



FEDAMON: A Forecast-Based, Error-Bounded and Data-Aware Approach to Continuous Distributed Monitoring

Downloaded from: <https://research.chalmers.se>, 2026-03-08 05:11 UTC

Citation for the original published paper (version of record):

Zhang, Y., Duvignau, R. (2025). FEDAMON: A Forecast-Based, Error-Bounded and Data-Aware Approach to Continuous Distributed Monitoring. DEBS 2025 - Proceedings of the 19th ACM International Conference on Distributed and Event-based Systems: 39-50. <http://dx.doi.org/10.1145/3701717.3730544>

N.B. When citing this work, cite the original published paper.

FEDAMON: A Forecast-Based, Error-Bounded and Data-Aware Approach to Continuous Distributed Monitoring

Yixing Zhang
yixing@chalmers.se

Chalmers University of Technology and University of
Gothenburg
Gothenburg, Sweden

Romarc Duvigau
duvignau@chalmers.se

Chalmers University of Technology and University of
Gothenburg
Gothenburg, Sweden

ABSTRACT

Efficiently monitoring distributed systems is critical for applications such as data center load balancing, fleet management, and smart grid energy optimization. Traditional continuous monitoring solutions often require significant communication overhead, straining network resources. This paper addresses the continuous distributed monitoring problem, where a central coordinator needs to track statistics from numerous distributed nodes in real-time. We propose a novel forecast-based, error-bounded, and data-aware approach that significantly reduces communication costs while maintaining accurate monitoring. Instead of transmitting all observed values to the central coordinator, our event-based monitoring leverages lightweight forecasting models at edge nodes. Both the coordinator and distributed nodes predict the evolution of local values, communicating only when deviations exceed a predefined error threshold. To adapt to dynamically changing trends in data streams, we introduce a data-aware model selection strategy that optimizes the balance between communication frequency and monitoring accuracy. Our solution is evaluated on diverse datasets and results demonstrate a substantial reduction in communication overhead with minimal impacts on accuracy, outperforming baseline monitoring regarding communication complexity, e.g., sending, on average, only 10% of baseline update events while maintaining less than 2% average error across all monitored streams. Furthermore, we show that our standard parameter solution even surpasses the best calibrated single models, achieving up to a 17% improvement in communication overhead with identical guarantees on maximum error. Optimizing the control factor in data-aware approach leads to a 13% improvement in performance, reducing error by 1%, without incurring additional communication costs. We believe our approach offers a scalable and efficient solution, enabling fully automatic, real-time monitoring with optimized performance.

CCS CONCEPTS

• **Applied computing** → *Event-driven architectures*; • **Networks** → **Network monitoring**; *Data center networks*; • **Theory of computation** → *Distributed algorithms*; • **Information systems** → *Data streams*.

Please use nonacm option or ACM Engage class to enable CC licenses
This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

DEBS '25, June 10–13, 2025, Gothenburg, Sweden

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1332-3/2025/06

<https://doi.org/10.1145/3701717.3730544>



KEYWORDS

continuous monitoring, distributed data streams, network monitoring, distributed tracking, data-aware approaches

ACM Reference Format:

Yixing Zhang and Romarc Duvigau. 2025. FEDAMON: A Forecast-Based, Error-Bounded and Data-Aware Approach to Continuous Distributed Monitoring. In *The 19th ACM International Conference on Distributed and Event-based Systems (DEBS '25)*, June 10–13, 2025, Gothenburg, Sweden. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3701717.3730544>

1 INTRODUCTION

The proliferation of connected devices has led to an unprecedented surge in the volume and velocity of data being generated across modern networks. Today's network infrastructure faces the challenge of managing high-velocity and high-volume data generated by billions of internet-connected devices, reflecting the pervasive reality of the Internet of Things (IoT) era [1]. Thus, reducing the amount of continuously transmitted data has become imperative for saving energy [3] and enhancing performance in modern packet-processing architectures [13, 19]. Efficiently and continuously monitoring large-scale distributed systems is a crucial challenge with numerous applications, ranging from extending the lifespan of battery-powered devices [11] in sensor networks to improving load balancing in data centers [2, 18].

