Applying ML to SDN: Use-Cases and Ongoing Experiments

Ongoing Research

Albert Cabellos (UPC/BarcelonaTech, Spain)
Prof. Jean Walrand (UC Berkeley)

albert.cabellos@gmail.com

IETF 94 – Yokohama – NMLRG – November 2015

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Berkeley UNIVERSITY OF CALIFORNIA

Thanks to:

- Shyam Parekh
- David Meyer
- Fabio Maino, Hugo Latapie, Chris Cassar, and John Evans
- Sharon Barkai
- Victor Muntés
- **Albert Mestres**, Eduard Alarcón, Pere Barlet, Alberto Rodríguez

Motivation and Goals

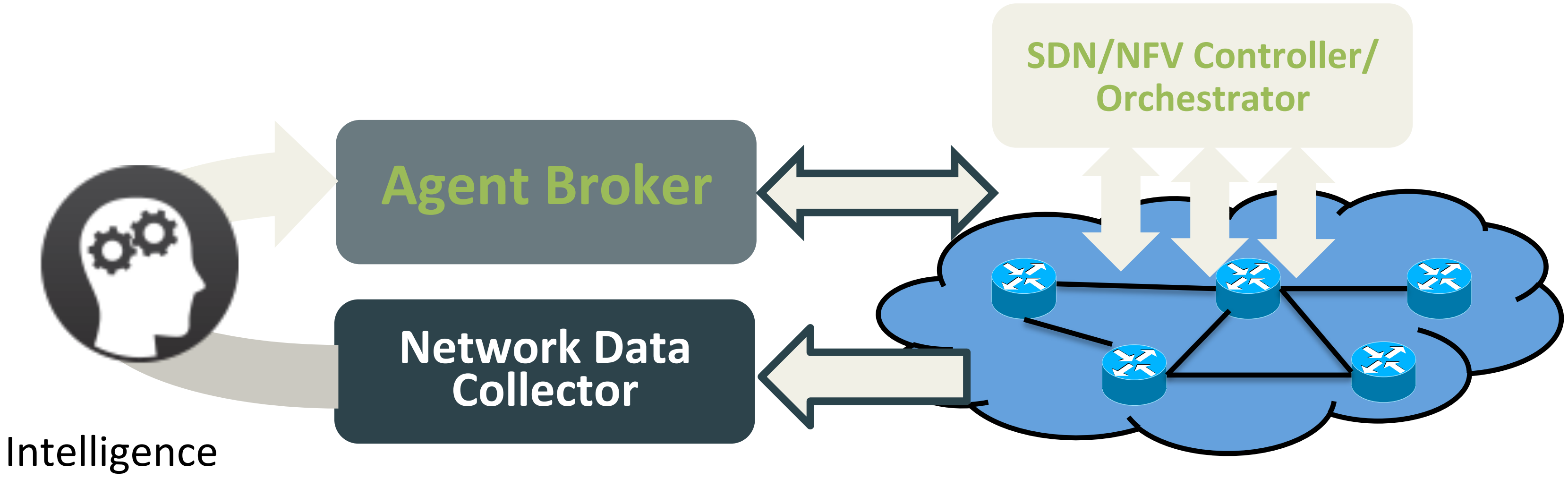
Scientific Objectives

- Apply ML techniques to Networking:
  - Control (fast dynamics)
    - E.g, routing, resource allocation (NFV/SFC), PCE, optimization, congestion detection
  - Management (slow dynamics)
    - E.g., network planning, resource management, load estimation
  - Recommendation mechanisms
- Towards self-driving networks
- Out of the scope
  - Anomaly Detection
  - Traffic Classification

Why now?

- Traditionally networks have been distributed systems
  - Partial view and actuation capabilities
- Beyond programmability, SDN provides centralization:
  - Full view of the network
  - Full actuation capabilities
- Data-Plane nodes are currently equipped with computing and storage capabilities
  - Beyond SNMP/Netflow monitoring

Overview of the Architecture



- Based on the data obtained by the Data Collector (topology, traffic, performance metrics) the system <u>learns</u> and <u>builds</u> a <u>network model</u>
- The network model is used to provide (e.g.,):
  - Optimized control
  - Autonomic Management
  - Recommendation

