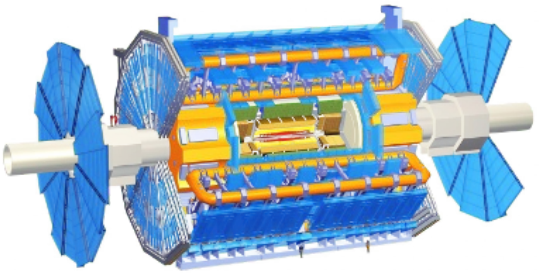
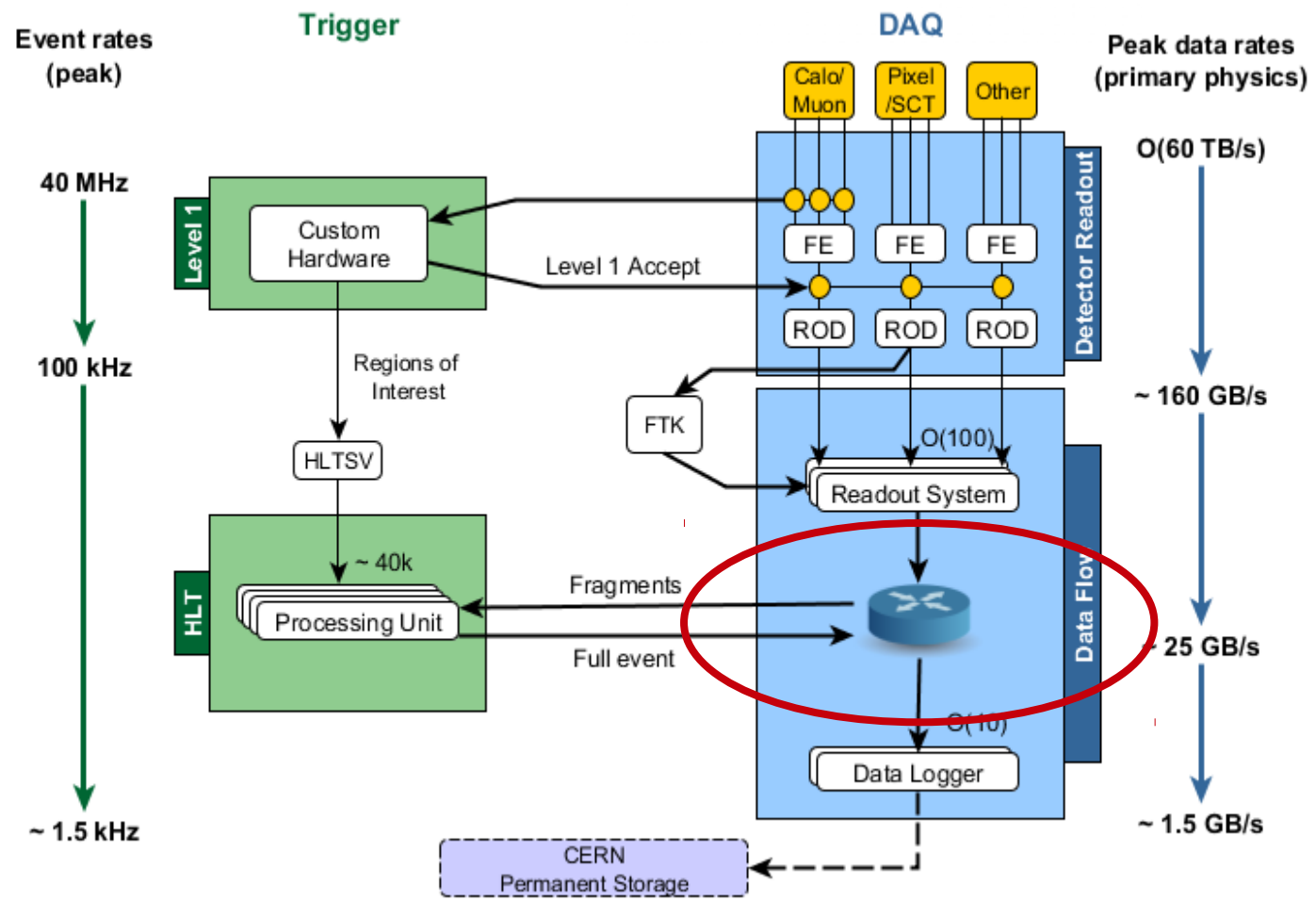
Machine Learning Techniques in the ATLAS TDAQ Network Monitoring System

Oskar Wyszyński

Introduction





Main Challenges in Network Monitoring:

- Link failure detection with low number of false positives

  How to distinguish between human interference and failing connection ?