This paper focuses on the Continuous Distributed Monitoring (CDM) problem, where a central coordinator C monitors a distributed system in a synchronous manner, retrieving observed values from all sensors during each round. The primary goal is to minimize the amount of data transmitted between distributed nodes and C while maintaining accurate monitoring. The field of CDM is a vibrant research area, with various approaches aiming to reduce network costs. In recent years, significant research has been devoted to developing communication-efficient algorithms across various models for problems like (approximate) event counting [8], computing item frequencies [9], and identifying the most popular items [36]. We focus here on the **All-Values-Tracking (AVT) problem**, i.e., keeping track of all observed distributed values at each round, an important task with numerous practical applications [13, 23, 24]. AVT focuses on tracking every single value from every node, and as such, offers a generic solution to CDM since by having C observing values from all nodes, any aggregation function at C can be computed based on the estimates.

The existing literature highlights several approaches to communication-efficient distributed monitoring, such as prediction-based models and data compression techniques. Prediction-based

frameworks, like the one in [22], employ forecasting models to minimize data transmission, but often rely on static modeling choices and do not account for the evolving nature of data streams. Similarly, prediction-based data reduction in Wireless Sensor Networks (WSNs) [11] assumes shared communication mediums and leverages inter-node correlations, which limits the applicability of per-node prediction schemes in more general distributed settings. Data compression techniques [30], while effective at reducing communication, fail to provide real-time monitoring capabilities, which are critical for applications requiring up-to-date resource tracking. Thus, there remains a gap in designing a scalable, adaptive monitoring approach that minimizes communication overhead while maintaining real-time accuracy, particularly in distributed systems with private communication channels and loosely correlated data streams, applicable to many Cyber-Physical Systems (CPS) and modern distributed environments.

1.1 Aim, Motivations and Challenges

To avoid continuously flooding the network with monitoring messages, we aim to reduce communication costs by allowing a small, bounded error in the tracked statistics at the coordinator. This is a challenging problem, as worst-case scenarios may require all data to be transmitted at all times. By embedding the monitoring logic close to data sources and leveraging the predictable nature of data evolution, we seek to drastically reduce network transmissions while incurring only a small reduction in tracking accuracy. Additionally, the dynamic nature of data streams necessitates data-aware monitoring model selection to account for changing data characteristics, thereby enhancing monitoring performances.

Despite advancements in distributed monitoring, current state-of-the-art solutions [8, 23, 24, 33] often prioritize worst-case scenarios, focusing on tight theoretical bounds for communication complexity or near-optimal online algorithms. However, these approaches fall short in practical settings, where real-time data stream values typically exhibit strong temporal correlations. For instance, continuously tracking all node values remains a significant challenge due to the dynamic nature of the data and the requirement for lightweight approaches—excessive computational demands for monitoring would negate the benefits of reduced communication. Furthermore, CPS and IoT devices operate under stringent energy and computational constraints. In this context, monitoring algorithms must be efficient not only in communication but also in computation to ensure low energy consumption and sustain system-wide energy efficiency. This creates a pressing need for algorithms that are both lightweight and capable of leveraging temporal correlations in data streams. A shift towards data-driven, data-aware monitoring solutions is essential to address these challenges and meet the demands of real-world applications.

1.2 Contributions

In this paper, we present (1) a novel **forecast-based, error-bounded approach** for continuous distributed monitoring. Our system enables both the coordinator and distributed nodes to predict the evolution of local values, reducing communication by transmitting updates only when observed values deviate beyond a pre-defined error threshold. This strategy leverages simple yet effective

Table 1: Summary of notations used in the paper.

