Antagonist

https://datatracker.ietf.org/doc/draft-netana-nmop-network-anomaly-semantics/
https://datatracker.ietf.org/doc/draft-netana-nmop-network-anomaly-lifecycle/

IETF 121 – Dublin

NMOP WG

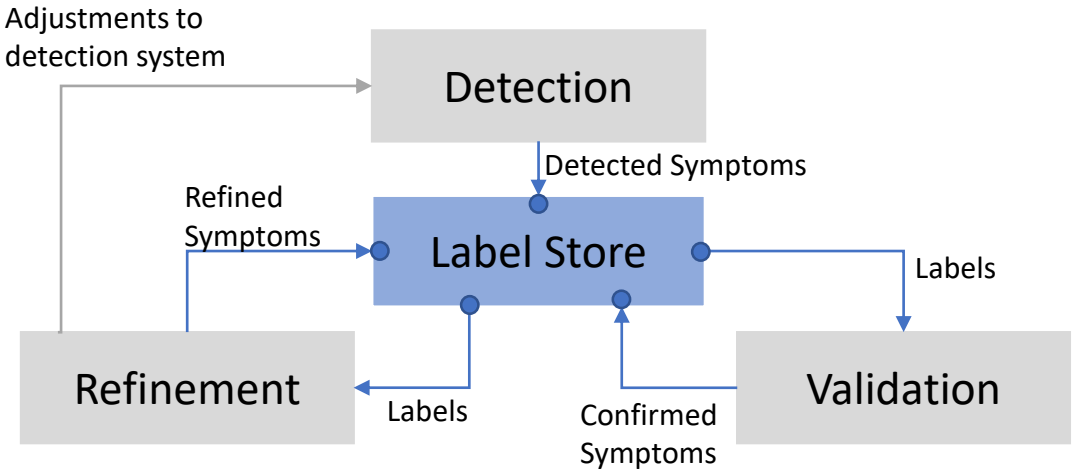
5th November 2024

Presenter: Vincenzo Riccobene

Team: Vincenzo Riccobene (Huawei), Thomas Graf & Wanting Du (Swisscom), Alex Huang Feng (INSA Lyon), Antonio Roberto

Antagonist – ANomaly TAGging ON hISTorical data

- Antagonist is a **Label Store for network anomaly detection** (https://github.com/vriccobene/antagonist)
- It addresses two main challenges in the network anomaly detection domain:
  - The **creation of labelled datasets** to train and evaluate anomaly detection algorithms and technologies
  - The **support for the human-in-the-loop paradigm** for what concerns the automated anomaly detection process.



Detection: Continuous monitoring of the network through Network Telemetry [RFC9232] and runtime identification of symptoms.

Validation: Verify if the detected symptoms are actually associated to a network incident or issue.

Refinement: Adjustment of the detection mechanism, to improve accuracy. This can include: update of the symptom expressions for SAIN, and retraining AI-based Network Anomaly Detection models

