Self-Driving Networks: Overlay Routing Optimization

Deliverable 6: Final deliverable
Spring 2017

Patricia Sampedro Garcia

Director: Albert Cabellos

GEP Tutor: Ferran Sabate
Gestio de Projectes (Gep)

Facultat d'Informatica de Barcelona (FIB)
Universitat Politecnica de Catalunya (UPC) - BarcelonaTech

27th March 2017

Contents

1 Introduction **4**

2 Background **4**
- 2.1 Software Defined Networking: SDN 4
- 2.2 Overlay Network 5
- 2.3 Reinforcement Learning:RL 6

3 State-of-the-art **6**
- 3.1 A Knowledge Plane for the Internet 6
- 3.2 A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks 7
- 3.3 Knowledge-Defined Networking: KDN 8
- 3.4 Uses of previous results 10

4 Context **10**
- 4.1 Areas of interest 10
  - 4.1.1 Prototype 11
  - 4.1.2 Routing Optimization: 11
- 4.2 Stakeholders 12
  - 4.2.1 Audience 12
  - 4.2.2 Users 13
  - 4.2.3 Beneficiaries 13

5 Scope of the project **13**
- 5.1 Scope 13
- 5.2 Methodology and rigor 13
  - 5.2.1 Development tools 14
  - 5.2.2 Tools to monitor evolution 14
  - 5.2.3 Validation of results 14
- 5.3 Obstacles and risks on the project 15
  - 5.3.1 Optimizer 15
  - 5.3.2 Prototype 15

6 Schedule **15**
- 6.1 Estimated project duration 15
- 6.2 Considerations 15

7 Project planning **16**
- 7.1 Life cycle 16
- 7.2 Main Tasks 17
  - 7.2.1 Initial work 17
  - 7.2.2 Optimizer 18
  - 7.2.3 Prototype 19

7.2.4 Documentation 19
7.3 Gantt Chart 20

8 Action plan **20**

9 Cost estimation **21**
9.1 Direct Costs 21
9.1.1 Initial Work 22
9.1.2 Optimizer 23
9.1.3 Prototype 24
9.1.4 Documentation 25
9.1.5 Total Direct Cost 25
9.2 Indirect Costs 26
9.3 Direct And Indirect Costs 26
9.4 Contingency Costs 26
9.5 Incidental Costs 27
9.6 Total Cost and Control Management 27

10 Sustainability Report **28**
10.1 Environmental 28
10.2 Economic 28
10.3 Social 29
10.4 Sustainability Matrix 29

List of Figures

1 SDN Layers 5
2 Overlay Network 6
3 Proposed architecture 7
4 KDN planes 8
5 KDN operational loop 9
6 Architecture of our project using KDN paradigm 10
7 Architecture of our project using KDN paradigm 11
8 Architecture of our project using KDN paradigm and RL . . 12
9 Gantt chart. It is generated using *TeamGantt* (8) 20

List of Tables

1 Direct Costs 22
2 Direct Costs 23
3 Direct Costs 24
4 Direct Costs 25

5 Total Direct Costs 25
6 Indirect Costs 26
7 Direct and Indirect Costs 26
8 Contingency Costs 27
9 Incidental Costs 27
10 Total Costs 28
11 Sustainability Matrix 29

1 Introduction

One of the primary goals of the field of networking is achieved a mechanism that selects the routing which has the minimum delay for a specific traffic. Recently, significant progress has been made by combining a centrally managed architecture, software-defined networking (SDN), with a centralized optimizer which is controlled by an operator (Hartert, 2015)[1].

However, while this architecture needs a human interaction, Mestres defined on 'Knowledge Defined Networking'(KDN)[2] a paradigm that combines SDN with machine learning to provide an automated network.

The aim purpose of this project is to make a prototype of Self-driving routing on an overlay network using the paradigm of the KDN.

In order to develop the first prototype of a network that learns and optimizes routing automatically we consider three grand challenges:

1. Seek optimization techniques
2. Develop a prototype, using the optimizer module that we created on 1), with existing open-source mechanism
3. Compare the results with state-of-the-art mechanisms.

2 Background

In this section, we will define the basics concepts and paradigms that are needed in order to understand this project.

2.1 Software Defined Networking: SDN

Software-defined networking is an architecture that separates the logic control to the physical devices, allowing the centralize control of the network. This paradigm is divided into three layers:

- **SDN Applications** are programs that communicate behaviors and needed resources with the SDN Controller via application programming interface (APIs). They have an abstract view of the network, including statistics and events.
- **SDN Controller** is a logical entity that receives instructions or requirements from the SDN Application layer and forwards them to the networking components. On the other hand, this entity also obtains

information about the network from the hardware devices and provide it to the SDN Application.

- **SDN Networking Devices**. The aim of this layer is to control the forwarding and data processing capabilities for the network.

