CS 3251: Computer Networks I April 3, 2013

Problem Set 3: Transport and Application Layers

Instructor: Prof. Nick Feamster *College of Computing, Georgia Tech*

Turn in your writeup and talk on **April 17, 2013** by 11:59pm. *Please upload your solutions to T-Square. Other forms of submission will not be accepted!* We will be providing more information about how to turn in your assignment as the due date approaches.

1 Web Caching and Performance

1. Explain the difference between Web proxying and caching. Why would you want to run a non-caching Web proxy? What benefits are provided by a caching Web proxy?

2. List three different Web proxies online that you might want to use for different purposes. For the ones that have public services running, configure your Web browser to use the proxy. Do you notice any difference in performance for any of them?

3. Occasionally, companies, businesses, and governments will re-route your traffic through a proxy in a "transparent" fashion, without you knowing it. What kind of tests could you run to determine that your Web requests were being routed through a Web proxy? (There is no single right answer; think about what types of discrepancies proxies might introduce.)

4. Go to `http://webpagetest.org//` and create a test for `http://nytimes.com`. Include a snapshot of the report in your assignment. Include information about what location your request was made from, as well as which browser was used for the test (Note: not *your* browser, but the Web browser that `http://webpagetest.org/` uses. For each of the questions below, please *explain how you computed your answer*.

    - What is your best guess at how `http://webpagetest.org/` works? Suppose you wanted to run this type of test from many different locations. How would you design such a service?
    - What was the total loading time for the page?
    - How many total objects were downloaded?
    - How many DNS lookups were required to load all of the resources for this page?
    - Some objects were returned with a 302 error code. What is a 302 error code, and why do you think it was returned for certain objects?
    - What is "time to first byte", and why is it an important metric?
    - How many distinct locations were the Web page resources downloaded from? For those labeled "Akamai", can you find the location of those resources by doing a traceroute, IP lookup, etc.? Where is that content relative to where the test was performed?
    - How many unique domains is the browser connected to at any time? How many connections does the browser make to each domain at any time? How does the number of parallel connections affect page load time? What impact might parallel HTTP connections have on "fairness" to other clients and Web browsers?

- What determines the order in which the resources for this page are fetched and loaded?

5. Suppose that a client has an "origin server" for a Web site that is 300 ms away (such might be the case for a user in a developing country), but that a local cache is 10 ms away. Suppose that the page contains static content that can be retrieved in five round trips to the server that contains the content over a TCP connection.

   - If all of the static content resides in the local cache, how much time would be saved in loading the page?
   - Suppose only 50% of the static content were in the cache. How much time would be saved to load the content?
   - What types of steps might you perform in designing a Web cache to improve the cache "hit rate"?
   - Suppose you have a site that also has dynamic content. Describe how you would use a "reverse proxy" to improve the loading time for a Web page.

2 Programming Assignment: Simple HTTP Proxy

In the previous assignment, you wrote a client and server program to do a TCP file transfer. This follow-up assignment builds up on the previous one to implement a HTTP Proxy that can handle an HTTP GET request. *We will run moss on your code to check for copying, so please be sure to write your own code.*

Most HTTP transactions can be broken down to the following sequences (RFC 2616):

1. Client establishes a connection with the server (a TCP connection)
2. The client issues a request, by sending a line of text using HTTP method (e.g., GET,POST etc. which were discussed in class), with the version of HTTP.
3. The server sends back a response status code, a reason phrase (description of the response code) and the message body containing the response.
4. The connection is closed, unless persistent connections are used.

HTTP Proxy: In contrast to the ordinary client-server model, where an HTTP client directly communicates with the Web server, in certain situations (such as in a content distribution network) the Web client might send a Web request to a proxy.

A proxy server is an intermediary between the web server and the client, which implies it serves the pages that would be served by another Web server. Your task is to implement a Web proxy that can accept and forward request to the original server. *Your server only needs to be able to handle one HTTP request at a time.* (Unless you want to do more, of course, which we will certainly welcome. :-)

The only type of HTTP request you need to support with your proxy is a GET request. The GET request addressed to a proxy server must have an absolute (not relative) URI. You should also implement "Bad Request" (400) error code for an invalid request, "Not Found" (404), and "Not

Implemented" 501 error code if the server does not support the functionality required to fill the request.

You should write your Web proxy in C or Python with the file structure and Makefile as explained in Socket Programming Assignment. Do not hardcode the port number that your proxy is running on. *If you are using Python, you are not allowed to use any libraries that implement the Web proxy for you.*

You can use your Web browser to test your Web Proxy by setting the Web proxy settings in the configuration of your Web browser and issuing an HTTP request from the command line.

To test your proxy, we will enter the request type, absolute URL (beginning with HTTP), and HTTP protocol with the version. You can also use a simple command line tool such as `curl` to fetch a URL (without fetching all of the embedded objects). For the test case, we will attempt to use your proxy to fetch a simple HTML file with no embedded objects from a local server at Georgia Tech.

Design question: Extending your proxy. Suppose that you want to extend your Web proxy to handle multiple concurrent HTTP requests (either from a single browser that opens multiple connections to the server, or from multiple clients.

1. Why would you want your proxy to support multiple simultaneous clients? (Hint: Think about caching.)
2. Why would you want your proxy to support multiple simultaneous requests from the same client?
3. How would you change your proxy code to support multiple concurrent HTTP requests? (Hint: Think about the things discussed in the sockets programming lecture, where several options were discussed.) What libraries or function calls would you use? Be as specific as you can; if you'd like to write the code to support concurrent requests, you're welcome to do so, although you're not required to do this. An explanation of what you *would* do will suffice.