Probabilistic Time Series Forecasting for Adaptive Monitoring in Edge Computing Environments

Dominik Scheinert*, Babak Sistani Zadeh Aghdam*, Soeren Becker*, Odej Kao*, and Lauritz Thamsen†
* Technische Universität Berlin, Germany, {firstname.lastname}@tu-berlin.de
† University of Glasgow, United Kingdom, lauritz.thamsen@glasgow.ac.uk

***Abstract*—With increasingly more computation being shifted to the edge of the network, monitoring of critical infrastructures, such as intermediate processing nodes in autonomous driving, is further complicated due to the typically resource-constrained environments. In order to reduce the resource overhead on the network link imposed by monitoring, various methods have been discussed that either follow a filtering approach for data-emitting devices or conduct dynamic sampling based on employed prediction models. Still, existing methods are mainly requiring adaptive monitoring on edge devices, which demands device reconfigurations, utilizes additional resources, and limits the sophistication of employed models.**

In this paper, we propose a sampling-based and cloud-located approach that internally utilizes probabilistic forecasts and hence provides means of quantifying model uncertainties, which can be used for contextualized adaptations of sampling frequencies and consequently relieves constrained network resources. We evaluate our prototype implementation for the monitoring pipeline on a publicly available streaming dataset and demonstrate its positive impact on resource efficiency in a method comparison.

***Index Terms*—Adaptive Monitoring, Data Reduction, Time Series Forecasting, Resource Management, Edge Computing**

I. Introduction

The amount of devices and sensors deployed in the Internet of Things (IoT) is increasing steadily. At the same time, this coincides with a rapid increase of generated data by the employed devices. Traditional cloud computing architectures encounter problems when trying to cope with this increasing scale, as new use cases, e.g. smart cities and manufacturing, digital health care, or autonomous driving pose considerable challenges to the underlying infrastructure. Especially in the aforementioned domains, the amount of collected and transferred data to the cloud adds an additional burden on possibly unreliable network connections and renders latency-bounded and bandwidth-intensive applications infeasible [1], [2].

In order to unburden the network and improve overall communication efficiency, the edge computing paradigm has been gaining momentum in the past years. By shifting computing capabilities closer to the actual data sources at the edge of the network, such environments enable the processing of data on edge devices in highly distributed architectures [3], [4]. The employed edge devices are typically resource-constrained and remotely located. Thus, they can i.e. easily be overloaded and are vulnerable to damage or theft. Since these critical infrastructures can have a decisive impact on everyday life, continuous monitoring is required in order to detect problems early on and assure the expected functionality [5]. However, constantly transmitting all the monitoring data, especially with high frequency, consumes noticeable network bandwidth [6] and can thus in turn aggravate network congestions or even service interrupts [6]–[9]. Therefore, the transmission rate of monitoring data is often reduced [6], i.e. by adaptively adjusting the monitoring rate [6], [10], [11]. Several approaches [12], [13] embed and employ the adaptive functionality directly on the edge devices, and although this can yield promising results, it also results in further processing load on already resource-constrained nodes.

Hence, in this paper, we are proposing an adaptive monitoring approach that is deployed on cloud nodes and chooses the monitoring frequency based on forecasting future monitoring metrics. Exploiting probabilistic forecasting models to decide whether to fetch monitoring data from edge nodes or to rely on the forecasting output, we aim to reduce monitoring traffic and at the same time still allow for optimizing and automating operations based on accurate monitoring metrics. Accordingly, our approach considers the variability of the data over time and retrains periodically in order to prevent the drifting of the forecasting model [14].

Contributions. The contributions of this paper are:

- A design for a system that minimizes data transmission rates in resource-constrained environments via sampling-based adaptive monitoring. The internally used probabilistic forecasting method allows for the assessment of predictions and hence conditional sampling.
- A prototypical implementation of our adaptive monitoring routine which follows the outlined principles of our proposed system design and is therefore representative.
- An evaluation of our implementation on a publicly available streaming dataset and comparison to a related method. We demonstrate the effective reduction of data transmission rates while retaining accurate metric data estimates, and discuss the implications of our findings.

