TSDN-Enabled Network Assurance: A Cognitive Fault Detection Architecture

Danish Rafique, Thomas Szyrkowiec, Helmut Griesser, Achim Autenrieth, and Joerg-Peter Elbers

ADVA Optical Networking SE, Munich, Germany, drafique@advaoptical.com

Abstract We propose and demonstrate a cognitive fault detection architecture for intelligent network assurance. Our framework both detects and identifies significant faults, and outperforms conventional fixed threshold-triggered operations, both in terms of detection accuracy and proactive reaction time.

Introduction

Network assurance is a key operational requirement of modern optical communication systems. Traditionally, this has been achieved by introducing redundancy in network architectures, service level agreements (SLAs) based on conservative designs, etc. resulting in high capital expenditure. On the other hand, with the inception of 5G technologies – and subsequent front-haul and back-haul dynamic connectivity requirements, the task of network assurance is becoming increasingly complex, and necessitates novel architectures incorporating automated, flexible and reliable – potentially open – network management solutions. In this context, software-defined networking (SDN) proposes centralized network controllers to enhance network programmability by separation of data and control plane. Likewise, network monitoring approaches have been reported using distributed and centralized frameworks[1].

However, typical network management functions make use of pre-determined threshold-based triggers for configuration, restoration, planning, etc. This often leads to underutilization of network resources due to pessimistic design conditions. Furthermore, service and network behaviour can evolve in an unpredictable manner, and catering such faults using fixed thresholds not only exposes the network to unacceptable SLA breaches, but is also a non-scalable approach – with increasing network complexity. While anomaly-based virtual service and capacity planning has been proposed in literature[2,3], network assurance involving automated fault detection remains largely unexplored.

In this paper we propose a transport SDN (TSDN)-integrated cognitive fault detection architecture, incorporating data analytics based on advanced machine learning methods. In particular, we disclose various types of real-life network fault use cases, extracted from our sample customer network, identify them using the proactive fault detection (PFD) framework, and compare its efficiency with traditional condition-based set point detection.

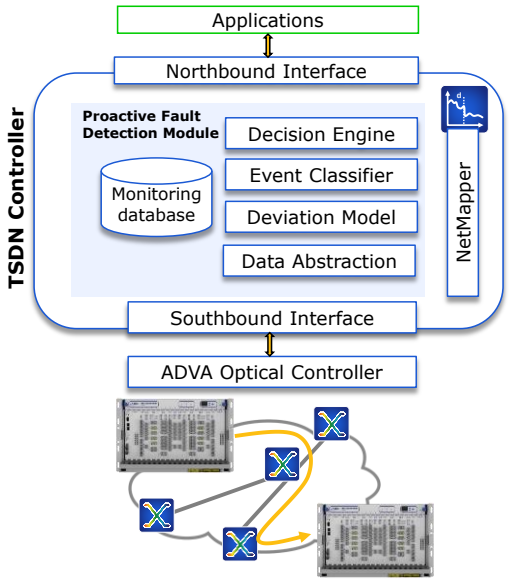

Fig. 1: TSDN controller with PFD architecture. Underlying network/service (arrow) is exemplary, and representative of sample network. PFD: Proactive fault detection

Overall Architecture

Our proposed architecture aims to proactively detect potential failures in real-time, replacing the static pre-defined fault thresholds. Fig. 1 depicts the software architecture for our approach, where the PFD framework is located within the TSDN controller. The monitoring data is collected every 15minutes through the southbound interface (SBI) – NETCONF, which is abstracted and stored in a database. The engine performs fault detection and classification, generates fault layer information (by mapping the metadata), fault locations, and maps the machine learning outcome to an internal decision engine followed by respective application (not specified here), exposed via RESTCONF to the northbound interface (NBI). The optical controller is an ADVA controller, whereas hardware components – in the given example –, comprise of ADVA FSP3000 network elements. Note that we consider a TSDN-integrated approach for our framework, however, the proposed architecture may be disintegrated, decentralized or used as part of an orchestrator as well. Furthermore,

while we test the proposed approach on features monitored at layer 0, the proposed cognitive architecture is scalable, and may be reused with, for example, packet flows (using OpenFlow, etc.).

Algorithm and Fault Types

Tab. 1 details the cognitive fault detection and classification approach. The goal of this procedure is to retrieve and process data blocks (partitioned data), make abnormality decision, prune the points which are insignificant, send the results to the next cycle, and eventually decide on true positives. The algorithm takes as input the monitoring data, and outputs labels for normal and abnormal operation. The details of the steps are as follows: The core of PFD framework traverses through the monitored data, and applies block based deviation and classification tests for a given set. The decision engine predicts true abnormal behaviour using neural networks based classifier, trained using historical fault patterns. Fig. 2 shows the implemented workflow. The fault types typically observed in commercial networks are classified in Tab. 2. *The reported fault categories are feature agnostic, and applicable to multiple layers in communication stack*. Network faults can take many forms including a spike due to a short-lived event, a gradual change in behaviour, a state change due to certain configuration changes, and finally localized abnormalities indicating potential faults.

