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

Alessandro Gaballo
Supervisors: Flavio Esposito (Saint Louis University)*, Guido Marchetto†

Extended Abstract

Mobile edge computing is based on the idea of moving computational power to the edge of the network with the goal improving application performance or to augment mobile node with the computational capability of a cloud. In this context, computation offloading — the process of transferring tasks to a different platform — is a common practice, especially to empower Internet of Things (IoT) devices, which have limited computation capabilities and often run on low-power requirements. The offloading process is challenging; examples of such challenges include making optimal decisions on the physical node or link location that hosts the offloaded tasks or traffic), the offloading granularity (e.g., offload an entire task, or merely a subset of it) and the request requirements in terms of computational power.

Despite several solutions proposed for offloading, to the best of our knowledge, an offloading architecture that describes the set of necessary and sufficient invariances for offloading does not yet exist. To this end, we design and prototype a policy-based architecture for offloading of both mobile tasks and edge traffic. Our architecture 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 our architecture is the offloading logic block, responsible for managing both task and traffic offloading according to the logic specified by the client. The idea is to have on the offloading server a set of policies that can be chosen and developed by the client itself.

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

In the first part of the thesis, we focus on defining such architecture and the task offloading protocol. In the second part of the thesis instead we focus on comparing traffic offloading strategies, and we introduce the novel contribution of a novel application of a Deep Learning method called Long Short-Term Memory (LSTM), used to overwrite classical routing protocols.

In our architecture, our protocol messages are defined using the Google Protocol Buffers abstract syntax notation [1]. The protocol helps clients to specify task requirements, such as, CPU, memory footprint and latency. The protocol supports also specification of the geolocation where we wish to perform the task. Finally, executable files, such as (Java JAR or Python EGG) can be piggybacked in the request as payload; Finally, it is possible to specify the criterion, i.e., the policy upon which the offloading process should be based. With these messages, a typical offloading flow would be as follows:

1. the client (mobile edge device) sends the offloading request to the offloading server
2. the offloading server acknowledges and parses 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.

Current (traffic) offloading strategies use standard routing algorithms such as OSPF are known to be performance unaware. OSPF for example considers only the nominal interface speed and according to it, computes the shortest paths for all the destinations using a method based on Dijkstra's algorithm. At the edge of the network, when short latency is often a requirement, especially in critical scenarios such as natural disasters where the infrastructure is unreliable or unavailable, using traditional routing policies could result in poor application performance in terms of latency and throughput.

For this reason, in the second part of this thesis, we design and implement a performance-aware traffic offloading i.e., routing strategy, to be used as

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

offloading logic in the architecture described earlier, but easily extendable to other scenarios. More specifically, we leverage deep learning theory to develop a path prediction system capable of changing its output according to the network conditions.

The idea is to introduce awareness needed to optimize routing decisions, by using traffic patterns in the prediction process: in this case we use the packet counter on the routers as performance indicator. The underlying intuition is that packet distribution reflects the network conditions, with a high packet counter being an indicator of an "overloaded" router and viceversa. For the prediction task we use Deep Learning, for its proven capabilities of solving numerous inference tasks.

In particular, we use a recent improvement of traditional Recurrent Neural Network called Long Short-Term Memory (LSTM). LSTM is capable of memorizing temporal data patterns with the goal of learning how traffic patterns evolve and route accordingly. LSTM is a supervised learning approach, requiring therefore something to learn from; since we are solving a routing problem, we use OSPF as a ground truth in the training process. We correlate the routing information with traffic patterns. The way OSPF works is by simply computing the shortest path according to a cost function, in this case the network interface speed. Our insight is that the traffic distribution on routers change according to the configuration that OSPF has converged to. The idea is to learn from as many OSPF different configurations as possible, so that our system could learn and predict different path depending on different network conditions.

Our LSTM-based approach show a good learning capability, achieving an average accuracy of more than 98%. This means that our neural network is capable of learning and emulating the OSPF behavior 98% of the time. To evaluate the impact of our LSTM solution, we compare our system to OSPF in case of lossy link that dropped packets with a certain rate. Our system shows the capability of avoiding, in certain cases, the lossy links, achieving up to 10% less retransmissions with respect to OSPF under the same network conditions. Our results suggest that deep learning could be useful to solve traffic optimization problems.