Outline. Section II discusses the related work. Section III elaborates on the idea and proposes a system for sampling-based and adaptive monitoring, whereas Section IV concretizes on the modeling approach for probabilistic forecasting of metrics. Section V presents the preliminary results of our comparison with a related method, and a discussion of general requirements of our approach. Section VI concludes the paper.

II. Related Work

This section discusses various related methods for adaptive monitoring as well as their differences from our method.

A. Adaptive Filtering

Solely emitting data values when they significantly differ from past data values is a strategy implementable on the device level and referred to as adaptive filtering. JCatascopia [15] is a monitoring framework that adjusts the filtering range in regard to a user-defined threshold, where the threshold defines the percentage of the data that should be filtered. The ATOM framework [16] follows a similar approach and additionally sends the median values of a subset of metrics when the values of these metrics did not change more than a threshold in a certain period of time. Again another data filtering system [17] conducts data filtering when previously observed patterns in the data are maintained, which is achieved by training a classifier on past data and using it on new data.

In contrast, our proposed framework only assumes accessible metric endpoints on the source nodes, which makes our approach agnostic against the concrete set of metrics that shall be modeled. Running the modeling solution on sink nodes further allows for the employment of more sophisticated methods due to non-existent battery constraints.

B. Adaptive Sampling

Adjusting the interval of sampling target devices in a dynamic manner, based on observed data characteristics, is a strategy called adaptive sampling. PayLess [18] is an adaptive monitoring framework that adjusts the sampling frequency by an operation with a constant definable value based on the difference between the current and the previous monitoring data and a predefined threshold. Another work [19] is based on the violation-likelihood detection method: The likelihood of not detecting a violation between two successive data points is calculated, which is used as an indicator, together with a user-defined threshold, for either establishing a fixed interval or conducting more frequent monitoring. FAST [20] is a framework that evaluates the need of adapting the sampling interval at each time step and adjusts the sampling frequency based on the error between a prior and a posterior estimation. With EASA [21], the authors propose an energy-aware method that attempts to determine the optimal sampling frequency and takes the battery level of target IoT devices into consideration.

With our approach using probabilistic forecasts and a robust model update strategy, we tackle limitations such as the negligence of evolutionary data streams, missing or insufficient model update strategies, and non-existent uncertainty handling.

C. Hybrid Algorithms

This category encompasses methods that either combine adaptive filtering and adaptive sampling, or optimize not only the amount of transmitted data. ADMin [11] is a framework that aims to reduce the data which is produced in a network and also reduce energy consumption by devices. Data is published over the network solely when a shift is detected in the data stream. The same authors also propose the AdaM framework [22], which measures the streaming data variability and evolution alongside employing two algorithms for adaptive sampling and adaptive filtering in order to reduce the monitoring data disseminated throughout the network. With the AM-DR framework [13], the authors attempt to reduce the data transmission between the sink and sensor nodes by predicting readings at both the source and sink nodes and transmitting sensor data when the difference between predicted and observed values exceeds a predefined threshold. The SETAR framework [23] employs a forecasting model for both the data aggregation layer and the source node, and in case the forecasted metrics differ more than a threshold from the actual values, the transmission of the actual values is triggered. In other works [24], [25], the authors propose to fit a linear model on the recent sensed values and only send the updates of the model parameters to the sink node. A new model is computed and distributed among nodes if the predicted values continue to deviate from the actual sensed values. Efficient sampling and hence reduced data transmission rates are also the result of a distributed Active Learning framework [26] for IoT applications deployed on multi-layer infrastructures.

The majority of hybrid algorithms require hardware or software modifications for both the source node and sink node, which makes their application challenging and less straightforward. With our approach, we are relieving the IoT devices of interest and hence simplify the operation.

III. Proposed System

This section elaborates on our envisioned system for adaptive monitoring in resource-constrained environments via a sampling-based approach. It is further illustrated in Figure 1.