Proof-of-Concept and Discussions

The experiment was carried out using diverse configurations extracted from ADVA's sample customer network, where various patterns were simulated, as discussed in Tab. 2. The labels represent abnormal behaviour which may or may not lead to potential failures. The physical layer received optical power levels were pulled in real-time via the optical controller, and the proposed

Tab. 1: Algorithm for cognitive fault detection framework, processing monitored data as an input and generating proactive fault information as output

INPUT: $\boldsymbol{X}_t^w \leftarrow metrics\ (\boldsymbol{X}_1^w{:}\boldsymbol{X}_t^w)$,
$\mathcal{L} = \langle \ell_1, \ell_2, \ldots \ell_m \rangle \in \mathbb{R}$
$where\ \boldsymbol{X}_t^w {:}\Leftrightarrow \{\boldsymbol{X}_t^w {:} w, t \in \mathbb{Z}^+\} \in \mathbb{R}$
OUTPUT: $\mathbf{R}{:}\Leftrightarrow\{r_k{:}k \in \mathbb{N}\}$

1. *for all blocks in* $\boldsymbol{X}_t^w$ *do*
2. $K \leftarrow getBlock\ (\boldsymbol{X}_t^w)$
3. $\forall\ k \in K\ do\ \rightarrow\ c_k \equiv \emptyset$
4. *for all point* $k \in \boldsymbol{X}_t^w$
5. $D_k \equiv DeviationCalc()$
6. $c_k \equiv ClassificationTest()$
7. $c_K \leftarrow (c_1{:}c_k)$
8. $\boldsymbol{C} \leftarrow (c_1{:}c_K)$
9. $R \equiv getDecision(\mathcal{L}, C)$
10. $f{:}\boldsymbol{R} \rightarrow \boldsymbol{Y}[Configuration, Alarm, Service]$
11. *defined as NetMapper*$(\boldsymbol{R})$

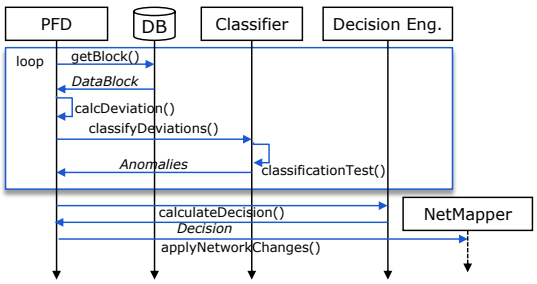

Fig. 2: Sequence flow implementation

architecture was executed on a host server. The monitored data was normalized to same scale (-1dBm) for better visualization, and this action did not have any impact on the performance. We evaluated two scenarios: one employing the PFD engine, where the true potential failure scenarios were cognitively predicted, and other where PFD was replaced by a fixed pre-determined fault determination threshold. Note that since we normalized our data, a single threshold was used for comparison, wherease in practice several threshold are required to be *defined* and *maintained* for various network configurations.

Fig. 3 shows various fault patterns, as described in Tab. 2, for different time traces. Fig. 3a depicts the performance of threshold-triggered fault detection, where label I is fully detected, labels' III and IV are only partially detected, whereas label II remains undetected. In comparison, Fig. 3b illustrates results from the proposed cognitive architecture, where all labels are largely detected, except for three (one) potential faults in label II (IV). It is worth mentioning that while the detection rate of faults is an important metric, the fundamental operational requirement is the ability to react to such potential failures in time.

From Tab. 3 it can be observed that our proposed architecture outperforms conventional set-point based detection, both in terms of detection rate and associated response time (defined as time from detection to typical alarm level) – for any given fault label. Specifically, label IV may be adequately addressed before actual failure, with a response time of 96hrs, as opposed to no

Tab. 2: Definitions for typical network fault patterns

Fault Label	Description
I	Point abnormalities due to random flash events and may lead to abrupt device damage
II	Local abnormalities indicating potential flaws with potential long-term impact on service performance
III	Steady abnormalities due to preceding system configuration changes, and may lead to damage and/or consistent performance loss
IV	Ramp abnormalities representing gradual system and/or service distortion possibilities

