Predicting network load with machine learning for VNF placement

Proposal for a masters thesis

Christoph Kaiser

24.06.2019

Computer Networks Group
Universität Paderborn

Problem description & Motivation

- Virtual Network Functions
- Placement and scaling algorithms
  - Create new VNFs if needed
  - Distribute VNFs in the physical network
- A problem of these algorithms is that they can be reactive (e.g. [Dräxler et.al. 2018])
  - Scaling and placement start too late
  - If VNF is not already running, startup time is added to processing time
  - Processing time up to tens of seconds [Mijumbi et.al. 2017]

Possible Solutions

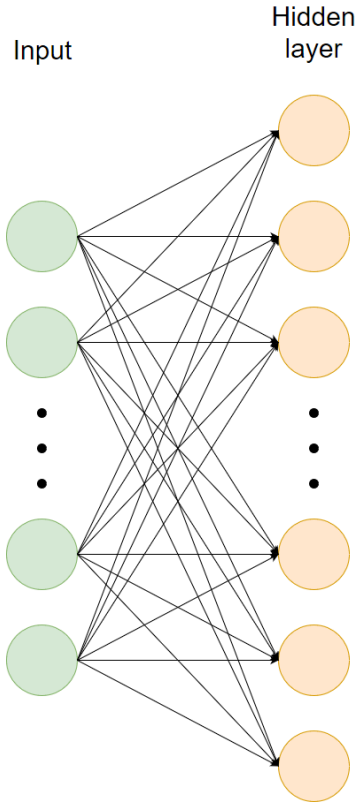
- Naive approach: start all VNFs on all nodes
  - Defeats purpose of VNFs
  - Waste of resources
  - VNFs interfere with each other
- Instead use machine learning models to predict traffic
  - Can compensate for startup times
  - Shut down VNFs earlier
  - Keep VNFs running

Data

- Multiple sources of traces exist
  - For example from Google and Facebook
  - Download an analyze
  - Many GB up to TB of data
- Data must be modified
  - Add service chain
  - Determine realistic chain
  - Network structure most likely not part of data
  - Reconstruction of network from traces not possible
  - Remove unneeded data
- Data will be prepared with R or Python
- Also possible to generate traces

Models

- What library to choose
  - Tensorflow or Keras
- Machine learning models
  - LSTMs
  - Neural Network
  - Modelling the input data
    - No standard way of modelling the input data
    - Input data can only have limited size
    - Number of inputs is static
- Statistical models
  - ARIMA



Models

- How should the predictions look like
  - Time
    - Point in time
      - e.g. traffic in 10s
    - Range of time
      - e.g. traffic in 5s for the next 10s
  - The load of services
    - A list of services
    - Just one service
  - Prediction for the whole network or just one node

Training

- Splitting data for training and testing
  - Split data in three sets
    - Training, validation, testing
  - Usual way of training neural networks
  - How much data needed for training is not clear
- Maybe consider training at runtime

Testing

- Gather test data and evaluate
  - Also using R or Python
- Many different evaluations possible
  - Comparison between point and range prediction
  - How far into the future do the predictions keep its quality

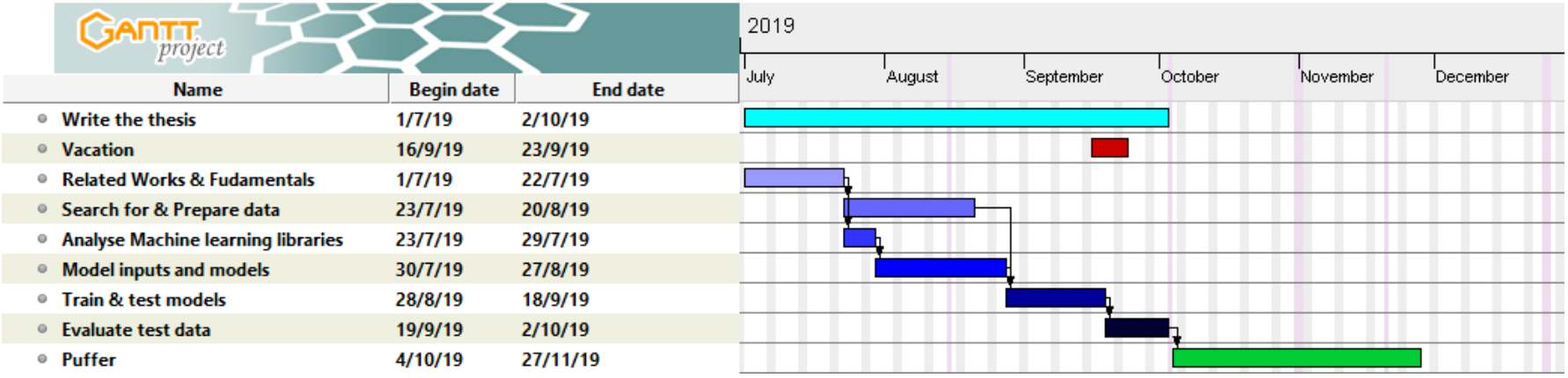
Summary – major tasks

1. Gather, sort and modify data
2. Model inputs
3. Model outputs
4. Build models
5. Training and testing

Structure of the thesis

1. Introduction
2. Related Work & Fundamentals
3. Gathering Data
    1. Download and analyze
    2. Preparing the data
4. Model Building
    1. Choosing a library
    2. Modeling the input data
    3. Building the models
5. Training and testing
    1. Training the models
    2. Evaluating the test data

Timetable



Name	Begin date	End date
Write the thesis	1/7/19	2/10/19
Vacation	16/9/19	23/9/19
Related Works & Fudamentals	1/7/19	22/7/19
Search for & Prepare data	23/7/19	20/8/19
Analyse Machine learning libraries	23/7/19	29/7/19
Model inputs and models	30/7/19	27/8/19
Train & test models	28/8/19	18/9/19
Evaluate test data	19/9/19	2/10/19
Puffer	4/10/19	27/11/19

Questions?

References

- R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba. Topologyaware prediction of virtual network function resource requirements. IEEE Transactions on Network and Service Management, 14(1):106-120, March 2017.
- S. Dräxler, S. Schneider, and H. Karl. Scaling and placing bidirectional services with stateful virtual and physical network functions. In 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pages 123-131, June 2018.