Use-Cases

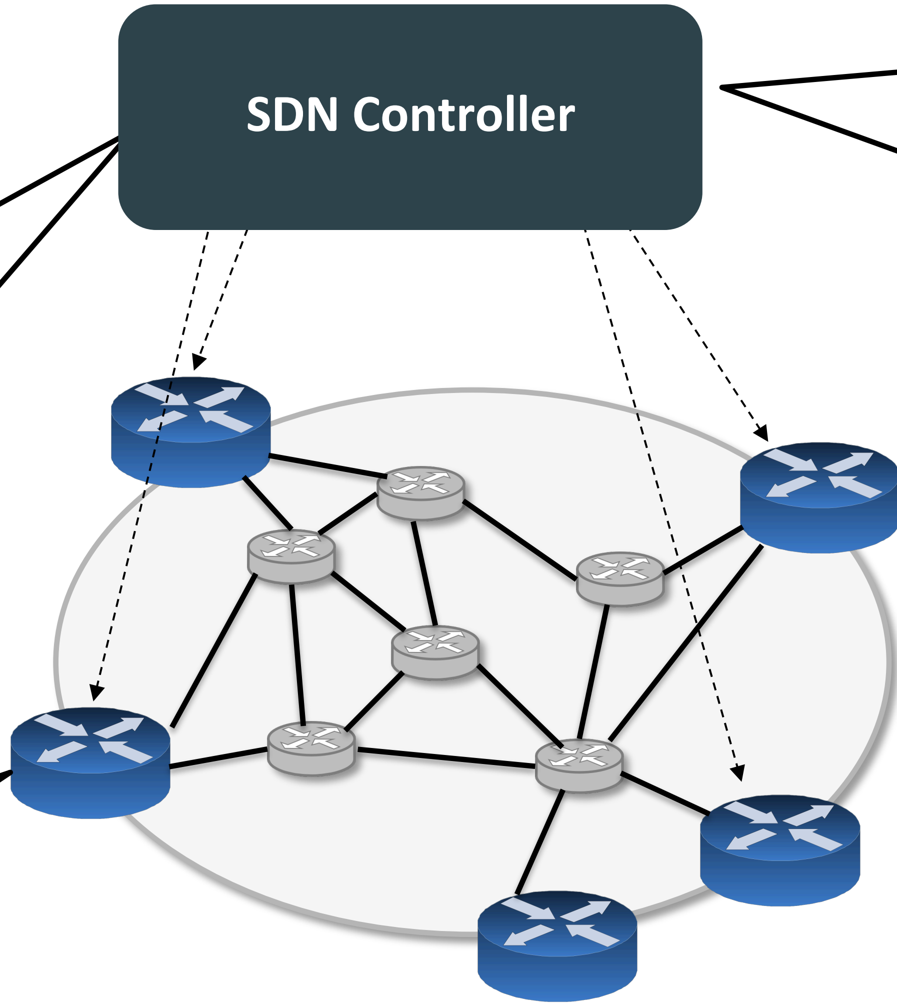
Use-case 1: Overlay Routing

- Overlay nodes use an **unknown** underlay, why?
  - Underlay is provided by another company
  - Underlay is legacy, does not provide monitoring/actuation capabilities
  - VPN over different providers (branch/offices)
- Underlay offers a set of paths to reach the destinations
  - E.g, LISP, Segment Routing or any other mechanism
- Problem: Traffic to paths allocation such that:
  - Load-balances traffic over the underlay
  - Provides delay guarantees
  - Other restrictions may apply

Learn how to route

Use-case 1: Overlay Routing

Control overlay routing to:
- Load Balance underlay
- Delay Guarantees

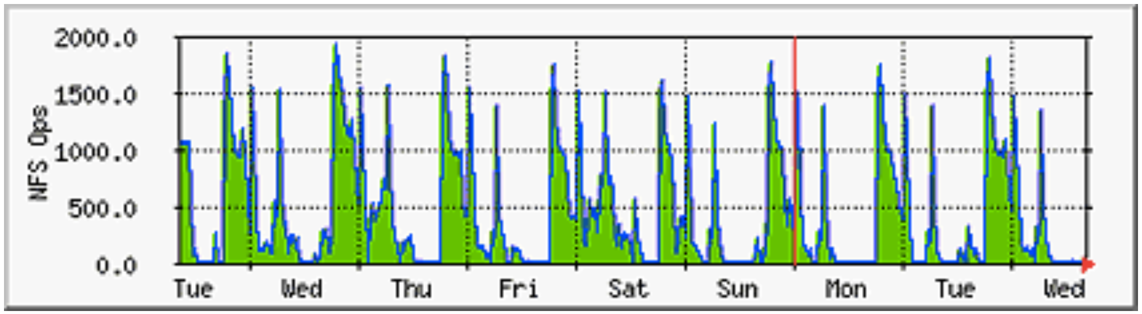


Learns about past routing decisions and the resulting underlay performance

Which underlay paths should I choose to send traffic?

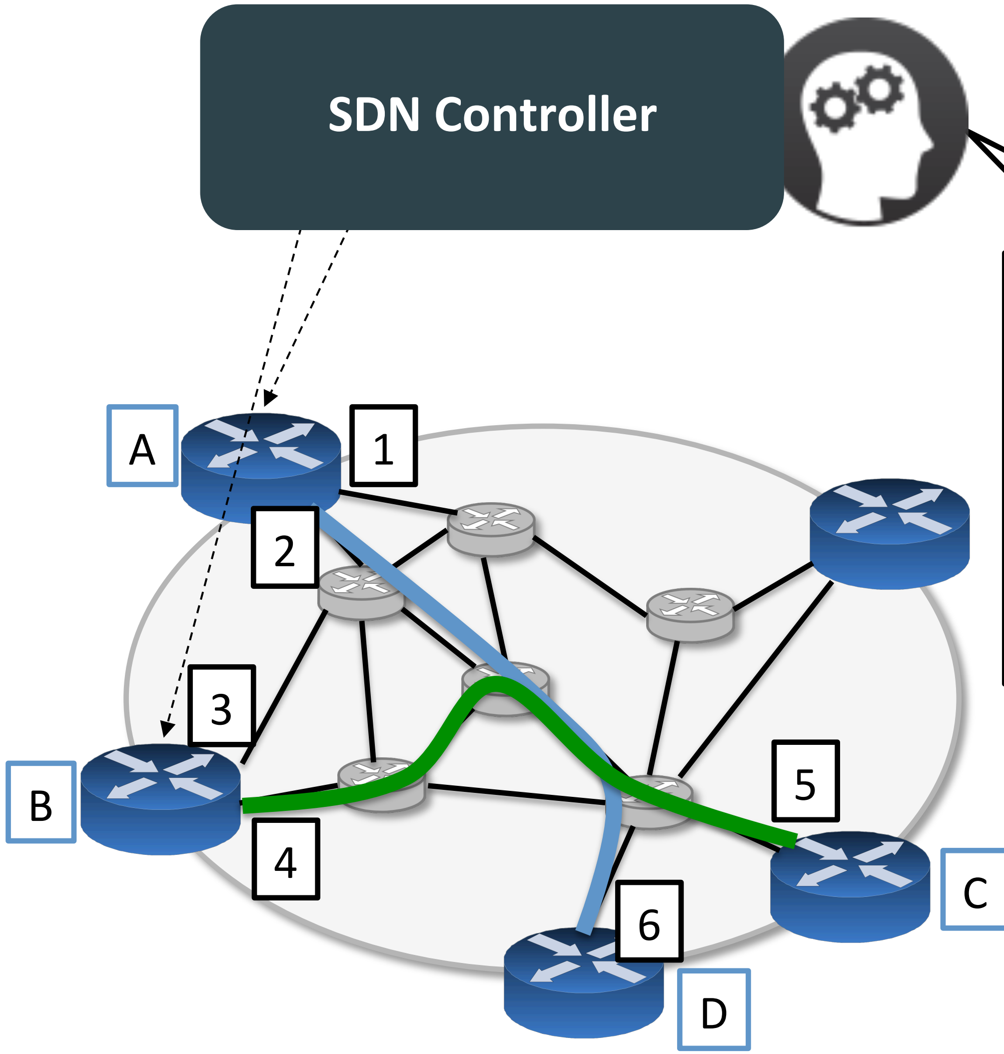
Use-case 1: Overlay Routing (I)

