

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

Alessandro Gaballo

February 2, 2018

Supervisor: Flavio Esposito - Saint Louis University



How many times have you delegated a task?

WHO CAN HELP ME?

WHO IS FASTER?



How should tasks be offloaded?

- MEC offloading orchestration is a complex problem
- There is not a unified abstraction about offloading mechanisms
- Current strategies are performance unaware
- SDN & KDN could help

*MEC = Mobile Edge Computing, SDN = Software Defined Networking,
KDN = Knowledge Defined Networking*

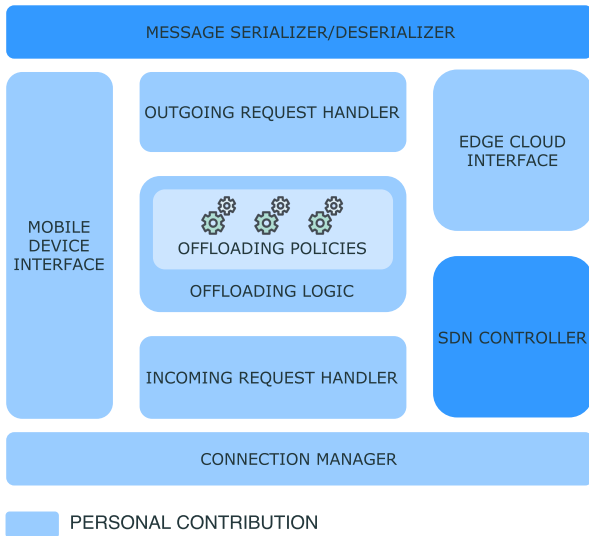
Offloading architecture

- Architecture abstraction definition to address the complexity problem
- Knowledge Plane to support network management
- Performance aware traffic steering

- Offloading Architecture
- Results
- Limitations

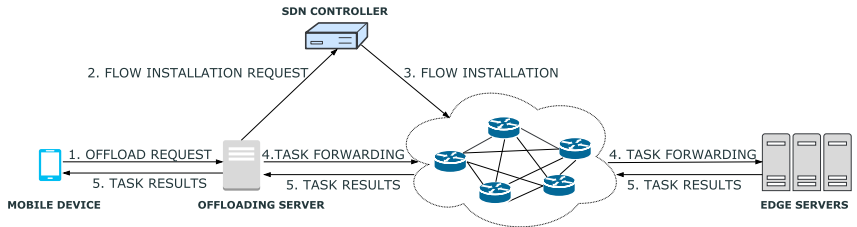
- Offloading Architecture
 - Abstraction
 - Task Offloading Protocol
 - Path Prediction via LSTM
- Results
- Limitations

Offloading architecture



- Offloading Architecture
 - Abstraction
 - Task Offloading Protocol
 - Path Prediction via LSTM
- Results
- Limitations

Task offloading protocol



The protocol allows the client to specify:

- task requirements such as CPU, memory and latency
- offloading logic (e.g. LSTM path predictor, nearest server)

Talk overview

- Offloading Architecture
 - Abstraction
 - Task Offloading Protocol
 - Path Prediction via LSTM
 - Overview
 - Dataset
 - LSTM Architecture
- Results
- Limitations

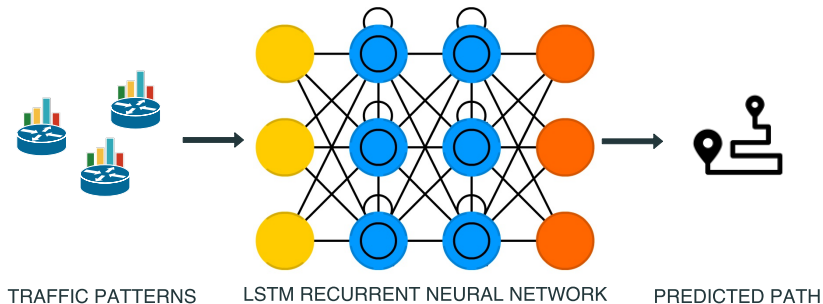
Machine learning is a powerful tool for inference tasks



IDEA: Routing problem as inference problem

How to determine the best path?

