Homework 3

Due: 17 April 2023

Problem 3.1

HTTP was designed to be stateless: at the protocol level, nothing is "remembered" from one request to the next. Any application that wants to connect a current request with a previous one needs to do their own work to do so. With that in mind:

a. What might be different in a version of HTTP that was designed to run over UDP? Why would the designers of HTTP have chosen to implement it over TCP instead of UDP?

b. Having chosen TCP, what are some properties of TCP that they could have made use of, but didn't?

Problem 3.2

Many applications, like web browsers, word processors, and photo editors, let the user have many tasks open at the same time (whether in different windows or in different tabs). It would certainly be possible to design such a program to be a single process running one thread per task, or to design it to fork a separate process for each task. (Ignore for the moment the possibility of a hybrid solution, or one with multiple tasks/documents per thread.)

What tradeoffs are being made in this decision? Identify and explain at least one advantage in each direction.

Problem 3.3

Software like Google Docs can be set up to have multiple people with access to edit the same document, potentially at the same time. When people use such a tool to collaborate on a document, typically they will coordinate, either with an audio channel for a synchronous editing session or perhaps a group chat or email thread for more asynchronous work.

Often, it all works out. Describe at least three ways that groups like this coordinate their editing (or, for one or two of the three, describe a way that a group might *fail* to coordinate sufficiently). In each case, make use of the technical terminology of thread/process synchronisation as appropriate.