DNS, HTTP

CS 168 – Fall 2024 – Discussion 10

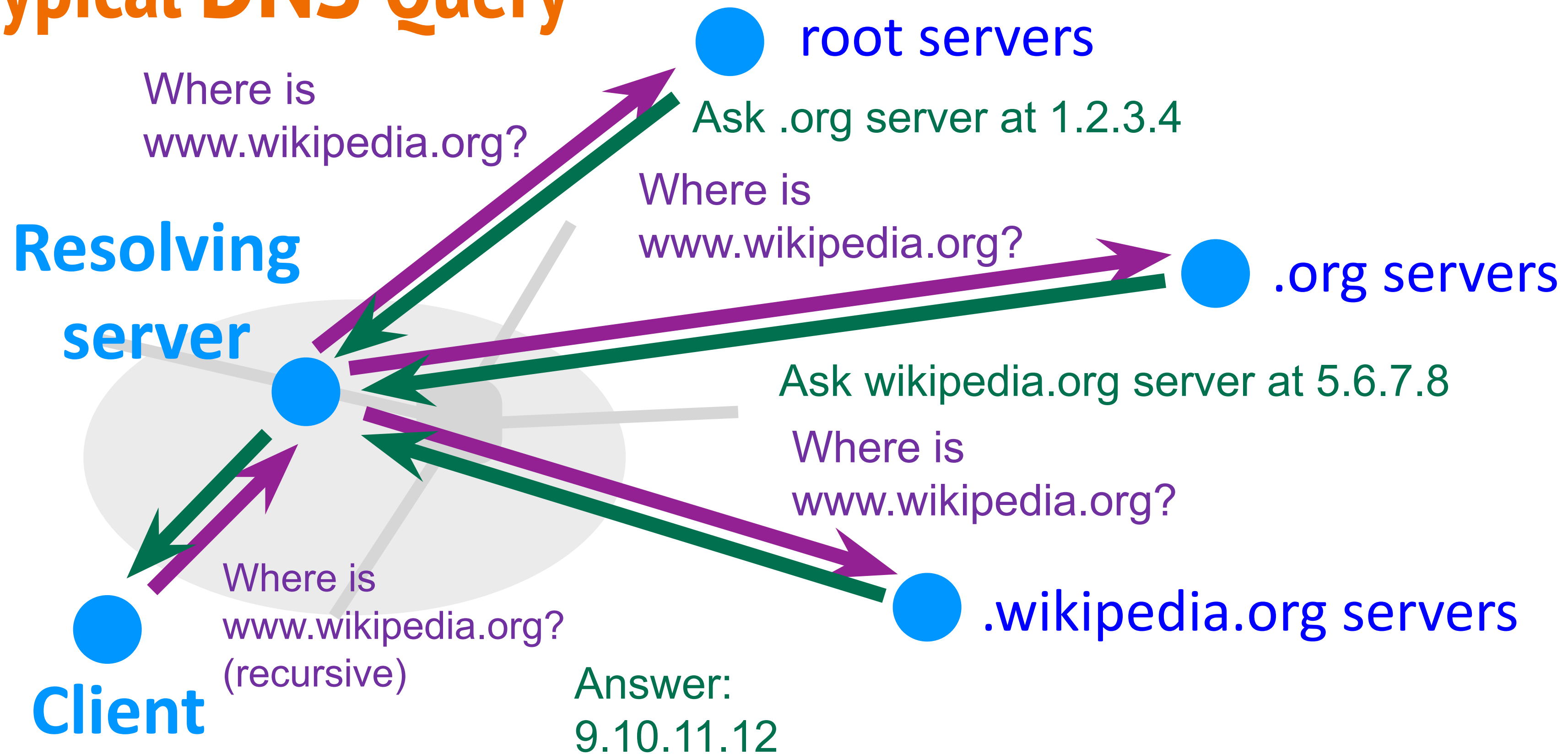
Agenda

- Logistics
- DNS Caching
- Anatomy of a URL
- HTTP
- CDN Caching

Logistics

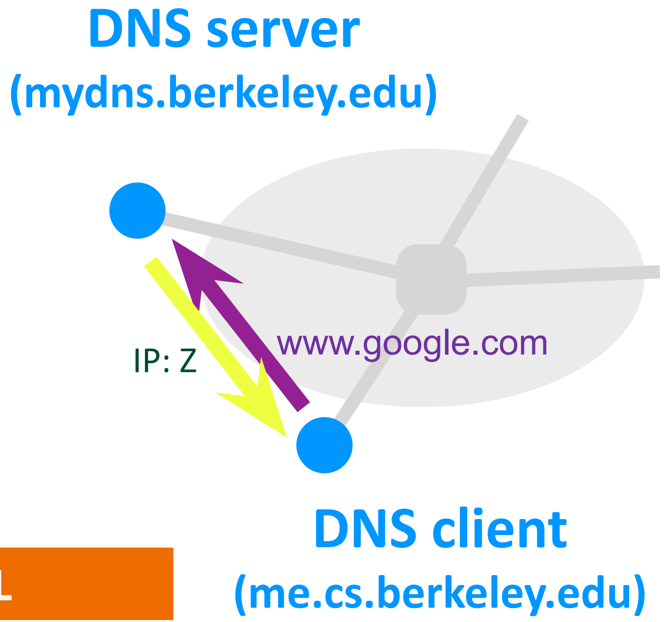
- Project 3A was due Friday, November 1st
- Project 3B due Tuesday, November 12th
- No discussion next Monday due to Veteran’s Day
- No office hours next Monday due to Veteran’s Day

Typical DNS Query



Caching DNS Responses

- DNS responses cached in DNS resolvers
  - Expire after TTL (time-to-live)
- Most popular sites visited often
  - Top sites frequently cached. Fast!



Hostname	IP	TTL
www.google.com	Z	60 min

Anatomy of a URL

`scheme://host[:port]/path/resource`

Scheme	Usually a protocol like (http, ftp, https, smtp, rtsp, *etc*.)
host	DNS hostname or an IP address
port	Defaults to protocol's standard port *e.g.* http: 80 https: 443
path	Traditionally reflects the file system