- Detecting anomalies in over half million of parameters

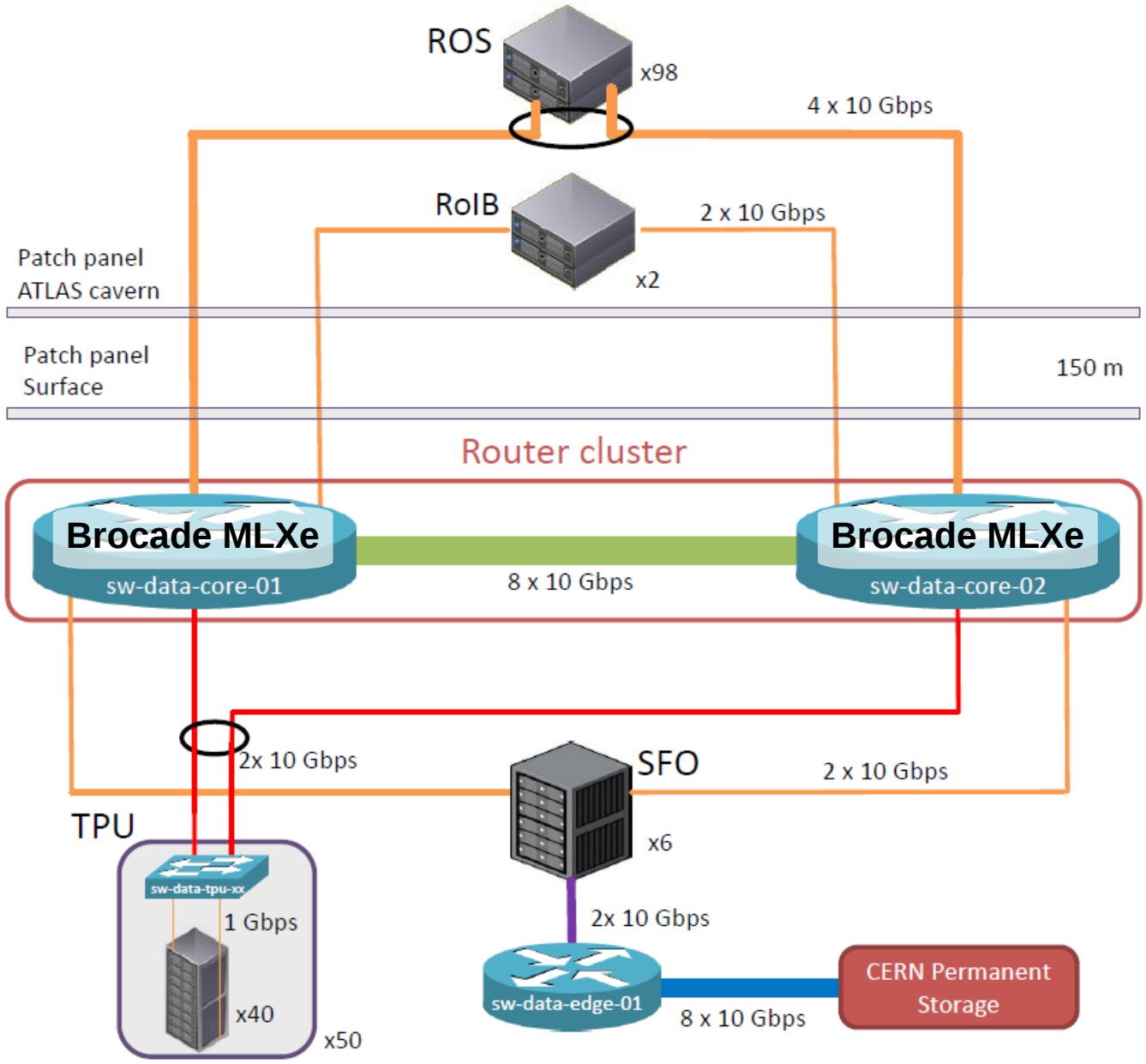
  How to define correct value ranges for parameters in dynamic systems ?

- Root Cause Analysis

  How to define primary cause ?
  Not addressed by this talk

Data Collection Network

- **High throughput**
- **High availability network. No single point of failure due to redundancy**
- **More than 500 x 10 GbE ports**
- **2 Brocade MLXe core devices in cluster mode**
  - Load balanced between cores
  - Transfer of ~ 15 GB/s per core
- **Large number of network devices**
  - ~ 200 switches and routers
  - ~ 6000 network cards
- **Half million of variables to monitor**



Network monitoring

Machine Learning

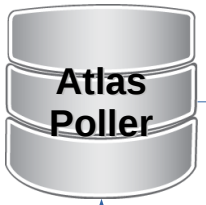
Introductory study of machine learning techniques application for the following problems:

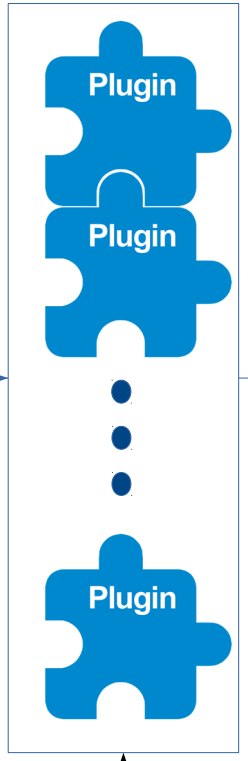
1) Anomaly Detection in the network → simple density-based technique

2) Detection, identification and filtration of interface flapping incidents → Radial Basis Function (RBF) Kernels and a decision tree