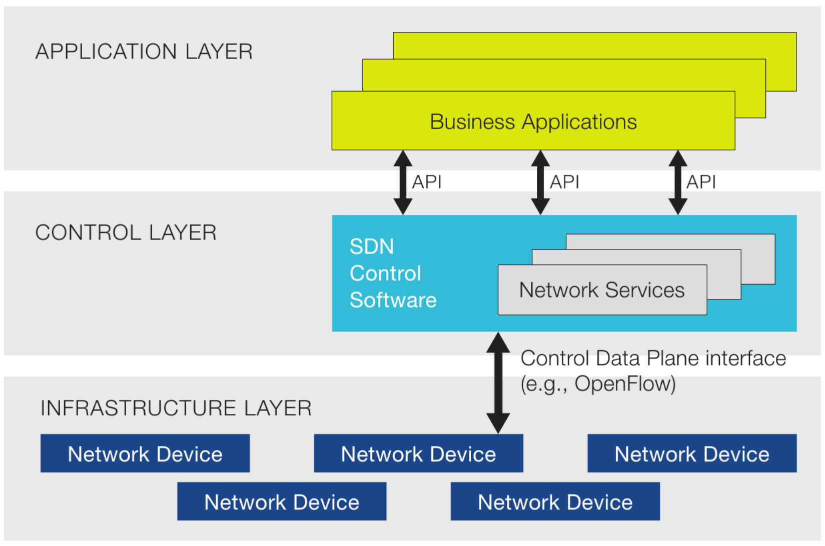


Figure 1: SDN Layers
Source: Website: SdxCentral. Inside SDN architecture (6)

In this project, we will use an open source Software-Defined Networking project called OpenDayLight, that is a collaborative and industry-supported framework.

2.2 Overlay Network

An overlay network is a telecommunication network which is built on top of another network, called underlay network. The nodes that below to the overlay network are connected by a virtual or logical link to the underlay.

Using this method, it is able to create layers of the network, it can be used to provide different and discrete virtualized networks on top of the underlying network. The benefits are creating a new application with a higher level of security.

We used this approach for two reasons. First of all, Overlay network give us a simplified version of our network. Secondly, in some real situation, SDN doesn't have all the information about the network, so it is a better approach to the industry problems.

4

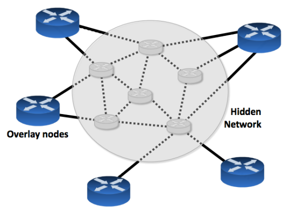


Figure 2: Overlay Network
Source: Website: SdxCentral. What is overlay networking? (5)

2.3 Reinforcement Learning:RL

Reinforcement learning is a semi-supervised learning where the agent learns with the interaction of the environment. Models on reinforcement Learning are used to be Markov Decision Problem, where the agent can visit a finite number of states, and in every state, it received a numerical reward. Every state can be reached selecting an action, according to a policy, from another particular state. The aim of RL has selected the actions that maximized the expected accumulative reward of the agent.

3 State-of-the-art

In this section, we will summarize the 3 papers that had more influence on the topic's election: 'Knowledge Plane for the Internet' by D.Clark [2], 'Knowledge-Defined Networking' by Mestre.[1] and 'A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks' by Hartert[3]. On the last section, we will describe the main gaps that are still to be covered in the literature which justify the need for carrying out this project.

3.1 A Knowledge Plane for the Internet

D.Clark [1] proposed a new construction called Knowledge Plane (KP) that relies on Cognitive System to manage the network. The main advantages of this proposal are: fault diagnosis and mitigation, automatic (re)configuration and optimize the data network.

During this article, D.Clark presents an architecture and multiple uses cases for that paradigm. However this is just a theoretical approach due to, it is designed for distributed system, which is contradictory of the application of

Machine Learning. This contradiction lies on the fact that Machine Learning requires a global view of the problem but using distributed systems it can only achieve a partial view, so every node only has his own view and a limited control on the network.

On section 3.3, we will explain the integration of this KP using SDN, which provides a centralize point of view.

3.2 A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks

In this paper, it is purposed a new architecture based on overlay-underlay network and a centralized optimizer called DEFO [3], which translate the goals written in a high-level programming language into compliant network configurations.

We defined this overlay-underlay paradigm in the following way:

1. Underlay: On this physical layer, devices like routers, switches.. lies in order to ensure network-wide reachability
2. Overlay: It is called the connectivity layer where an operator defines a path for specific flow. On the most probable scenarios once in the network lifetime.

Additionally, DEFO controls the optimization layer following high-level goals from operators written on a small Scala DSL.DEFO has been proved to have an extraordinary performance in comparison with other state-of-art, like Segment Routing on top of IS-IS, RSVP-TE, or OpenFlow, nevertheless, it is not able to make a decision without the human interaction. On the following point, it will define a paradigm that allows the network be optimized and control by itself.

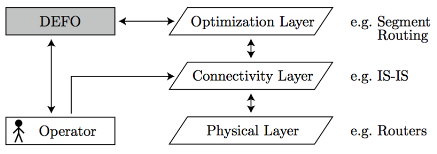


Figure 1: Proposed architecture.

Figure 3: Proposed architecture
Source: Scientific paper: ’A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks ’ (3)

3.3 Knowledge-Defined Networking: KDN

Knowledge-Define Networking[2] is a paradigm for control and operate a network using artificial intelligence. In what follows we will first define the architecture and then we will describe how operates.

We will explain the architecture that it is shown in Fig.3 in more details:

1. **Data Plane**. Elements on this layer are network devices that operate without knowing the other layers on the architecture and rely on the other planes to populate their forwarding tables and update their configuration.

2. **Control Plane** Its main role is to handle the matching and processing rules on the data plane.In SDN this is usually a job of the SDN controller.

3. **Management Plane**.It is responsible for defining the network topology and updating the configuration of network devices. On SDN paradigm this role is also assigned to SDN controller. Management Plane also has the function of monitoring the network, collecting telemetry information and keeping a historical record of the network state from the data plane in order to provide critical network analytics.

4. **Knowledge Plane**. This is the plane that was originally proposed by Clark[1](see point 3.1). In KDN paradigm, KP uses the control and management planes to achieve a complete view and control over the network. Also, it uses the network analytic stored on the management plane to learn about it, so as to create knowledge and take the decision (automatically or using an operator).