- Typically paths exhibit a seasonal behaviour



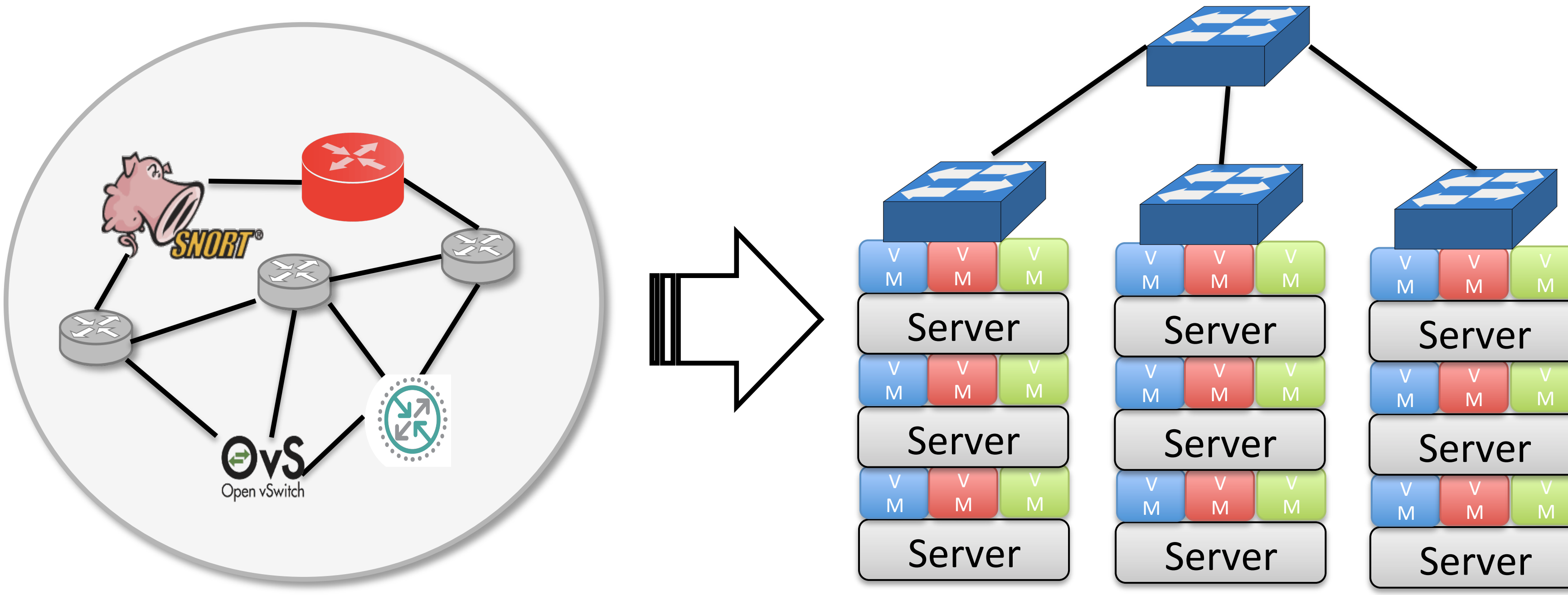
- Use time-series to model the performance of a path over time based on the past
  - Predict future (minutes, hours) performance
- Improve this by learning correlations with other events
  - Calendar Day: Holidays, etc
  - Other network related metrics (jitter, utilization, etc)
- Can we predict failures/blackouts?

Use-case 1: Overlay Routing (II)



When Router A sends traffic over path 2-5 and Router B over path 4-6 I see an increase on the delay for both paths. I should try other paths.