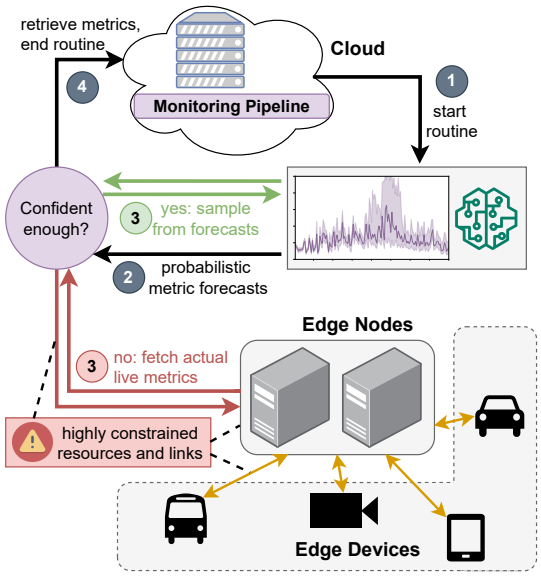

Fig. 1. Overview of the envisioned system. Limited computing resources and network capacity characterize the edge environment. Hence, we reduce the overall network usage by conditionally sampling from either a target node or a model distribution, based on the models' confidence in its predictions.

A. Probabilistic Adaptive Sampling

Monitoring of services or devices in resource-constrained environments comes with its own challenges. In order to reduce the additional bandwidth usage caused by monitoring, methods have been proposed in the past that seek a minimization of transferred information. Yet, most of them require changes to the respective source node and/or sink node, or provide insufficient means of dealing with imprecise predictions and the evolution of data streams over time. For applicability in real-world scenarios, it is therefore desirable to design an approach that shifts control back to the respective sink node(s), requires no changes on the device level and is hence fairly agnostic, and employs a sophisticated strategy for dealing with model predictions and variability in data streams. Consequently, we demand a sampling-based approach that utilizes probabilistic forecasting to realize adaptive monitoring.

B. Envisioned System

Assuming that target devices expose relevant metrics by factory default or this functionality can be retrofitted with manageable effort, the amount of transmitted data can be reduced using a sampling-based approach, i.e., by employing a prediction model on a sink node. Once sufficient data has been collected via initial frequent sampling, a probabilistic prediction model can be trained and used for predicting future values together with a notion of prediction uncertainty. From there on, at any point in time, the range of possible values predicted by the model is evaluated. In case of significant model uncertainty, actual metric data is sampled from the source node, whereas otherwise, we sample from the aforementioned value range of the model. With this envisioned system, the overall data transmission rate can be reduced and a conscious strategy for dealing with uncertainties is enabled.

IV. Resource-Efficient Adaptive Monitoring

This section presents our approach to adaptive monitoring in edge computing environments using probabilistic time series forecasts. The approach is generally sketched in Figure 2 and explained in more detail in the following paragraphs.

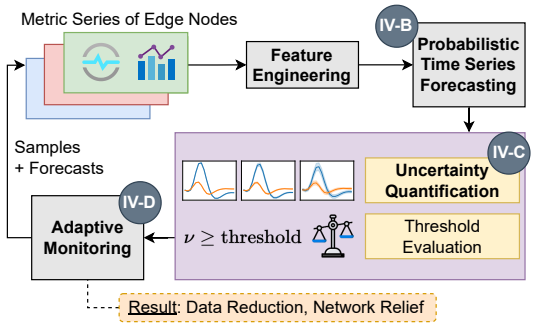

Fig. 2. The proposed pipeline. Time series data of actual metrics are processed, enriched, and used as forecasting model input. The model output is then further analyzed and evaluated with regard to adaptive monitoring.