Notation	Description
C	Coordinator node
$\mathbb{S}, n = \mathbb{S} $	Set of distributed nodes and its size
$s^i \in \mathbb{S}$	i -th distributed node
v_t^i, \hat{v}_t^i	value observed, estimated on s^i at time t
ϵ	Absolute error upper-bound
D	Maximum buffer size
T	Total no. of monitoring timesteps
Q	Total no. of update messages sent
ρ	Communication ratio
$R_{\mathcal{D}}$	Range of dataset \mathcal{D}
$R_{\mathcal{E}}, R_{\mathcal{G}}, R_{\mathcal{I}}, R_{\mathcal{A}}$	Range of Ericsson, Geolife, IntelLab, ACSF1
W	Window size for score calculations
α	Control factor for score calculations
ξ	Model selection criteria

forecasting models, such as the Auto-Regressive (AR) model, Least Mean Square (LMS) filter [20, 31], and Piecewise-Linear Approximation (PLA) [12], to minimize computational complexity at the nodes. Additionally, we introduce (2) a **data-aware mechanism** that dynamically selects the most suitable forecasting model based on current data characteristics. By evaluating multiple candidate models, the coordinator optimizes the trade-off between communication cost and monitoring accuracy by updating the distributed nodes with appropriate model parameters as needed. To validate our approach, we conduct experiments on four large datasets spanning diverse environments, including CPU usage in mobile architectures, vehicular sensor data, temperature readings in sensor networks, and energy consumption of home appliances. The results demonstrate significant reductions in communication costs with minimal impact on real-time monitoring accuracy, highlighting the practical benefits of our framework in real-world applications.

1.3 Paper structure

The rest of this paper is structured as follows. Section 2 provides an overview of related work in continuous distributed monitoring and forecasting approaches. Section 3 presents our proposed forecast-based monitoring framework, including the design of error-bounded communication protocols. Strategies for data-aware model selection are introduced in Section 4. Section 5 discusses the results of our extensive performance evaluation, highlighting the effectiveness of our approach across various scenarios and comparing it to baseline solutions. Finally, Section 6 concludes the paper by summarizing our contributions and discussing future works.

2 BACKGROUND AND RELATED WORK

In this section, we introduce our system model and compare it to other related monitoring models, highlighting the differences in key assumptions and characteristics. We then outline the primary problem in focus and present the existing solutions solving it.

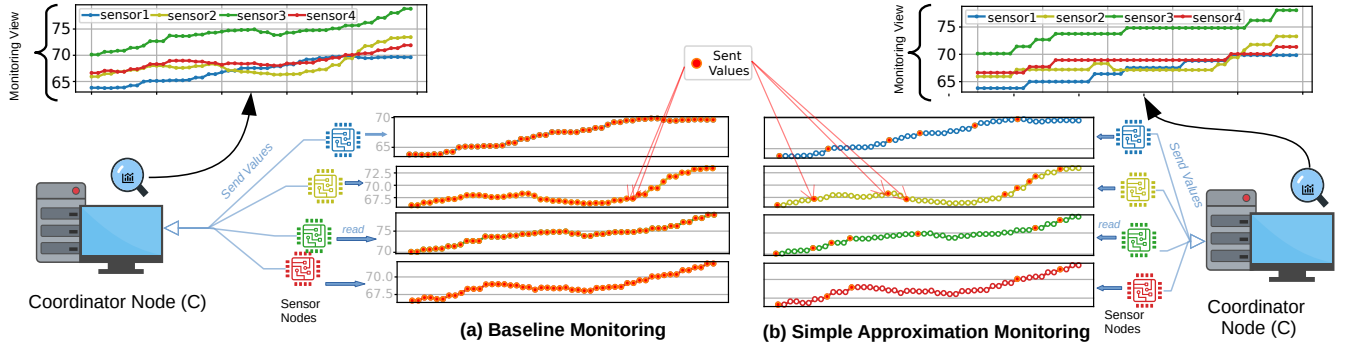


Figure 1: Illustration of continuous monitoring comparing *baseline monitoring (a)* [25] with a *simple approximation (b)* [28] strategies, both on 4 distributed sensor nodes with monitoring views of each coordinator.