Use-case 2: NFV/SFC Resource Management



- Assign VNFs to VM
- Place VM in Server

} Resource Management

Learn the cost of a VNF (delay/computing)

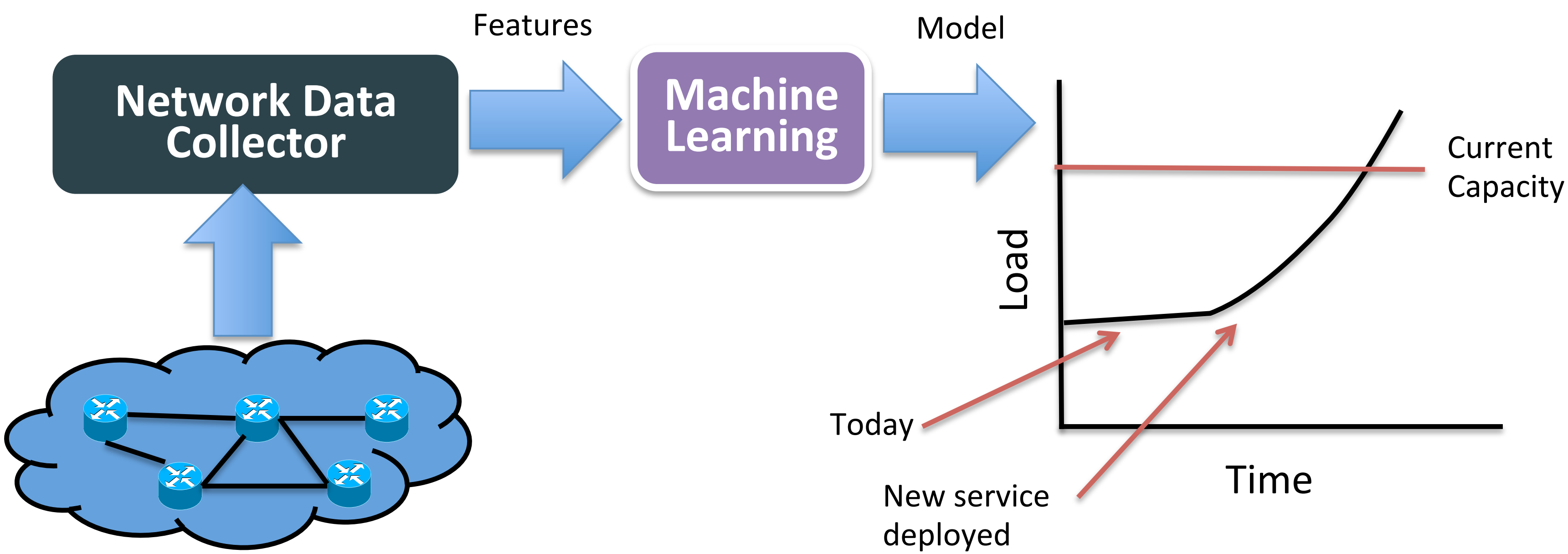
Use-case 3: Load Estimation

- Predict the load of the networking infrastructure
- Long-term:
  - Estimate when an infraestructure upgrade is needed
  - Reduce infraestructure replacement costs
- Short-term
  - Anticipate congestion/performance degradation
  - Just-in-time provisioning vs. Just-in-case

Learn the relation between client/services and load of a network

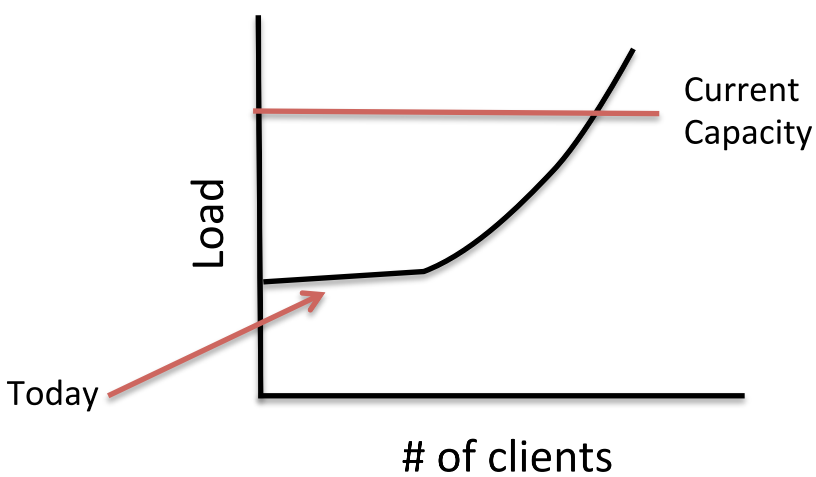
Use-case 3: Load Estimation

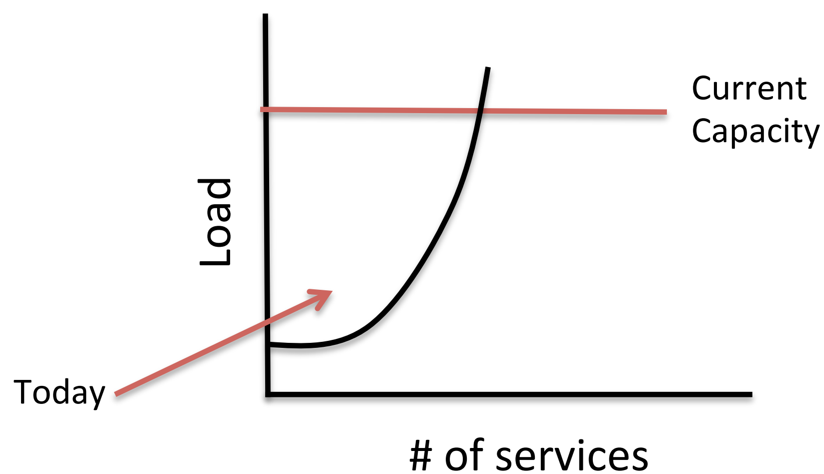
- Overview of the architecture
- System is trained off-line and used on/off-line