A. Preliminaries

When collected over time, metric data can provide an abstract representation of the state of each system component. As in our previous work [27], we define metric data as multivariate time series, i.e. a temporally ordered sequence of vectors $S = (S_t \in \mathbb{R}^d : t = 1, 2, \ldots, T)$, where d is the number of metrics and T defines the last sample time stamp. For $S_b^a = (S_a, S_{a+1}, \ldots, S_b)$, we denote indices a and b with $a \leq b$ and $0 \leq a, b \leq T$ as time series boundaries in order to slice a given series S_T^0 and acquire a subseries S_b^a. Additionally, we use the notion $S(i)$ to refer to a particular metric dimension i, with $1 \leq i \leq d$. With $\hat{S}$, we furthermore refer to a forecasted multivariate time series.

B. Probabilistic Time Series Forecasting

The problem with commonly employed deterministic forecasting approaches is that an indication of the certainty of model outputs is missing. In our adaptive sampling setting, this hinders the quality assessment of model forecasts and would only allow for an evaluation in retrospect. Conveniently, we can make use of probabilistic forecasts to tackle this limitation. Here, the underlying idea is that of *quantile regression*, where the loss is formally defined and computed as

$$L_\rho(y, \hat{y}) = \rho . f(y - \hat{y}) + (1 - \rho) . f(\hat{y} - y), \tag{1}$$

with $\rho \in (0, 1)$ being the quantile, $f(x) = max(0, x)$ a smoothing function of predicted values, y the ground truth, and $\hat{y}$ the corresponding predicted sample. Consequently, in the case of our previously defined multivariate time series, the total model loss is then calculated across time (T), quantiles (P), and the user-defined prediction horizon K as:

$$\sum_t^T \sum_\rho^P L_\rho \left(S_{t+K}^t, \hat{S}_{t+K}^t \right). \tag{2}$$

By training a model with this loss function and adapting the model parameters accordingly, we can assess the certainty of model predictions later, which is imperative for our approach to adaptive monitoring.

C. Uncertainty Quantification

We attempt to train a model on all metrics of a target system, which results in a multivariate time series where we can possibly exploit correlations between individual metrics. Next, for each individual metric, we regulate the corresponding sampling frequency, which demands a suitable criterion. In order to quantify the uncertainty of each metric, we utilize the standard deviation as a criterion to quantify the variability of predicted samples across all quantiles. If the forecasted samples have variance more than a predefined threshold, intuitively, we consider the model outputs to be uncertain due to the wide range of possible values. If we sample N values from the model's learned distribution, then the uncertainty quantification process can be formulated as below:

$$\sigma_k = \sqrt{\frac{\sum_{i=1}^N (s_i - \mu_k)^2}{N}}, \quad \nu = \frac{\sum_{k=1}^K \sigma_k}{K} \tag{3}$$

Here, σ_k is the standard deviation of the forecasting samples at time k, $s_i = \hat{S}_k^k$ is the i-th sample, and μ_k is the average of all N samples at time step k. At the next step, the average of all standard deviations of all time steps in the forecasting windows length, i.e. K, is calculated as ν. Finally, the value of ν is compared with the defined threshold, which triggers a new sampling routine based on the outcome of this evaluation. Evidently, this uncertainty quantification can also be used as a trigger for model retrainings, e.g., if it is observed that defined thresholds are violated more frequently than before.

D. Adaptive Monitoring

With a strategy now in place for model training and conditional sampling, we can further elaborate on our overall routine for adaptive monitoring, for which we summarize the pseudocode in Algorithm 1. In the first step, all metrics are fetched from the respective target system for the duration of the defined input window length of the model. In the next step, the model predicts future metric values based on the given input for the duration of the defined forecasting horizon length. Afterward, as previously described, the uncertainty of each metric for the period of the forecasted time is calculated. If no metric with high uncertainty is found, then the next input of the model will be set to the recently forecasted series. Otherwise, our monitoring routine idles until the last forecasted time step with high certainty and then triggers the fetching of uncertain metrics. Subsequently, the fetched metrics and forecasted metrics are combined along the time dimension and used as the next model input.