resource	Identifies the desired resource Can also extend to program executions: `http://us.f413.mail.yahoo.com/ym/ShowLetter?box=%40B%40Bulk&MsgId=2604_1744106_29699_1123_1261_0_28917_3552_1289957100&Search=&Nhead=f&YY=31454&order=down&sort=date&pos=0&view=a&head=b`

HTTP

- Client-server architecture
  - Server is “always on” and “well known”
  - Clients initiate contact to server

- Synchronous request/reply protocol
  - “Synchronous” means same HTTP session used for request and reply
  - Runs over TCP, Port 80

Requests and Responses

Request

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

Response

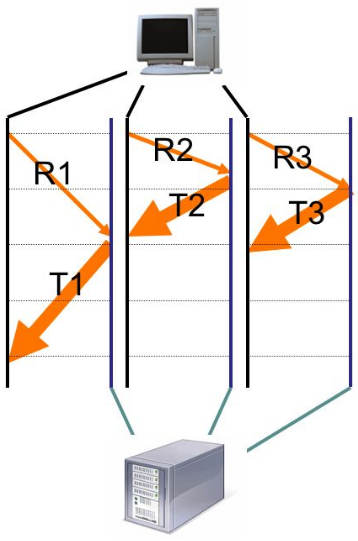
```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 2006 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 2006 ...
Content-Length: 6821
Content-Type: text/html
(blank line)
data data data data data ...
```

Performance!

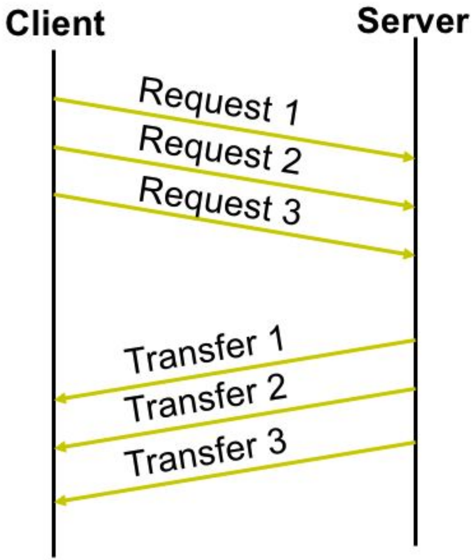
- A wise person once said "Architect for flexibility, engineer for performance"
- We have an architected solution – let's explore ways to make it go fast

