

# On the use of streaming telemetry data for network health monitoring and anomaly detection

Experiences from the AI-NET-PROTECT field trial

Karoly Makonyi<sup>\*</sup>, Henrik Abrahamsson<sup>†</sup>, Daniel Henriksson<sup>‡</sup>, David Hock<sup>§</sup>,  
Stefan Kremling<sup>§</sup>, Fabian Lipp<sup>§</sup>, Jonathan Salisbury<sup>\*</sup> and Johan Sandell<sup>¶</sup>

<sup>\*</sup>Savantic

<sup>†</sup>RISE

<sup>‡</sup>Lunet

<sup>§</sup>Infosim

<sup>¶</sup>Waystream

**Abstract**—Traditionally, the Simple Network Management Protocol (SNMP) has been used for network monitoring. Today, streaming network telemetry is a better choice for high-speed, high-resolution performance data, enabling AI-based anomaly detection and opening up for proactive maintenance. However, there is a scalability challenge in using streaming telemetry data since it can involve many parameters, high data collection frequency, and large amounts of data. This paper first highlights a use case where telemetry data outperforms traditional SNMP. The paper then presents a data collection pipeline and experiences from a field trial in a fiber access network where the target was to increase network availability as well as to simplify operations. The field trial shows that it is feasible to collect streaming telemetry data from more than a thousand switches and continuously monitor and visualize the network health status.

## I. INTRODUCTION

Communication networks are today vital for society and network resilience is therefore crucial. Disturbances or outages can have a dramatic impact on those affected. Network health monitoring is important to provide early warnings and enable proactive fault and anomaly detection.

AI-NET-PROTECT<sup>1</sup> is a CELTIC-NEXT flagship project with 36 partners from four countries. The project objective is to provide automated resilience and secure networks. The results highlighted in this paper comes from collaboration between Savantic, Lunet, Waystream, Infosim and RISE. We investigate scalable solutions for network performance metric collection, and the use of streaming telemetry data for network health monitoring and anomaly detection. This is an industrial paper with focus on practical implementation and evaluation.

Traditionally, the Simple Network Management Protocol (SNMP) has been used for network monitoring. Today, streaming network telemetry is a promising alternative for collecting detailed performance data on shorter timescales. In Section II we explain the difference between SNMP versus streaming network telemetry, and highlight a use case where telemetry data outperforms traditional SNMP.

Using streaming network telemetry to collect high-speed, high-resolution performance data can enable AI-based anomaly detection and proactive maintenance. However, in practice, when deployed in a real network with hundreds of switches, there is a scalability challenge in using streaming telemetry data to monitor hundreds or thousand of parameters with high data collection frequency. In Section III we present a scalable data pipeline that can collect telemetry data from a large number of access switches. We present results and experiences from a live network field trial in a fiber access network. The field trial shows that, with our data pipeline, it is feasible to collect streaming telemetry data from more than a thousand switches and continuously monitor and visualize the network health status.

## II. SNMP VERSUS STREAMING NETWORK TELEMETRY

In network management, there are two fundamental different paradigms for acquiring data from network devices. The first is a pull system, which uses well-established protocols such as the Simple Network Management Protocol (SNMP) to periodically poll devices in order to gather data about their states (Figure 1a). This occurs at regular, pre-determined intervals, usually in the range of several minutes. The SNMP protocol is now over 30 years old and the need for alternatives has increased due to the growing complexity of modern networks. Further, the SNMP protocol is criticized for its low granularity and the inevitable resources required from devices that are constantly being polled.