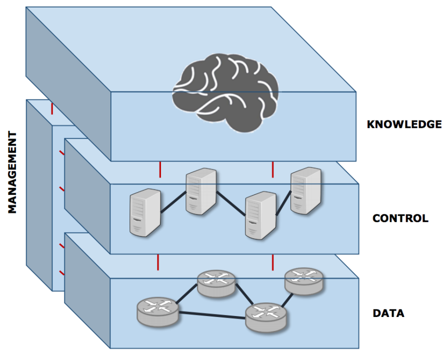


Figure 4: KDN planes
Source: Scientific paper: Knowledge-Defined Networking (2)

On Fig. 4 it can see the steps on the main KDN. On the following lines we will explain it, step by step:

1. **Forwarding Elements SDN Controller → Analytics Platform**. Analytics Platform used Forwarding Elements and SDN controller in order to have a complete view of the network and management state. Also, it monitors the data plane on real-time.
2. **Analytics Platform → Machine Learning**. Machine learning uses the information on Analytics Platform for the sake of creating Knowledge.
3. **Machine Learning → Intent Language**.Machine learning is being able to make the decision on how to act on the network. These decisions are forwarding using and Intent-driven language, a declarative language.
4. **Intent Language → SDN controller**. Automatic system and network agents use Intent Language to express their decision.Once SDN received the declarative primitive, renders it into specific imperative control action.
5. **SDN controller → Forwarding Elements** The imperative control action, that we created in the last point are pushed to the forwarding elements in order to program the data plane.

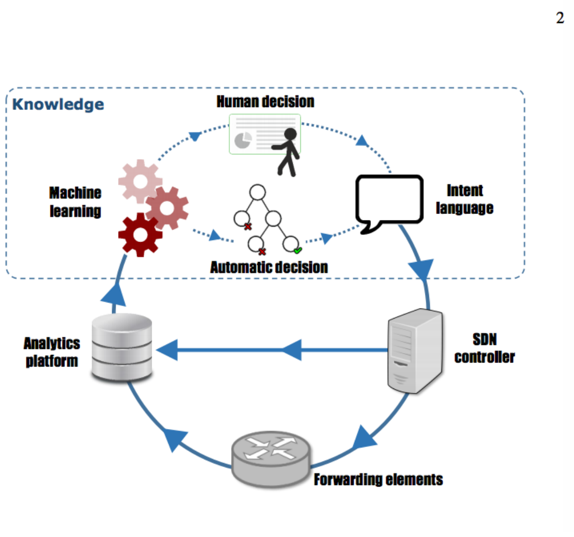


Figure 5: KDN operational loop
Source: Scientific paper: Knowledge-Defined Networking (2)

3.4 Uses of previous results

On one hand, KDN is the first time in the literature that presents a feasible architecture for an automatic network, however, it has never been prototyped.

On the other hand, DEFO is the optimizer controller with the best performance results but is facing this problem from another point of view where it is needed the human interaction.

Our aim goal is trying to create an optimize module, that we will compare with DEFO, but without the need of an operator, using the paradigm of KDN for the automatic network.

Our architecture incorporates an optimized module to the Knowledge plane, using the following schema:

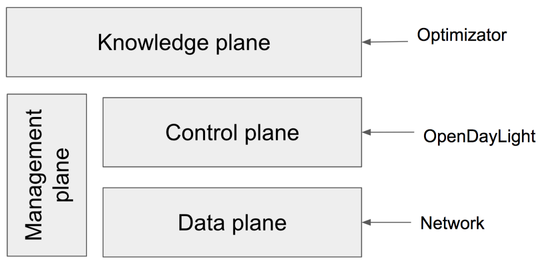


Figure 6: Architecture of our project using KDN paradigm
Source: Personal compilation

4 Context

In this section, we are going to focused on the areas of interest for this project. We will also explain who are the stakeholders that are going to be affected by this project

4.1 Areas of interest

In this project we have two main areas of interest: the creation of an optimize routing program and the integration of this model with KDN, in order to verify our results.

4.1.1 Prototype

This objective is the most technical one, so we will have to integrate the optimizer module with the open-source SDN called OpenDaylight. Moreover, we will integrate a traffic generator and a network simulator. For the network simulator we will be using OMNET++ and for the traffic generator, we will use TREX. TREX is an open-source traffic-generator created and maintenance by Cisco. The architecture of our prototype will be:

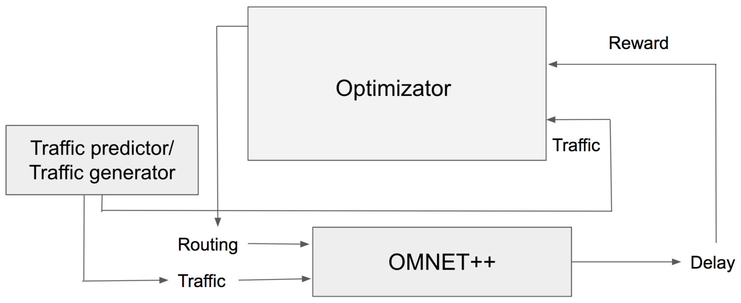


Figure 7: Architecture of our project using KDN paradigm
Source: Personal compilation

4.1.2 Routing Optimization:

Nowadays, routing optimization is still one of the problems without solutions.

The model that we are using under Overlay-Underlay paradigm, consist in:

- We only have control over the overlay network, that contains N nodes
- Every node sends and receives traffic
- Every node on the overlay is connected with the underlay network only with 2 links.

