An Architecture for Detection of Anomalies in Deterministic Time within Real-Time Communication Networks

Christian Maier, Jia Lei Du, Stefan Fahrthofer, Peter Dorfinger
Intelligent Connectivity
Salzburg Research Forschungsgesellschaft mbH
Salzburg, Austria
email: {christian.maier, jia.du, stefan.fahrthofer, peter.dorfinger}@salzburgresearch.at

***Abstract*—Anomaly detection is a classical and important topic within the domain of communication networks. For critical applications, the time it takes to detect an anomaly and to respond to it is an important metric of an anomaly detection system. To address this, we propose an end-to-end real-time anomaly detection architecture. In this architecture, the collection and transmission of the required data, the analysis of this data in a machine learning model and the subsequent reaction when an anomaly is detected, are carried out in deterministic time. We study two different use cases where this architecture may be applied in the future and we investigate a demo implementation of one of these use cases as a proof of concept for the proposed architecture. This contributes to the future application of machine learning for anomaly detection in deterministic time in time critical application areas.**

***Keywords*—machine learning; real-time Ethernet; anomaly detection.**

I. Introduction

Real-time communication networks and the application of Machine Learning (ML) methods to such networks is a current research area in the networking domain. Anomaly detection is one example where ML methods have been successfully applied to detect abnormal behaviour in (non-real-time) networks. With the increasing relevance of real-time communication networks in cyber-physical systems in critical domains, such as manufacturing and smart energy grids, solutions for applying ML approaches to real-time networks will become highly important in the future. In such a setting, particularly the timely detection of anomalies will play a significant role.

ML approaches can be roughly subdivided into the areas of supervised, unsupervised and reinforcement learning. One of the main advantages of unsupervised learning is that it is not required to manually label data sets in advance, making it particularly suitable for anomaly detection tasks. The ML models most commonly used for anomaly detection are Autoencoder Neural Networks (ANNs), which are a special type of an artificial neural network.

In this paper, we aim to combine these ingredients (real-time communication networks and ML) to provide an architecture for detection of anomalies in deterministic time through unsupervised ML in real-time Ethernet networks.

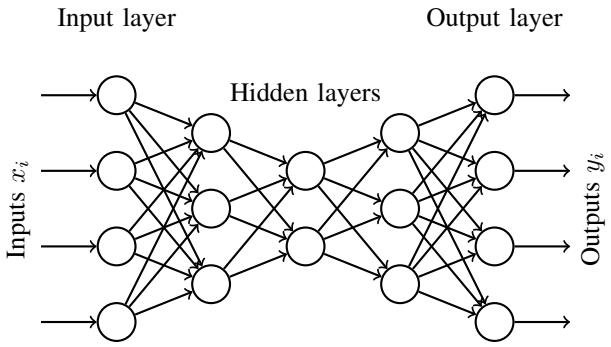

Fig. 1. Schematic overview of an ANN.

The remainder of this paper is organised as follows: Section II presents work related to this paper. The concept of an ANN is recalled in Section III. Section IV then introduces the proposed architecture and considers two use cases where this architecture can be applied. A demo implementation of one of these use cases is described in Section V. Section VI concludes the paper and gives hints to future work.

II. Related Work

Zhao et al. [6] propose a network anomaly detection system based on ML. This system operates in real-time. Unlike our work, however, they do not consider a scenario in which the network itself operates in real-time. A similar approach can also be found in [4].

Work related to this paper comes from real-time networks on the one hand and from anomaly detection via ML on the other hand. The demo implementation presented in this paper is based on the scenario considered by Farthofer et al. [5], which studies redundant disjoint paths in real-time Ethernet networks.

There is a growing amount of literature which investigates the application of ML techniques to problems in the domain of communication networks, and in particular to anomaly detection. Boutaba et al. [1] hints to classical approaches for anomaly detection systems and surveys on ML approaches for

 ISBN: 978-1-61208-926-3

anomaly detection. Casas [3] compares different ML models via their performance in the analysis of network measurements for detection of network attacks and anomaly detection. The survey in [2] investigates ML for cyber-security analytics.

