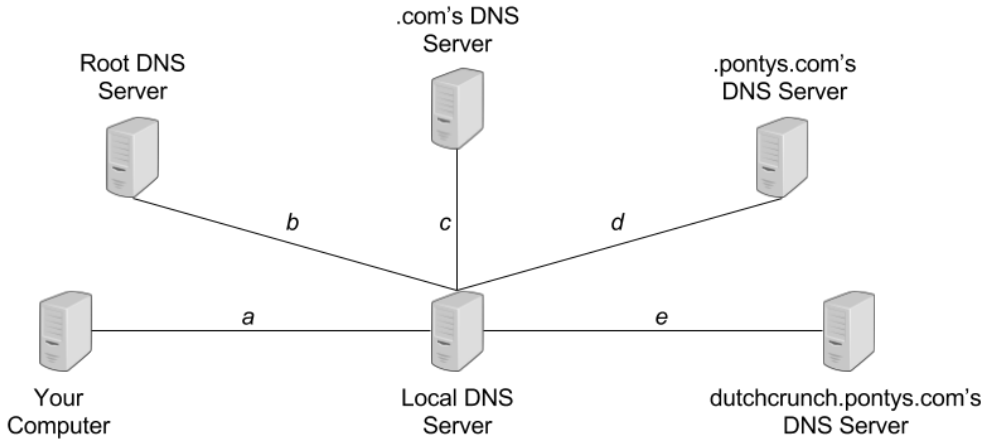
CS 168 Introduction to the Internet
Fall 2017 Scott Shenker

Discussion 10: DNS and Web

1 DNS

Pontague's Sandwiches, a deeply cherished sandwich shop in the Bay Area, is about to go out of business. However, being an adept Internet user with a particular eye for detail, you have realized that their website, www.dutchcrunch.pontys.com, is still operational and accepting online orders for the next T minutes. Consider the following setup of DNS servers, with annotated latencies between servers:



Assume that the latency between your computer and the Pontague's server is t, that once you send an order for a sandwich you must wait for a confirmation response from Pontague's before issuing another, and that your computer never stores any state about Pontague's IP address. In each of the following three scenarios below, determine how many sandwiches you can order before Pontague's closes for good:

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

Solution: Your computer begins by issuing a DNS query for www.dutchcrunch.pontys.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, .pontys.com's DNS server, and dutchcrunch.pontys.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.dutchcrunch.pontys.js, which responds with an order of confirmation, a transaction 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.dutchcrunch.pontys.com, and so the number of orders we can make in this time is

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

(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.dutchcrunch.pontys.com is cached, it will remain cached in our local DNS server until the Pontague'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:

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

(3) Let $T = 600$ seconds and $a = b = c = d = e = t = 1$ second. Your local DNS server caches responses with a finite time-to-live of 30 seconds.

Solution: When you make your first DNS query, the response gets cached in your local DNS server at time $a+2b+2c+2d+2e = 9$ seconds, and will remain cached for the next 30 seconds, until time 39.

Once the response is cached, your local DNS server sends it to your computer (which arrives at time 10) and then you issue a sandwich request (and receive confirmation at time 12). This gives you $39-12 = 27$ additional seconds to make more requests until the TTL for the cached entry expires, during which time you can make $\lceil \frac{27}{4} \rceil = 7$ additional sandwich requests, with the final request being completed at time 40.

At this point, the cached DNS response has expired, and 40 seconds have elapsed. The events above can repeat a maximum of $\frac{600}{40} = 15$ times in our 600 second window of opportunity, and so we can order at most $15 \cdot (7+1) = \boxed{120 \text{ sandwiches}}$. Not bad!

2 Performance

We want to download two files, both of size M. However, to download two files, we need to first download a webpage of size P. Assume the following:

- SYN, ACK, SYNACK, and HTTP request packets are small and take time z to reach their destination.
- Each of our HTTP connections can achieve throughput T for sending files and web pages across the network unless there are concurrent connections, in which case each connection's throughput is divided by the number of concurrent connections.
- You never need to wait for TCP connections to terminate.

For each of the following scenarios, compute the total time to download the web page and both media files.

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

Solution:

1. $2z$ (SYN + SYNACK) $+z$ (ACK/HTTP request) + $(\frac{P}{T}+z)$ (actual webpage)
2. $2z$ (SYN + SYNACK) $+z$ (ACK/HTTP request) + $(\frac{M}{T}+z)$ (first media file)

3. $2z$ (SYN + SYNACK) $+z$ (ACK/HTTP request) + $(\frac{M}{T}+z)$ (second media file)

Total $= \boxed{12z+\frac{P}{T}+2\frac{M}{T}}$

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

Solution:

1. $2z$ (SYN + SYNACK) + z (ACK/HTTP request) + $(\frac{P}{T}+z)$ (actual webpage)
2. $2z$ (SYNs + SYNACKs) + z (ACKs/HTTP requests) + $(\frac{M}{(\frac{T}{2})}+z)$

Total $= \boxed{8z+\frac{P}{T}+2\frac{M}{T}}$

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

Solution:

1. $2z$ (SYN + SYNACK) $+z$ (ACK / HTTP request) $+(\frac{P}{T}+z)$ (actual webpage)
2. z (HTTP request) $+(\frac{M}{T}+z)$ (first media file)
3. z (HTTP request) $+(\frac{M}{T}+z)$ (second media file)

Total $= \boxed{8z+\frac{P}{T}+2\frac{M}{T}}$

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

Solution:

1. $2z$ (SYN + SYNACK) $+z$ (ACK / HTTP request) + $(\frac{P}{T}+z)$ (actual webpage)
2. z (Both HTTP requests)
3. $\frac{M}{T}$ (first media file)
4. $(\frac{M}{T}+z)$ (second media file)

Total $= \boxed{6z+\frac{P}{T}+2\frac{M}{T}}$

Note that above, the second media file can begin to be sent immediately after the first is pushed onto the wire; hence, we don't need to account for the propagation delay of the first file in our calculation.

(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?

Solution: If the media files are very small, then transmission delay is small so propagation delay dominates. If the media files are very large, then transmission delay is large, and so dominates.