Use-case 3: Load Estimation

- ML produces models correlating a set of relevant metrics:





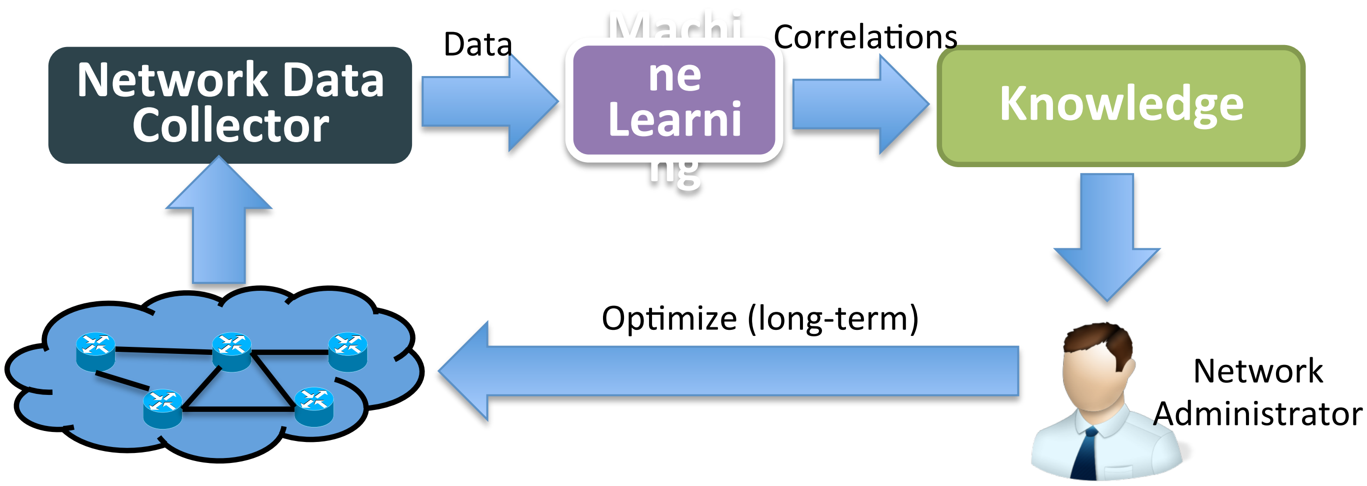
- Simple (linear) estimates about the trends in #clients, #services can be used using historical data

Use-case 3: Load Estimation

- Statistical Machine Learning
- Suggested network features (incomplete list):
  - Clients (active, types, temporal evolution…)
  - Services (type, # VMs, cross-dependency…)
  - Performance metrics (utilization of the links, latency of the applications, jitter…)
  - Fixed data: Topology, etc
- The accuracy of the model can be tested using historical data

Use-case 4: Knowledge extraction

- Data→Knolwedge
- The system finds correlations and creates knowledge
- Used by humans to optimize the infraestructure



Correlate relevant network events

Use-case 4: Knowledge extraction

- Examples of knowledge

```
Interface GE1/1 on node N is congested each tuesday at
around 8pm, services X, Y and Z have a large number of
clients
```

```
A high number of BGP UPDATES messages are sent,
Interface GE1/2 flappes
```

```
Jitter in Interface GE1/2 is high, service X, Y, Z
latency is high, clients for service Y is higher than
the average
```

Use-case 4: Knowledge extraction

- K-means, PCA and Correlation Analysis techniques
- Suggested network features (incomplete list):
  - Clients (active, types, temporal evolution…)
  - Services (type, # VMs, cross-dependency…)
  - Performance metrics (utilization of the links, latency of the applications, jitter…)
  - Signaling events (BGP messages, BGP states, used routes…)
  - Interface stats (packets, jitter, delay, …)
  - Fixed data: Topology, etc

Use-case 4: Knowledge extraction

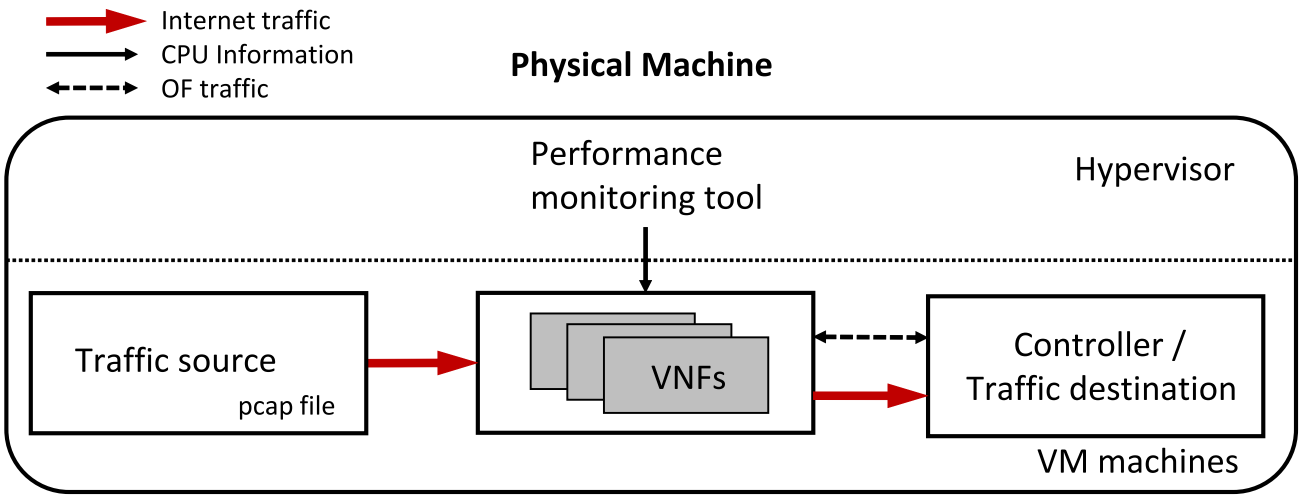
- Training is performed by network administrators selecting relevant events:
  - Interfaces flappes
  - High number of BGP_UPDATES/WITHRAWL over a period of time
  - Latency above average
- ML finds correlation around such data events
- Creates knowledge