Use traffic pattern as performance indicator.
Perform path prediction with machine learning.



Long Short-Term Memory (LSTM)

LSTM is an evolution of recurrent neural networks (RNNs) capable of memorizing data temporal patterns

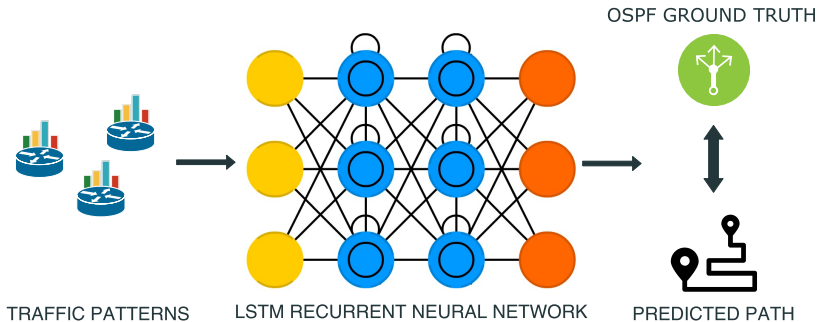
Objective:

Learn how traffic patterns evolve and route accordingly

Learning from who?

LSTM RNNs are a supervised learning method, they require data to learn from.

We use OSPF routing decisions as a ground truth.



Problem formulation

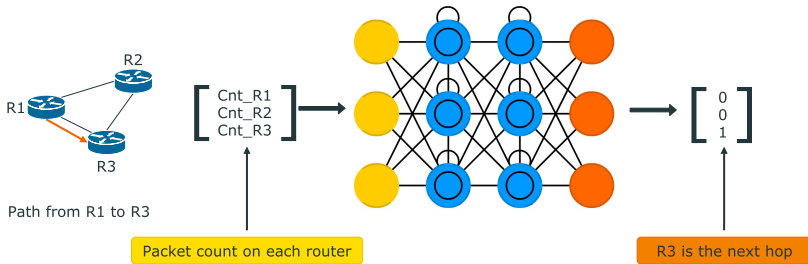
To use a LSTM we must define the model input and output.

Input:

Incoming packets count
on each router

Output:

One-hot encoded vector
with the next hop in the path



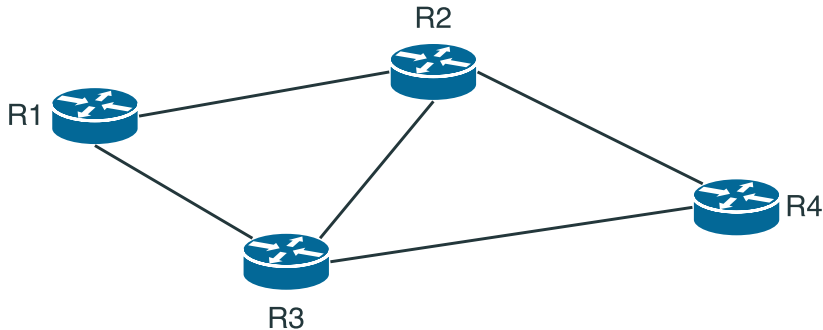
Path prediction process

Training a single model for all the targets in the topology is not feasible.

SOLUTION: train a model for all the source-destination pairs in the network.

To compute the whole path we iteratively use the model of the predicted next hop until the destination is reached.