2.1 Continuous Distributed Monitoring

2.1.1 CDM System Model. The CDM problem involves the task of continuously gathering information at a central location, or *Coordinator* (denoted as C), i.e., a sink node that collects all observed values by a set of distributed *nodes*. Unlike traditional monitoring approaches, CDM emphasizes the optimization of communication efficiency over computational performance. Refer to Table 1 for the notations used throughout the rest of the paper.

We consider a set of n distributed nodes, or sensors $\mathcal{S} = \{s^1, s^2, \dots, s^n\}$, where each node s^i observes a discrete stream of values $v_t^i \in \mathbb{R}^+$ at discrete timesteps $t \in \mathbb{N}$. The coordinator C is connected to each node via a bi-directional channel, and direct communication between nodes is assumed not to take place. The distributed nodes can be thought of as routers or worker nodes in a data center, IoT devices in a sensor network, or vehicles in a vehicular network. Figure 1 presents an example of communication between C and the sensor nodes. For each node s^i , rounds are assumed to have a constant duration d_i , such that time t (or round t) refers to the interval starting at real-time $t \cdot d_i$ and ending at $(t + 1) \cdot d_i$. The objective of CDM is to compute a monitoring function f at C using distributed inputs, while minimizing communication overhead. The function f can track all values individually or be the result of an aggregation (e.g., maximum or average) of the sensor nodes' readings, e.g. CPU usage of worker nodes in a data center, temperature or humidity from a number of IoT devices, or vehicle's position and usage for a fleet management application.

A *monitoring decision* is defined as the action taken by a node after recording a measurement — to either inform C or remain silent. The key challenge in CDM lies in reducing communication overhead while ensuring that the monitoring function is accurately computed at the coordinator. The *monitoring view* refers to C 's (approximated) view over the complete set of tracked values and is updated at every round. Values observed at the nodes are treated as observation events. Based on the monitoring decision, each node determines whether to trigger an update event or not.

2.1.2 Other Related CDM Models. Cormode's Model [8] is among the most well-known frameworks for distributed monitoring. This model focuses on enabling C to compute an aggregate function over the union of streams from n observers, each directly connected to

C via a bidirectional channel. In this context, an *item* refers to an application-specific *type* of observation detected at a remote site, which is then considered an event. The model operates asynchronously, where each observation triggers an event, followed by a monitoring decision and a message transmission to update the function f at C . The key strength of this model lies in leveraging the interchangeability of events to efficiently compute classical data mining functions. However, there is a major limitation that prevents its application to our setting, namely treating f as a function over the union of streams, thereby ignoring the identity of the nodes (i.e., the same event from different nodes is treated identically), whereas our problem requires tracking each node individually.

The Distributed Data Streams Model [6] introduces a variation of the CDM model with two notable differences: observers communicate with C via unidirectional channels, and they are constrained to use sublinear space, meaning they can only maintain approximate summaries or *data sketches* [26] of their input streams. Due to the unidirectional communication, observers must rely solely on their own observations to make monitoring decisions, without any feedback from C . While the space limitations are effective for counting problems (like top- k items or frequency tracking), they are less applicable in scenarios where historical data is critical, which is often the case in our context. Additionally, the restriction to unidirectional channels significantly reduces the efficiency of monitoring algorithms and no practical system justifies this limitation.

The Distributed Online Tracking Model [33] is another unidirectional, synchronous framework that also introduces intermediary nodes (or relays) between observers and C . These relays, as cluster heads in WSNs [11], are responsible for local aggregation before forwarding data to C . This model emphasizes competitive analysis, comparing its online algorithms against an optimal offline solution. It suffers from assuming a unidirectional communication channel, a requirement to achieve online performance bounds.