In contrast, network streaming telemetry uses a radical different approach by which network devices continuously stream a vast amount of data in near real-time. The Network Management System (NMS) or any other data collector can then subscribe to the data stream and specific metrics (Figure 1b). In the telecommunications industry, having access to a more granular and near real-time view of metrics could make the difference to quickly detect bottlenecks or hotspots and diagnose the problem more effectively. Finally, this leads to a consistently high quality of service. With telemetry, near-

<sup>1</sup><https://protect.ai-net.tech>

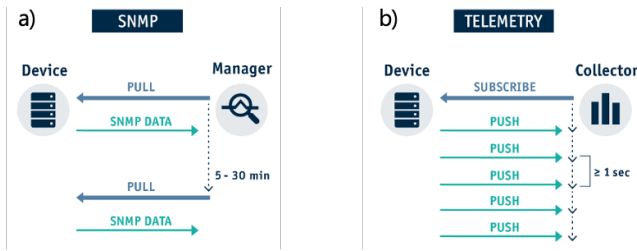


Fig. 1: a) SNMP (pull-based) and b) telemetry (publish-subscribe based) data acquisition

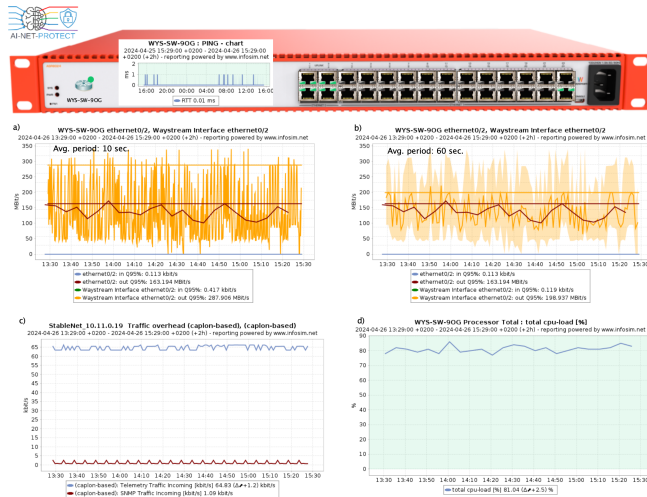


Fig. 2: Dashboard from AI-NET-PROTECT Demo Setup. a) SNMP 5 min. (red) and telemetry 10s (yellow) measurement of an interface. b) same measurement as in a) but with an average of the telemetry measurement to 60s. c) network traffic related to snmp measurement (red) and telemetry measurement (blue). d) total cpu load for the device during operation.

real time data and lower device-side resource requirements are certainly important improvements.

We investigated differences between SNMP and telemetry using a Waystream access switch that include streaming telemetry data covering system parameters, resource utilization, interface data and service performance. In total, we monitored 6 interfaces (in/out octets, in/out errors, in/out discards, op status) using both, snmp and telemetry. For each interface we also monitor sfp parameters (rx/tx power, bit rate and temperature) via telemetry only. We also monitor environmental measures (4 temperature and 4 fan sensors) via telemetry as well as the system load. Processor metrics (CPU load, uptime) are measured via SNMP. For a comprehensive study we varied the polling interval for SNMP and the telemetry streaming frequency of the device and investigated the influences and effects on device and network resources. The test setup allows also the creation of specific traffic load patterns, which can be used to assess the impact on short-term event detection accuracy. Figure 2 depicts the device and a dashboard from a demo setup. Figure 2 a)

illustrates the utilization of the Ethernet0/2 interface for a total measurement range of 2 hours. The red curve represents the SNMP measurement with a sampling interval of 5 minutes, while the yellow curve depicts the telemetry measurement with a sampling interval of 10 seconds. It is evident that the higher resolution of the telemetry measurement can capture the traffic peaks more effectively. Figure 2 b) illustrates the same measurement as in Figure 2a) but with an average period for the telemetry measurement of 60 seconds. Figure 2 c) shows the total network traffic associated with both measurement methods, with the blue curve representing telemetry and red curve representing SNMP.

### III. EXPERIENCES FROM THE LUNET FIELD TRIAL

Lunet builds and operates the optic fiber network in the city of Luleå. The city network currently consists of more than 1500 (Waystream) access switches that serve more than 22000 households. The network has a ring topology where network switches are connected in access node rings through 10 Gbit/s connections, which in turn are connected to core nodes. Downstream from each switch there are 1 Gbit/s connection to the households.