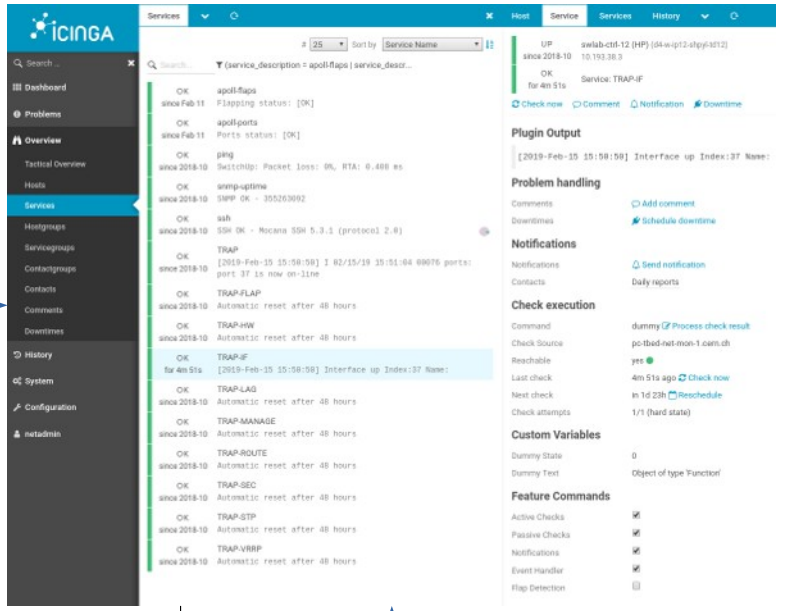
Network monitoring

Software Environment



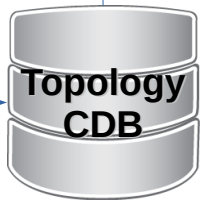


Passive
&
Active
Checks



Notifications

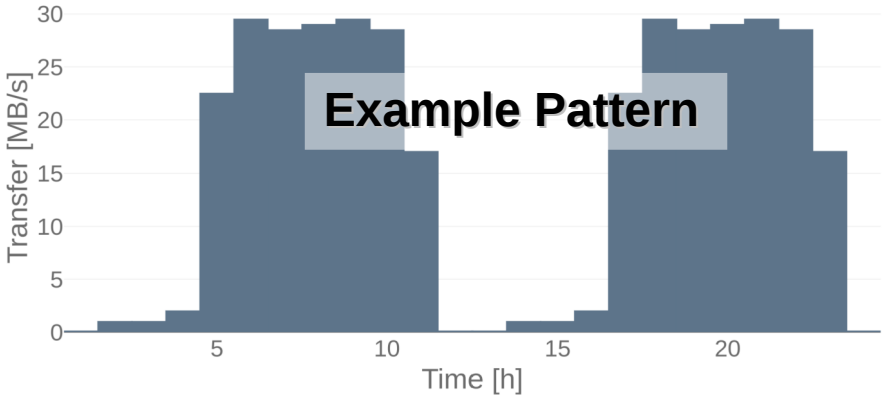
- Algorithms were implemented as Icinga (version 2) plugins
- Fed with external data as well as from Icinga itself



Anomaly Detection

Network Dynamics

- Few data transfer patterns
- The most challenging is the pattern that follows LHC cycle
  - Data taking can last up to 30h
  - Few-hours gap between data taking sessions
  - Most of monitored variables are following this pattern
- Half a million of variables to monitor
  - Arbitrary thresholds not feasible due to changing conditions
  - Adaptation to changing conditions are required, in order to reduce false alarm rate
  - Exploring learning rate as an adaptation implementation
- Uniformity of hardware is exploited. Comparing anomaly in time to anomaly in a group



Two-stage detection:

1) Anomalies in time – monitoring a parameter over time, comparing it to a individual model

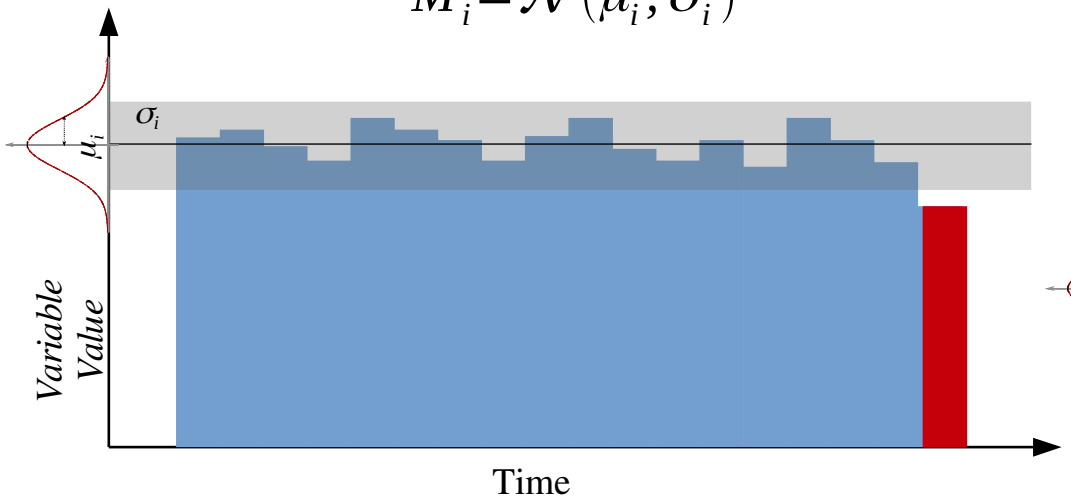
2) Anomalies in groups – comparing output value of individual variables with other members of a group.