Lastly, the Online Monitoring Model [23, 24] addresses the problem of tracking (approximately) the top- k greatest values from distributed nodes in a synchronous environment. Within each monitoring round, nodes are allowed to send a polylogarithmic number of messages to the coordinator, with the option of performing synchronized *sub-rounds*. This model also incorporates a broadcast channel, counted as a single message, to enhance communication.

However, this synchronization assumption limits its applicability to specific types of distributed systems, and no experimental validation was provided in prior studies.

2.2 The All-Values-Tracking Problem

The AVT problem consists in maintaining at C an approximation of the current value for every node and every monitoring round, where observed values are assumed to arrive in discrete asynchronous rounds. See Figure 1 for an illustration of AVT monitoring with 4 distributed nodes, one coordinator C and 50 rounds; 2 monitoring strategies “baseline monitoring [25]” and “simple approximation [28]” are presented, cf. § 2.3 and § 3.3 and the monitoring views at C . We would like to highlight that when a highly communication-efficient solution for AVT is presented, it can be extended to efficiently monitor any function f at C . This can be achieved without the need to fine tune the monitoring logic to f , by simply continuously computing f over the (approximated) tracked values at C . As this work is focusing on a general CDM setting rather than providing solutions for a specific monitoring function f , there might exist more efficient monitoring algorithms for some f . Our design thus offers *query flexibility* by allowing new aggregation queries to be implemented without redesigning the monitoring system, for instance abnormal values can be easily detected when all values are being tracked. Beyond supporting any aggregation functions, observe that tracking all values from all nodes in real-time is also essential in many practical scenarios. For example, in a smart city scenario, vehicle tracking [4] requires real-time access to the position of each single node to enable traffic flow optimization, safety monitoring, and incident detection. Similarly, in sensor networks, IoT devices often need to report raw, full-resolution readings (e.g. environmental or health monitoring [35]) rather than data summaries, as the original values feature anomalies or correlations that are required to the later analysis.

While there are efficient solutions in the CDM space [8, 9, 15, 32, 36] that significantly reduce data transfer over the network, these methods are not applicable to the AVT problem. This limitation arises because these approaches rely on the interchangeability of data sources – meaning that a value observed at one node can be treated as equivalent to that observed at another. However, in the AVT setting, each individual data stream must be tracked separately.

One of the most practical approaches to AVT includes *adaptive filtering* [27] (transmitting only values that fall outside a predefined per-node interval). It has been extensively studied in the context of IoT devices [16]. While it yields significant reductions in communication costs, it often masks a loss in accuracy—whether in value-accuracy by filtering minor fluctuations or in time-accuracy. However, as demonstrated in our evaluation, it can be further refined by considering both data variation (not all nodes benefit equally from the same model) and data evolution (prediction models may need to adapt over time to better align with changing data trends).

One particularly well-studied technique is Geometric Monitoring [15, 32], which enables efficient threshold-based monitoring of functions computed over network-wide aggregates. However, these global approaches are not directly applicable to AVT which

focuses on per-node data granularity. In addition to analytical approaches, heuristic solutions for distributed queries have also been explored [27] as well as algorithms for tracking popular items [5]. These heuristic methods prioritize reducing communication costs while maintaining acceptable accuracy but are generally limited to specific query types and do not address the challenges of per-node tracking inherent in AVT.

To summarize, the inherent complexity of AVT lies in the necessity to monitor all individual streams independently, without leveraging data redundancy across nodes. Thus, there is no one-size-fits-all solution for arbitrary input data, which previous CDM approaches have generally targeted. The distributed nature of AVT compounds the challenge, requiring solutions that prioritize node-to-coordinator communication while remaining agnostic to the presence or absence of other nodes. This necessitates more sophisticated, communication-efficient strategies that can handle the non-transferable nature of events across distributed nodes.

