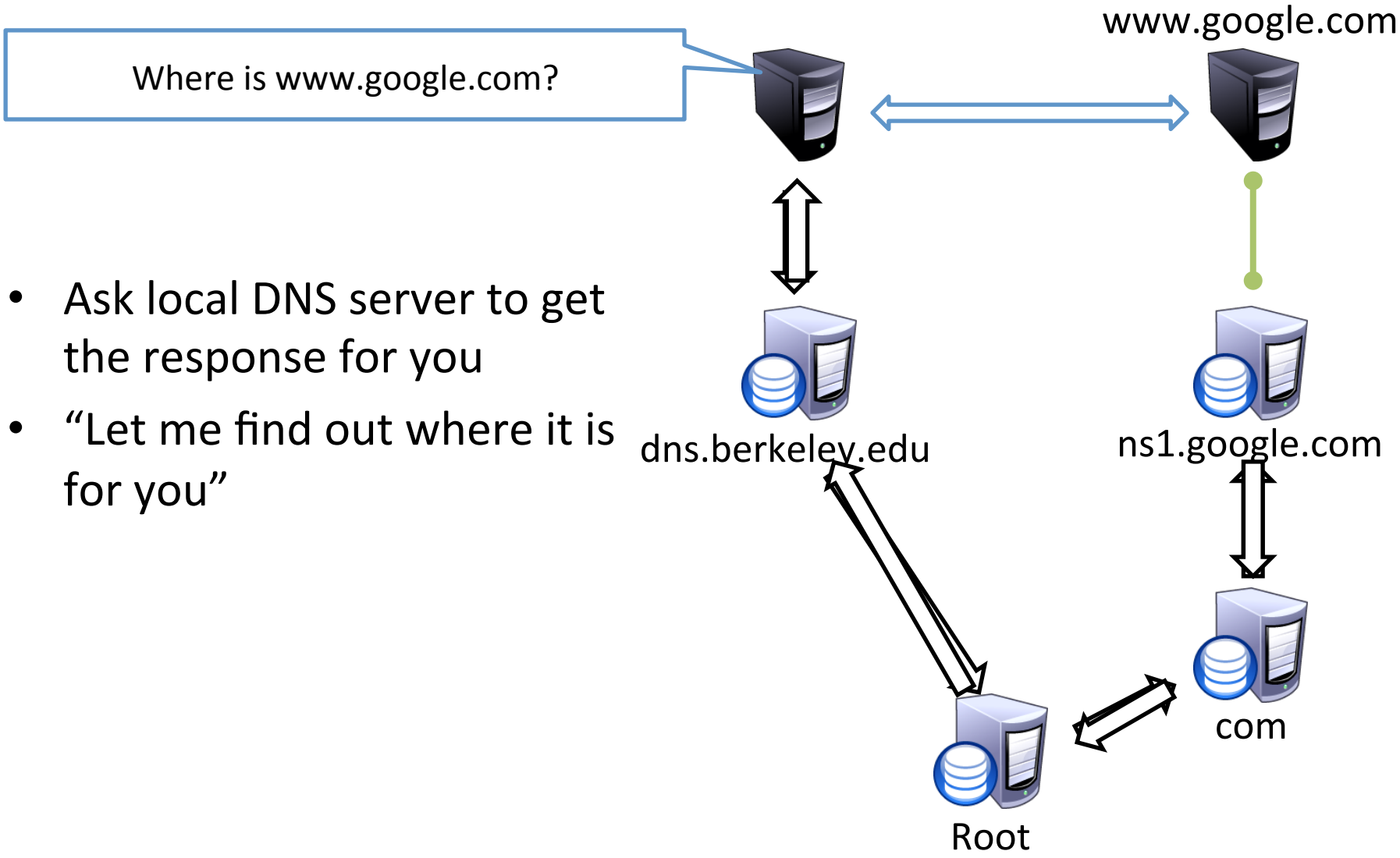
Hierarchical Namespace



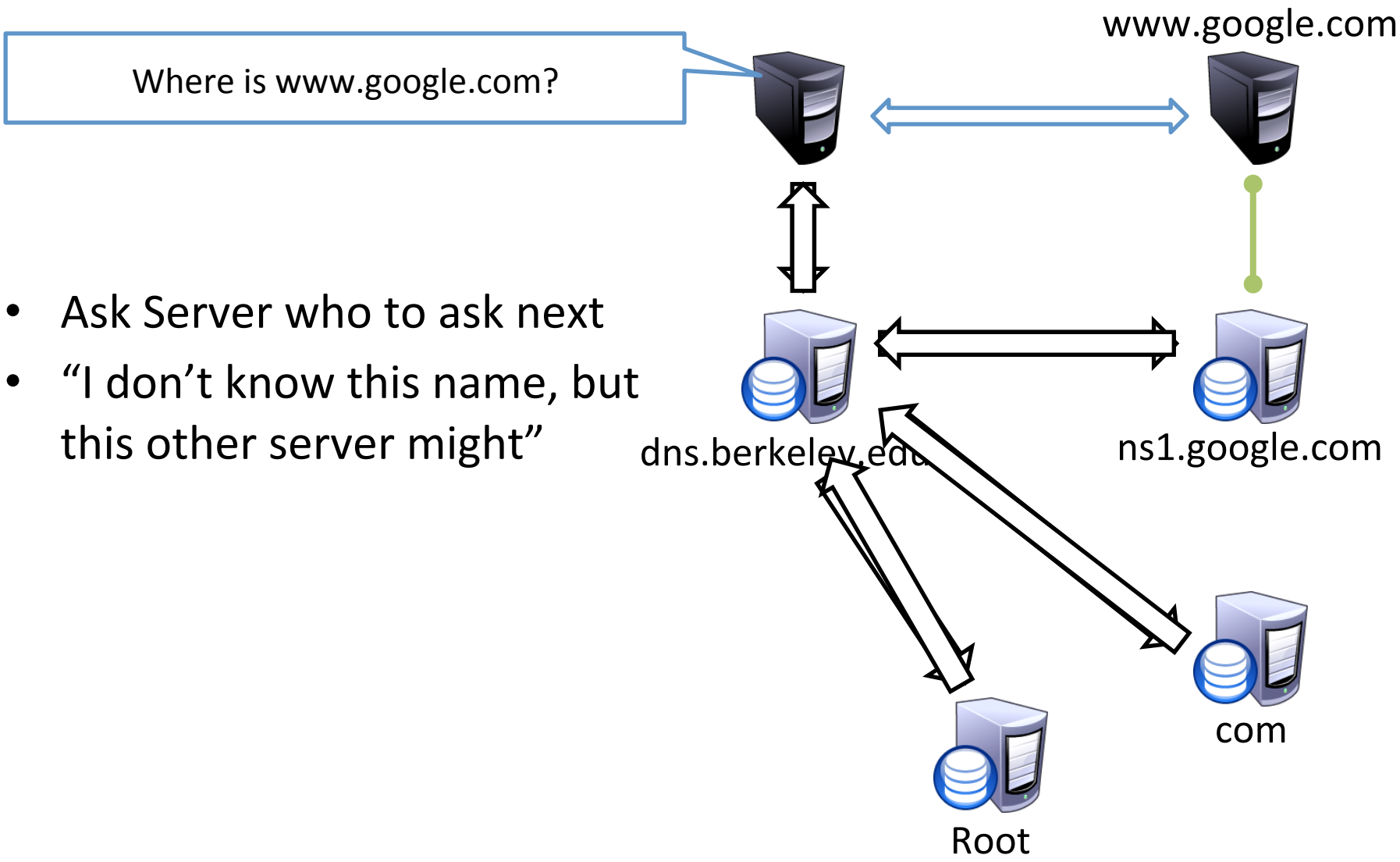
- "Top Level Domains" are at the top
- Domains are subtrees
  - e.g.: .edu, berkeley.edu, eecs.berkeley.edu
- Name is leaf-to-root path
  - instr.eecs.berkeley.edu
- Name collisions trivially avoided!
  - each domain's responsibility

Recursive DNS Query

- Ask local DNS server to get the response for you
- "Let me find out where it is for you"



Iterative DNS query



- 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.)

