Machine Problem 4: Network Programming – A Multi-Threaded HTTP Downloader

(Due on Apr. 23, submit the zip file through eCampus)

The Assignment

For this lab, you will implement a multi-threaded HTTP downloader using C/C++ and socket programming.

1. Study the HTTP protocol: Hypertext Transfer Protocol -- HTTP/1.1: http://tools.ietf.org/html/rfc2616. Understand how HTTP works (e.g., HTTP GET request and response). Write a simple socket program `getweb` that can download any **single** file with a given URL (e.g., http://faculty.cse.tamu.edu/guofei/paper/PoisonAmplifier-RAID12.pdf) through HTTP.
2. Study how to speed up the downloader by using parallel partial downloading (in multiple threads) of the *single* file/URL. Read RFC2616 and this article (http://benramsey.com/blog/2008/05/206-partial-content-and-range-requests/). Make sure you understand the following:
   - HTTP HEAD and GET Request
   - HTTP Response and Content-Length & Accept-Ranges Headers
   - HTTP GET Request with Range Header

   Sketch the design in your report, and implement a multi-threaded version of the HTTP downloader `mgetweb`. (Kind reminder: your `mcopyfile2` program in Machine Problem 2 can be a basis for this new program.)
3. Measure the performance of these two programs when downloading different sizes of files (ranging from MB to GB, for which you can choose some Linux distribution/images on the Internet as testing files). How much can your multi-threaded downloader speed up? Also measure the performance of `mgetweb` by change the number of threads. In all the measurements, please use graphs to show the results and explain the best configuration.

4. [*BONUS task, 20 additional points*]. Following the similar principle, can you implement a multi-threaded FTP (File Transfer Protocol) downloader in sockets and C/C++?
5. [*BONUS task, 50 additional points*] The basic goal of this bonus task is to contribute your tool to the open-source community (this implies a high-quality requirement). To work on this bonus task, *please let me know first*, and (with my permission) a team (with up to 2 members) is allowed to work on this task.

What to Hand In

- The source code of programs: e.g., `getweb, mgetweb.`
- A detailed report that describes your design, results, findings, and thoughts. It should contains answers to all the questions above, detailed procedure with necessary screen captures, issues/problems encountered and how you fix them, understanding of systems after the lab, and any lesson you have learned and thoughts you have about the lab.