Anomaly Detection

Example Models

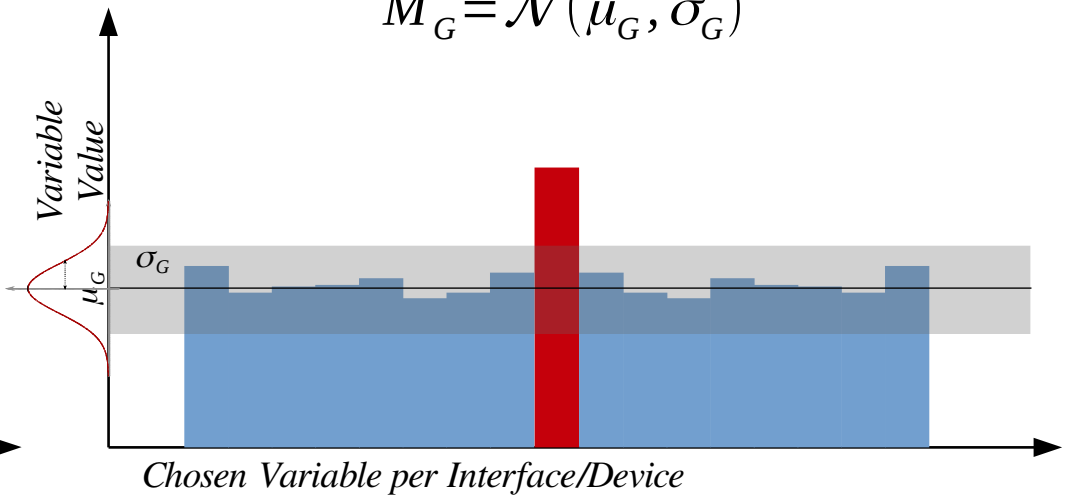
Model for Individual variable

$$M_i = \mathcal{N}(\mu_i, \sigma_i^2)$$



Model for group of variables

$$M_G = \mathcal{N}(\mu_G, \sigma_G^2)$$



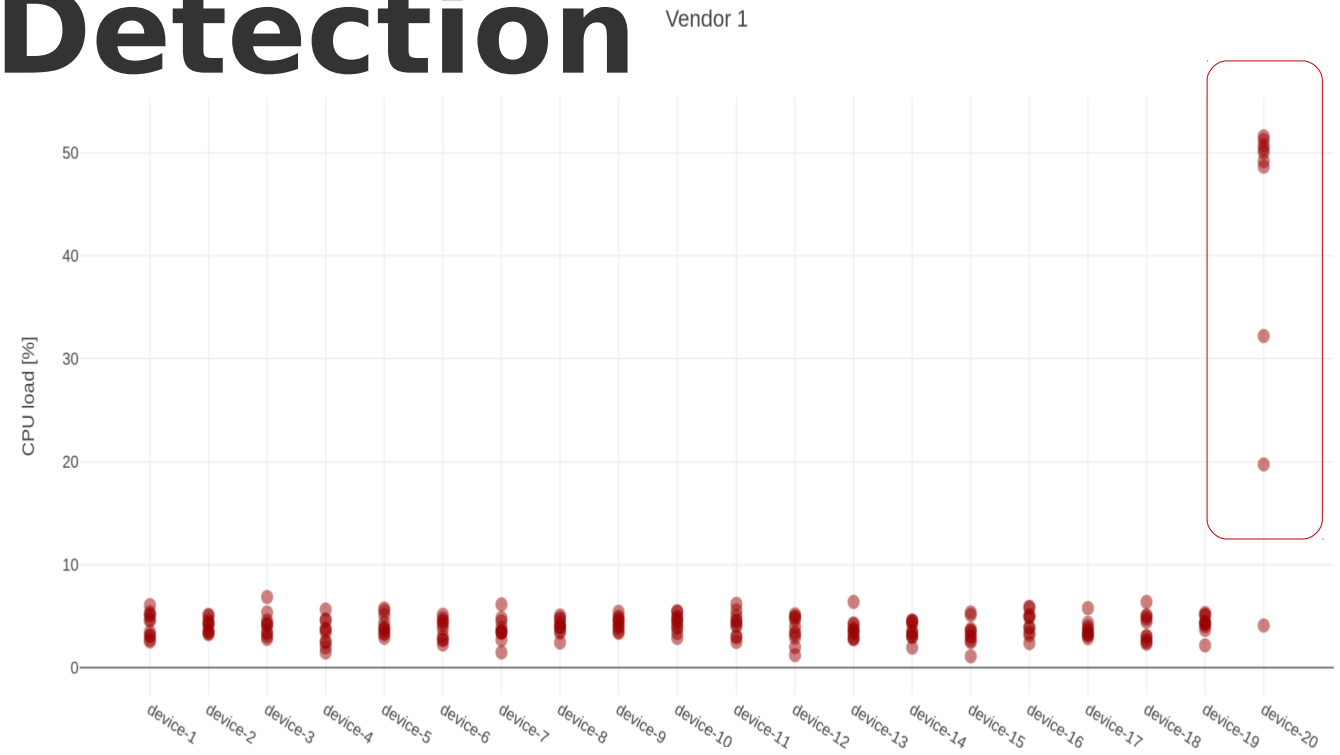
- **515**k variables and **63** groups defined so far
- Vast number of false alarms dismissed
  - up to **150**k dismissed per run
  - up to **600**k dismissed per day
  - maximal registered reduction: **97**%
- Computational overhead is negligible
- Models generated based on device database
  - new devices detected automatically
  - automatic group assignment