#### A. Collecting telemetry data at scale

Continuously collecting high-resolution telemetry data can provide operators with a detailed view of the state of the network, and opens up for AI-based anomaly detection and proactive maintenance. However, for deployment in a real network with many network elements, there might be a scalability challenge in using streaming telemetry data to monitor hundreds of parameters with high data collection frequency. To investigate this, we have conducted a field trial in the Lunet network. We have implemented an efficient data pipeline, and we have gradually connected more and more access switches. We monitor 1500 parameters every 10 or 60 seconds depending on their configuration. The field trial shows that, with our data pipeline, it is feasible to collect streaming telemetry data from more than a thousand switches and continuously monitor and visualize the network health status.

#### B. Implementing an efficient data pipeline

The data pipeline is outlined in Figure 3. It is build on four open-source components: Kafka, Telegraf, PostgreSQL, and Apache Superset. Kafka is an open-source distributed event streaming platform. Telegraf is a server-based agent for collecting and sending metrics and events. PostgreSQL is relational database and Apache Superset is a platform for data exploration and visualization.

The data is collected by Telegraf clients and aggregated by Kafka Streams. The data from the Kafka topics are received by a Python script that resamples the data for 1 minute aggregation time and filters it to have the quantities that are necessary for the outlier detection (up- and downstream bit rates, drop-rates, packet size distributions, number of unicast and multicast packets, and more). The selected and pre-processed data is

fed back to Kafka and then consumed by a processing unit that groups and runs outlier detection algorithms on the data. The resulting outlier score together with the above mentioned selected/resampled data is stored in a PostgreSQL database. With this data pipeline presented above it is feasible to collect streaming telemetry data from more than a thousand switches using a single server, however for scalability purposes every part in the pipeline is modular, containerised and can be duplicated to remove bottlenecks and increase the potential throughput of the pipeline (as an example: due to the large number of read out servers a bottleneck was hit on the parsing of the raw telemetry data but all that was needed to alleviate this was to spin up another instance of the parser and split the flow in two at that point). Because of the modular and containerised architecture there is a very high theoretical limit on the throughput on the pipeline if it were upgraded even though most of the parts are relatively slow and un-optimised Python scripts.

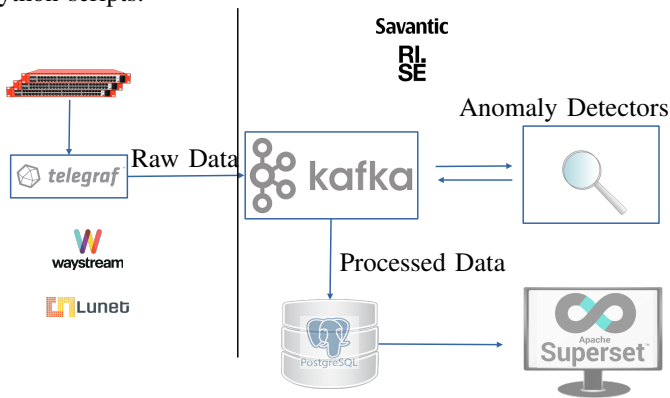


Fig. 3: Data collection pipeline for monitoring and visualization. The switches are provided by Waystream, the Telegraf clients were configured by Lunet. The rest of the pipeline, included the outlier detector algorithms were developed by RISE and Savantic.

In addition to that the resulting anomaly scores are displayed for help to the operators to monitor the status of the network and identify possible errors more efficiently via representation of the data (outlier scores) visually. This is the main reason to attach a visualisation unit on the top of the above mentioned data-pipeline, using Apache Superset. A visualisation unit fetches data from the PostgreSQL database and creates a visualized representation of the data. Examples from the dashboard are shown in Figure 5.

