PROGRES - Mini-Project 1

Sébastien Tixeuil - Sebastien.Tixeuil@lip6.fr

Exercise 1 - TCP Relay

We are trying to program in Python a relay mechanism between a client and a server using TCP protocol. A relay is a program that acts like a server towards the client, and like a client towards the server. It retransmits to the server all the data that the client would normally have transmitted to the server, and it retransmits to the client all the data that the server would normally have transmitted to the client.

1. From the code examples seen in PROGRES class for the TCP client and the TCP server, program the relay mechanism in Python using the `socket` library. The server's IP address and port number are assumed to be provided to the relay.
2. Test the relay with client and server programs using TCP, on the same machine and on different machines; make sure everything is working properly.
3. Test the relay with several clients (or several instances of the same client) that make simultaneous requests on the server (via the relay); make sure everything is working properly.

Exercice 2 - HTTP Relay

We are trying to program in Python a relay that is dedicated to the HTTP protocol. It is therefore considered that all exchanges between the client and the server (via the proxy) use the HTTP protocol. The client is therefore typically a web browser, and the server is therefore typically a web server.

1. Modify the TCP relay developed for exercise 1 so that it acts as an HTTP *cache*: the first time a URI is supplied as an argument to GET in an HTTP request, the relay retransmits the request to the server, and locally stores its response in a file; if the same URI is subsequently requested by the same client or another client, the relay sends the file that it has stored locally directly to the client without making a new request to the server.
2. Modify the TCP relay developed in Exercise 1 to act as an HTTP logger. More precisely, all client GET requests are archived in a log file (including the URI that was requested to the client), all non-empty server responses to GET requests are archived in the log file. The log file must contain enough information to perform audits: given a URI (or part of a URI), we want to be able to find the IP addresses of clients who have obtained a non-empty response from a server concerning this URI. Write a Python program to perform such an audit.
3. Modify the TCP relay to act as an HTTP censor. The relay now has a list of banned sites (supplied as an input to the proxy). If the URI requested in a client's GET request contains a link to a banned site, that link is replaced by a "Forbidden" message in the body of the response. If the URI is not banned, the client receives a page as requested. The IP addresses of clients that requested a forbidden server must be logged in a dedicated file by the proxy.
4. Test the HTTP relays made, first individually (a relay between a client and a server), then by chaining them (for example, a client is connected to an HTTP cache that is connected to an HTTP logger, which itself is connected to the server). Check that several relays connected to the same other one (as clients) still operate properly.