Predicting Traffic Patterns in Software Defined Networks

Liva Giovanni - liva.giovanni@spes.uniud.it
Hermann Hellwagner - @aau.at
Marino Miculan - @uniud.it

May 28, 2015

Abstract

Just the prediction part

1 Introduction

The predictability of network traffic is the main aim of the thesis. Usually, there are two different category of network prediction: short and long period predictions. The short forecast is used to guess values in terms of seconds or minutes. Instead, the long one is adopted to estimate the future workload. Therefore, it favors the possibility to produce better planning and decision. To be able to predict future load of a network, we have to create a model of its behaviors. On the changing of a model we have different characteristics such as the correctness of the prediction and its adaptability.

There are two type of models: **Supervised** and **Unsupervised**. The *Supervised* algorithms takes as a input a set of objects and the desired output. From the input, the learning algorithm analyzes the training data and produces an inferred function that is checked with the one passed in input. The internal structure of the model is changed according to the error between the forecast and the desired result. Instead, the *Unsupervised* learning tries to find hidden structure in unlabeled data. The difference with the *Supervised* learning algorithms is that there is no error or reward signal to evaluate a potential solution.

We focus over the long term prediction and only supervised classifier. The decision of which classifier chose has been taken conducting an experiment in a small simulated network. We simulate a normal scenario of daily network usage through a network of 4 nodes. We repeat the simulation thirty times collecting at each execution statistics of the links utilization in terms of network bandwidth and the load of the switches. From this information, we have created different dataset changing the features used and the numbers of the last observations. Then, we have test the prediction precision and recall of different algorithm at the varying of the distinct datasets.
The division of the work is designed in four different phases

- Observer
- Analyzing
- Plan
- Execute

A graphical representation how the four module are interconnected is given in Figure 1.

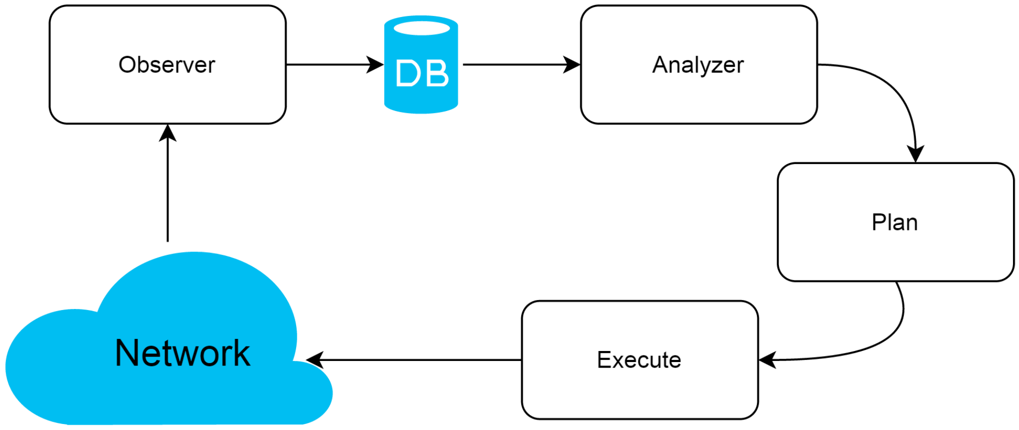

Figure 1: The different phases of the prediction

The first module, **Observer**, is implemented as a daemon in the cloud application. Every few minute it launches a python script which queries the network controller and then stores the result inside the database. The information saved regards the network load, switches and flows.

The second one, **Analyzing**, is done looking through the information inside the database. A java application collects the data from the database and converts the knowledge in the Attribute-Relation File Format (ARFF). This format is an ASCII text file that describes a list of instances sharing a set of attributes. The application depends on the *weka* (Waikato Environment for Knowledge Analysis) package, a well known suite of learning machine algorithms developed by the University of Waikato. The ARFF files are read by the weka package and used to produce the model that is adopted to make predictions.

The third element, **Plan**, is demanded to the Administrator. He or she can write rules to specify what to do when a particular event occurs. In the cloud application the administrator has the possibility to create the rules that are stored inside the FloodLight controller.

The last phase, **Execute**, is implemented by the controller. It monitors the network and every few minutes it makes predictions using the previously generated model. When it perceives from a forecast that some rules can be applied, it fires them.

Every module is uncoupled from the others. This design decision of modularity gives us the possibility to change or upgrade every module whenever there is the necessity. This feature is crucial for the prediction phase. We can test new classifiers or the addiction of new features in a separate and controlled network without affecting the production one. Moreover, we can hot swapping the model using the cloud interface. We have designed the controller to work with a model for each switch. This decision brings the possibility to predict when a particular node will be overloaded with more precision and recall.

2 Weka

2.1 Configuration

classification learning is appropriate for any problem where deducing a classification is useful and the classification is easy to determine. Net -¿ Mininet Observer -¿ Daemon Analyzer -¿ Weka -¿ Struttura modulare -¿ Cambiare algo quando si vuole Plan -¿ Operator w/ Web Interface config the FloodLight Module Execute -¿ FloodLight Module exe the rules

2.2 DataSet

2.3 Evaluation

2.4 Result

Printare i dati in excell, dare qualche valore

2.5 Discussion

Avere 50% di successo -¿ 10 volte meglio di sparare a caso 1/21 = 4.76%