Two sets of unsupervised anomaly detection algorithms were employed in the study [1]. The first set focused on univariate outlier detection. A forecasting approach was adopted to anticipate the downstream bitrate and the share of active households, which represents the percentage of households per switch/ring receiving traffic above a specified threshold. Two forecasting methods were utilized in this regard. Firstly, the *Naïve* method assumed similarity between the current time slot (lets say: Tuesday from 11:23 till 11:24) and the corresponding slot from the previous week. Secondly, forecasting was con-

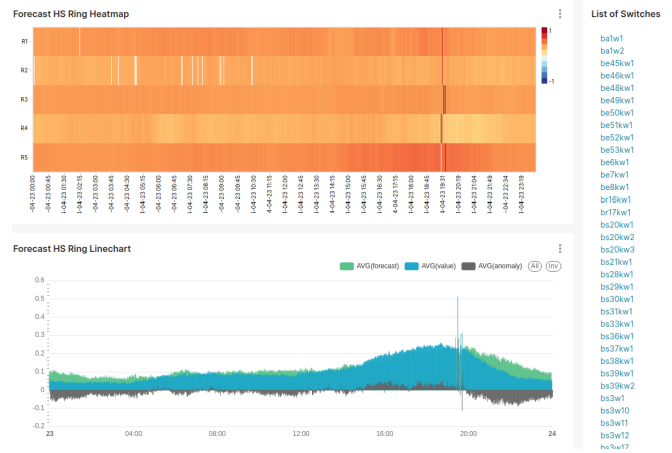


Fig. 4: Monitoring network status and anomalies using telemetry data. The figure shows the share of households that are active and use their network connection. The heatmap shows deviations between the forecasted and actual data in five rings (Y-axis) as a function of time (1 day, 1 minute granularity). The line-plot shows the forecasted and actual data as a function of time (same period, same granularity) as well as the difference between them.

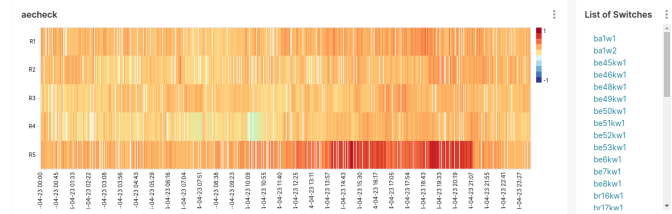


Fig. 5: Monitoring network status and anomalies using telemetry data. The figure shows the anomaly scores gained from multivariate autoencoder method. The heatmap shows deviations between the reconstructed and actual data in five rings (Y-axis) as a function of time (1 day, 1 minute granularity)

ducted using Facebook’s Prophet [2]. Any disparities between the forecasted and actual data were considered indicative of unusual events.

The second set of algorithms targeted multi-variate outlier detection. Two distinct approaches were employed for this purpose. Firstly, an autoencoder-based method was utilized. This involved utilizing an encoder-decoder pair to identify an optimal hidden representation of the input data. The input data had  $N \times M \times P$  dimensions, where  $N$  represented the number of time slices (spanning 4 weeks with 1-minute resolution),  $M$  indicated the number of studied ports and  $P$  denoted the number of features. The disparity between the raw input and the reconstructed output matrix served as an indicator of unusual behavior. The system was trained on outlier-free data, resulting in a matrix with elements close to zero when there were no outliers. However, if the input contained data not present in the training set (outliers), the reconstruction would be imperfect, leading to discrepancies in the difference array.

Secondly, outlier detection was performed using an ensemble of outlier detectors. This ensemble included various detectors such as Copula-based Outlier Detector[3], Local Outlier Factor[4], PCA-based outlier detector, and nearest-neighbor ensembles-based Isolation-based anomaly detector[5]. These detectors were implemented in the PyOD library[6]. The scores generated by individual detectors were aggregated into a single outlier score, providing a comprehensive assessment of anomalous behavior.

In the field trial we have 1026 switches connected to the data pipeline. The amount of data received by Kafka is approximately 1GB per switch per day, so in total approximately 1TB per day for all switches. For the anomaly detection in this field trial we process only a subset of the parameters, corresponding to 32 GB/day for the 1026 switches.