In this project, we have 3 main variables

- Routing R: Matrix NxN where it is specified α for each pair of the node. We defined α as the percentage of routing that is sent for the first link that connects this node with the underlay.
- Traffic T: Matrix NxN where we establish the quantity of traffic per each pair of nodes.

- Delay D: Matrix NxN for each pair of nodes.

In order to obtain the routing that minimizes the delay we will be using Reinforcement Learning, following this schema:

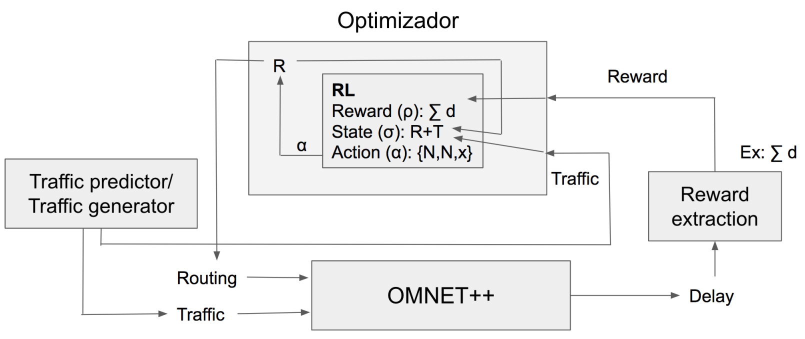


Figure 8: Architecture of our project using KDN paradigm and RL
Source: Personal compilation

On Fig.7 it is showed that our optimizer uses the delay and the traffic in order to learn which is the best router topology. The action space that we are using is all the possible combinations per Routing and Traffic and the actions are: The individual item on row I and column J increase or decrease in Y units, so it is a continuous action space. Our main goal is build the program that could solve this problem using that representation.

4.2 Stakeholders

On the following sections, we will explain the stakeholders of our project.

4.2.1 Audience

The consumers that our application plans to target is people who work and investigates on networking field. The goal of this project is converting KDN in an industry-supported paradigm, due to, the main target is the researchers on industry.

As our prototype is a use case of KDN, the most likely people to purchase this product is the writers on that article: professors of Berkeley and UPC as well as researchers working on Cisco and HP.

4.2.2 Users

The main users of our interfaces are other researchers who will be using our prototype to develop other application on the KP plane using the KDN paradigm. Also, it will be used by people who work on network optimizing the routing, in order, to do it more efficiency

4.2.3 Beneficiaries

The beneficiaries from our final product will be the people that use the network as users, so the delay for loading a page/send an email etc will be lower. Also, it will make easier the work for people who works on optimizing the routing for a specific case.

5 Scope of the project

5.1 Scope

This project consists of creating a prototype for self-driving routing based on KDN. The project will be completed by October 2017. It will include a module that learns how to optimize the routing by itself and a prototype using this module, in order to compare with the state-of-art results.

5.2 Methodology and rigor

In order to develop the project, we have planned a development schedule that is going to help us in its creation.

First of all, we wanted to assure that the project was something that we will be able to do. Before deciding on choosing this project we wanted to be sure that make an optimizer that learns without operator it could be possible. For that reason, we make a simple test using Sequential Least Squares Programming(SLSP), with the implementation that the library scipy provides (https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize .minimize.html), with positive results. After that, we realize that we will be able to do the optimizer module.

During the first two months (March and April) we want to research on better ways to do the self-driving routing, more probably Reinforcement Learning.

After that, we will decide if we finally use Reinforcement Learning or more classical algorithms like SLSP for the optimizer module. In this moment and for the next two month we will be working on developing the prototype.

If we follow this schema, we will finish the technical part at the end of June, so we will have a couple of months, until October, to write the documentation.

5.2.1 Development tools

For doing the optimizer module we will use Python, in particular, the libraries Theano and Keras for reinforcement learning. In order to build the prototype will be using OpenDaylight, OMNET++ and TREX.

5.2.2 Tools to monitor evolution

My situation on the developing on the project is special for two reasons: I'm doing this project in collaboration with two Ph.D. candidates: Albert Mestres and Giorgio Stampa and I am also living outside of Spain.

For this situation, my supervisor, Albert Cabellos, organizes two hangouts every week in order to track the progress. On Monday, Albert Mestres, Giorgio Stampa and I talk about what we think that we should do this week and on Thursday, professor Albert Cabellos, Albert Mestres, Giorgio Stampa and I discuss the results that we had obtained during this week and Albert Cabellos guides us for the future steps.

On the side of tools for track the development of the project, we are going to use Git and Github, so they allow us to document all the changes that are done to the code. Also, we use Google drive and email to share what we are doing, as well as, to ask each other opinion.

5.2.3 Validation of results

We will be using our prototype in order to compare the results of our optimized module with the state-of-the-art mechanism, that we explained in section 3.2.

This article will be the only one that we will compare our results to because it compares many state-of-art technologies.This means that we won't have to use other articles, as this one will provide sufficient comparison to justify our results.

5.3 Obstacles and risks on the project

During this project is possible that we will face some problems. On the next lines we will explain the risk that we are taking on each part and the possible consequences of that:

5.3.1 Optimizer

Our first idea is to use Reinforcement Learning to solve that problem, but we can find ourselves in the situation where this representation of the states and the action is not the most adequate. Another probable scenario will be faced with the fact that RL is not the best method to solve it or even that we are not able to find a method that solves it with the requisites that we want.