Algorithm 1 Pseudocode of adaptive monitoring routine

```
N                                          ▷ forecasting horizon length
L                                          ▷ input window length
FS                                         ▷ forecasted series
FM                                         ▷ fetched metrics
UM                                         ▷ metrics with high uncertainty
IS ← fetchAllMetrics(length = L)           ▷ input to model
LFT ← 0                                    ▷ last forecasted time step
while TRUE do
    FS ← forecast(input = IS)
    UM ← getUncertainMetrics(input = FS)
    if isEmpty(UM) then
        IS ← FS
        LFT ← getLastTimeStep(FS)
    else
        idle(until = LFT)                  ▷ wait for next interval
        FM ← fetchMetrics(metrics = UM)
        IS ← temporalMerge(FM, FS)
    end if
end while
```

In summary, the aforementioned pipeline allows for targeted sampling of individual metrics and makes use of probabilistic forecasts if sufficient knowledge is available.

V. Preliminary Results

In this section, we examine a prototypical implementation of our monitoring pipeline, called *AM-PF*, obtain preliminary experimental results, and discuss our findings in detail.

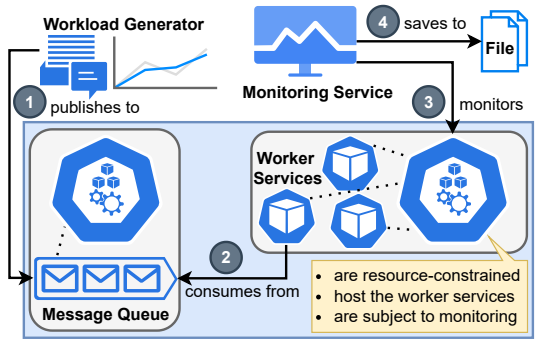

Fig. 3. Illustration of our experiment setup for the data acquisition.

A. Data Acquisition

A dataset is required for the evaluation of our proposed framework. For the sake of highlighting the resource overhead of frequent monitoring, we consider a sampling frequency of 1 second. Our general goal is to obtain realistic metrics of a node with an underlying predictive pattern, which requires a corresponding workload. The base workload is created by means of employing 10 python services that are running on the respective node. Each service consumes messages from an external RabbitMQ message queue, and upon message receipt, several processes are in parallel running basic operations in order to stress relevant resources such as CPU, memory, etc. These operations are by design dependent on the message rate so that varying patterns can be simulated. The metrics of the host machine are eventually gathered via Prometheus queries and saved to file. As a source for publishing to the message queue, we employ the IoT Vehicles experiment dataset created and published in [28], which reports amounts of moving cars at intervals of 1 second. At each point in time, the corresponding vehicle amount is read and used for publishing the same amount of messages to the message queue. The whole procedure is illustrated in Figure 3.

B. Prototype Pipeline

We implement a prototype of our envisioned pipeline. First, the values of each variable of the potentially multivariate time series are min-max normalized along the time dimension, where the boundaries are determined from the respective training data and used during inference as well. We further use as additional features cyclical encodings of *SecondOfMinute* and *MonthOfYear*, as well as custom encodings for 1) *MinuteOfDay* and 2) separating work days and the weekend.

The neural network is implemented using the Darts[1] library, composed of two stacked LSTM layers with a dropout layer in-between and a final linear layer at the end, trained by employing a quantile regression loss, and evaluated with respect to the ρ-risk metric [29]. Optimized model hyperparameters are found via a hyperparameter tuning approach based on grid search. The investigated values are listed in Table I, with the best ones found highlighted in bold. The model is subsequently

[1]https://unit8co.github.io/darts/, accessed: October 2022

TABLE I
MODEL HYPEROPTIMIZATION

Configuration and Search Space	
Learning rate	0.001
#Epochs	max. 20
Input dimension	{**300**, 600}
Output dimension	{**300**, 600}
Hidden dimension	{25, **75**}
Batch size	{**256**, 512}
Dropout rate	{5%, 10%, **20%**}

