Computer Networks - Homework 2

JJ McCauley
2/17/25

Chapter 1's Problems: 3,7,8,9,22

3 Transport and Application-layer protocols

In order for the HTTP client to retrieve the IP address from a URL, it must use the **DNS** protocol on the application layer to translate the URL to a valid IP. These DNS queries are then sent through the UDP in the transport layer. Once the IP address is acquired, the client is then able to establish a HTTP connection through TCP to receive the document.

7 Calculating total time with RTT

In order to calculate the total time, we must add **DNS Lookup Time** and **TCP Connection Time**. In order to find total DNS delay, we must consider that visits for n DNS servers can be considered as $\text{RTT}_1 + \text{RTT}_2 + ... + \text{RTT}_\text{n}$. Therefore:

$$\text{Total DNS Delay} = \text{RTT}_1 + \text{RTT}_2 + ... + \text{RTT}_\text{n}$$

Then, we must consider the TCP Connection Establishment. Since RTT_0 represents the time it takes for the client to send the HTTP request, as well as the time it takes for the client to receive the request, we can note the delay in this phase as:

$$2 * \text{RTT}_0$$

Therefore, the total overall time can be denoted as the following:

$$\textbf{Total Time} = (\text{RTT}_1 + \text{RTT}_2 + ... + \text{RTT}_\text{n}) + 2 * \text{RTT}_0$$

8 HTTP Transmission Time with Multiple Objects

8.1 Non-persistent HTTP with no parallel TCP connections

With non-persistent HTTP and no TCP connections, then each object would need to be fetched over a separate TCP connection. Therefore, since there are 8 objects, then the total time would be $8 *$ time for one object . Since a connection must be made to send and receive the HTML file, with each transfer taking one RTT, then we know that each object will require two RTTs for each object. Therefore, **the total time for this scenario would be** $2 * 8 = 16$ **RTTs**.

8.2 Non-persistent HTTP with 6 parallel connections

Since this approach uses non-persistent HTTP, then 2 connections will still be needed to fetch an HTML object (one for client to request, one for client to receive). Since **6** objects can be fetched at once, and there are 8 objects, we know that it must take two rounds of fetching to retrieve all files (first fetch for the first six objects, second fetch for the remaining two). Since each fetch will take two RTTs (request & receive object), and we must account for the initial TCP connection, then **the total time for this scenario would be** 2 **RTTs** + 2 **RTTs** + 2 **RTTs** = 6 **RTTs**, which accounts for the two object fetches and the initial connection. This is significantly more efficient than the previous approach.

8.3 Persistent HTTP

When using persistent HTTP, then all HTML objects can be transferred at once. Therefore, it will take 1 RTT for HTML files (transfer all 8 in one go), and will require an RTT for the client requesting and receiving, respectively. Therefore, **the total time for this scenario would be** 1 **RTTs** + 1 **RTTs** + 1 **RTTs** = 3 **RTTs**. This approach is the most efficient by far, when compared to the previous approaches ($3 < 6 < 16$).

9 Calculating Total Average Response Time

Firstly, we know that the average object size is 1,000,000 bits, the request rate is 16 requests/sec, and the internet delay is 3 seconds.

9.1 Total Average Response Time without Cache

We know that the average access delay is models as

$$\frac{\Delta}{1-\Delta\beta}$$

where $\Delta = \frac{\text{object size}}{\text{access link rate}} = \frac{\text{1,000,000 bits}}{100*10^6\text{bps}} = .01$ sec and $\beta = .01 \text{ sec} * 16$ requests per second $= .16$. Plugging this into the equation, we get:

$$\frac{.01}{1-.16} = \frac{.01}{.84} = .011905 \text{ seconds}$$

Then, we can use the known internet delay of 3 seconds to derive the following answer:

$$T_{\textbf{avg, no cache}} = \text{Average Access Delay} + \text{Internet Delay} = .011905 \text{ sec} + 3 \text{ sec} = 3.011905 \textbf{ seconds}$$

9.2 Total Average Response Time with Cache

Now with cache with a miss rate of 0.4, we can recalculate the total average response time knowing that only the misses need to be considered when calculating β. To achieve this new number, we can multiply the request rate (16 requests per second) by the miss rate (.4), such that

$$\beta = 16 \text{ requests per second} * 0.4 \text{ miss rate} = 6.4 \text{ requests/second.}$$

Then, we can multiply this by Δ such that 6.4 requests per second $* .01 = .064$. Now, we can plug this into the Access Delay formula as follows:

$$\text{Access Delay} = \frac{.01}{1-.064} = \frac{.01}{.936} = .01068 \text{ seconds}$$