Example: computing the path from R1 to R4



Step 1 - model = R1-R4 → next hop = R2

Step 2 - model = R2-R4 → next hop = R3

Step 3 - model = R3-R4 → next hop = R4

Computed path: R1 - R2 - R3 - R4

Talk overview

- Offloading Architecture
 - Abstraction
 - Task Offloading Protocol
 - Path Prediction via LSTM
 - Overview
 - Dataset
 - LSTM Architecture
- Results
- Limitations

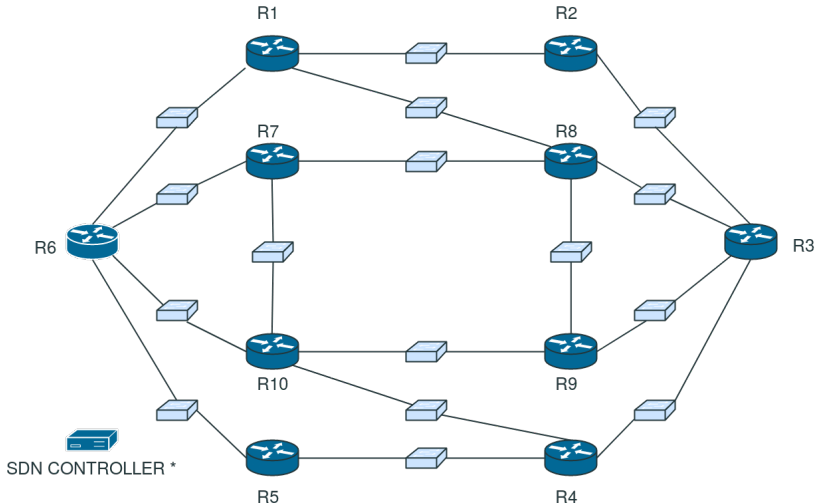
To train our model we need:

- network topology
- routing algorithm
- packet counter

We could not find any public dataset suited to our needs so we create our own.

Network topology

We create this topology using MiniNeXt



* All switches are connected to the SDN controller

Routing algorithm

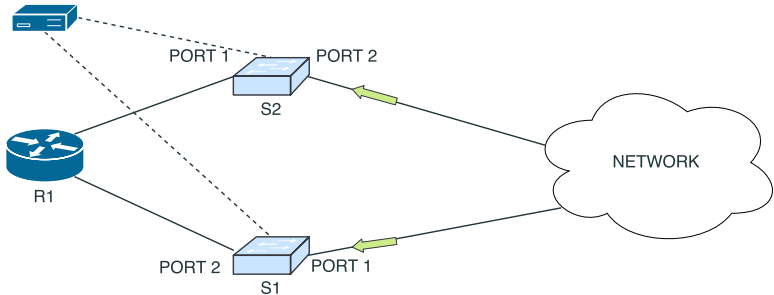
To run routing algorithms on MiniNeXt nodes we use Quagga. Quagga is a routing suite providing different routing algorithms (e.g OSPF, IS-IS, RIP).

We choose OSPF because of its wide adoption as iBGP. We also take in consideration Equal-Cost Multi Path (ECMP) to compare the performance.


Packet counter

The SDN controller –Ryu– is responsible of retrieving the packet count.

SDN CONTROLLER



Dataset generation steps

- Initialize the topology with different link speed
 - Simulate traffic for a certain duration
 - Save packet count and routing tables
 - Stop the traffic simulation and tear down the network
- 

- Offloading Architecture
 - Abstraction
 - Task Offloading Protocol
 - Path Prediction via LSTM
 - Overview
 - Dataset
 - LSTM Architecture
- Results
- Limitations

How to decide the LSTM structure?

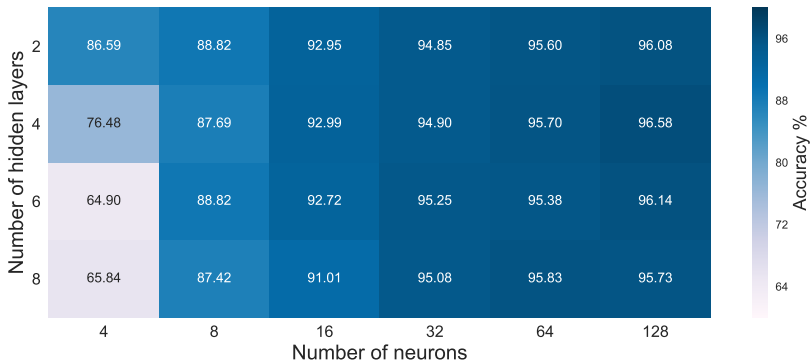
The number of layers and neurons cannot be decided a priori. Cross validation is a powerful technique to determine the model parameters.