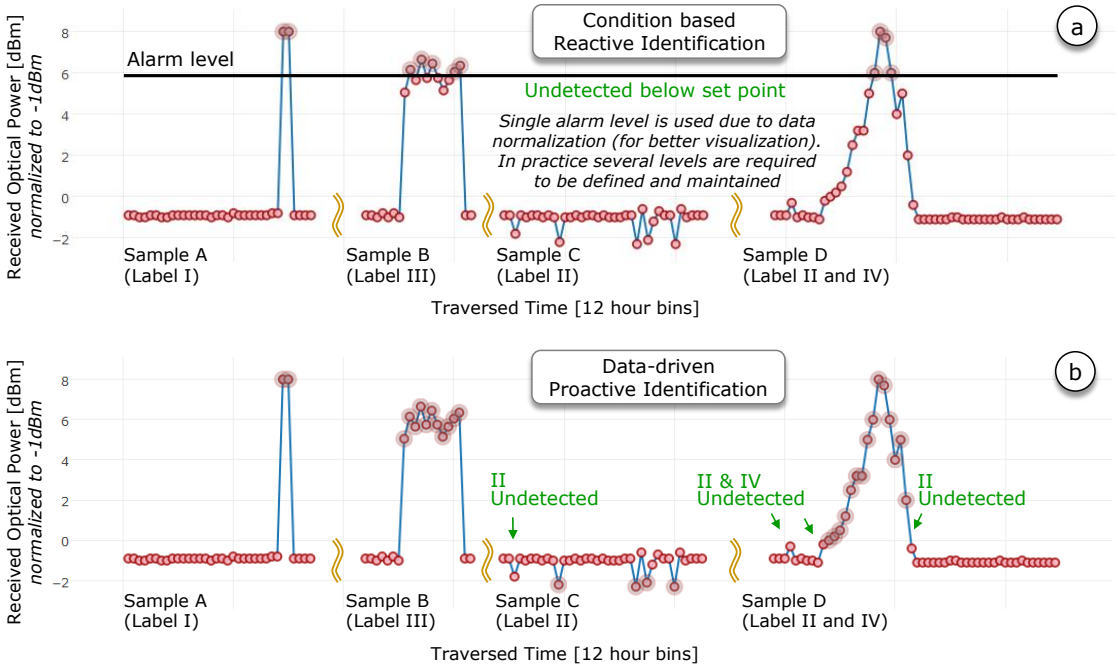

Fig. 3: Monitored layer 0 optical received power levels as a function of traversed time (aggregated to 12 hour bins). a) Condition based fault detection, b) Data based fault detection (PFD engine). Highlighted symbols (opacity) represent detected faults. Broken lines represent different data samples. For fault label definitions see Tab. 2

response time at all from threshold-based detection. Likewise, label III alarms may be issued at least 10hrs prior to failure, whereas reaction time to label II depends upon its evolution behaviour. While label I is detected by both methods, it allows no reaction time due to its peak response. Finally, the trade-off between accuracy of label detection and true predicted faults is shown in the last column, where a ratio of 1 shows 100% detection of actual faults predictors (determined based on a NN model). Qualitatively this means whether the decision engine incorporated in PFD framework correctly labels significant and nonsignificant fault behaviours, based on historically identified fault patterns. It can be seen that while PFD performs exceedingly well, compared to condition-based method, label IV leads to minor over-detection.

Tab. 3: Quantitative summary of Fig. 3 (first two data columns), showing performance evaluation of data-driven and condition-based methods. Third data column shows ratio of detected labels in Fig. 3, and actual (true) fault predictors (determined by the decision engine)

		Fault Label Detection Rate [%]	Proactive Reaction Time [hours]	Detected / True Faults
Condition-based	Label I	100	0	1
Data-driven		100	0	1
Condition-based	Label II	0	0	0
Data-driven		~57	>100*	0.57
Condition-based	Label III	~45	0	0.45
Data-driven		100	10	1
Condition-based	Label IV	25	0	.25
Data-driven		~93	96	1.25

*assuming label IV-like symmetric behavior

Conclusions

We reported a cognitive fault detection architecture – integrated in TSDN framework – for network assurance. The proposed framework not only allows for simpler network management, getting rid of multiple fixed set point *definition* and *maintenance*; but also significantly improves the proactive fault response time, compared to conventional threshold-based failure detection.

The goal of this work is to introduce a generic cognitive assurance architecture, neither limited to presented network configuration nor to the dataset, and application use cases across different network layers are underway.

Acknowledgements

This work was performed in the framework of the CELTIC EUREKA project SENDATE-Secure-DCI (Project ID C2015/3-4), and it is partly funded by the German BMBF (Project ID 16KIS0477K).

References

[1] A. Vela, et al., "Reducing virtual network reconfiguration and traffic losses under multiple traffic anomalies," Proc. ACP, AF3E.5, Wuhan, (2016).

[2] S. Yan, et al., "Multilayer network analytics with SDN-based monitoring framework," J. Opt. Commun. Netw., Vol. **9**, no. 2, p. A271 (2017).

[3] F. Morales, et al., "Incremental capacity planning in optical transport networks based on periodic performance metrics," Proc. ICTON, We.A3.2, Trento (2016).