Then, we can calculate the total delay when a cache hit occurs (knowing internet delay) through the following calculation:

$$T_{\text{cache miss}} = .01068 \text{ sec} + 3 \text{ sec} = 3.01068 \text{ sec}$$

Know that we know the response time for cache misses, we can calculate the **total average time when caching** by finding the weighted sum via the following formula:

$$T_{\textbf{avg, cache}} = (.6)(0\text{sec}) + (.4)(T_{\text{cache miss}}) = 0 + .4 * 3.01068 = 1.2043 \text{ seconds}$$

Note that cached responses will require 0 seconds for a hit, so it can essentially be disregarded. As we can see, when compared to the previous response, Caching can be significantly faster and reduce the total average response time immensely.

22 Minimum Distribution Time: Client-Server & P2P

For this question, we know that we need to distribute a file of $F = 20$ Gbits to N peers, where $N = 10, 100, 1000$. Additionally, we know it has an upload rate of $u = 300$ Kbps, 700 Kbps, 2 Mbps, and an download rate of $d = 2$ Mbps.
In order to find Client-Server distribution T_{cs} and P2P distribution T_{p2p}, we will use the following equations:

$$T_{\text{cs}} = \max\{\frac{F}{u}, \frac{F}{d}, \frac{NF}{u}\}$$

$$T_{\text{p2p}} = \max\{\frac{F}{u}, \frac{F}{d}, \frac{NF}{u + Nu}\}$$

Prior to computing, we must firstly convert the units into bits or bits/sec, which will be used for the equation. For example, we must convert F such that $F = 20 * 10^9$ bits.
Now, we can utilize these equations for each of the N and u cases in order to find the total computation time for both Client Server and P2P.

22.1 Client-Server Distribution Results

In the client-server architecture, only the server's upload rate u_{s} is to be considered when performing the calculations, with $u_{\text{s}} = 30 * 10^6$ bits/second. Therefore, we can derive the following table using N:

N	$\frac{F}{d}$	$\frac{F}{d}$	$\frac{NF}{u}$	T_{cs} (seconds)
10	667	10,000	6,667	**10,000**
100	667	10,000	66,667	**66,667**
1000	667	10,000	666,667	**666,667**

Table 1: Client Server Distribution Results

where T_{cs} is derived from the maximum of the previous columns. We can then use these values in the final chart.

22.2 P2P Distribution

In the P2P architecture, the upload rates u must be considered. When plugging in the given numbers (with units adjusted) with each N and u, and finding the max of the three fractions (as shown in the equation above), the following tables are produced:

N	T_{p2p} (s)
10	**10,000**
100	**33,333**
1000	**60,606**

Table 2: P2P Distribution Results for $u = 300 \times 10^3$ bps

N	T_{p2p} (s)
10	**10,000**
100	**20,000**
1000	**27,397**

Table 3: P2P Distribution Results for $u = 700$ bps

N	T_{p2p} (s)
10	**10,000**
100	**10,000**
1000	**10,000**

Table 4: P2P Distribution Results for $u = 2000$ bps

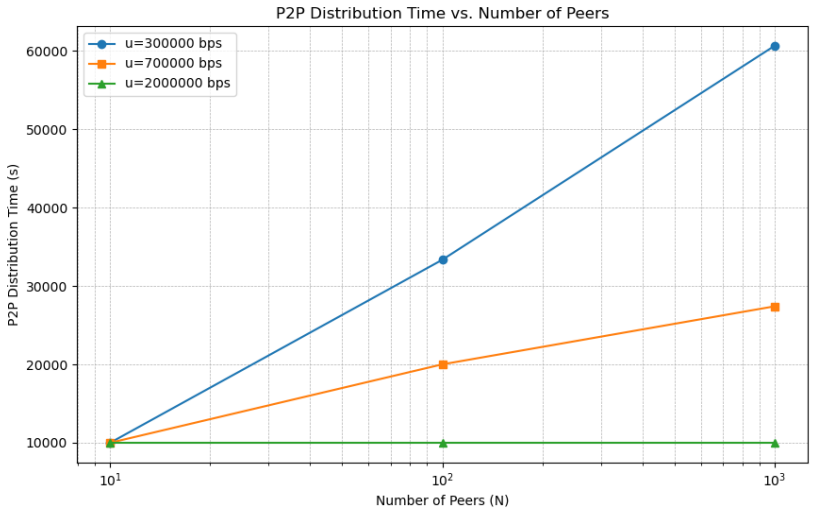

Figure 1: P2P Distribution Graph

As we can see, the P2P architecture significantly outperforms the Client-Server model. The speed of the P2P architecture increases as N peers increases, and decreases as the upload rate increases.