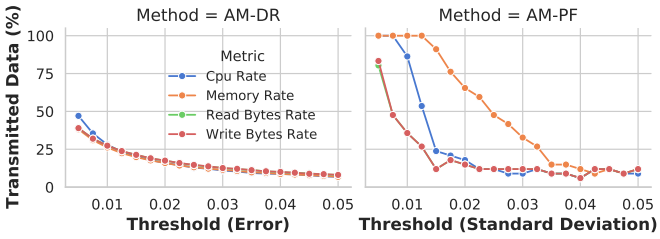

Fig. 4. Comparison of frameworks with regard to transmitted data. Though not directly comparable, it can be observed how individual metrics are handled with varying levels of sensitivity, based on their inherent characteristics.

fully-trained with the best found hyperparameters using early stopping, where the training of the model is stopped if the loss of the model on the validation data does not decrease more than 0.001 after 5 steps. The goodness of the hereby received trained model is further verified via K-fold cross-validation.

C. Baseline

We compare our method against the AM-DR framework [13] and make use of its publicly available implementation[2]. In short, this framework employs models both on a source node and respective sink node, such that the former has to transmit only its immediate sensed values that deviate significantly from the predicted values. We investigate different maximum error thresholds for this method, while all other parameters are set as reported in the original publication. Though not directly comparable to our approach due to a different take on the problem, it is insightful to observe the general saving potential on data transmissions as well as implications for metric reconstruction accuracy at sink nodes.

D. Evaluation Setup

We use the previously acquired data to evaluate both approaches by indexing the respective data file by time and extracting relevant input sequences as arguments to the models, i.e., no additional services are involved in this simplified scenario which favors a detailed model comparison.

For both methods, we evaluate thresholds from 0.005 to 0.05, with a step size of 0.0025. For AM-DR, this threshold translates to the maximum error allowed, whereas, for our approach, the threshold marks the largest standard deviation tolerable. Due to the different meanings of the threshold in both methods, its configuration is not directly comparable but still allows for insights and the derivation of recommendations.

In terms of evaluation metrics, we are interested in the percentage of transmitted data given various threshold configurations as well as the hereby inflicted implications on the prediction accuracy and metric reconstruction at the sink.

E. Results

At any point in time, the confidence of the model in its predictions is evaluated and compared against a threshold. Naturally, the choice of the threshold has an impact on data transmission rates and prediction accuracies, as it controls the monitoring frequency as well as the amount of transmitted data. In our experiments, we for instance observe that with a more conservative threshold of 0.0225, our framework is fetching real metrics more frequently when compared to a used opportunistic threshold of 0.0475. While the latter requires less fetching and hence relieves network bandwidth usage, it leads to a less accurate observable result. This manifests itself in higher accumulated Mean Squared Error (MSE) values for higher thresholds (up to 5% increase in our example), which can also greatly vary across metrics.

The percentage of transmitted data in relation to used thresholds is illustrated in Figure 4. For both methods (the results are not directly comparable), we observe that less data is transmitted with increasing thresholds. Worth mentioning is that for AM-DR, the reduction is comparably smooth and also similar across metrics, whereas, for our framework, we observe that the decline is initially very steady and also dependent on the specific metric, which indicates that individual metric characteristics are taken into consideration, which favors a more metric-specific adaptation of sampling frequencies.

Lastly, we also present the prediction errors in relation to used thresholds in Figure 5. Here, we use the Symmetric Mean Absolute Percentage Error (SMAPE) since it allows for an easy-to-interpret percentage error as well as comparison across metrics. As expected, the SMAPE is increasing for both methods with rising threshold values, since fewer actual metrics are considered and hence the accuracy of inferred metrics is affected. Noticeably, the SMAPE values of AM-DR are steadily increasing and tend to fan out over time, whereas the SMAPE values of AM-PF are more volatile in the beginning and tend to converge toward a common value.

Summarizing our results, we find that a suitable configuration of AM-PF yields a similar performance as related methods, while requiring no changes to target edge devices.

