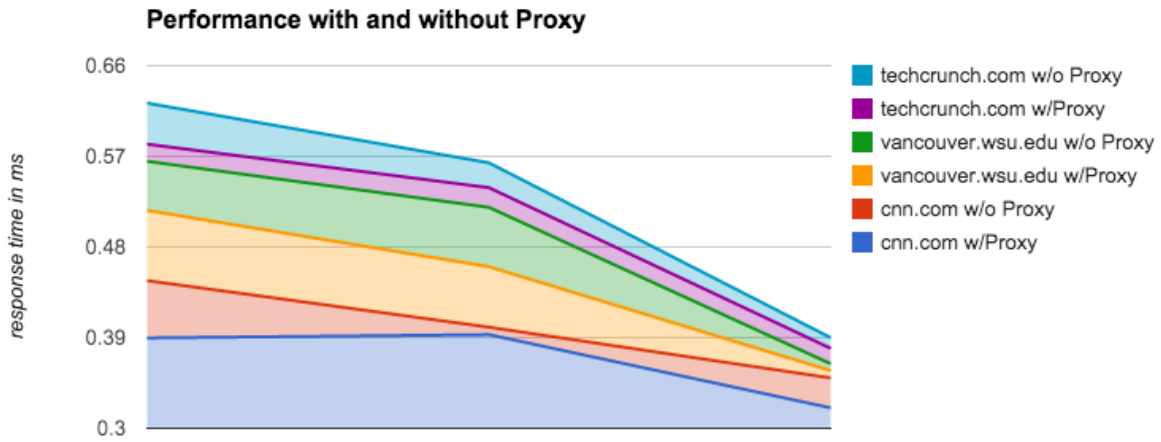
David Parrott
CS 455 Program 2 - HTTP Proxy w/ Cache
Due 9/22/14
Fall 2014

Proxy Design:
The proxy is designed to run on any internet accessible Unix machine. Upon execution a socket is opened with the port specified on the CLI when the script is executed, or a port specified as DEFAULT_PORT in the header. When a client attempts to connect to the proxy server a thread is spawned to handle the request. The thread handler reads all of the data from the client and assumes that any data sent from the client is a properly formatted HTTP request. There is no contingency to handle data that is not correctly formatted; however, a try/except is used to catch errors that will result from the data sent by the client not being a properly formatted HTTP request. The thread handler does not assume that the client's browser is set to not use a local cache and as such it attempts to remove If-Modified-Since and If-None-Match entries within the header sent by the client.

Once there is reasonable assurance that the HTTP request is formatted correctly it is sent to parse_request which returns the hostname, IP, request (GET, POST, CONNECT, etc), and url. The request is sent to STDOUT as the only information the proxy displays during operation. The url is used as a key to determine if a given file has been cached already. If it is not found in the cache a new socket is opened to the server, the request is sent, response received, file cached, sent on to the client and finally the sockets to both the server and the client are closed. If the file is already in the cache the If-Modified-Since field is altered to contain the date and time that the file was cached by the proxy before it is sent to the server. If the server responds with a 304 response code the cached file is sent back to the client; otherwise, the file is requested, cached and sent on to the client before the sockets to both the client and the server are closed.

In order to have a more readable implementation custom classes Dictionary and Entry were created. Dictionary contains a Python dictionary called cache that will use a url as a key and an Entry as the value, a deque to keep track of the order pages were cached in and a threading lock. There are several methods used to edit and access the contents of the Dictionary elements: insert, delete, exists, get, remove_lru. Insert checks if the cache is full and either adds the entry to the cache and the url to the deque or else removes the oldest entry from the cache and deque before adding it. Delete removes an entry associated with a given url from both the cache and the deque. Exists returns true if the entry is in the cache. get returns the entry associated with a given url. Remove_lru removes the oldest entry from the cache and deque. An Entry contains the time the entry was cached, the contents of the file associated with a url and the url itself. While each other the methods in the Dictionary class contain threading locks since only native Python libraries and data structures are used they are inherently thread safe.



The data shown above indicates that response times were better with the proxy than without it. In order to improve response times further more efficient methods to parse the requests from the client could be used. A sequence of regular expressions instead of repeated calls to split as well as iterating over lists to search for strings should dramatically improve performance.