However all of that, the project will be finished, due to, we have proved that exists a mechanism, SLQP, that can solve it. The main inconvenient of this method is the lack of scalability. So, even though, we will be able to create the first prototype using the KDN paradigm, optimize problem will still be a non-solve problem

5.3.2 Prototype

The main risk that we are taking on this part, is my lack of knowledge about networking. I will have to learn about it in general and in particular the most recent technologies like SDN. Also, I will work with technologies that I have never used before: OMNET++, TREX and OpenDayLight.

6 Schedule

6.1 Estimated project duration

The estimated project duration is 7 month, from the end of February 2017 until begins of October 2017.

6.2 Considerations

As it is a research project, it may be possible that the initial plan changes and we have to reconsider some of the technologies that we are using, as well as, the algorithms.

From 19th of August until 2nd of September, the project will stop for 2 weeks during summer holidays.

7 Project planning

We can define our project as a Chaotic Project, so we don't have a clear requirement or a clear solution for that. Also, the feedback from the client (director of this TFG) will be high and the requirements will change often using that feedback. Moreover, we don't have experience in developing this kind of projects.

Taking all of that into consideration, we think that the most appropriate software methodology is Extreme Programming (XP), as it has proven itself very successful for projects that have frequent changes of the requirements, which make an influence on the final functionality.

Extreme programming, as well as any other agile methods, is based on an iterative and incremental process that will change with the client's feedback. In order to follow this schema, you divide the program in cycles, where every cycle takes around 1-2 weeks. At the beginning of the cycle, you define a set of objectives and storylines (small tasks) that you will do. At the end of the cycle, the objectives have to be achieved.

The choice of extreme programming lies on the fact that during every cycle you are able to change the requirements defined at the beginning of the cycle. This property is not common in other agile methodologies.

On the following sections, we will provide an overall sketch of the phases we will be going through and will set a list of goals we want to achieve.

7.1 Life cycle

On Extreme Programming every iteration consists of 4 phases:

- Planning: Developers and clients meet together and define the requirements. After that, development team divides it on small features and prepare the plan, time and cost for doing this iteration.
- Designing: The team decides the principal design guides, for example, standards for class names etc.
- Coding: It is the most important phase of Extreme Programming and it has to follow the design guidelines.

- Testing: In Extreme Programming, TDD is integrated regular development philosophy where team implements unit and integration test to eliminate bugs and have a proper code coverage.
- Listening: On all the development phase, it is has a continuous feedback from the client. On every feedback, the requirements are revised and redefined if needed.

On our specific project, every iteration will start on Thursday, when we meet with our Professor to report and consider difficulties we are facing. After that, we will define plans for the next week and the new requirements and fix the priorities. Based on the priorities, we will start developing during the week with a constant communication using email and hangouts. This iteration will finish the next Wednesday.

7.2 Main Tasks

We can divided our task on 4 different sections: Initial work, Optimizer, Prototype and Documentation.

7.2.1 Initial work

On this section, we will do an overview of the project in order to define a plan, do requirement analysis and evaluate the viability. This phase will take around one month.

- Task 1: Context and scope of the project
- Task 2: Project planning
- Task 3: Budget and sustainability

The resources that we will use:

- Human Resources
  - **Project manager** will be responsible for the strategy, road-map and documentation for that product.
- Hardware Resources
  - **Personal computer** (Intel Core i7 2.2 GHz, 16GB RAM, 256GB SSD): used in all tasks for this project
- Software Resources
  - **macOS El Capitan v.10.11.6**: used in all the tasks.
  - **LaTeX:** used to write the documentation.

    - ***TeamGantt*** *(8)*: used for producing Gantt chart.

7.2.2 Optimizer

During this phase, we will learn about methods for optimising the routing.

- Task 1: Find a method for optimising the routing (One example could be Reinforcement Learning). During this process, we have 3 steps:
    - Learn about the method
    - Find if this method is suitable for our objective using a theoretical approach. If it is not, we will find another method and repeat the steps again.
- Task 2: Build the optimizer module. On this task, we will also check the accuracy of the optimizer module and decided if it is valid or not. If not, repeat Task 1 for another Optimizer. The precedent for this task is Task 1 of this section.

The resources that we use are:

- Human Resources
    - **Project manager** will be the mutual point between development group and client.
    - **Networking engineer** will be needed for insights on the main properties that defines the routing patterns of a network.
    - **Data Science** whose main tasks will be defining the method that will enable us to optimise the routing.
    - **Software developer** will build the optimise module.
- Hardware Resources
    - **Personal computer** (Intel Core i7 2.2 GHz, 16GB RAM, 256GB SSD): used in all tasks for this project
- Software Resources
    - **macOS El Capitan v.10.11.6**: used in all the tasks.
    - ***Python***: used for building the optimizer module.

We expected that the Task 1 and 2 will be repeat several times in order to achieve the optimizer with higher accuracy.

7.2.3 Prototype

On this section we will learn about networking.

- Task 1: Learn about Software Defined Networking and OpenDaylight
- Task 2: Learn about traffic generation and network simulation.
- Task 3: Combine SDN, traffic generation, network simulation and optimizer module in order to build our prototype. In order to start this task, we will have to finish all the tasks on the optimizer block.
- Task 4: Test the system to ensure robustness and proficiency of the prototype. The preceding task for this task is the task 3 of this section.

The resources that we use are:

- Human Resources
  - **Project manager** will be the mutual point between development group and client.
  - **Software developer** will build the prototype and will be responsible for all the test tasks.
- Hardware Resources
  - **Personal computer** (Intel Core i7 2.2 GHz, 16GB RAM, 256GB SSD): used in all tasks for this project
- Software Resources
  - **macOS El Capitan v.10.11.6**: used in all the tasks.
  - ***Python***: used for building the optimizer module.
  - ***OpenDaylight*** *(11)*: on open-source SDN platform used for the prototype.
  - ***OMNET++*** *(9)*: Network Simulator used for the prototype.
  - ***TREX*** *(10)*: traffic generator used for the prototype.