We test 24 configurations and average the results of 10 runs.

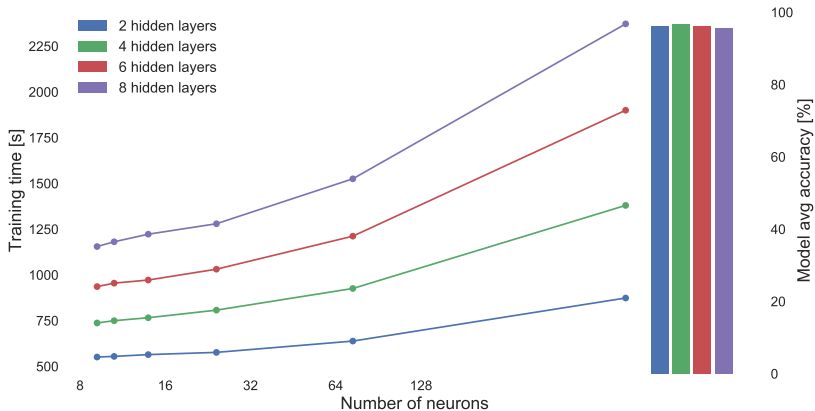


Accuracy: more neurons or more layers?

Neurons affect performance more than hidden layers

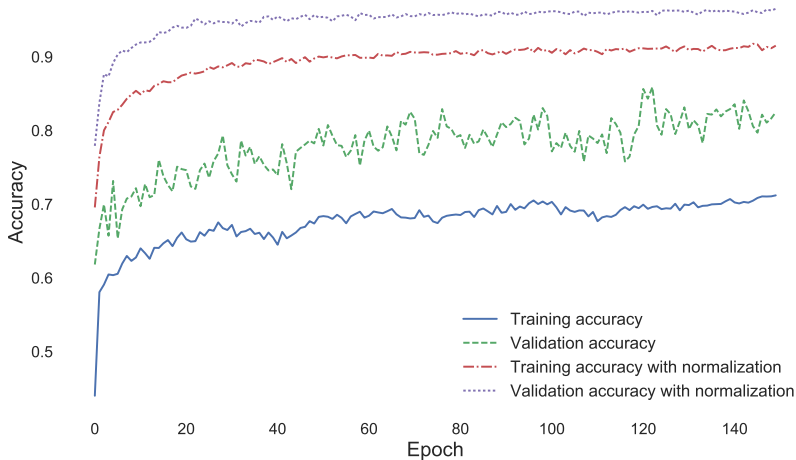


Finding the compromise: training time



* Accuracy shown only for models with 128 neurons

Tuning the network: input normalization



* Accuracy shown only for models with 128 neurons

- Offloading Architecture
- Results
- Limitations

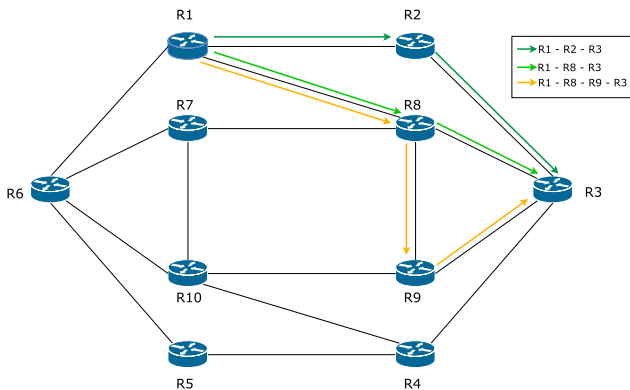
The system is able to emulate OSPF

We test the system's ability to behave like OSPF by averaging the performance of all the models on the test set.