Request Patterns

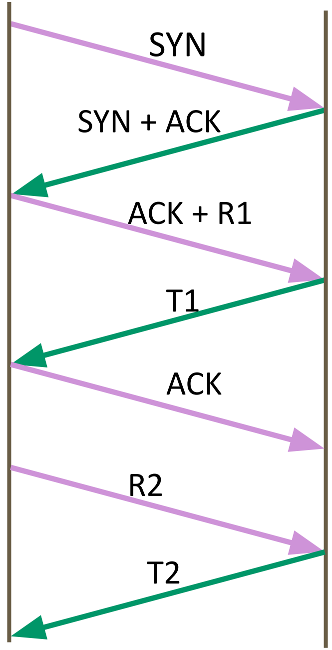
Concurrent



Pipelined



Persistent



Caching: How

- Idea: Replication
  - Replicate the content across multiple copies to reduce bottlenecks
- Implementation: content delivery networks (CDNs)
  - The client content provider modifies its content so that embedded URLs reference the new domains.
  - e.g.: *http://www.netflix.com/tiger_king.jpg* might become *http://a1386.g.akamai.net/tiger_king.jpg*

Caching: CNAMEs

- Instead of using a weird URL, use CNAMEs!
  - List picture as *http://cdn.netflix.com/tiger_king.jpg*
  - The authoritative server for this is controlled by netflix
  - CNAME record aliases *cdn.netflix.com* to *a1386.g.akamai.net*
- Akamai handles sending to the closest server because *a1386.g.akamai.net* is under an Akamai nameserver

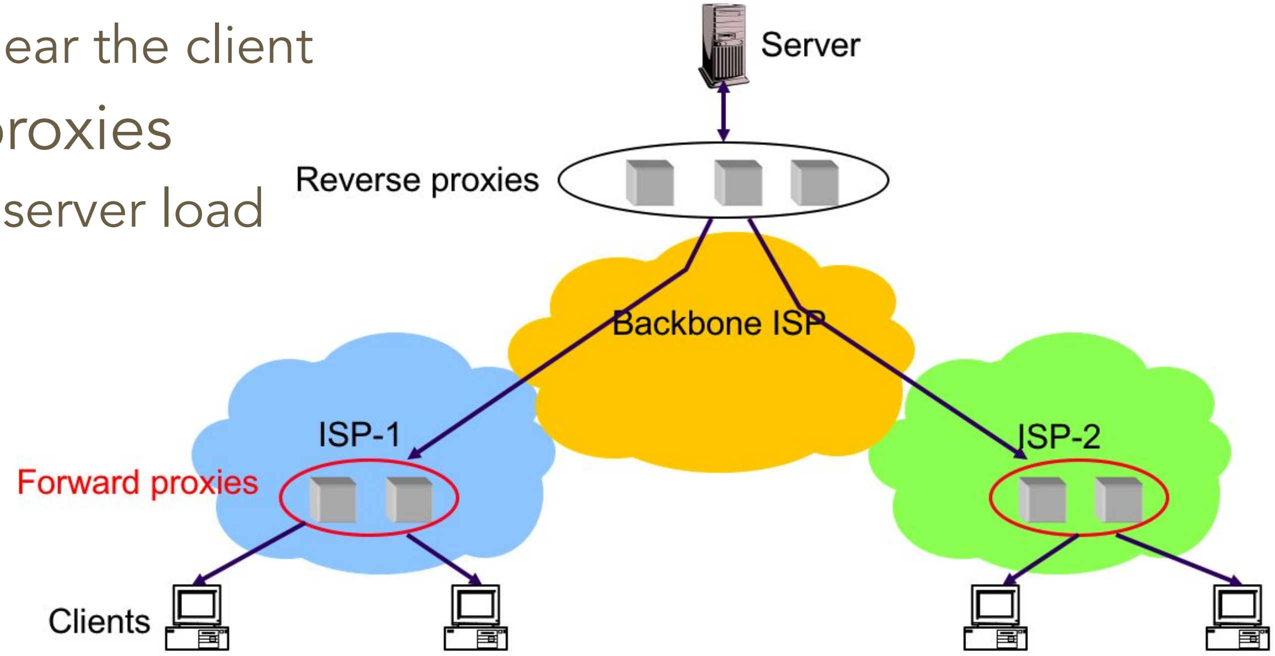
- Only change to the HTML is *http://www.netflix.com/tiger_king.jpg* became *http://cdn.netflix.com/tiger_king.jpg*