Experimental Results (<u>**ongoing**</u>)

How to experimentally demonstrate such use-cases with the available data?

Load estimation of a VNF

VNF Load Experiment



$$f(traffic\_features) = CPU$$

- Can we predict the load of a VNF?
- Predictor: Traffic
- Predicted: CPU and Delay
- VNF is a black box

VNF Load Experiment

- 3-layer Artificial Neural Network
  - 10-node hidden layer
  - 70 input features
- Tested with real-world traffic
- 70 traffic-features

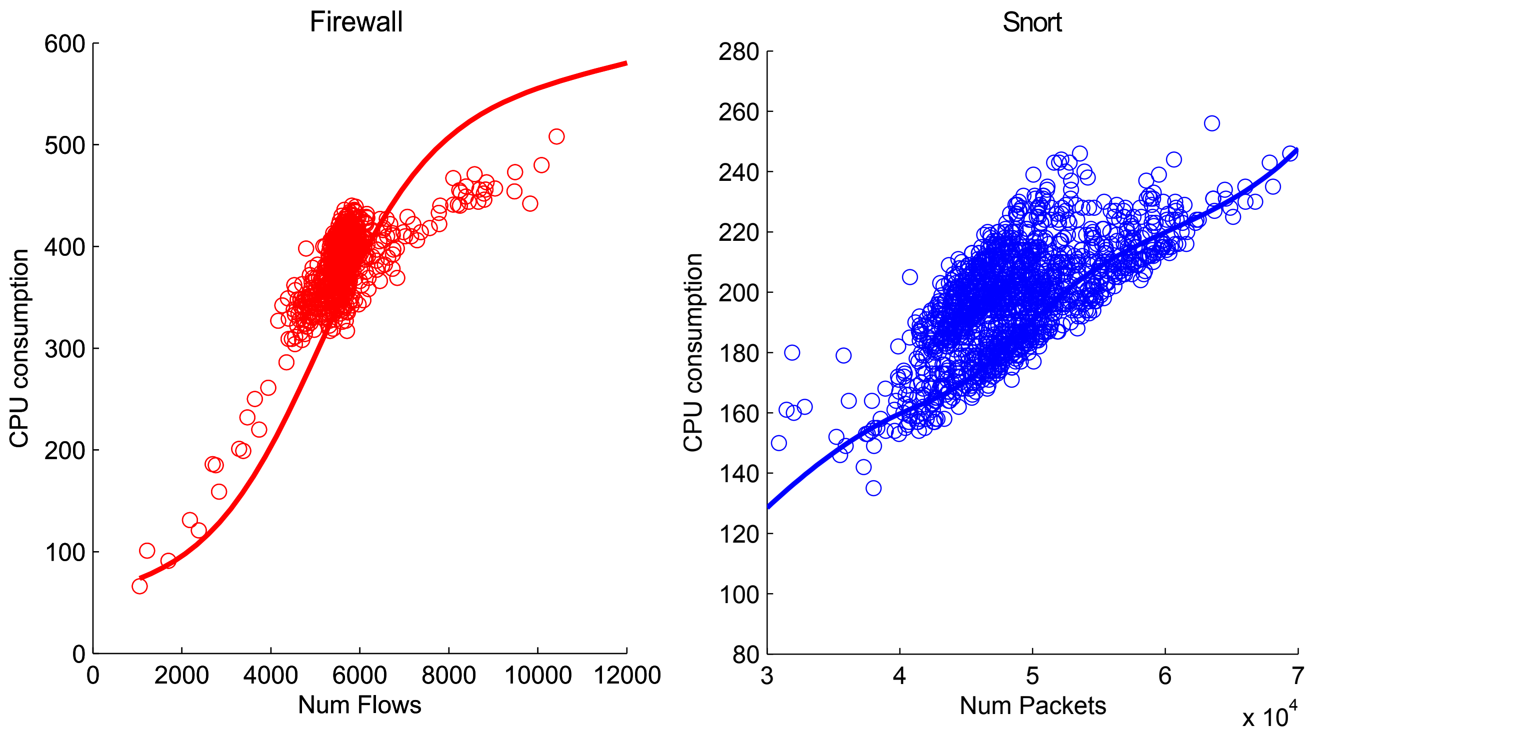
- VNFs:
  - OVS (switch)
  - OVS (fw)
  - Snort

VNF Load Experiment: Traffic Features

- 70 traffic features
- Can be computed at line-speed
- Typically available by default in many networking equipment

