DESIGN

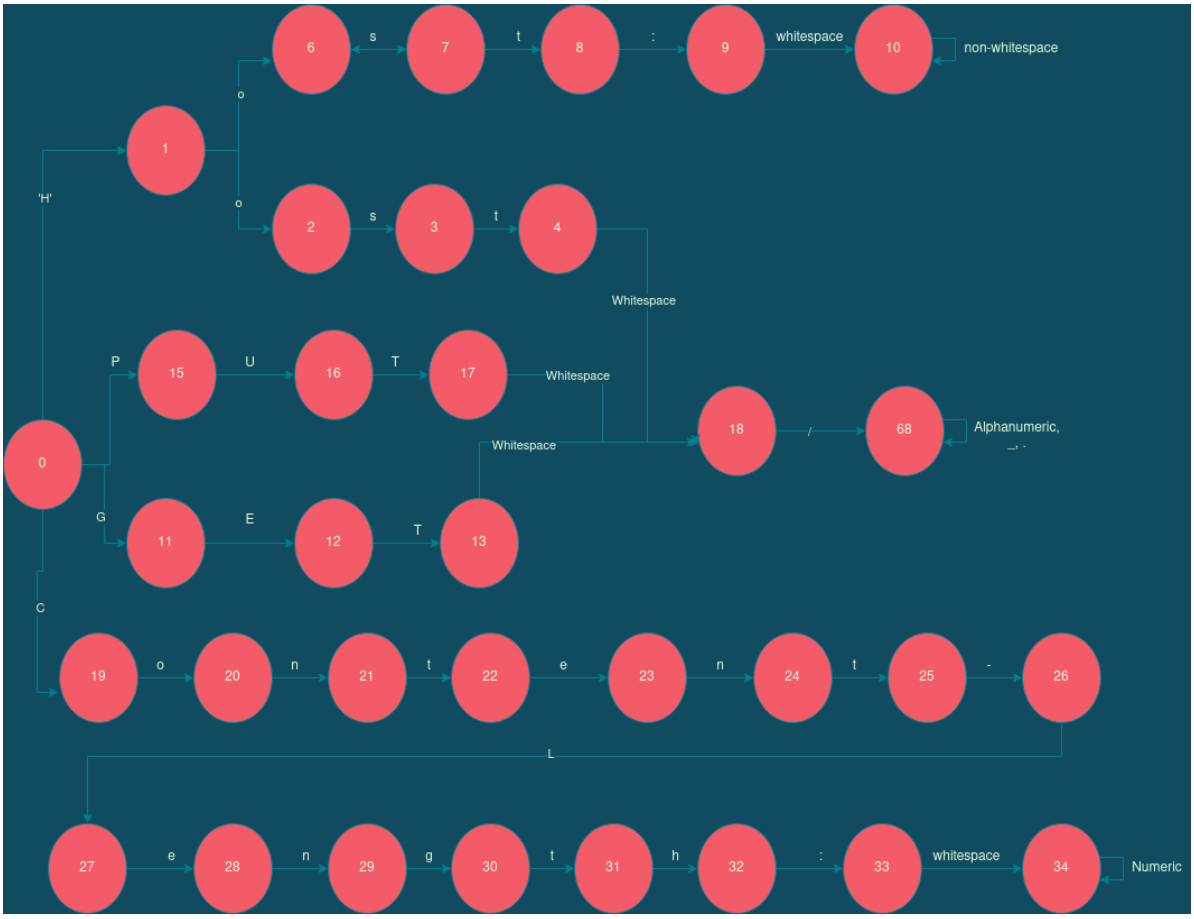
Abstract/General Algorithm

A dispatcher thread picks up connections and hands them off to worker threads. Each worker thread that picks up a connection then checks the request to ensure that it's of the proper format, and is requesting an item that aligns with the specifications. If so, the worker thread must increment the number of calls made to the proxy, such that if the number equals R, the worker thread can probe all of the servers for their healthchecks and balance the load accordingly. After incrementing the number of calls made to the proxy, and incrementing the number of calls/fails to the httpserver, the worker thread(s) will then check if the file is in the cache of the proxy. If it is, then the proxy must check the modification time of the file. If the file's modification time matches the modtime returned by a head request to the targeted HTTP server, then the proxy responds immediately with that stored file. If not, or if the file is not in the cache, but can be stored there, then the thread locks the critical region, adds the file to the cache (dequeuing if necessary), and moves on to make the get request. From there, we rinse and repeat.

Complex algorithms

Response/Request parsing

Every section of my program that must parse through a response or a request does so with an NFA, which is functionally identical to the NFA I had for asgn2. For reference:



Caching Logic

I used a linked-list based queue for the cache. If the cache is full, and a new request is made for a new file that is not already in the cache, but could be, then the oldest file (w.r.t when the file was added to the cache) is removed from the cache, and then the new file is appended to the end of the cache.

Load Balancing Logic

I use an ordered array for each server, and organized each server into a struct. Each server object has an associated port number, problematic value, number of calls made, and number of fails made. Using the information above, I find the best available server on each request made to the proxy using a one-pass min-search. One important thing to note is that the order is determined first by the problematicism of the server, then the number of calls, and finally the number of fails. If all of those values are equivalent for any two servers, then I don't consider their particular order w.r.t each other to be of significance, and thus leave them in the order that they were originally in.