#### IV. LESSONS LEARNED AND FUTURE WORK

We have investigated differences between SNMP and telemetry using a Waystream access switch. This showed how high resolution telemetry measurements can better capture the traffic load patterns and give a more detailed view of the state of the network. In the Lunet field trial we have learned how an efficient data pipeline can be implemented. We have learned that it is feasible to collect streaming telemetry data from more than a thousand switches and continuously monitor and visualize the network health status.

Going forward we have identified possibilities for improvements in terms of measurement resolution, data collection pipeline and anomaly detection. For the data pipeline several performance optimization are possible and planned. Much of the current data pipeline is built up of scripts written in Python which perform complex and time consuming data grouping tasks. These scripts will be optimized and rewritten in a more efficient programming language. For the outlier detection each port, switch and ring needs their own pre-trained models. In the current implementation, the models are kept in memory and this sets a limit of the number of ports that can be processed at the same time. Loading the models on demand and releasing them would alleviate this limitation based on memory usage.

Concerning anomaly detection, the described tool is at the moment using unsupervised outlier detection methods. It is reasonable to believe that the detected outliers in most of the cases are not harmful. To decide whether a given outlier is harmful or anomalous, labelled data and supervised learning is needed. One way to get labelled data is via labelling the outliers that are detected by the unsupervised methods. This labelling process would have to be done by field experts. For future work, our tool can be further developed to support operators in the process of labelling network event as anomalies. This process will give us fingerprints of various error situations. We will assess whether there might be a need for further improved measurement resolution in the access switches to address specific use cases and modify the measurement set-up accordingly.

Network monitoring is a broad field and an active area of research with many different techniques and approaches [7], [8], [9], [10]. In this industrial paper we have focused on the practical implementation of a data pipeline, experiences from a field trial in an operational network, and a feasibility study that shows that it is possible to continuously collect high-resolution telemetry data from a large number of switches. Interesting questions for future work also include how much of the data needs to be collected centrally by an operator for network monitoring, techniques for collecting the important data rather than collecting all the data, and what monitoring tasks and anomaly detection can be done distributed without telemetry data having to leave the switch.

Transferring data pre-processing steps or even entire tasks such as anomaly detection to simple and low performance hardware on the edge will continue to be a focus in the near future due to new technology achievements in hardware and software architectures (e.g. containerization). The separation of computationally intensive steps (e.g. model training) on correspondingly powerful hardware and the simple application is also a promising approach in many other applications in order to keep the amount of data to be transported manageable. This enables new data strategies, especially for higher-level management software systems, in which only relevant data is archived centrally and persistently, and other data is already aggregated at the edge. Preliminary work on this is already ongoing.

#### V. ACKNOWLEDGEMENT

This work has been done within the framework of the CELTIC-NEXT project AI-NET-PROTECT supported by Sweden's innovation agency VINNOVA (2020-03506) and BMBF (German Federal Ministry of Education and Research), funding ID 16KIS1290.

#### REFERENCES

- [1] H. Abrahamsson, D. Henriksson, K. Makonyi, D. M. Hurtado, and J. Sandell, "Towards automated and proactive anomaly detection in a fiber access network," in *Proceedings of 18th Swedish National Computer Networking and Cloud Computing Workshop (SNCNW)*, 2023.
- [2] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.
- [3] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "COPOD: copula-based outlier detection," in *IEEE ICDM*, 2020.
- [4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, 2000.
- [5] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells, "Isolation-based anomaly detection using nearest-neighbor ensembles," *Computational Intelligence*, vol. 34, no. 4, 2018.
- [6] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.
- [7] S. Lee, K. Levanti, and H. S. Kim, "Network monitoring: Present and future," *Computer Networks*, vol. 65, pp. 84–98, 2014.
- [8] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (ntma): A survey," *Computer Communications*, vol. 170, pp. 19–41, 2021.
- [9] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, and P. Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Transactions on Network and Service Management*, vol. 16, 2019.
- [10] M. Yu, "Network telemetry: towards a top-down approach," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 1, p. 11–17, feb 2019.