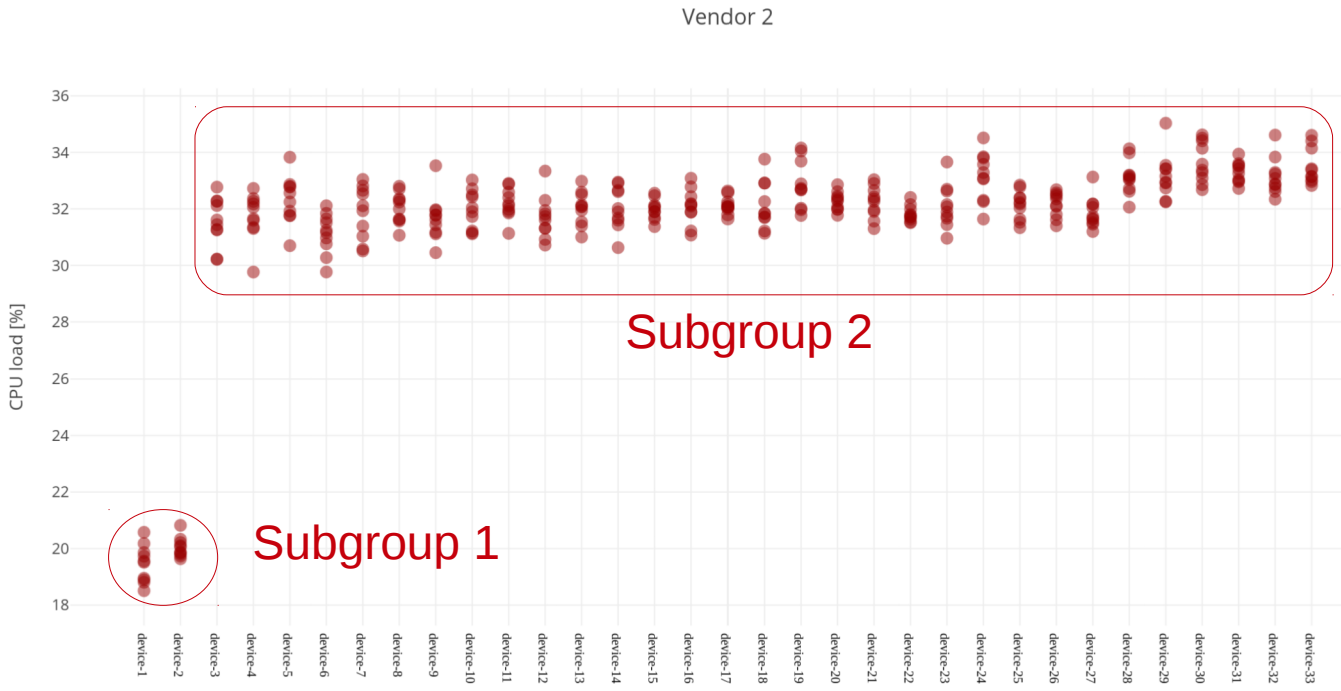
Group Anomaly \ Individual Anomaly	False	True
False	OK	OK
True	Warning	Critical

Anomaly Detection

Examples

- Anomaly detected (red square) for device-20 within Vendor 1 group of “CPU load” variables
- The high CPU utilization caused by renegade snmp monitoring process





- All devices within Vendor 2 group present correct behavior
- Due to significant number of unused ports, a subgroup must be defined for device-1 and device-2
- Clusterization can be employed for finding subgroups

Anomaly Detection

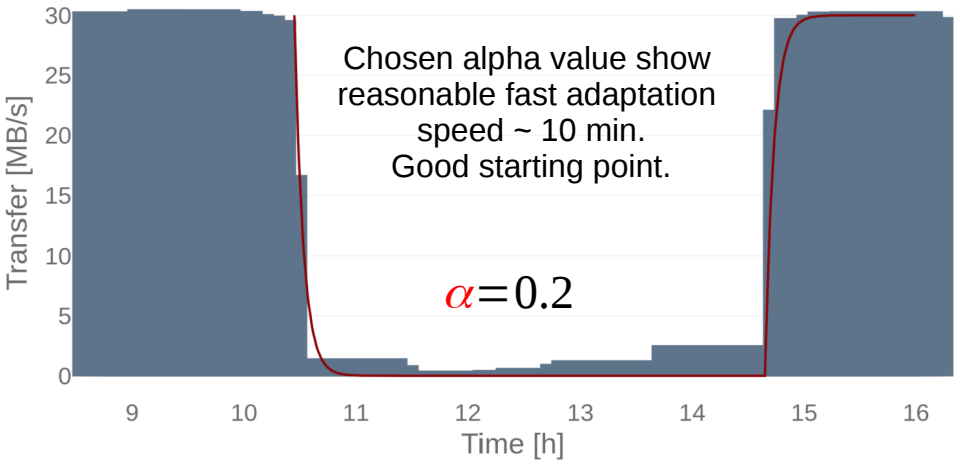
Learning Rate

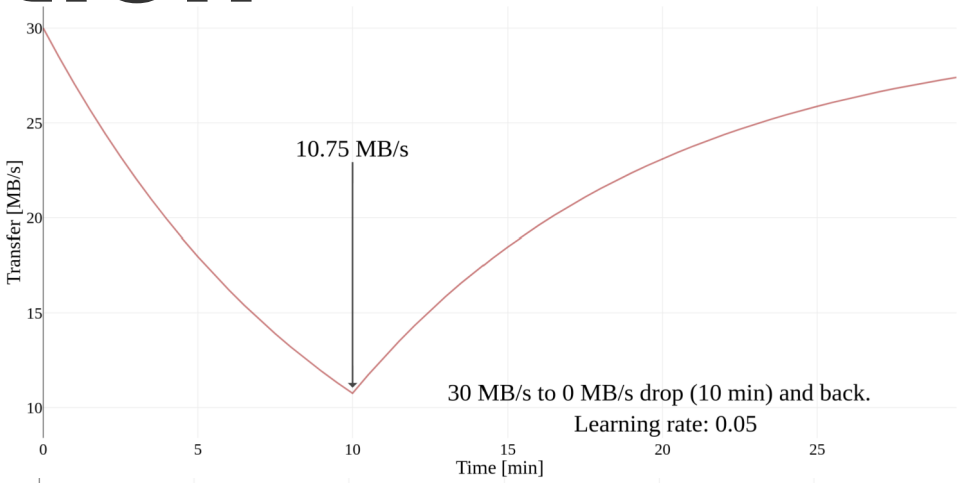
Adaptation is needed, to prevent spending tremendous amount of time on fine-tunning thresholds that determine behavior of current dynamic system e.g. trunk load balancing.