Caching: Specifics in HTTP

- GET Request header:
  - `If-Modified-Since` – returns "not modified" if resource not modified since specified time
  - `Cache-Control: no-cache` – ignore all caches; always get resource directly from server (think force refresh)
- Response header:
  - `Cache-Control: max-age=<ttl>` – TTL: how long to cache the resource
  - `Cache-Control: no-store` – Don't cache this
- When making request, if within the TTL, just load cached resource… otherwise, send with *if-modified.*
  - Server will either send a HTTP 304 ("Not Modified") or HTTP 200 (changed, and here's the new data)

Caching: Where

- Forward Proxies
  - Cache near the client
- Reverse proxies
  - Reduce server load



Feedback Form:
https://tinyurl.com/cs168-disc-fa24

Worksheet

Question 1

1. CDNs lower the latency for users.

2. HTTP 1.0 Requests are not human readable.

3. A server responds with only a header for an ”If-Modified-Since” request on an object that has not been recently modified.

4. Pipelined connections are frequently used in practice.

Question 2 Hints

- How long does it take to establish the connection?
- How long does it take to receive the webpage / media files?

Question 2

(1) Sequential requests with non-persistent TCP connections.

Question 2

(2) Concurrent requests with non-persistent TCP connections.

Question 2

(3) Sequential requests with a single persistent TCP connection.

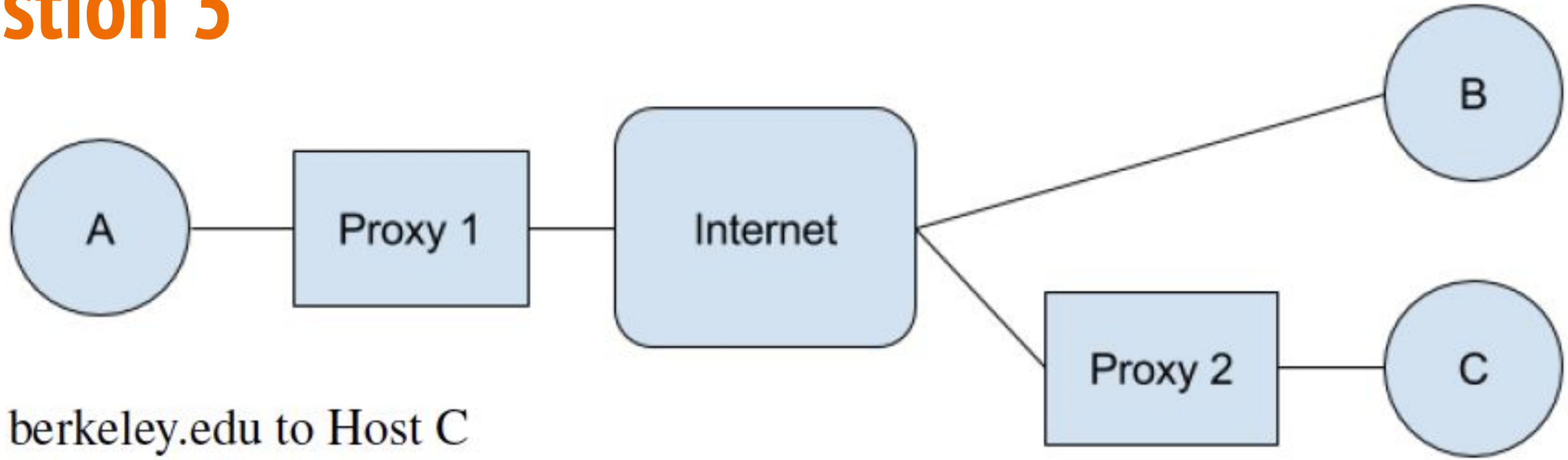
Question 2

(4) Pipelined requests within a single persistent TCP connection.