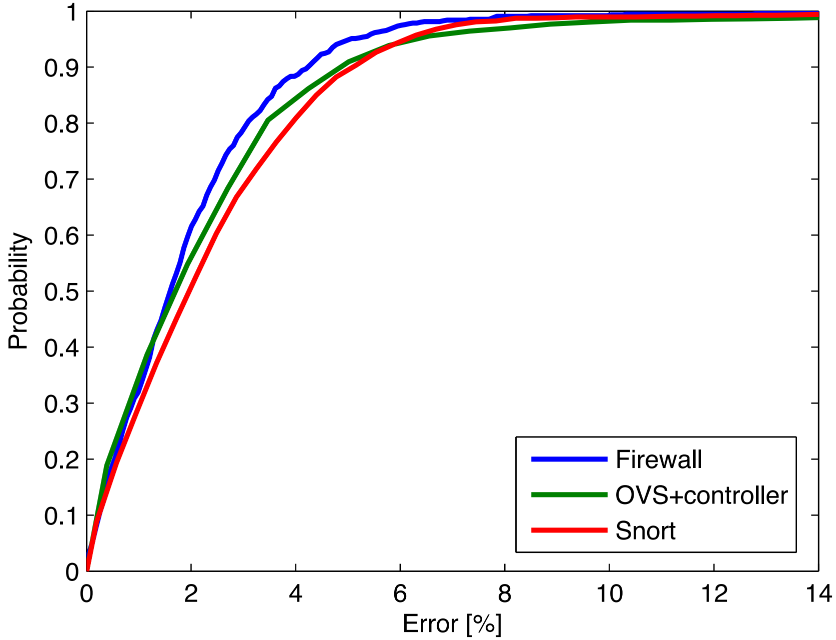
```
numPackets:
totalBytes:
avgInterAT:
stdInterAT:
avgLength:
stdLength:
ipSrc:
ipDst:
ipSrcDst:
ipv4:
ipv6:
icmp4:
icmp6:
otherL3:
ipMaskSrc[30]
….
```

Is the model trivial/linear?



Error of the model < 5%

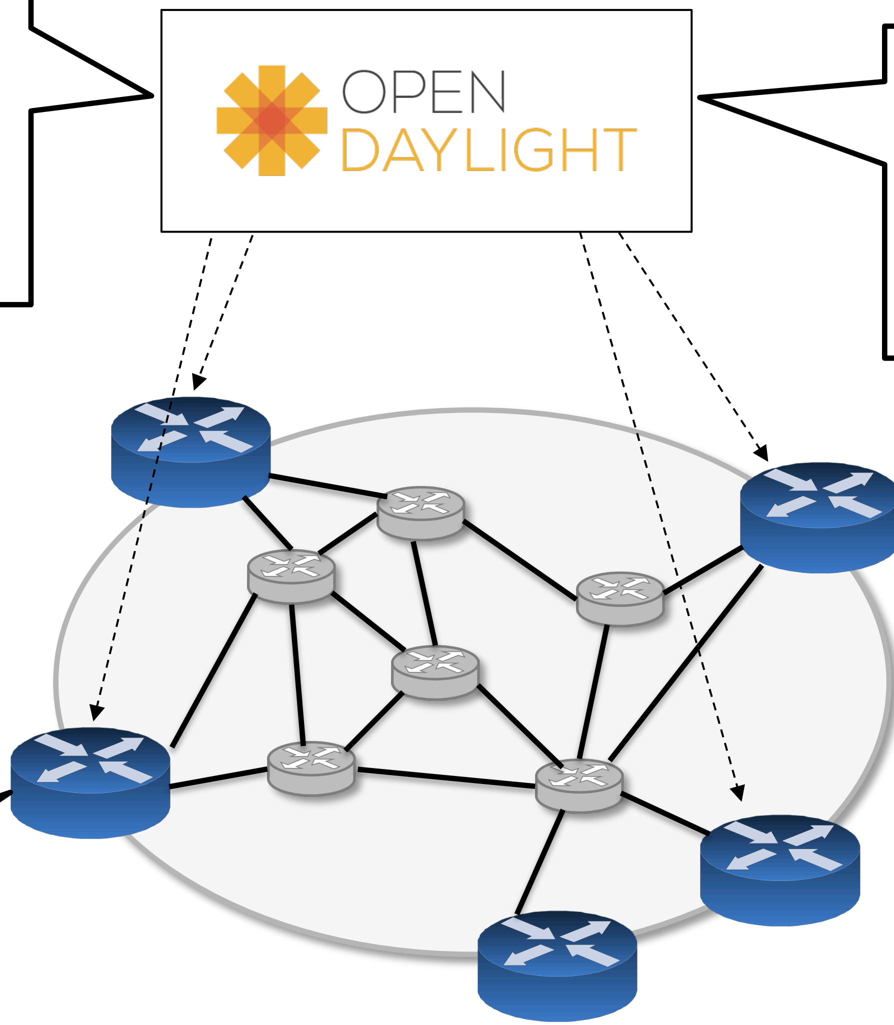
- VNFs depend on different features according to:
  - Type of VNF
  - Configuration of VNF
- Offline learning also possible for many scenarios



Overlay Routing (ongoing)

Overlay Routing

Let me see what happens when I send traffic through here…



I learnt how to route! Even if I don´t seen the underlay!

Which underlay paths should I choose to send traffic?

