Architectures and Distributed Algorithms for a Knowledge-Plane Supporting Edge Computing Applications

Alessandro Gaballo

EURECOM - Data Science and Engineering

Professional Thesis Report (confidential)

Research carried out at Saint Louis University under the supervision of Professor Flavio Esposito

January 2018

This thesis is dedicated to
someone
for some special reason

Acknowledgements

plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle.

Abstract

The spreading of smart mobile devices and the development of IoT has brought computation power in everyone's pocket, allowing people to perform simple task with their smartphones. There are some tasks however, which are more computationally expensive and therefore cannot be performed on a mobile device; for these type of tasks computation offloading represents a valid solution. Computation offloading is the process of transferring computation tasks to another platform for reasons such as limited computation resources and energy saving, which are evident in mobile and IoT devices. Typically, tasks are offloaded to powerful machines in the cloud, however for time sensitive applications the cloud could be too far, this is why the edge computing paradigm is spreading. Edge computing moves the processing from the core to the edge of the network, improving user experience by reducing the perceived latency. One way to reduce latency is to chose a proper path to the destination; paths advertised by the routing algorithms are not always the best because they don't take into account the network conditions. Our conjecture is that cooperative routing-based methods will steer edge traffic with (statistically) better quality, i.e., shorter end-to-end delays than reaction-based methods, such as Cartographer [8]. A way to perform path prediction is to use machine learning techniques, which in the last few years have spreaded and changed the way software is made [4]. Machine learning is the field of computer science that gives computers the ability to learn without being explicitly programmed [7], by extracting patterns from data. Recently, many goal have been achieved with the use of machine learning in the field of computer vision, gaming, text and speech recognition, but not much has been done in networking: this work is a first approach to machine learning in networking for path prediction. In this project, we present a system architecture to manage the offloading process of a task at the edge of the network: this architecture has a modular design so that the offloading policies could be easily plugged into the system. As part of the offloading

policies we implement a path prediction system based on a deep neural network, with the aim of reducing the latency with respect to the standard routing policies.

Abstract

The spreading of smart mobile devices and the development of IoT has brought computation power in everyone's pocket, allowing people to perform simple task with their smartphones. There are some tasks however, which are more computationally expensive and therefore cannot be performed on a mobile device; for these type of tasks computation offloading represents a valid solution. Computation offloading is the process of transferring computation tasks to another platform for reasons such as limited computation resources and energy saving, which are evident in mobile and IoT devices. Typically, tasks are offloaded to powerful machines in the cloud, however for time sensitive applications the cloud could be too far, this is why the edge computing paradigm is spreading. Edge computing moves the processing from the core to the edge of the network, improving user experience by reducing the perceived latency. One way to reduce latency is to chose a proper path to the destination; paths advertised by the routing algorithms are not always the best because they don't take into account the network conditions. Our conjecture is that cooperative routing-based methods will steer edge traffic with (statistically) better quality, i.e., shorter end-to-end delays than reaction-based methods, such as Cartographer [8]. A way to perform path prediction is to use machine learning techniques, which in the last few years have spreaded and changed the way software is made [4]. Machine learning is the field of computer science that gives computers the ability to learn without being explicitly programmed [7], by extracting patterns from data. Recently, many goal have been achieved with the use of machine learning in the field of computer vision, gaming, text and speech recognition, but not much has been done in networking: this work is a first approach to machine learning in networking for path prediction. In this project, we present a system architecture to manage the offloading process of a task at the edge of the network: this architecture has a modular design so that the offloading policies could be easily plugged into the system. As part of the offloading

policies we implement a path prediction system based on a deep neural network, with the aim of reducing the latency with respect to the standard routing policies.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Data-intensive computing requires seamless processing power which is often not available at the network-edge but rather hosted in the cloud platforms. The huge amount of mobile and IOT devices that has become available in the past few years, is able to produce a massive quantity of data, which contributes to the very famous world of big data. The majority of these devices do not have or can not handle the computational requirements to process the data they capture and so leave to the cloud the responsability to perform the computations. This process of transferring computation tasks to another platform is called computation offloading and it is crucial to the edge devices because it results in lower processing time and energy consumption. In critical scenarios, such as natural disasters, not only computation offloading becomes necessary, but latency requirements become more strict, making paths management essential to satisfy these requisites. In this work we develop an architecture to be deployed at the edge of the network to assist the offloading process and make use of machine learning techniques to perform path management. The goal is to outperform the classic routing policies in terms of latency and througput by learning implicit patterns in network traces.