7.2.4 Documentation

This section will take around 2 month and the main task will be writing the documentation for our TFG.

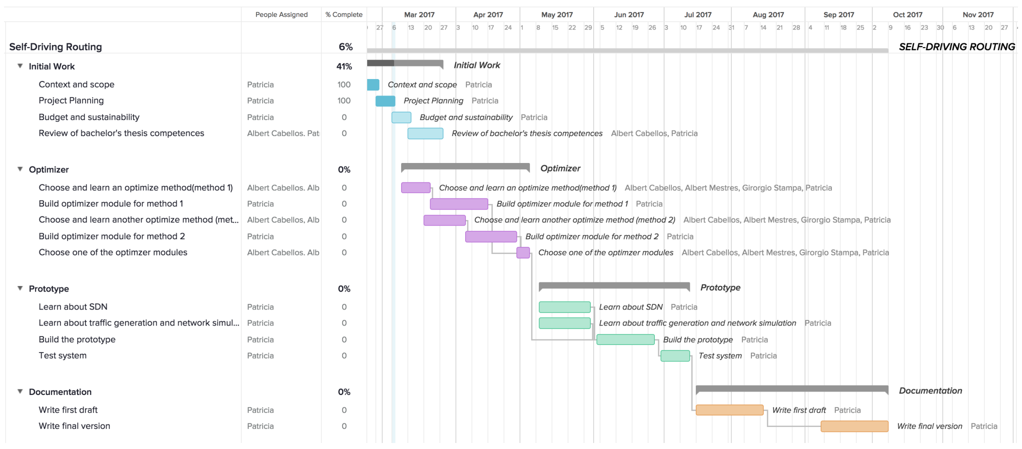
- Task 1: Write first draft of the report for our project
- Task 2: Write final report

The resources that we will use:

- Human Resources
  - **Project manager** will be responsible for the strategy, road-map and documentation for that product.
- Hardware Resources
  - **Personal computer** (Intel Core i7 2.2 GHz, 16GB RAM, 256GB SSD): used in all tasks for this project
- Software Resources
  - **macOS El Capitan v.10.11.6**: used in all the tasks.
  - **LaTeX:** used to write the documentation.
  - ***TeamGantt** (8)*: used for producing Gantt chart.

7.3 Gantt Chart

On this section, we summarise our tasks using a Gantt Chart. Moreover on this chart, we have included the order of precedence for the tasks.



	People Assigned	% Complete
Self-Driving Routing		**6%**
▼ **Initial Work**		**41%**
Context and scope	Patricia	100
Project Planning	Patricia	100
Budget and sustainability	Patricia	0
Review of bachelor's thesis competences	Albert Cabellos. Pat	0
▼ **Optimizer**		**0%**
Choose and learn an optimize method(method 1)	Albert Cabellos. Alb	0
Build optimizer module for method 1	Patricia	0
Choose and learn another optimize method (met...	Albert Cabellos, Alb	0
Build optimizer module for method 2	Patricia	0
Choose one of the optimzer modules	Albert Cabellos. Alb	0
▼ **Prototype**		**0%**
Learn about SDN	Patricia	0
Learn about traffic generation and network simul...	Patricia	0
Build the prototype	Patricia	0
Test system	Patricia	0
▼ **Documentation**		**0%**
Write first draft	Patricia	0
Write final version	Patricia	0

Mar 2017
Apr 2017
May 2017
Jun 2017
Jul 2017
Aug 2017
Sep 2017
Oct 2017
Nov 2017
SELF-DRIVING ROUTING
Initial Work
Context and scope Patricia
Project Planning Patricia
Budget and sustainability Patricia
Review of bachelor's thesis competences Albert Cabellos, Patricia
Optimizer
Choose and learn an optimize method(method 1) Albert Cabellos, Albert Mestres, Girorgio Stampa, Patricia
Build optimizer module for method 1 Patricia
Choose and learn another optimize method (method 2) Albert Cabellos, Albert Mestres, Girorgio Stampa, Patricia
Build optimizer module for method 2 Patricia
Choose one of the optimzer modules Albert Cabellos, Albert Mestres, Girorgio Stampa, Patricia
Prototype
Learn about SDN Patricia
Learn about traffic generation and network simulation Patricia
Build the prototype Patricia
Test system Patricia
Documentation
Write first draft Patricia
Write final version Patricia

Figure 9: Gantt chart. It is generated using *TeamGantt* (8)
Source: Personal compilation

8 Action plan

Our action plan is defined on the Gantt Chart. We've just mentioned several times, that we are not sure if exists a solution that can optimize by itself the routing.

On the subsection 'Methodology and rigor' we explained that it is possible to optimise the routing using linear programming. But this approach doesn't fit in our present requirements, due to its bottleneck on scalability and also it is not self-optimized. If we are not be able to find a optimizer by the deadline for the Optimizer section, we will redefine our requirements and use linear programming for building the prototype. We can redefine our requirements in any moment of the project so we are using Extreme Programming methodologies.

Our deadline is fixed but we are not sure if our original plan of designing a solution for self-driving routing is possible at all. However, we need to build a prototype and complete the project.