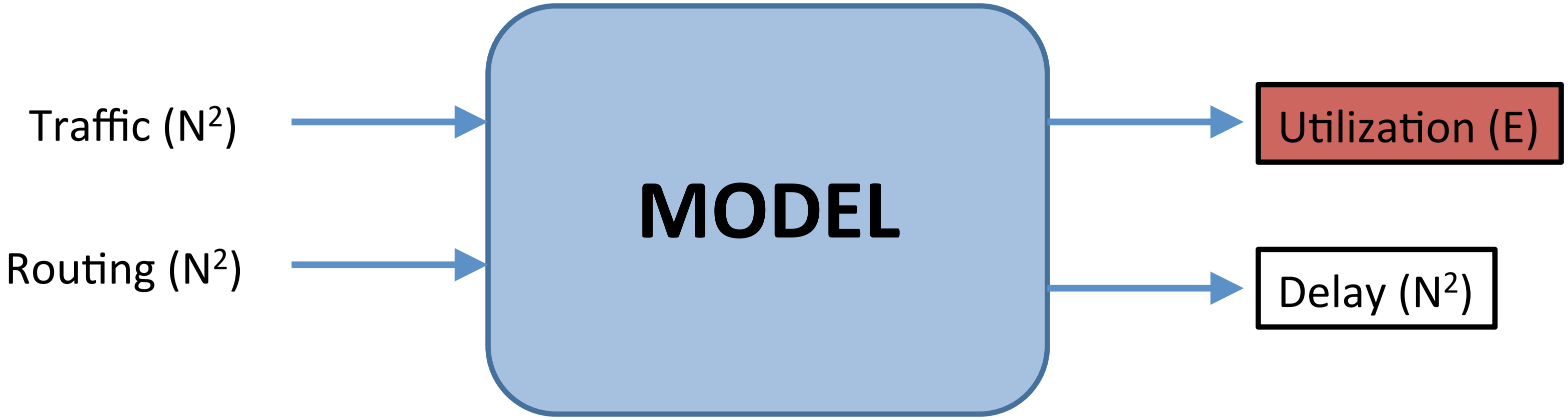
Network Model (I)

- There are 196 pairs of nodes (N^2)
  - Random uniformly distributed traffic
- Simple Internet2 topology (14 nodes, 22 links)
- Five routing options among nodes
    - 1: Shortest path
    - 2: Equally distributed among possible paths*
    - 3: 2/3 shortest path + 1/3 2^{nd} shortest path
    - 4: 4/5 shortest path + 1/5 2^{nd} shortest path
    - 5: 1/2 1^{st} path + 1/3 2^{nd} path + 1/6 3^{rd} path
  - Randomly chosen

*: limited to 10 paths and delay <= 2·minDelay

- Train the system with 10000 random samples

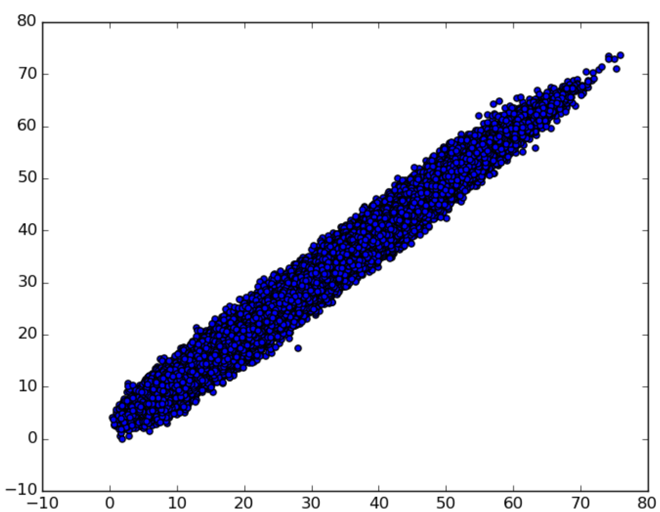
Network Model (II)

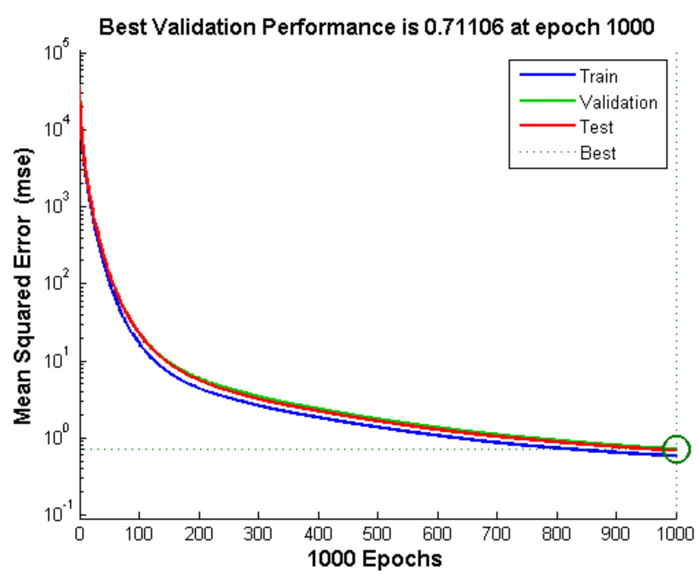


Topology is hidden

Results (ongoing)

- Artificial Neural Network (1 hidden layer)
  - 196 input features
  - 200 nodes in the hidden layer
  - Topology is hidden for ANN
  - **Results only for one traffic policy**





Conclusions

- Use-cases: where traditional models are impractical:
  - Computationally too expensive
  - Hidden variables
  - Not accurate
- Paradigm shift on how we manage and run our networks
  - Unprecedented optimization
  - Lower management costs
  - Towards self-driving networks