F. Discussion

Our findings demonstrate that probabilistic forecasts can be used to motivate a sampling-based monitoring approach that is able to adapt to data stream changes and reflect on its own recommendations. By tuning a configurable threshold parameter, the amount of transmitted data as well as the hereby possible metric accuracy can be controlled and balanced.

[2]https://github.com/YasminFathy/AMDRIoT, accessed: October 2022

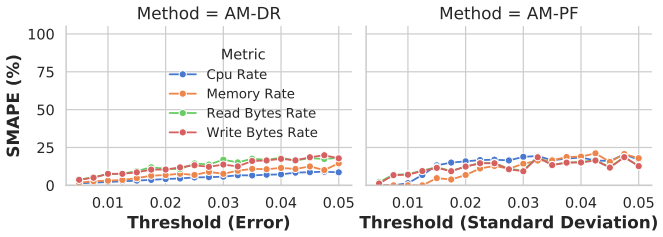

Fig. 5. Comparison of frameworks with regard to reconstruction accuracy. The prediction discrepancy is measured using the SMAPE metric. It can be observed that both methods expose a different convergence behavior.

For the application of our adaptive monitoring framework in real-world scenarios, we deem the following things important. First, it is required that target edge nodes are offering metric endpoints which can be scraped – though a reasonable assumption, some edge nodes might need adaptations if they previously only followed a push-based messaging principle. Another aspect to keep in mind is the configuration of the forecasting horizon – it is advisable to choose a horizon that allows reacting flexibly to any changes that may occur on the edge node. Furthermore, a resource shortage on the edge node and a need for efficient resource usage must be given, otherwise, the employment of related methods like AM-DR (with their respective overhead) might be more conceivable. Lastly, while having designed the framework for use in resource-constrained edge environments, we assume that sufficient resources are available at the sink node in the cloud for the model training.

VI. Conclusion

The primary goal of this work is to demonstrate the applicability of probabilistic time series forecasting for adaptive monitoring in edge computing environments. To this end, we envision a system that realizes adaptive monitoring in a sampling-based fashion using the aforementioned technique, such that overall network usage is reduced and constrained resources are relieved. Towards this goal, we implemented an approach for adaptive sampling based on probabilistic forecasts, and evaluated it in experiments and against a method from related work. We find that our solution is generally able to reduce the amount of data transmission and furthermore provides means of automating the retraining process.

In the future, we plan to leverage our findings and make use of the proposed methods in the context of recent research on carbon-aware computing in edge environments.

Acknowledgments

This work has been supported through grants by the German Federal Ministry of Education and Research (BMBF) as BIFOLD (funding mark 01IS18025A) and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as FONDA (Project 414984028, SFB 1404).

References

[1] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE*, 2018.

[2] S. Becker, F. Schmidt, and O. Kao, "Edgepier: P2p-based container image distribution in edge computing environments," in *IPCCC*. IEEE, 2021.

[3] A. J. Ferrer, S. Becker, F. Schmidt, L. Thamsen, and O. Kao, "Towards a cognitive compute continuum: An architecture for ad-hoc self-managed swarms," in *CCGrid*. IEEE, 2021.

[4] S. Becker, D. Scheinert, F. Schmidt, and O. Kao, "Efficient runtime profiling for black-box machine learning services on sensor streams," in *ICFEC*. IEEE, 2022.

[5] S. Becker, F. Schmidt, A. Gulenko, A. Acker, and O. Kao, "Towards aiops in edge computing environments," in *BigData*. IEEE, 2020.

[6] P. Venkateswaran, C. Hsu, S. Mehrotra, and N. Venkatasubramanian, "REAM: resource efficient adaptive monitoring of community spaces at the edge using reinforcement learning," in *SMARTCOMP*. IEEE, 2020.

[7] P. Lou, L. Shi, X. Zhang, Z. Xiao, and J. Yan, "A data-driven adaptive sampling method based on edge computing," *Sensors*, 2020.