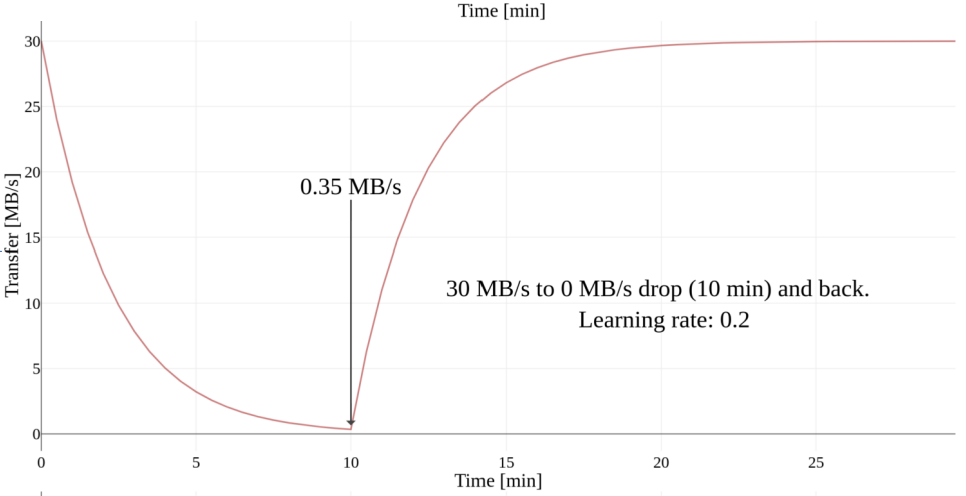
This proved to be time consuming task, typically resulting in increasing tolerance and consequently decreasing sensitivity. In order to adapt to changing conditions, a learning rate constant is used.

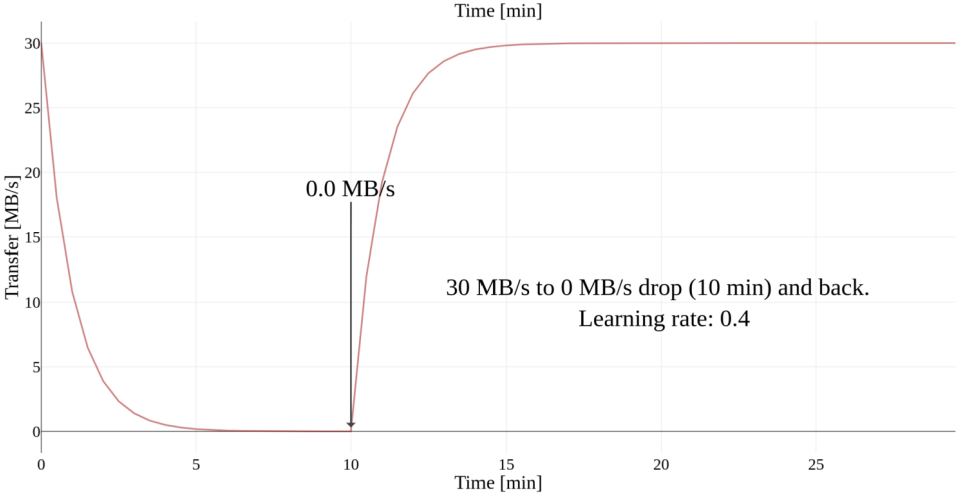
$$\hat{\mu}_{(t)} = \hat{\mu}_{(t-1)}(1-\alpha) + \alpha\mu_{(t)}$$

The alpha value is a **trade-off** between fast adaptation vs alarm rate of individual parameters.