1.2 Related Work

Cyber-foraging is a highly complex problem since it requires to take into consideration multiple issues. Lewis and Lago [6] dive into those issues and present several tactics to tackle them. Tactics are divided in *functional* and *non-functional*, with the former identifying the elements that are necessary to meet Cyber-foraging requirements and the latter the ones that are architecture specific. Functional tactics cover computation

offload, data staging, surrogate provisioning and discovery; non-functional tactics deal with resource optimization, fault tolerance, scalability and security. They describe a simple architecture which include an *Offload Client* running on the edge device and an *Offload Server* running on the surrogate (cloud or local servers).

Wang et. al [9] report the state-of-the-art efforts in mobile offloading. More than ten architectures are described, each one in a different possible offloading situation. In particular the reported works face the single/multiple servers as offloading destination scenario, the online/offline methods for server load balancing, the devices mobility support, the static/dynamic offloading partitioning and the partitioning granularity.

In the last few years machine learning is being used to solve various challenges but it's not being widely adopted in networking problems; however, many people are trying to change this tendency. Malmos [3] is a mobile offloading scheduler that uses machine learning techniques to decide whether mobile computations should be offloaded to external resources or executed locally. A machine learning classifier is used to mark tasks for local or remote execution based on application and network information. To handle the dynamics of the network an online training mechanism is used so that the system is able to adapt to the network conditions. Malmos has proven to have higher scheduling performances than static policies under various network conditions.

In [2] machine learning is used for computation offloading in mobile edge networks. Regression is used to predict the energy consumption during the offloading process as well as the time to for the access point to receive the payload. Available servers are represented in a feature space according to a hyper-profile, then K-NN (K-nearest neighbor) is used to determine the closest server based on metrics related to the hyper-profile. By using K-NN, if an application needs to partition a task into multiple parts onto multiple servers, one should simply vary the value of K.

Kato et. al [5] show the use of deep learning techniques for network traffic control. A DNN (deep neural network) is used for the prediction of a router next hop. The decision is based on the number of inbound packets in a router at a given time and OSPF paths are used for training. By combining the next hop decision for each router the system is able to predict the whole path from source to destination. Results show that the system is able to improve performances in terms of signaling overhead, throughput and average per hop delay with respect to the classic OSPF algorithm.

Another use of machine learning in networking is described in [1]. Bui, Zhu, Pescap & Botta designed a system for a long horizon end-to-end delay forecast. The idea is to use measured samples of end-to-end delays to create a model for long horizon

forecast. Considering the set of samples as a discrete-time signal, wavelet transform is applied which results in two groups of coefficient. A NN (neural network) and a K-NN classifier are then used to predict the coefficients. Once again ML techniques seem to provide good results when applied to networking.

1.3 Structure

Chapter 2 contains a brief background about machine learning general notions, and the main techniques used in this project.

Chapter 3 describes the personal contribution to this project, including the detail about the architecture and the implementation of the path predictor.

Chapter 4 shows considerations and results of the implemented system.

Chapter 5 presents comments about the outcome of the project and possible future implementations.

Chapter 2

Background

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Chapter 3

Contribution

Bla

Chapter 4

Conclusions and Results

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Appendix A

Sample Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Appendix B

Sample Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bibliography

[1] V. Bui, W. Zhu, A. Pescape, and A. Botta. Long horizon end-to-end delay forecasts: A multi-step-ahead hybrid approach. In *2007 12th IEEE Symposium on Computers and Communications*, pages 825–832, July 2007.

[2] Andrew Crutcher, Caleb Kochy, Kyle Colemanz, Jon Patmanx, Flavio Esposito, and Prasad Calyamx. Under submission: Hyperprofile-based computation offloading formobile edge networks. 2017.

[3] H. Eom, R. Figueiredo, H. Cai, Y. Zhang, and G. Huang. Malmos: Machine learning-based mobile offloading scheduler with online training. In *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pages 51–60, March 2015.

[4] Andrej Karpathy. Software 2.0. https://medium.com/@karpathy/software-2-0-a64152b37c35.

[5] N. Kato, Z. M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani. The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective. *IEEE Wireless Communications*, 24(3):146–153, 2017.

[6] Grace A. Lewis and Patricia Lago. A catalog of architectural tactics for cyber-foraging. In *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, QoSA '15, pages 53–62, New York, NY, USA, 2015. ACM.

[7] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, July 1959.

[8] Patrick Shuff. Building a billion user load balancer. Dublin, 2015. USENIX Association.

[9] Jianyu Wang, Jianli Pan, Flavio Esposito, Prasad Calyam, Zhicheng Yang, and Prasant Mohapatra. Under submission: Edge cloud offloading algorithms: Issues, methods and perspectives. 2017.