[8] Z. Zhai, K. Xiang, L. Zhao, B. Cheng, J. Qian, and J. Wu, "Iot-recsm - resource-constrained smart service migration framework for iot edge computing environment," *Sensors*, 2020.

[9] L. Ma, "One layer for all: Efficient system security monitoring for edge servers," in *IPCCC*. IEEE, 2021.

[10] D. Trihinas, G. Pallis, and M. D. Dikaiakos, "Adam: An adaptive monitoring framework for sampling and filtering on iot devices," in *BigData*. IEEE, 2015.

[11] ——, "Admin: Adaptive monitoring dissemination for the internet of things," in *INFOCOM*. IEEE, 2017.

[12] S. Tata, M. Mohamed, and A. Megahed, "An optimization approach for adaptive monitoring in iot environments," in *SCC*, X. F. Liu and U. Bellur, Eds. IEEE, 2017.

[13] Y. Fathy, P. M. Barnaghi, and R. Tafazolli, "An adaptive method for data reduction in the internet of things," in *WF-IoT*. IEEE, 2018.

[14] R. C. Cavalcante, L. L. Minku, and A. L. I. Oliveira, "FEDD: feature extraction for explicit concept drift detection in time series," in *IJCNN*. IEEE, 2016.

[15] D. Trihinas, G. Pallis, and M. D. Dikaiakos, "Jcatascopia: Monitoring elastically adaptive applications in the cloud," in *CCGrid*. IEEE, 2014.

[16] M. Du and F. Li, "ATOM: automated tracking, orchestration and monitoring of resource usage in infrastructure as a service systems," in *BigData*. IEEE, 2015.

[17] D. Kim, Y. Jeong, and S. Kim, "Data-filtering system to avoid total data distortion in iot networking," *Symmetry*, 2017.

[18] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "Payless: A low cost network monitoring framework for software defined networks," in *NOMS*. IEEE, 2014.

[19] S. Meng and L. Liu, "Enhanced monitoring-as-a-service for effective cloud management," *IEEE Trans. Computers*, 2013.

[20] L. Fan and L. Xiong, "An adaptive approach to real-time aggregate monitoring with differential privacy," *IEEE Trans. Knowl. Data Eng.*, 2014.

[21] B. Srbinovski, M. Magno, F. E. Murphy, V. Pakrashi, and E. M. Popovici, "An energy aware adaptive sampling algorithm for energy harvesting WSN with energy hungry sensors," *Sensors*, 2016.

[22] D. Trihinas, G. Pallis, and M. D. Dikaiakos, "Low-cost adaptive monitoring techniques for the internet of things," *IEEE Trans. Serv. Comput.*, 2021.

[23] I. B. Arbi, F. Derbel, and F. Strakosch, "Forecasting methods to reduce energy consumption in WSN," in *I2MTC*. IEEE, 2017.

[24] L. Leal, M. Lemos, C. Carvalho, and R. Filho, "Avoiding data traffic on smart grid communication system," in *ECSA*. MDPI, 2014.

[25] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "Practical data prediction for real-world wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, 2015.

[26] S. Nedelkoski, L. Thamsen, I. Verbitskiy, and O. Kao, "Multilayer active learning for efficient learning and resource usage in distributed iot architectures," in *EDGE*. IEEE, 2019.

[27] D. Scheinert, A. Acker, L. Thamsen, M. K. Geldenhuys, and O. Kao, "Learning dependencies in distributed cloud applications to identify and localize anomalies," in *CloudIntelligence*. IEEE, 2021.

[28] M. Geldenhuys, B. J. J. Pfister, D. Scheinert, L. Thamsen, and O. Kao, "Khaos: Dynamically optimizing checkpointing for dependable distributed stream processing," in *FedCSIS*. IEEE, 2022.

[29] M. W. Seeger, D. Salinas, and V. Flunkert, "Bayesian intermittent demand forecasting for large inventories," in *NeurIPS*. IEEE, 2016.