Question 2

(5) We have been assuming that the throughput for sending media files is T for a single connection, and $\frac{T}{n}$ for n concurrent connections. However, depending on the size of the media files, we can make more inferences about how fast we can send the media files. If the media files are very small, what kind of delay would dominate the time it would take to send them? What if the files are very large?

Question 3

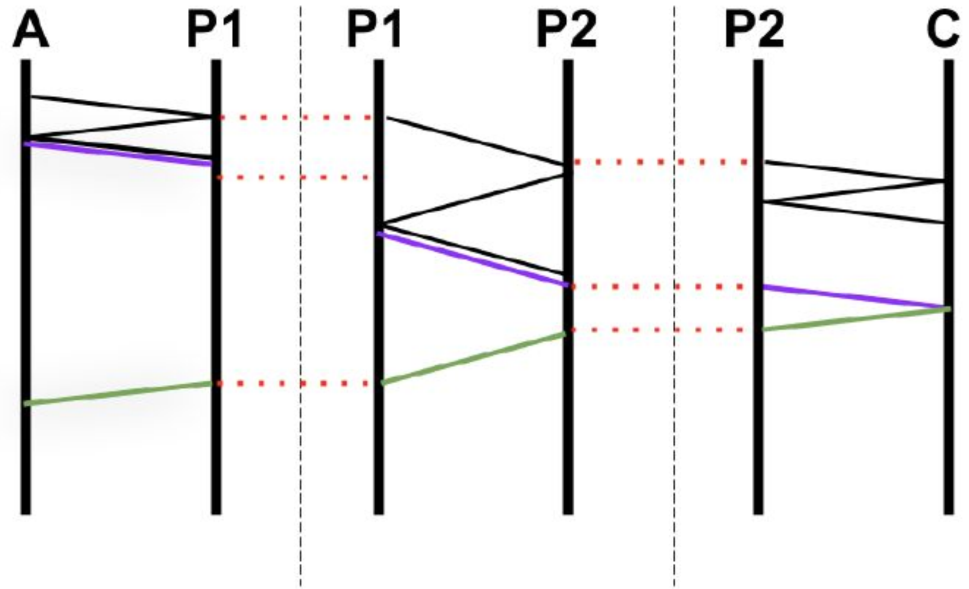


- berkeley.edu to Host C
- eecs.berkeley.edu to Host C
- stanford.edu to Host B
- mit.edu to Host B
- stanford.edu to Host B
- berkeley.edu to Host C

1. Sent Sequentially
2. Sent concurrently

3.1

$$
\begin{aligned}
&2(S+2L_C+L)+2(S+2L_B+L)+4L+S+2(L_C-L)+L=\\
&2S+4L_C+2L+2S+4L_B+2L+4L+S+2L_C-2L+L=\\
&5S+6L_C+4L_B+7L=\\
&5(5L+2I)+6(3L+I)+4(2L+I)+7L=\\
&25L+10I+18L+6I+8L+4I+7L=\\
&58L+20I
\end{aligned}
$$

