Designing a Controller + Broker

Sina Kamali
University of Waterloo
sinakamali@uwaterloo.ca

I. Introduction

In this document, we will go over the design details of the controller + broker segment. We will also point out how these two components can be detached if ever needed.

II. Design

In this section we will go over the overall design patters used in designing the controller and broker. An overview of the system can be seen in figure 1

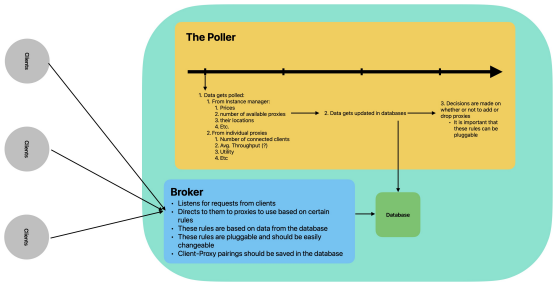

Fig. 1. overall design

A. Designing the Controller

The controller is in charge of getting information from underlying parties (such as the instance manager and proxies), and updating the database accordingly. The controller gathers these information by periodically polling data endpoints on the proxies and the instance manager. The instance manager and the controller communicate using to different endpoints:

- **A GET endpoint on the instance manager**: for the controller to query and pull information from when it's first starting to gather the initial data regarding current available proxies. The controller uses the gathered data to initialize it's database.
- **A POST endpoint on the controller**: for the instance manager (and the rejuvenation algorithm) to post (or in other words, push) new changes to available proxies, migrations, and reclaims. After the new data is posted to the controller, it rearranges it's users accordingly and gives them migration notices.

After gathering data, the controller stores them in a database that can be also accessed by the broker. This data will later be used by the broker to assign proxies to clients. This database follows a simple relational schema is seen in figure 2

Since the database was an important component of the controller, we decided to against using the famous Flask library to implement our web server, and instead chose the more complex alternative Django [1]. Django offers a great internal ORM which is fast and secure, which can help managing the database a much more streamlined process.

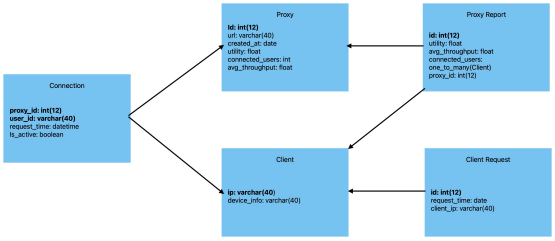

Fig. 2. database schema

B. Designing the Broker

The broker is the main component that is in charge of assigning proxies to client. Our broker is designed in a way that can easily be integrated with any proxy assignment algorithm.

In this paper we decided to use the proxy assignment algorithm introduced in this [2] work by Nasr et al. To implement this paper, we used the same formula and attributes they used in their evaluation. All the information needed for the calculation of those features are extracted from the entries in the dataset.

III. How to decouple the broker and the controller

To detach the broker and controller systems, one can easily use a third node for a database and then give access to the detached broker and controller systems.

References

[1] *Django*. Django Project. URL: https : / / www . djangoproject.com/ (visited on 01/10/2024).
[2] Milad Nasr et al. "Enemy At the Gateways: Censorship-Resilient Proxy Distribution Using Game Theory." In: *NDSS*. 2019.