SOFTENG 710 Assignment 2

Jordan Thoms 1203117

1 Testing

I have tested these programs in the ground floor computing lab (compsci building), with their Ubuntu Linux 11.10 installation.

2 Run Instructions

I have included scripts to run the programs - from a terminal, cd to FragmentHttpServer and execute run.sh, do the same for FragmentHttpCache and HttpClient. I've also included scripts for part 1's HttpServer and HttpCache. (As they both listen on the same ports, the Part I and Part II programs should not be run simultaneously.)

3 Caching Technique for Part II

I used a system where the Http cache first sends a HEAD request to the server, which returns a list of the blocks in the file, identified by MD5 hash. The cache then checks to see which of those blocks are not cached, and sends a GET request to the server with the blocks it needs in a header, if necessary (so it requests all the needed blocks in one request). The server replies to this request with a JSON object mapping each of the block identifiers to a base64 string representing that block's binary content. JSON was used for simplicity in this case - in a production system a binary protocol would probably be used as that would remove the time and space overhead of the base64 encoding.

Once the cache has all the necessary blocks, it reconstitutes the file by appending the blocks together, using the values from the cache.

To avoid the issue of changes early in the file causing the block boundaries to change and then needing to reload the whole file, I used the Rabin formula as described in Value-Based Web Caching (S. C. Rhea, K. Liang and E. Brewer, 2003). When the Rabin formula , modulus 2048, is zero, a block boundary is placed. There is also a minimum block size of 128, and a maximum of 4096, in case an unusual or malicious input is being processed. This results in a high cache efficiency.