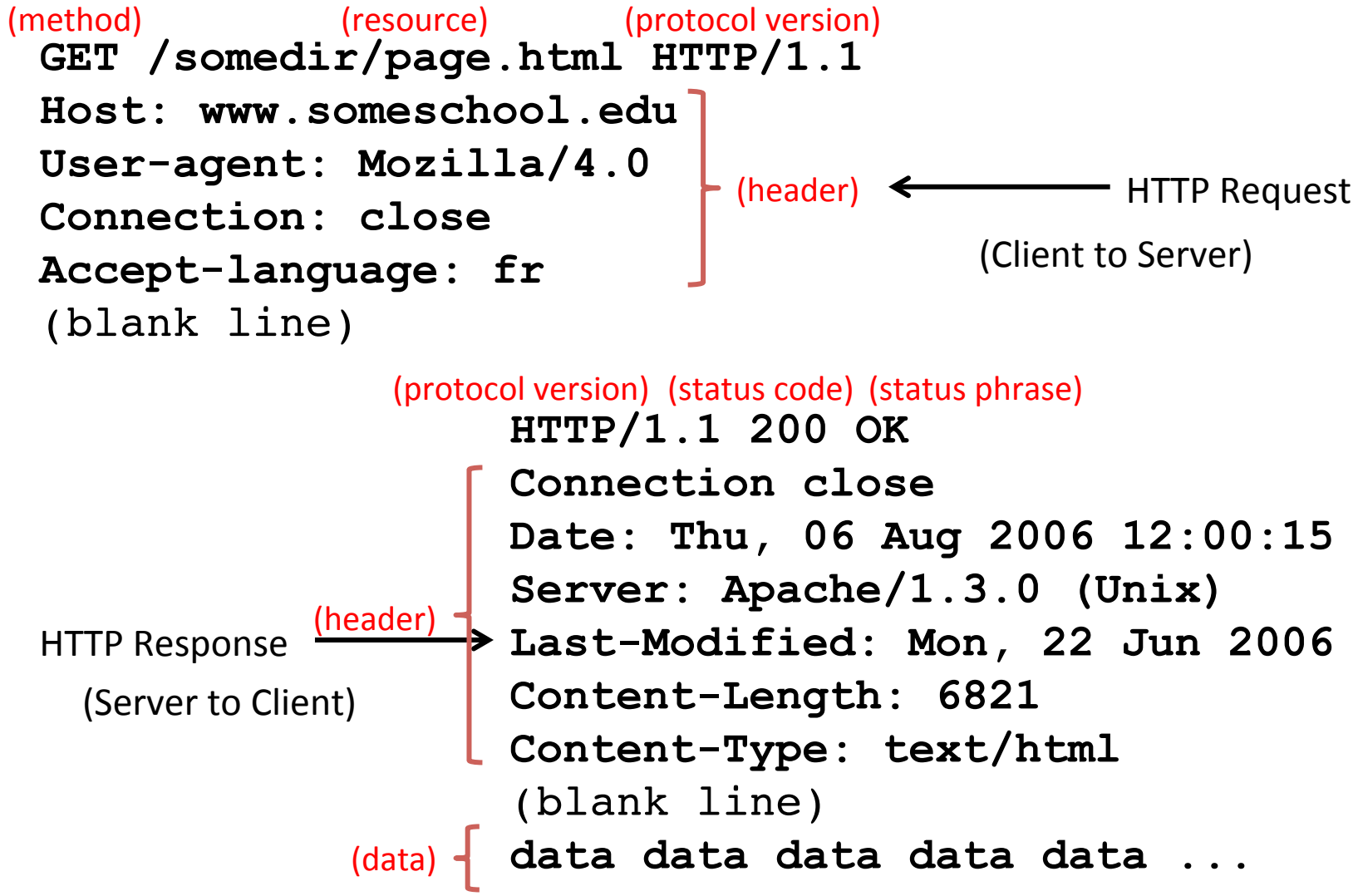
- Type = CNAME: (-> Canonical NAME)
  - name = hostname
  - value = canonical name
- Type = MX: (-> Mail eXchanger)
  - name = domain in email address
  - value = canonical name(s) of mail server(s)

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



HTTP's stateless-ness

- Pros?
  - Scalable
  - Easier to h
  - Order of r
- Cons?
  - Can't keep
- Solution?
  - Client-side

Cookies!

rofiles…)

HTTP Performance: Non-persistent TCP Connection

1 RTT

TCP SYN

TCP SYN-ACK

1 RTT
+ transmission

TCP ACK + HTTP REQUEST

TCP ACK + HTTP RESPONSE

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

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

2

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

3

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

4

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

2

Cheeseburger, French Fries, and Medium Drink

WWW.IN-N-OUT.COM

Q2

c	Page	Media 1	Media 2	Media 3	Total
Sequential requests with non-persistent TCP connections	1 RTT (TCP) 1 RTT (HTTP) P/T	1 RTT (TCP) 1 RTT (HTTP) M/T	1 RTT (TCP) 1 RTT (HTTP) M/T	1 RTT (TCP) 1 RTT (HTTP) M/T	8 RTTs + P/T + 3M/T

Q3

c	**Page**	**Media 1**	**Media 2**	**Media 3**	**Total**
Concurrent with non-persistent TCP connections	1 RTT (TCP) 1 RTT (HTTP) P/T	1 RTT (TCP) 1 RTT (HTTP) M/(T/3)	*1 RTT (TCP)* *1 RTT (HTTP)* *M/(T/3)*	*1 RTT (TCP)* *1 RTT (HTTP)* *M/(T/3)*	4 RTTs + P/T + 3M/T

<table>
<tr><th>c</th><th>Page</th><th>Media 1</th><th>Media 2</th><th>Media 3</th><th>Total</th></tr>
<tr><td rowspan="2">Sequential with a single persistent TCP connection</td><td colspan="4">1 RTT (TCP)</td><td rowspan="2">5 RTTs
+ P/T
+ 3M/T</td></tr>
<tr><td>1 RTT (HTTP)
P/T</td><td>1 RTT (HTTP)
M/T</td><td>1 RTT (HTTP)
M/T</td><td>1 RTT (HTTP)
M/T</td></tr>
<tr><td rowspan="4">Pipelined within a single persistent TCP connection</td><td colspan="4">1 RTT (TCP)</td><td rowspan="4">3 RTTs
+ P/T
+ 3M/T</td></tr>
<tr><td>1 RTT (HTTP)
P/T</td><td colspan="3"></td></tr>
<tr><td rowspan="2"></td><td colspan="3">1 RTT (HTTP)</td></tr>
<tr><td>M/T</td><td>M/T</td><td>M/T</td></tr>
</table>