ECE 358 - A3

Ariel Weingarten (20366892)
Alexander Maguire (20396195)

May 29, 2014

1 Question 1

1.1 Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case?

No, where sense is taken to mean will there be an increase in the total download time. For the initial download both serial and parallel instances of HTTP will perform exactly the same as there is only one file to retrieve. Afterwards, the total download time will still be the same for parallel and serial instances of HTTP. The line is saturated by any one download. We could split it N ways, but would just be obtaining each individual file more slowly because there is no unused bandwidth to capitalize one. Saturation can be seen by comparing our transfer rate of 150 bits/s to the data packet size of 100,000 bits for objects that are 100,000,000 bits.

The only difference between serial and parallel instances of HTTP in this case is that after the initial object has been downloaded, parallel instances of HTTP will result in N objects being retrieved simultaneously but more slowly, whereas serial instances will retrieve individual objects more quickly, but can only make progress on one object at a time. If more was known about the target objects, then perhaps one of serial or parallel could be seen as more desireably based on this observation.

1.2 Now consider persistent HTTP. Do you expect significant gains over the non-persistent case?

With persistant HTTP, all that is saved is the overhead of creating new connections. Since HTTP makes use of TCP, we know that there is a three-way handshake to set up new connections. Each control packet is 200 bits, making the total overhead 600 bits per connection. It takes $\frac{600[bits]}{150bits/s} = 4$ seconds to handle this overhead per connection. It takes $\frac{100000bits}{150[bits/s]} \approx 666.67$ seconds to transfer a data packet (which happen to be the same size as our objects).

With 11 objects, the total download time of data is $t_{data} = 666.67[s] \times 11 \approx 7333$ seconds. Using persistent HTTP, we only need to set up a connection once. This adds 4 seconds to the download time, giving $t_{persistant} = 7337$ seconds. Using non-persistant HTTP we need to set up 11 connections. This adds 44 seconds to the download time, giving $t_{non-persistant} = 7377$ seconds.

With these values we can see that persistant HTTP is $1-\frac{7337}{7377} = .005422258\%$ faster. This is arguably an insignificant speedup.

1.3 Do Bob's parallel connections help him get Web pages more quickly?

Yes. Let n_{bob} denote the number of connections that

1.4 What is the impact on Bob?

2 Question 2

In BitTorrent, suppose Alice provides chunks to Bob throughout a 30-second interval.

Will Bob necessarily return the favour and provide chunks to Alice in the same interval? Why or why not?

Necessarily, no, Bob won't return the favour if Alice is a seed and doesn't require any chunks.

Furthermore, if Alice is only connected to Bob, but Bob is connected to the entire swarm, it's feasible that Alice will send data to Bob, but be a trivial source of chunks to Bob, so he will send to more prominent members of the swarm.

In the absense of these two cases, however, it is very likely that Bob will reciprocate chunks to Alice.

3 Question 3

3.1 Show $D_{cs} \geq max\{\frac{NF}{u_s}, \frac{F}{d_{min}}\}$ is a tight bound.

Assume an ideal network (no unexpected latency or dropped packets) – which is the best possible case for D_{cs}. Since the upload transmission rate to a client must equal the download transmission rate from the server, there are two cases when determining the bottleneck in the network.

The first is that centralized server's upload speed is the bottleneck. In this case, the server must upload F bits to N peers, each of whom can download faster than the server can upload. In this case, the bottle neck is the total data uploaded divided by the upload rate:

$$D_{cs} = \frac{NF}{u_s}$$

The other bottleneck case is a client has a slower download rate than the server can upload. In this case, the total duration is the size of one file over the slowest download rate.

$$D_{cs} = \frac{F}{d_{min}}$$

Combining these bottlenecks, it is clear that whichever duration of the two is larger will be the total time in an ideal network. Of course, it is unlikely the network be ideal, validating our usage of the inequality:

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

3.2 Show $D_{P2P} \geq max\{\frac{F}{u_s}, \frac{NF}{u_s+u}\}$ is a tight bound.

Again, assume an ideal network. The P2P model also has two potential bottlenecks:

If the original seeder s's upload speed is the bottleneck of the network, the swarm can transfer each chunk to the entire network faster than the seeder can provide chunks. By the time the seeder has uploaded its last chunk for a total of F bits, the network has the entire file. The total transmission time is thus described as:

$$D_{P2P} = \frac{F}{u_s}$$

The other case is that swarm's health is the bottleneck. Here, the total upload power of the network is $u_s + u$ where u is the sum of the individual peers' upload rates.

Since there are NF bits to send, and it is likely that $\forall i : d_i >> u_i$, the upload speed of the network will be the bottleneck of the file share. The total transmission time is thus:

$$D_{P2P} = \frac{NF}{u_s + u}$$

Again, only one of these cases may be the bottleneck, so we are justified in maximizing them with one another:

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{NF}{u_s + u}\right\}$$

Our assumptions are an ideal case, making it likely that this is indeed a lower bound, but still attainable.

4 Question 4

Which k peer identities should be maintained at each peer to minimize the query for a key on average?

The best strategy for splitting knowledge of k identities from a pool of N entities is circularly – with each peer maintaining identities evenly divided along the circularly linked peer list. This allows lookup of any node in $O(\log_k n)$, since each step is a k-section of the remaining peers).

The optimal peer $i \in [0..k)$ identity maintained by node n is described by the following:

$$p_n(i) = n + \begin{cases} i = 0, & 1 \\ else, & \lfloor i\frac{N-2}{k} \rfloor \end{cases} \mod N$$

5 Question 5

First you have to upload the image file (JPEG, PNG or PDF) from your computer to writeLaTeX using the upload link the project menu. Then use the includegraphics command to include it in your document. Use the figure environment and the caption command to add a number and a caption to your figure.