In case we can't design a new solution, in order to complete the prototype, we will use Linear Programming for Optimizer, so we will be on time for the Optimizer deadline. If we don't have enough time to finish the Prototype I will increase the number of hours per week for timely delivery. Now, I am working 25 hours per week(3 hours per weekday and 10 hours on weekends). So, if we have some potential deviation, I will work 34 hours per week (4 hours per weekday and 14 hours per weekends).

9 Cost estimation

9.1 Direct Costs

This section is divided using the activities that we described on the previous documentation.

The role of the product manager (12) and software engineer (13) corresponds to an undergrad student, so it is a junior salary. However Network engineer(14) and Data Scientist (15) are Ph.D. student, consequently, they have a senior salary.

In order to calculate the amortization for the personal computer, we use the following equation:

$$HL * \frac{PL}{YL * 365 * Hday}$$

HL = Hours using the laptop for this project

YL = Useful life in years

Hday = Hours per day using the laptop

PL= Price Laptop

We suppose that the life of our laptops is 6 years and that people usually use the laptop around 8 hours per day.

9.1.1 Initial Work

Human Resources						
	Role	Unit	Pay (€/h)	Hours/week	Estimated time (weeks)	Cost(€)
	Product Manager	1	30€	25	4	3000 €
Total	-	-	-	-	-	3000€

Hardware Resources					
	Item	Unit	Price/unit (€)	Useful life (years)	Amortization (€)
	Personal computer	1	2200€	6	12.56 €
Total	-	-	-	-	12.56 €

Software Resources					
	Product	Unit	Price/unit (€)	Useful life (years)	Amortization (€)
	macOs Sierra	3	0€	6	0 €
	Python	1	0€	-	0 €
	Latex	1	0€	-	0 €
	TeamGantt	1	0€	1	0 €
Total	-	-	-	-	0 €
Total	-	-	-	-	3012.56 €

Table 1: Direct Costs

9.1.2 Optimizer

Human Resources						
	Role	Unit	Pay (€/h)	Hours/week	Estimated time (weeks)	Cost(€)
	Software engineer	1	30€	25	8	6000 €
	Network engineer	1	39€	8	8	2496 €
	Data Scientist	1	50€	10	8	4000 €
Total	-	-	-	-	-	12496€

Hardware Resources					
	Item	Unit	Price/unit (€)	Useful life (years)	Amortization (€)
	Personal computer	3	2200€	6	43.20 €
Total	-	-	-	-	43.20 €

Software Resources					
	Product	Unit	Price/unit (€)	Useful life (years)	Amortization (€)
	macOs Sierra	3	0€	6	0 €
	Python	1	0€	-	0 €
Total	-	-	-	-	0 €
Total	-	-	-	-	12539.20 €

Table 2: Direct Costs

9.1.3 Prototype

Human Resources						
	Role	Unit	Pay (€/h)	Hours/week	Estimated time (weeks)	Cost(€)
	Software engineer	1	30€	25	10	7500 €
Total	-	-	-	-	-	7500 €

Hardware Resources					
	Item	Unit	Price/unit (€)	Useful life (years)	Amortization (€)
	Personal computer	1	2200€	6	31.39 €
Total	-	-	-	-	31.39 €

Software Resources					
	Product	Unit	Price/unit (€)	Useful life (years)	Amortization (€)
	macOs Sierra	1	0€	6	0 €
	OpenDayLight	1	0€	-	0 €
	Python	1	0€	-	0 €
	TREX	1	0€	-	0 €
	OMNET++	1	0€	-	0 €
Total	-	-	-	-	0 €
Total	-	-	-	-	7531.39 €

Table 3: Direct Costs

9.1.4 Documentation

Human Resources						
	Role	Unit	Pay (€/h)	Hours/week	Estimated time (weeks)	Cost(€)
	Product Manager	1	30€	25	8	6000 €
Total	-	-	-	-	-	6000€

Hardware Resources					
	Item	Unit	Price/unit (€)	Useful life (years)	Amortization (€)
	Personal computer	1	2200€	6	25.11 €
Total	-	-	-	-	25.11 €

Software Resources					
	Product	Unit	Price/unit (€)	Useful life (years)	Amortization (€)
	macOs Sierra	3	0€	6	0 €
	Python	1	0€	-	0 €
	Latex	1	0€	-	0 €
Total	-	-	-	-	0 €
Total	-	-	-	-	6025.11 €

Table 4: Direct Costs

9.1.5 Total Direct Cost

	Section	Cost
	Initial Work	3012.56
	Optimizer	12539.20
	Prototype	7531.39
	Documentation	6025.11
Total	-	29108.26 €

Table 5: Total Direct Costs

9.2 Indirect Costs

The main two indirect costs are electricity and Internet.

To obtain an approximation to the cost derivated from the electricity, it is used the following data:

Price of the Kwh in Spain: 0.12 €

Use of all the laptops: 894h

Power of the laptop: 61 W = 0.061 kW

It gives a cost of:

$$0.061kW * 894h * 0.11e/kWh = 5.99 \sim 6eur$$

For the Internet, it is paid 45€/ month. We calculate the amortization based on that we usually use it 8 hours per day:(45*hours_working_monthly)/(30*8)

	Item	Cost (€)
	Electricity	6€
	Internet	169.62€
Total	-	175.62 €

Table 6: Indirect Costs

9.3 Direct And Indirect Costs

	Type	Cost
	Direct	29108.26
	Indirect	175.62
Total	-	29283.88 €

Table 7: Direct and Indirect Costs

9.4 Contingency Costs

The contingency has been calculated as the 8% of the direct and indirect costs, due to as we have a high detail planned we don't expect a lot of deviations.

