CS 168 Introduction to the Internet: Architecture and Protocols
Fall 2022 Sylvia Ratnasamy

Discussion 11: DNS & Web

1 True or False

1. One of the goals of a CDN is to lower latency for clients.
2. HTTP 1.0 Requests are not human readable.
3. If an HTTP request includes the header field "If-Modified-Since", then the HTTP server responds with only a header when the requested object has not been recently modified
4. Pipelined connections are frequently used in practice.

2 Performance

We want to download a webpage. We must first download the HTML (size P). This HTML includes URLs for two embedded images of size M which need to then be loaded. Assume the following:

- SYN, ACK, SYNACK, and HTTP request packets are small and take time z to reach their destination in either direction.
- 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.

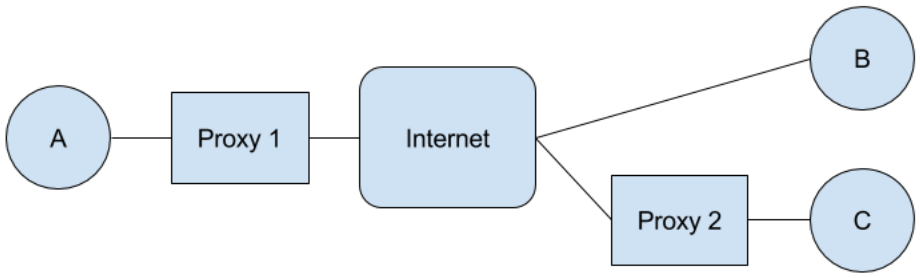
2. Concurrent requests with non-persistent TCP connections.

3. Sequential requests with a single persistent TCP connection.

4. Pipelined requests within a single persistent TCP connection.

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. Remember that the throughput for sending the media files depends on both its transmission delay and propagation delay. So far we ignored this finer granularity division but 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?

3 HTTP



Consider the (abstracted) network topology above. Hosts A and C are connected to HTTP proxies that cache the results of the last two HTTP requests they've seen to improve performance. The proxy will perform any TCP handshakes or teardowns with the client and server concurrently. That is, when it gets a SYN from a client, it will respond and immediately send a SYN to the server, and similarly for other messages.

As an example, let's suppose that A sends a request to B. A sends a SYN to B, which is intercepted by the proxy. The proxy sees the request is going to B, and initiates its own TCP handshake with B, all the while completing the original, separate handshake with A. By the time this has completed, there will be 2 TCP sessions. The first between A and the proxy, and the second between the proxy and B. When A sends a request, the proxy will forward it to B if it is not cached, or respond if it is cached. A similar process to the handshake is followed when A tears the connection down.

For the purposes of this problem, assume that the latency on each link is L, and the latency through the internet is I. Processing delay at all points and packet size is assumed to be negligible (don't consider transmission and processing delay). Assume that TCP connections use the 3 message teardown, and that no data is sent in ACK packets.

Suppose that Host A issues the following list of requests (in order):

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

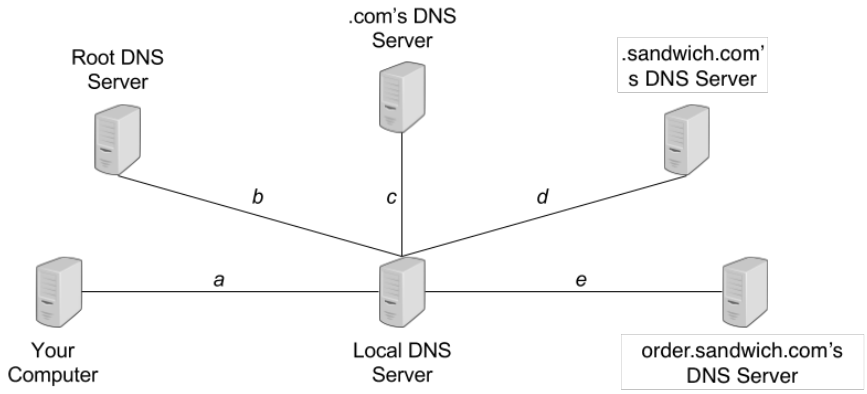
- mit.edu to Host B
- stanford.edu to Host B
- berkeley.edu to Host C

1. What is the total time to complete all requests if they are issued one at a time? That is, each completes before the next is started with a separate TCP session (no need to wait for the session to be torn down).

2. What is the total time to complete all requests if they are performed concurrently such that everything is done in parallel and there is no possibility for caching?

4 DNS

A sandwich ordering website www.order.sandwich.com is 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 website's server is t, that once you send an order for a sandwich you must wait for a confirmation response from the website before issuing another, and that your computer does not cache website's IP address.

In each of the following three scenarios below, determine how many sandwiches you can order for the next T minutes:

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

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

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.