III. Autoencoder Neural Networks

An ANN is a special type of a Multi-Layer Perceptron (MLP), consisting of an input layer, an output layer and several hidden layers $h_1, \ldots, h_k$ connecting them. Each hidden layer h_i consists of a certain number of neurons. In contrast to the usual structure of a MLP, in which most neurons are located in the centre of the network, an ANN becomes narrower towards the middle. Moreover, the number of outputs is equal to the number of inputs (Figure 1).

The goal of ANNs is to learn low dimensional representations of the input data $x_1, \ldots, x_n$. This is achieved by having at least one hidden layer, which has fewer neurons than the input layer. The error function is defined as

$$E = \sum_{i=1}^{n} (x_i - y_i)^2,$$

where $y_1, \ldots, y_n$ are the outputs of the ANN. Hence, the error function is a Mean Squared Error (MSE). The neural network thus learns to reproduce the inputs on the outputs. If this is achieved, the information on the thinnest hidden layer is the low dimensional representation of the input data. As we can observe, the learning process is of an unsupervised manner, which means that no labeling of the input data is required. This is an advantage, since a labeling process is usually elaborate.

An ANN can be interpreted as a composition $g \circ f$ of an encoding function f followed by a decoding function g. Here $f : \mathbb{R}^n \to \mathbb{R}^m$ is a function which computes the code $(z_1, \ldots, z_m) \in \mathbb{R}^m$ of the inputs $(x_1, \ldots, x_n)$. The function $g : \mathbb{R}^m \to \mathbb{R}^n$ restores the original vector $(x_1, \ldots, x_n)$ from this code.

The main issue with ANNs is that it is not clear a priori how many neurons are necessary in the thinnest hidden layer. On the one hand, one strives for a representation of the input data that is as low-dimensional as possible. On the other hand, the number of degrees of freedom in the input data is not known. For example, if there are indices $i \neq j$ such that x_i and x_j are uncorrelated, at least 2 neurons in the thinnest layer are necessary. This implies that one needs to figure out the number of correlations between the input data values $x_1, \ldots, x_n$. This can either be done from theoretical information of the input data (e.g., from physics, queueing theory, ..., depending on the type of data respectively use case) or in a purely experimental way. The latter can be done by performing training runs with different numbers of neurons and finally using the neural network with the lowest number of neurons that still has sufficient performance.

IV. Architecture

In this section, we propose an architecture for real-time detection of anomalies for security (intrusion detection) or safety (component failure prediction). We consider a scenario where an edge node sniffs traffic from the network. This traffic could either contain data about the network state itself (like delay values, jitter, packet loss, flow size, etc.) or data representing the state of machines/controllers connected to them. The traffic information is represented by an n-dimensional vector

$$\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n.$$

We assume that the real numbers x_i are normalised either to the unit interval or to have mean 0 and standard deviation 1. The vector $\mathbf{x}$ changes over time. Our aim is to examine these values for anomalies in the network or for hints to machines not working properly. To this end, we use an ANN at the edge node to learn low dimensional representations of the values $x_1, \ldots, x_n$. In other words, we use these values as inputs for the ANN and train it to reflect these values at the outputs. This training is done online. This means that we do not collect the data over an interval of time and train the network thereafter with the generated data set. Instead, we use the data immediately when it arrives at the edge node to update the weights and bias values of the neural network.