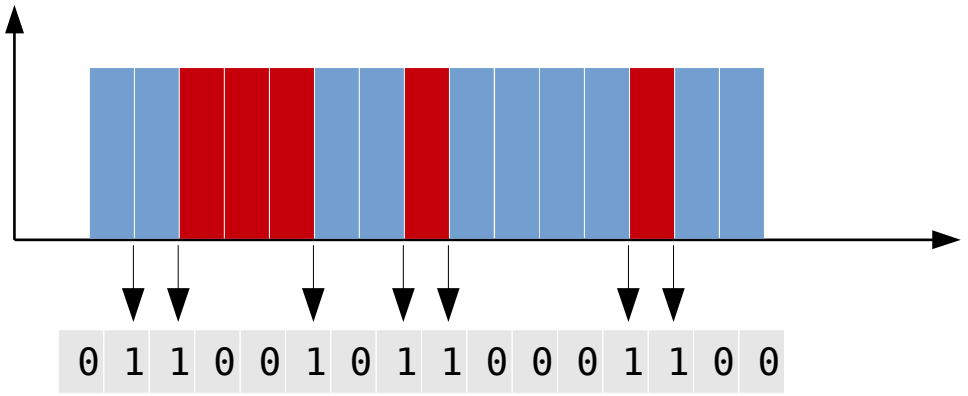




Link Flapping

Introduction

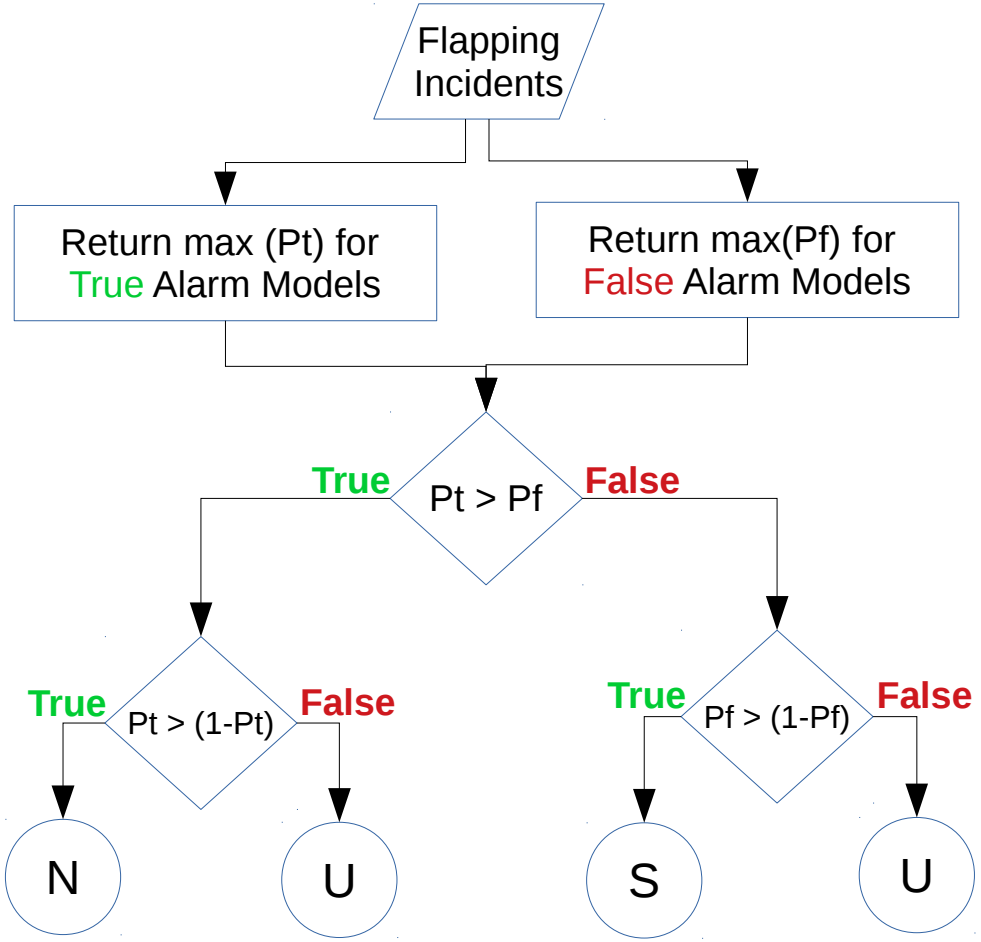
- Link flapping is a condition where a link speed alternates between discrete values e.g. 1Gb → 100Mb → Off → 1Gb
- It can be caused by:
  reboots, power-saving features, incorrect duplex configuration or signal integrity issues due to defect of physical connection
- Counting transitions within time window is often used method. However, it did not work well in our case, because time span of the pattern may not fit within window or number of transition can be as low as 2