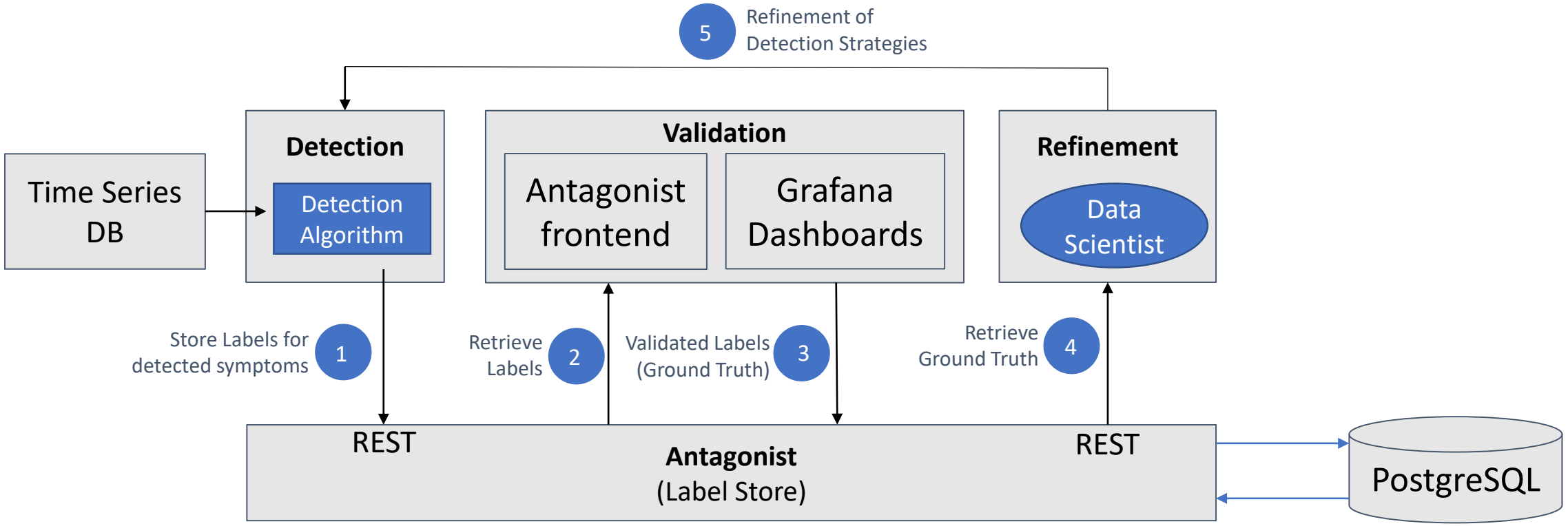
Goals of the project

- Define and validate suitable data models for label exchange between different actors and detection stages (API of the label store)
- Validate the data model in a wide set of use case scenarios
- Validate the data model with real network data

Work previously done on Antagonist

Antagonist – Development of Backend and Frontend to enable network experts to provide labels

- ✓ Implement Anomaly Label persistency and retrieval on PostgreSQL
- ✓ Implement REST API
- ✓ Integrate with timeseries data (via plugin mechanism)
- ✓ Implement Frontend GUI for data exploration, validation and refinement
- ✓ Implement automatic dashboard generation



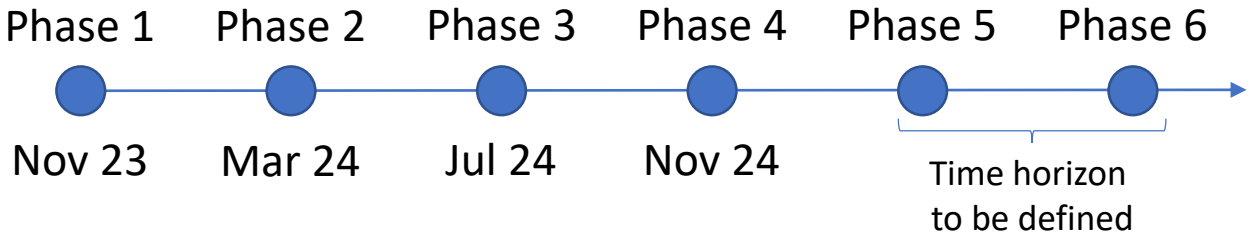
Experiment Roadmap

Types of detection algorithms

- ✓ Validation of Human label management
- ✓ Validation of ML-based anomaly detection
- ✓ **Validation of rule-based anomaly detection**
- ✓ **Formalize connection between metrics and symptoms**
- ✓ **Improve YANG data model**
- ❑ Integrate Antagonist in Swisscom lab

Roadmap

- ✓ Phase 1: Implement very basic PoC for data retrieval, API exposure and GUI (based on Grafana)
- ✓ Phase 2: Enhance GUI and extend API
- ✓ Phase 3: Validate the PoC with AIOps related data and a ML-based anomaly detector
- ✓ **Phase 4: Finalize validation of the PoC with SAIN [RFC 9417-9418] (as a rule-based anomaly detector)**
- ❑ Phase 5: Integrate with Swisscom Lab
- ❑ Phase 6: Finalize YANG data model



What was done for this hackathon iteration

- ✓Review YANG data models for the Label Store REST API
- ✓Created a new YANG data model for the notification of relevant states
- ✓Extended YANG main data model by mean of augmentation to support the different analysed use cases

What we learned

- The validation of the data model for SAIN [RFC 9418] (rule-based) detection system has been finalized.
- Taking into account all the experiments, it has been realized that there is a need to **make the semantic "pluggable"** in the data model, as different anomaly detection approaches might have different semantics.
- This has led to some significant changes to both the code and the drafts:
  - a clearer decoupling between the lifecycle and the semantic documents has been introduced
  - The API of the Label Store has been re-designed to allow integration of "pluggable" semantic
  - Identification of the need to open up the schema (which will require some more refactoring of the persistency layer).
- Also, the need for a clear indication of the monitored entity (service) was identified and addressed by using the same pluggable mechanism.