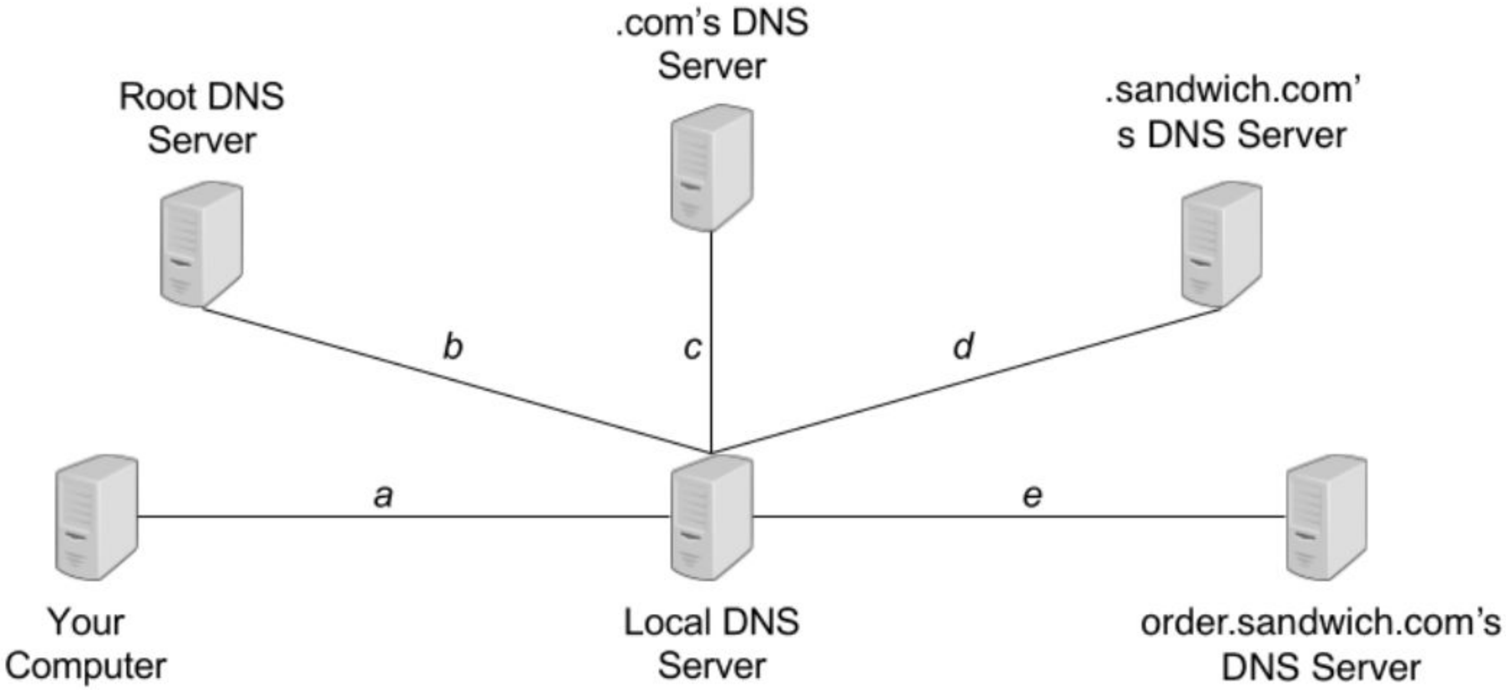


3.2

Solution: Since all the requests occur concurrently, there is no opportunity for caching by the proxies. Therefore the total time is just the max of all the times for the individual requests, which is the first request. It takes:

$S + 2L_C + L =$
$5L + 2I + 2(3L + I) + L =$
$5L + 2I + 6L + 2I + L =$
$12L + 4I$

Question 4: DNS



1. Your local DNS server doesn’t cache any information.

1. Your local DNS server doesn't cache any information.

Solution: Your computer begins by issuing a DNS query for www.order.sandwich.com to its local DNS server, which takes time a. Your local DNS server then iteratively queries the root DNS server, .com's DNS server, .sandwich.com's DNS server, and order.sandwtch.com's DNS server, which takes time $2b+2c+2d+2e$. It then returns the result of the query to you, which takes time a. Your computer can then issue a sandwich request to www.order.sandwich.com, which responds with an order confirmation, which takes time $2t$. The total time per query is hence

$$2a+2b+2c+2d+2e+2t$$

Each sandwich order takes this much time, since our local DNS server doesn't cache the IP address corresponding to www.order.sandwich.com, and so the number of orders we can make in this time is

$$\boxed{\text{\# sandwiches} = \left\lfloor \frac{T}{2a+2b+2c+2d+2e+2t} \right\rfloor}$$

(rounding down to the nearest integer with the floor function $\lfloor x \rfloor$).

2. Your local DNS server caches responses, with a time-to-live $L \geq T$.

Solution: If $L \geq T$, then this effectively means that once the result of the DNS query for www.order.sandwich.com is cached, it will remain cached in our local DNS server until the website's server ultimately goes down. The first query thus takes the same amount of time as a query from the previous part

$$2a + 2b + 2c + 2d + 2e + 2t$$

so long as this time is less than T. All subsequent queries only take time $2a + 2t$ thanks to caching. Hence we can model the number of sandwiches we get as follows:

$$\#\text{ sandwiches} = \begin{cases} 1 + \left\lfloor \frac{T-(2a+2b+2c+2d+2e+2t)}{2a+2t} \right\rfloor & T \geq 2a+2b+2c+2d+2e+2t \\ 0 & T < 2a+2b+2c+2d+2e+2t \end{cases}$$