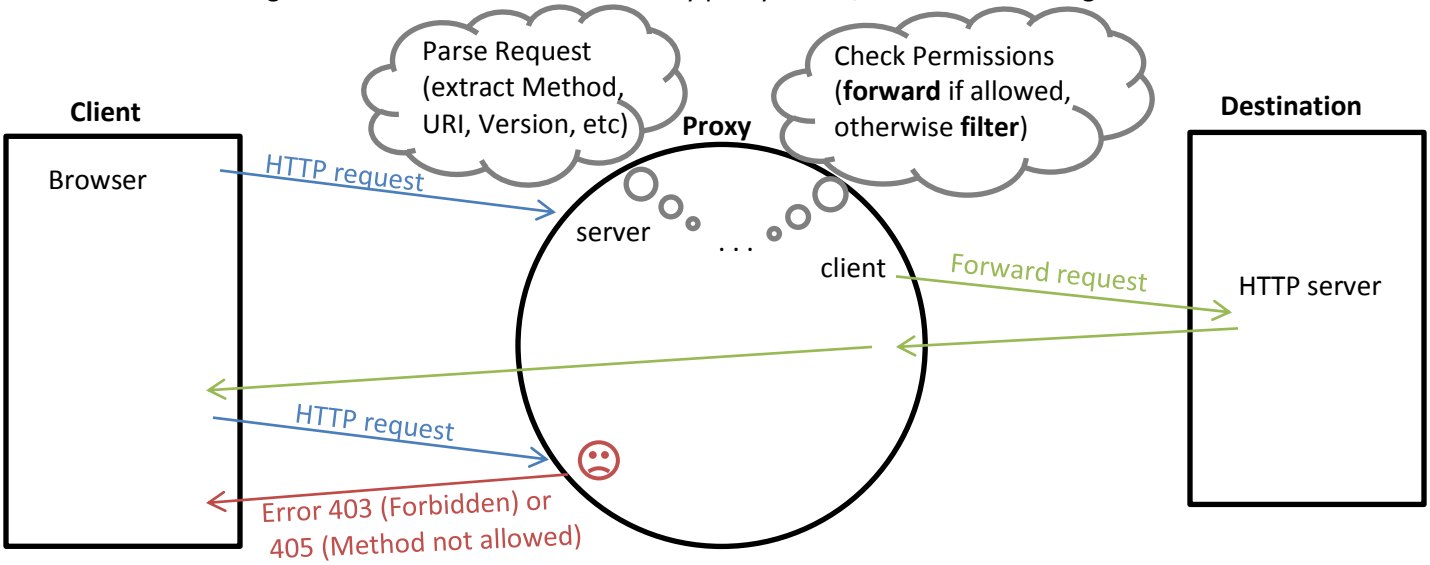
PROXY SERVER - DESIGN DOCUMENT
Ruben K Lopez | rklopez@ucsc.edu
CMPE 156: Network Programming
May 30, 2014

Proxy Server :

Filter requests based on an access control list.

To visualize how I designed the internal architecture of my proxy server, I've included a diagram:



This proxy server parses incoming HTTP requests (for relevant info) and forwards them to their destinations if allowed by the access control mechanism (i.e. by checking "permitted-sites.txt"). It should then forward any data received from the destination back to the BROWSER(client). The proxy server will Print and Log information for each request. It also returns error response codes to the BROWSER (if we Filter a request).

usage: *./proxyServer <port num> <access list>*
example: *./proxyServer 8080 permitted-sites.txt*

TCP connections on the BROWSER side are managed with our "connected" socket descriptor, which gets established with the return value of our first *accept().* This happens in the Parent process, before we start forking.

On the SERVER (or destination) side, the TCP connection is managed with our "temporary" socket descriptor, which gets created (and closed) every time we Forward a Request (and get the server's reply). This happens in the child process.

Concurrency is supported by forking.

As you can see in the diagram, the steps involved in **processing a request** (from the browser) include:

1. Reading from the Browser-side-socket
2. Parse-ing the HTTP Request (to extract the Method, URI, Version, Host, and Path).
3. Checking permissions
4. If PERMITTED, Write to Server-side-socket, Read reply from Server-side-socket, and then Write reply to Browser-side-socket.
5. If NOT PERMITTED, Write to Browser-side-socket (Error 403 or 405).

My design deals with **errors** by…

1. Checking for a -1 return code on a read() or write().
2. Using *shutdown(sock, SHUT_RDWR)* to essentially close a connection, so that we can prevent getting stuck in a read()-loop when trying to resolve a webpage (e.g. when we're trying to read the Server's reply).

Example Setup:

(Tested with Firefox Browser)

Browser: First, configure Firefox to "manually" use a proxy.
Specify the IP address(or hostname) and Port (e.g. IP: 127.0.0.1, Port: 8080)
(e.g. IP: galileo-2.soe.ucsc.edu, Port: 8080)
Then exit the browser.

Terminal: Next, run the proxy server:
./proxyServer 8080 permitted-sites.txt

Browser: Now open the browser and try connecting to these sites:
example.com
classes.soe.ucsc.edu/cmpe150/Fall99
classes.soe.ucsc.edu/cmpe110/
classes.soe.ucsc.edu/cmpe151/Spring14/index.html