Extensions to the basic relevant-state data model are provided in the form of augmentation of the "symptom" and "service" containers

```
module: ietf-relevant-state
  +--rw relevant-state
     +--rw id                yang:uuid
     +--rw description?      string
     +--rw start-time        yang:date-and-time
     +--rw end-time?         yang:date-and-time
     +--rw anomalies* [id version]
        +--rw id                              yang:uuid
        +--rw version                         yang:counter32
        +--rw state                           identityref
        +--rw description?                    string
        +--rw start-time                      yang:date-and-time
        +--rw end-time?                       yang:date-and-time
        +--rw confidence-score                score
        +--rw (pattern)?
        |  +--:(drop)
        |  |  +--rw drop?                     empty
        |  +--:(spike)
        |  |  +--rw spike?                    empty
        |  +--:(mean-shift)
        |  |  +--rw mean-shift?               empty
        |  +--:(seasonality-shift)
        |  |  +--rw seasonality-shift?        empty
        |  +--:(trend)
        |  |  +--rw trend?                    empty
        |  +--:(other)
        |     +--rw other?                    string
        +--rw annotator!
        |  +--rw name                  string
        |  +--rw (annotator-type)?
        |     +--:(human)
        |     |  +--rw human?          empty
        |     +--:(algorithm)
        |        +--rw algorithm?      empty
        +--rw symptom!
        |  +--rw id                    yang:uuid
        |  +--rw concern-score         score
        +--rw service!
           +--rw id    yang:uuid
```

Presence containers

Example of semantics

Cosmos Bright Lights Detector

```
augment /rsn:relevant-state/rsn:anomalies/rsn:symptom:
  +--rw action?              string
  +--rw reason?              string
  +--rw cause?               string
  +--rw (plane)?
     +--:(forwarding)
     |  +--rw forwarding?    empty
     +--:(control)
     |  +--rw control?       empty
     +--:(management)
        +--rw management?    empty
```

```
augment /rsn:relevant-state/rsn:anomalies/rsn:service:
  +--rw vpn-service-container
     +--rw vpn-service* [vpn-id]
        +--rw vpn-id       string
        +--rw vpn-name?    string
        +--rw site-ids*    string
```

```
augment /rsn:relevant-state/rsn:anomalies/rsn:service:
  +--rw vpn-node-termination-container
     +--rw vpn-node-termination* [hostname route-distinguisher]
        +--rw hostname                inet:host
        +--rw route-distinguisher     string
        +--rw peer-ip*                inet:ip-address
        +--rw next-hop*               inet:ip-address
        +--rw interface-id*           int32
```

SAIN (RFC 9418) Detector

```
augment /rsn:relevant-state/rsn:anomalies/rsn:symptom:
  +--rw symptom-name?       string
  +--rw relevant-metrics* [name]
     +--rw name    string
augment /rsn:relevant-state-notification/rsn:anomalies/rsn:symptom:
  +-- symptom-name?       string
  +-- relevant-metrics* [name]
     +-- name    string
```

```
augment /rsn:relevant-state/rsn:anomalies/rsn:service:
  +--rw subservice-id?     yang:uuid
  +--rw subservice-type?   string
augment /rsn:relevant-state-notification/rsn:anomalies/rsn:service:
  +-- subservice-id?     yang:uuid
  +-- subservice-type?   string
```

ML-based Detector

```
augment /rsn:relevant-state/rsn:anomalies/rsn:symptom:
  +--rw (behaviour)?
     +--:(too-high)
     |  +--rw too-high?         empty
     +--:(too-low)
     |  +--rw too-low?          empty
     +--:(missing-value)
        +--rw missing-value?    empty
```

```
augment /rsn:relevant-state/rsn:anomalies/rsn:service:
  +--rw metric-id?    yang:uuid
  +--rw hostname?     string
```

Discussion Points

Next steps:

- Refactor API Implementation based on new version of the data model
- Extend Persistency layer to accommodate flexibility of the models
- Integrate with Swisscom Lab

Open Questions:

- We are looking for feedback:
  - **Do you see anything that would require further extension based on your post-mortem or detection internal processes?**
- We are looking for cooperation:
  - **We are looking for more operators that want to validate anomaly detection. Want to work with us?**
  - **Is there any use case scenario that you would like to test / integrate with Antagonist (maybe at the next hackathon)?**