	Justification	Impact on Cost	Cost (€)
	Level of detail is accurate	8 %	2342.70€
Total	-	-	2342.70 €

Table 8: Contingency Costs

9.5 Incidental Costs

	Causes	Solution	Risk of occurences	Impact on cost (€)	Cost (€)
	Lack of time on Prototype section	Software Engineer will increase the working hours (During 4 weeks will work 34 hours per week)	25%	1080€	270.0€
Total	-	-	-	-	270.0 €

Table 9: Incidental Costs

9.6 Total Cost and Control Management

This aim of this project is to create a prototype using open source technologies. One deviation could be that the open source products are not suitable for our purpose, for example, we are not able to configure them in order to deploy our optimizer module. This has a low probability so the combination of OpenDayLight, OMNET++, and TREX, it is common on this field. Moreover, the most regular changes for this technologies are also open-source.

The main probable cause of increasing the budget will be that engineers have to work more hours. It has been already considered that on the incidental costs.

However, it is only considered that the engineers will be working more hours on the Prototype section because on the other section it will not be needed as it is explained on the following points:

- Optimizer: If we arrived at the deadline without any good optimizer, we will use Linear Programming to create the prototype
- Initial work and Documentation: We have reserved an enough quantity of time in order to be able to finish and edit it.

Depending on the experience of the software engineer, the working hours could be reduced. We can use the same argument for the Product Manager.

	Type	Cost
	Direct And Indirect	29283.87
	Contingency	2342.70
	Incidental	270.0 €
Total without VAT	-	31896.57 €
Total with 21% VAT	-	38594.85 €

Table 10: Total Costs

10 Sustainability Report

10.1 Environmental

During the development of the project, it is not used anything else than the personal computers. Consequently, the only environmental impact will be the electricity. It has been seen that the cost is not remarkable.

The main part of our project is to find a solution for a problem that nowadays doesn't exist a solution. Finding an environmental improvement for the current solutions will be on further steps (not included in this project) when this problem will be solved.

10.2 Economic

It has been presented a detailed planned and an estimated costs for the project, where it has been included human and material resources.

The solutions that have been presented until now, requires the presence of operators: people that are in charge of controlling and optimizing the network. If our approach is successful, the jobs on network operator will be decreased, so the network will do the most important task on its own, without any human help.

Nowadays, on network operations center you need at least 3 network operator per work shift (their medium salary is 50000€(16) per year) who will provide a watchful 24/7 eye on the networks. If this solution is finally implemented in real operation centers, it will only 3 network operation, one per every work shift. The cost of these centers will decrease on the half for human resources.

10.3 Social

This project will allow me to discover what is be a researcher and will help me to decide which is the path that I want to follow in the next years in my career.

KDN paradigm provides a future vision where all the issues that can occur on the network will be saved and analysis by the network. Using this tool the Network operator jobs will be easier, so they will have more information for the troubleshooting.

If this project is successful the delay on the network will be no longer a problem, so always will be the minimum delay possible. People who use network, like Internet, will increase their quality of life so, they will be able to realize their tasks, such send an email, load a page etc faster.

As we mentioned on the economic aspect, this project will make less necessary the Network Operator role, so a lot of their functions can be done by the network without human intervention.

10.4 Sustainability Matrix

	Project Development	Exploitation
Enviromental	10	0
Economic	8	6
Social	10	7

Table 11: Sustainability Matrix

References

[1] Clark, D., "A knowledge plane for the internet." Conference on Applications, technologies, architectures, and protocols for computer communications, ACM Proceedings of, 2003.

[2] Mestres,2016. Knowledge-Defined Networking.

[3] Hartert,2015. A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks in SIGCOMM 2015.

[4] Scholarpedia. Reinforcement Learning — Scholarpedia. Feb. 22, 2017. url: http://www.scholarpedia.org/article/Reinforcement_learning (visited on Feb. 22, 2017).

[5] SdxCentral. What is overlay networking?. Feb. 23, 2017. url: https://www.sdxcentral.com/sdn/definitions/what-is-overlay-networking/ (visited on Feb. 23, 2017).

[6] SdxCentral. Inside SDN architecture. Feb. 20, 2017. url: https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/ (visited on Feb. 20, 2017).

[7] SdxCentral. OpenDayLight Controller Feb. 19, 2017. url: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/opendaylight-controller/ (visited on Feb. 19, 2017).

[8] Teamgantt, online gantt chart software. https://www.teamgantt.com/. Online. Accessed: 2017-03-05.

[9] OMNET++, discrete Event Simulator. https://omnetpp.org/. Online. Accessed: 2017-03-05.

[10] TREX, Realistic traffic generator https://trex-tgn.cisco.com/. Online. Accessed: 2017-03-05.

[11] OpenDayLight, Open-source SDN platform https://www.opendaylight.org/. Online. Accessed: 2017-03-05.

[12] Product Manager Hourly Wages. http://www1.salary.com/Web-Product-Manager-I-hourly-wages.html. Online. Accessed: 2017-03-10.

[13] Software Engineer Hourly Wages. http://www1.salary.com/Software-Engineer-I-hourly-wages.html. Online. Accessed: 2017-03-10.

[14] Network Engineer Hourly Wages. http://www1.salary.com/Network-Engineer-II-hourly-wages.html. Online. Accessed: 2017-03-10.

[15] Data Scientist Hourly Wages. http://www1.salary.com/Data-Scientist-hourly-wages.html. Online. Accessed: 2017-03-10.

[16] Network operation salary. https://www.glassdoor.com/Salaries/network-operation-center-analyst-salary-SRCH$_KO0, 32.htm.Online.Accessed : 2017 - 03 - 10$.