Project 1 - Web server / client

Due Feb. 20th 23:55

This project should be done in pairs.

Introduction

In this project your goal is to write your own simple web browser/ webserver on top of a simulated network. The code on moodle simulates the different layers of the internet. For now each layer is simply passing the message down or up a layer without making any changes to it. In addition the network currently only allows for one connection between the source and the destination. To see how the network works, simply run the ServerApplication first and then run the ClientApplication. Everything you type into the client application should show up on the server side. The server will acknowledge each message with a “received” response.

Functionality

Although most of the work will be done in the application layer you are required to implement two changes to the lower layers to make your network appear real:

- Add transmission delay and propagation delay in the network layer. Since you are simply simulating the delay (and not its cause) you can just put some time delays in the code. The propagation delay should be constant for every message sent, while the transmission delay should depend on the size of the message.
- Add a version of the 3 way handshake in the transport layer when the application is asking to open a connection. You do not need to follow the exact format of TCP (for example, no need to add IP addresses or port numbers since there is only a single connection), but should have a transport layer protocol in which the client send a syn and the server ack’s before the first message request can be sent.
- Besides that there is no need at this point to add any additional headers in other layers.

The rest of the changes should be made in the application layer. Feel free to add additional classes if necessary:

- You should use the same general format for request and response messages as HTTP.
- Your client/server should at minimum support the GET method.
- You client/server should be able to use the if-modified header.

Your server should be able to produce the following responses when required. (Each one should be accompanied by an appropriate message).

- 200
- 404
- 304

You are required to add an additional improvement to your HTTP protocol. This could be whatever you like. You might be inspired by Google's SPDY (you do not need to implement it in exactly the same way, but it might inspire you).

You do not need to implement a full version of HTML. Instead have a simpler markup language that allows for only the following:

- Text to display
- Additional embedded text files

When displaying your HTML file simply display the text with the additional embedded text files.

Program Input:

Each program (client/server) should accept a few command line parameters:

The server should accept the propagation delay and the transmission delay (per byte). This can be added to the different layers constructor's parameters.
The client should accept the HTTP version (either 1.0 or 1.1), and the page requested. This should lead to a persistent or a non persistent connection.
Your server should support both depending on what the client is requesting.
In addition the client should print out the webpage received.
Finally the client should print out the time taken for the whole website to arrive.

Analysis:

Your analysis should include the comparison of the following.

Comparing persistent and nonpersistent download times of an entire "webpage".
Comparing using local caching vs. not using local caching.
Comparing any additional modifications you have added to your protocol.

For each one you should graph a few interesting comparisons on how things change with different transmission delays, propagation delays, file sizes and number of files. You do not need to look at all the parameter space, but think about experiments that would be interesting.

Submission:

Besides your code (which should be well commented), please submit the following report:

A short introduction to what the program is trying to achieve.
A description of the parts of the HTTP protocol you have implemented. This should include:
    - Syntax of the messages (both request and response). That is what are the fields and how they are organized.
    - Semantics of the different fields. That is what can be in the fields and how they are used. Here you should discuss what methods/response code you have used, in addition to which header lines you support.
    - The expected exchange of messages. If this is different for different HTTP versions, mention that here.
A description of your markup language and how it works.
A description of your code design and choice of data structures.
Correctness Results - In this section you should present results for 4 different runtimes of downloading a full “webpage”. (Persistent, Non Persistent, Cache (either persistent or not), and your additional method). They should be presented in the following way:
    - Describe the delays used for all the experiments.
    - Manually calculate the time it should take to download the webpage. Show your calculation.
    - Show the average time as measured by your algorithm to ensure correctness.
Results Analysis - Show a comparison of the different algorithms when changing different parameters. Make sure to present sme graphs which show how the time depends on different parameters.
Conclusion - Mention which algorithms work the best and under what conditions.
References - If necessary.
A section on what each member of the team did as part of the project.