As outlined before, the ANN learns two functions: An encoder function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a decoder function $g : \mathbb{R}^m \to \mathbb{R}^n$. The m-dimensional vector $f(x_1, \ldots, x_n)$ is the code of the input vector $\mathbf{x} = (x_1, \ldots, x_n)$. The training stops if

$$E = \sum_{i=1}^{n} (x_i - y_i)^2 < \epsilon$$

for some chosen threshold value ϵ, i.e., if the MSE is sufficiently small. Here, $y_1, \ldots, y_n$ are again the output values of the ANN. After that, the system enters a monitoring mode. In this monitoring mode, the neural network continues to be fed with the values $x_1, \ldots, x_n$. If

$$E = \sum_{i=1}^{n} (x_i - y_i)^2 \geq \delta$$

for some second fixed threshold δ (with $\delta > \epsilon$, e.g., $\delta = (r + 1)\epsilon$ for some positive integer r), it is assumed that an anomaly has been detected. Then, for example, a reconfiguration of the network can be initiated to protect the network and the devices.

The critical point is now that if all the connections in the network are real-time connections, then the detection of anomalies will be real-time as well. This is due to the fact that the computation of the output values of neural networks consists exclusively of a fixed number of additions, multiplications and applications of activation functions. Hence this can be done in deterministic time as well. Moreover, in suitable systems, also the reaction to an anomaly may be done in deterministic time. This then provides a closed deterministic anomaly detection end-to-end loop. In the following, we provide two use cases for this architecture:

A. Use Case 1

An edge-node collects information from various devices in a communication network based on real-time Ethernet (see Figure 2). We assume that all nodes in this network operate

 ISBN: 978-1-61208-926-3

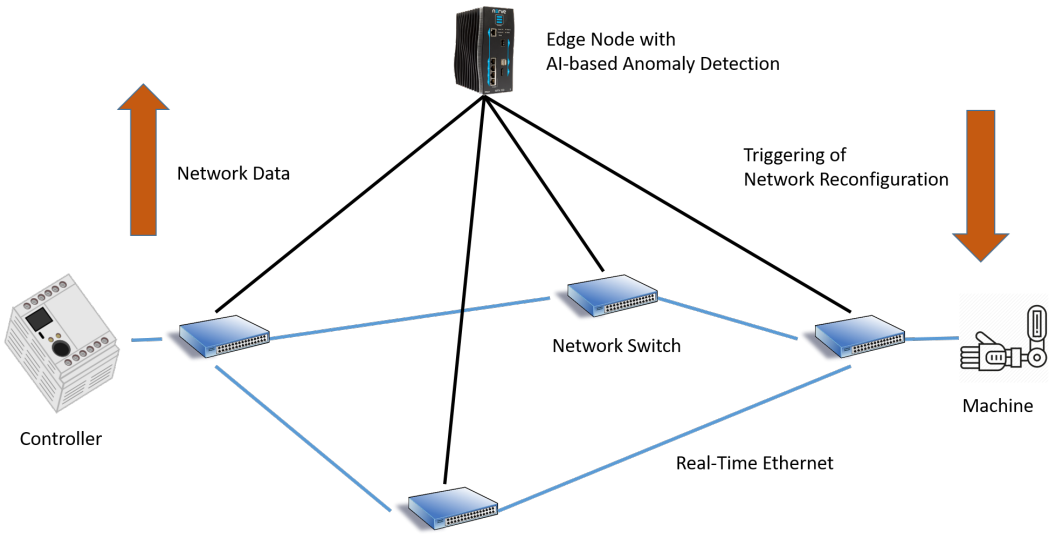

Fig. 2. Use Case 1: Anomaly detection of network data and real-time reconfiguration of the real-time Ethernet network