Link Flapping

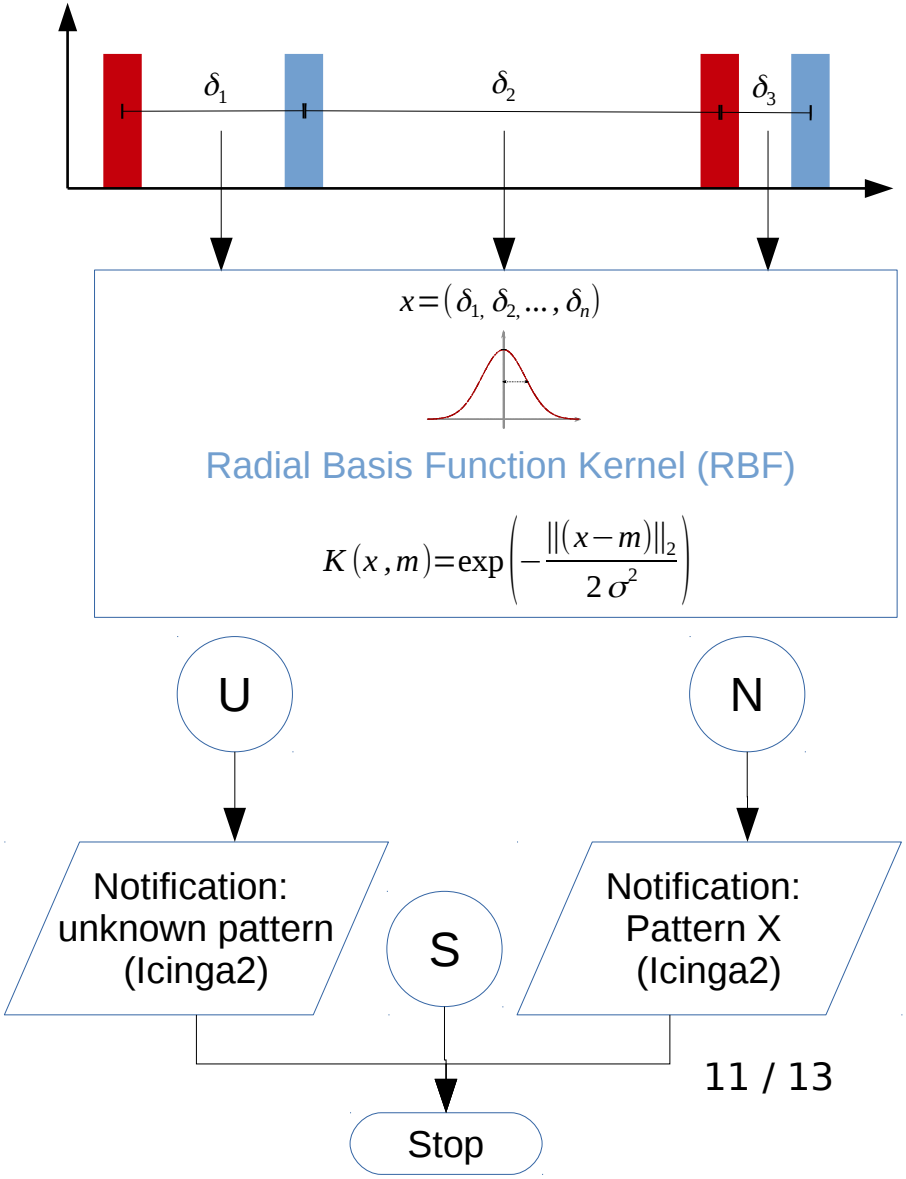
Decision Tree and Model

Trap incidents are provided in a form of variable length vectors of time differences. Those vectors are compared against various model, gathered in two groups: True alarms (modeled patterns of understood problems) and False alarms (modeled patterns of false incidents e.g. machine reboot)



Models must be created for each type of motherboard/NIC, switch, trap. Total number of models is estimated to be around 30.

However, in case of new failure pattern, a new model might be created.



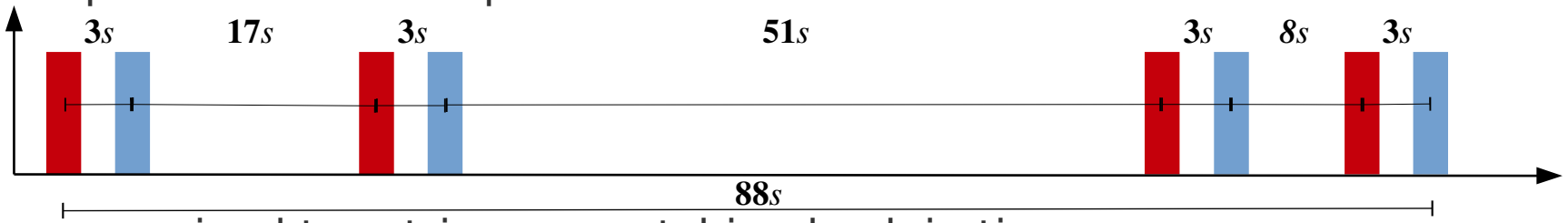
Link Flapping

Reboot Model

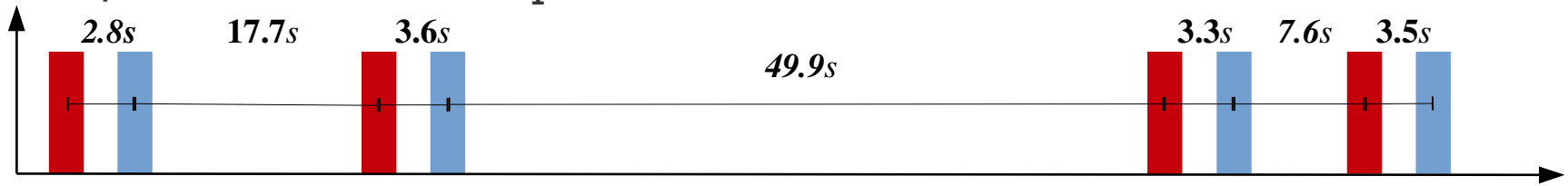
Radial Basis Function Kernel (RBF):

$$K(x,m)=\exp\left(-\frac{\|(x-m)\|_2}{2\sigma^2}\right)$$

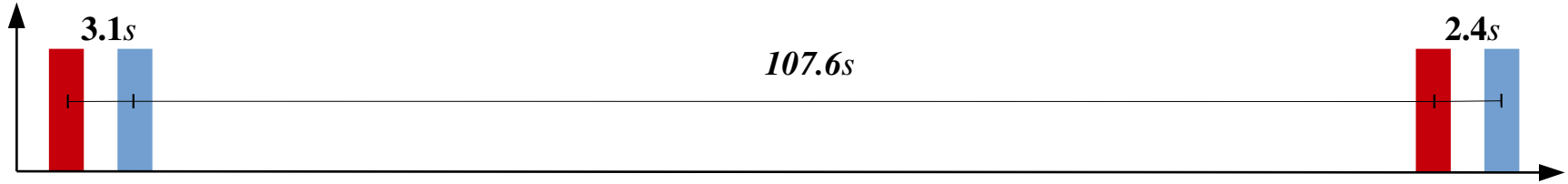
- The models was trained by rebooting machine and recording traps generated by a switch
- Example incident for Supermicro X10DRi motherboard:



- Every received trap triggers matching back in time
- Impact of different switch model as well as different firmware version is under evaluation process
- Example of estimated reboot models:
  - Supermicro X10DRi (σ_1= 1.63)**:**



  - Supermicro X10DRW-iT (σ_2= 2.27):



  - Similarity between models below … 0.01 %

Summary

- The right tool for the right task… simple machine learning techniques are still very useful
- Exploiting network homogeneity reduces amount of false alarms
- Many independent models improve extendability and maintenance
- Work still in progress, it is a step towards development of Root Cause Analysis (RCA) tool