An Architecture for Task and Traffic Offloading in Edge Computing via Deep Learning

Alessandro Gaballo
Supervisors: Flavio Esposito[*], Guido Marchetto[†]

1 Summary

Mobile edge computing is the idea of moving computational power to the edge of the network with the goal of avoiding network congestion and improving application performance in terms of latency. In this context, computation offloading - the process of transferring tasks to a different platform - is a common practice, especially when it comes to mobile and IoT devices, which have limited computation capabilities and power.

The offloading process is a complex problem with several factors to take into account: these factors include the server location, the offloading granularity and the request requirements in terms of computational power. Typically, especially in computer science, complex problems are addressed by defining common guidelines and strategies to face them; an example is the ISO/OSI stack for the internet. These guidelines usually take the form of an *architecture*, which in computer science defines the set of invariances and the separation of roles needed to complete a task. Unfortunatelly, at the best of our knowledge, there's no such thing for task offloading; this is why, as part of this work we prototype such architecture, defining its building blocks. These blocks include a connection manager to handle simultaneous connections with different client and servers, primitives for the communication between the parties and a SDN controller for a more advanced network management.

The main component of the prototyped architecture is what we call the *offloading logic*, responsible of manage the offloading process according to

[*]flavio.esposito@slu.edu
[†]guido.marchetto@polito.it

the logic specified by the client. The idea is to have on the offloading server - the server responsible of the offloading process - a set of offloading policies, that can be chosen and developed by the client itself, and then plugged into the system.

To support this architecture, we also develop a protocol that can be adopted in this context. A protocol requires the definition of the messages that are exchanged to make the system function: we implement this protocol using Google Protocol Buffers [1]. The implemented protocol addresses some of the problems aforementioned: first of all, it allows the client to specify the task requirements, in terms of CPU, memory footprint and latency, with this last component being particularly important in critical scenarios where latency is crucial. Secondly, the protocol supports the specification of the location of the task to be performed, by including the executable (JAR, EGG) in the request payload or by specifying the remote location of the task with a unique identifier, for server-less computing. Finally, it is possible to specify the criterion upon which the offloading process should be based.

With these messages, a typical offloading flow would be the following:

1. the client (mobile edge device) sends the offloading request to the offloading server
2. the offloading server acknowledges and parse the request
3. if necessary, the offloading server communicates with the SDN controller to install on the network nodes the flow required by the specified policy
4. the offloading server forwards the request to the edge servers
5. the edge servers receive the request, perform the task and send back the computation results.

[1] `https://developers.google.com/protocol-buffers/`