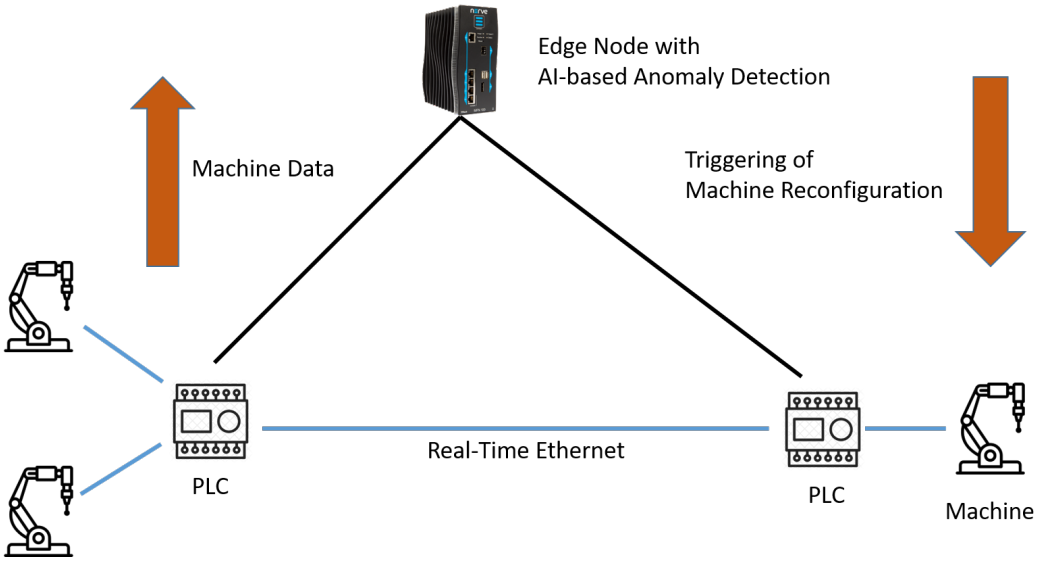

Fig. 3. Use Case 2: Anomaly detection of machine data and real-time reconfiguration of the industrial machines (network switches not explicitly shown in this figure)

in real-time as well. The network data may consist of delay values, jitter, traffic load and number or configurations of flows. The information is represented by an n-dimensional vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, which changes over time. An anomaly detection system based on an ANN at the edge-node is fed with the vector $\mathbf{x}$. As soon as an anomaly is detected, the edge-node could trigger building blocks based on Software Defined Networking (SDN) to reconfigure the network configuration. As an example, a flow, which uses a certain path p_1, could be reconfigured to use a path p_2 after an anomaly has been detected to circumnavigate an issue or attacker.

B. Use Case 2

In this use case, sensor values $x_1, \ldots, x_n$ of machines are sent over a real-time network to the edge node (see Figure 3). At the edge node, an ANN is trained online to reproduce the sensor values. A detected anomaly may for example indicate a machine error. And a quick detection and analysis in deterministic time could prevent damages to the involved machines or products within a previously specified reaction time.

V. Proof of Concept Implementation

A proof of concept implementation of the proposed architecture was realised in our laboratory (see Figure 4).

 ISBN: 978-1-61208-926-3

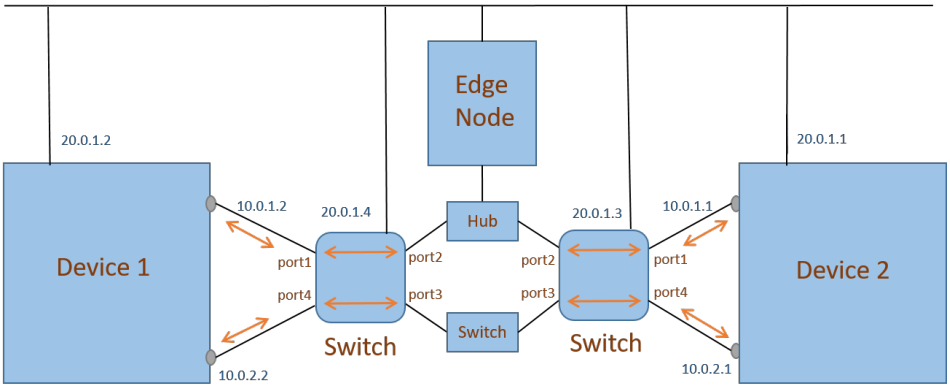

Fig. 4. Schematic overview of the realized proof of concept setup

Two real-time communication capable devices (representing controllers or machines) are connected through a real-time Ethernet network. To be able to demonstrate various scenarios, including for example fast failover, the devices are connected via two disjoint paths. In a separate network, an edge node running the ML algorithms (in our case the ANN) is connected to the network switches and devices to trigger network or controller/machine reconfigurations. Finally, the edge node is also directly connected to the real-time Ethernet network so it can mirror and listen into any communication between the two devices. For a demonstration, we simulated machine sensor/network information values by generating random real numbers $a_1, \ldots, a_m$ uniformly distributed in the interval $[0, 1]$ and by generating input values $x_1, \ldots, x_n$ of the form