The LSTM achieves an average accuracy of **98.71%**.

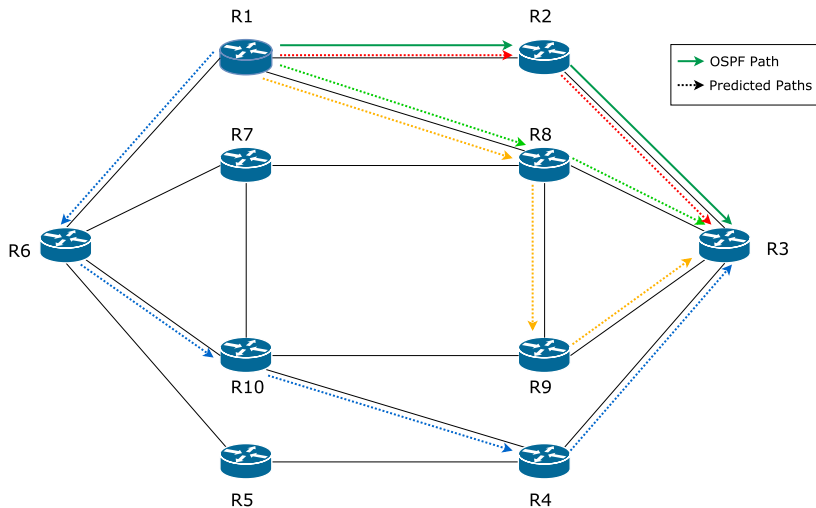
Is the system performance aware?

We select a target and analyze, through an example, how our system behaves differently from OSPF in case of link loss.



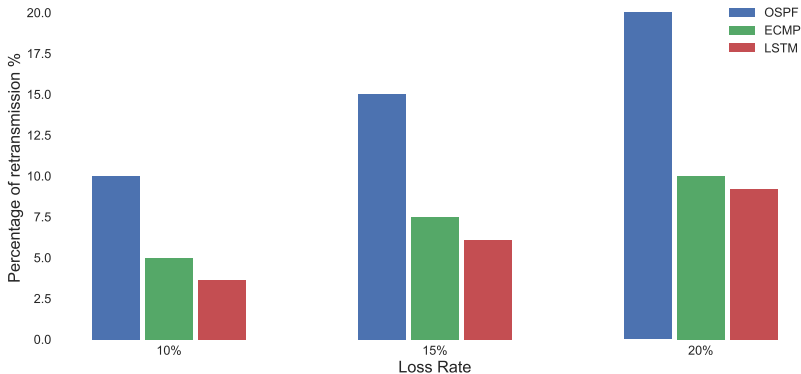
The system exhibits a dynamic behavior

The LSTM path predictor is able to suggest multiple paths



LSTMs outrun current approaches in terms of retransmission

In case of malfunctioning links, our system has a lower retransmission percentage than traditional routing



ECMP = Equal-Cost Multi Path, OSPF = Open Shortest-Path First, LSTM = Long Short-Term Memory

- Offloading Architecture
- Results
- Limitations

Limitations

Testbed:

- The functionalities of Mininet are limited
- The efficiency of the prediction system starts to decrease if the link loss is too high

Computation:

- The training process is slow
- The number of models to train is big
- The controller needs GBs of memory to handle the trained models

The current results are encouraging, however, to further investigate the problem we aim to:

- get rid of Mininet emulation environment constraints
- test the method on larger networks (e.g using GENI)
- run the testbed on a more scalable platform (e.g using GPU)
- explore more machine learning techniques (e.g reinforcement learning)

Take home messages

- Mobile edge computing is a complex problem with no unified vision
- We prototyped an architecture for MEC offloading orchestration
- We developed a machine learning-based, performance aware routing strategy that improves on classic iBGP mechanisms

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

Alessandro Gaballo

February 2, 2018

Supervisor: Flavio Esposito - Saint Louis University