$$(x_1, \ldots, x_n)^t = A \cdot (a_1^2, \ldots, a_m^2)^t + B \cdot (a_1, \ldots, a_m)^t + \mathbf{c}$$

Here $A, B \in \mathbb{R}^{n \times m}$ are randomly generated (but fixed) matrices and $\mathbf{c} \in \mathbb{R}^n$ is a fixed random column vector. This simulates the number of correlations between the values $x_1, \ldots, x_n$. Note that we thereby know the number of degrees of freedom, which is exactly equal to m. This number corresponds to the number of neurons needed in the thinnest layer of the neural network.

The neural network was implemented in Python, using the TensorFlow framework. We used a scenario with 16 network status information values $x_1, \ldots, x_{16}$. The number of degrees of freedom was set to 8. They were all normalised to have mean values 0 and standard deviation 1. The ANN consisted of 3 hidden layers, with 12 neurons in the first hidden layer, 8 neurons in the middle hidden layer and 12 neurons in the third hidden layer. The ANN achieved a performance of an MSE of 0.01 after an online training with $\approx 10^4$ generated samples $x_1, \ldots, x_{16}$. This MSE multiplied with 3 was then used for the monitoring mode as the offset value for the detection of an anomaly.

After an anomaly is detected by our ANN, a reconfiguration of network flows (e.g., shutting down a network path, switching to another network path) or a reconfiguration of the machines (e.g., new parameter settings) are triggered. This thus closes our deterministic anomaly detection loop, since all components operate in deterministic time.

VI. Conclusion and Future Work

We proposed an architecture for an end-to-end deterministic anomaly detection system in real-time networks. This is achieved via an online training of an ANN at an edge node. We provided two use cases where this architecture may be applied. We implemented the architecture in one of these use cases as a demo implementation, and thus delivered a proof-of-concept. In the future, measurements will be performed to evaluate the proposed architecture in terms of the actual reaction time from anomaly detection to network or machine reconfiguration. The results will then be compared to measurements in existing anomaly detection systems.

Acknowledgment

This research was partially funded by the Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) under the program “ICT of the Future” (https://iktderzukunft.at/en/) and by the WISS 2025 (Science and Innovation Strategy Salzburg 2025) project “IDALab” (20102-F1901166-KZP).

References

[1] R. Boutaba et al., “A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities,” Journal of Internet Services and Applications, 9(1), pp. 1—99, 2018.

[2] A. L. Buczak and E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” IEEE Communications Surveys & Tutorials, 18(2), pp. 1153—1176, 2016.

[3] P. Casas, “On the Analysis of Network Measurements Through Machine Learning: The Power of the Crowd,” Network Traffic Measurement and Analysis Conference (TMA), pp. 1—8, IEEE, 2018.

[4] Y. Du, J. Liu, F. Liu, and L. Chen, “A Real-time Anomalies Detection System based on Streaming Technology,” Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics, vol. 2, pp. 275—279, IEEE, 2014.

[5] S. Farthofer, D. T. B. Lima, and J. L. Du, “Application Layer Benefits of Redundant Disjoint Paths in a Real-Time Ethernet,” International Conference on Information and Communication Technology Convergence (ICTC), pp. 161—165, IEEE, 2020.

[6] S. Zhao, M. Chandrashekar, Y. Lee, and D. Medhi, “Real-time Network Anomaly Detection System using Machine Learning,” 11th International Conference on the Design of Reliable Communication Networks (DRCN), pp. 267—270, IEEE, 2015.

 ISBN: 978-1-61208-926-3