2.3 Forecast-based Monitoring

Simple Approaches. The most straightforward approach to CDM is for each node to directly forward all its observations to C , which is the **TinyDB** [25] strategy, serving as our baseline monitoring method. However, this strategy quickly becomes impractical as the number of observers or the volume of observations increases, leading to scalability issues [8, 13]. Two common approaches to address this challenge are polling the system less frequently and sampling values from only a subset of nodes. Reducing the polling frequency helps mitigate scalability issues but sacrifices some real-time aspects. The key drawback of polling lies in its dependency on the chosen frequency: a high polling rate can easily overload the network, while a lower rate may result in significant losses in monitoring accuracy [8]. Sampling, on the other hand, reduces monitoring overhead by collecting data from only a subset of nodes. While sampling can alleviate the communication burden, it is clearly unsuitable for the AVT problem, where monitoring every individual stream is essential.

State of the art. In [22], the authors outline a framework for communication-efficient distributed monitoring using predictive modeling. As in our proposal, the approach involves leveraging predictions at C and sending real values only when the deviation from predictions exceeds a predefined threshold. It employs an AR model whose parameters are transmitted instead of raw data. Initially, nodes run multiple AR models with varying lags and utilize a racing mechanism based on the Hoeffding bound to evaluate and discard underperforming models. Ultimately, each node maintains a single, optimized model, ensuring efficient and adaptive monitoring. The main differences with our proposed approach are: 1) utilizing a single AR model instead of considering the prediction model as “pluggin” which can have several possible implementations, and 2) disregarding the evolution of data streams contrary to our data-aware forecasting approach that adapts to different data and network conditions.

Prediction-based data reduction has also been extensively studied in the context of WSNs [11]. A notable difference from our setting is that WSN models typically assume a shared communication medium among sensor nodes, allowing values transmitted by

nearby sensors to aid in data reduction. This approach limits the feasibility of per-node prediction schemes [11], as they are often considered too energy-intensive to operate on individual sensor nodes. Instead, prediction tasks are usually offloaded to a cluster head or to C which limits the possible reduction in communication. For instance, in [7], the authors propose a probabilistic model to compute a probability density function over values on the nodes. It exploits spatial correlations among distributed nodes by exchanging information between nodes. In contrast, our work considers a more general distributed setting where communication channels between nodes and the coordinator are private. As their work is compared with TinyDB [25] and Simple Approximation [28], we also utilize these methods for comparison with our approach. Furthermore, the datasets we use often exhibit low correlations between the observations of distributed nodes, making our approach better suited for such scenarios. The related problem of data stream compression [12], particularly in sensor networks [30], is worth mentioning. The key distinction from the problem studied here lies in the “real-time” dimension of monitoring. If the sole focus were on the data itself, nodes could compress their observations and transmit the compressed data at large intervals, significantly reducing communication with the coordinator. However, this approach eliminates the coordinator’s ability to track all values in real-time, as it would need to wait for the next batch of compressed data to update its node-specific information. For many applications that require regular monitoring of resources, such a delay is unacceptable.

3 ERROR-BOUNDED FORECAST-BASED MONITORING

In this section, we introduce our monitoring framework, which is based on maintaining aligned time series prediction models at both the nodes and C . By treating the observations at each node as a time series, we can capture the underlying characteristics of the evolving patterns with limited required prior knowledge, thus is general to apply for any data stream. In summary, during each round, each node follows a monitoring strategy by making a monitoring decision regarding the incoming observations. Specifically, the node decides whether to inform C or remain silent by checking if the predicted value exceeds a predefined violation threshold.

3.1 Error-Bounded Value Tracking

It is evident that striving for an error-free solution to the AVT problem is impractical: due to the unpredictable nature of stream changes over time, the only viable approach to *perfect* AVT (i.e., error-free) would be to transmit every value that differs from the last shared value. Instead, we adopt a more efficient strategy by permitting the coordinator to tolerate a small, bounded error for each stream independently, with this error constrained in absolute terms by a predefined absolute error upper bound. That is, for any given times $t \geq 1$, the value \hat{v}_t^i that C has as estimate for the value on s^i is within ϵ of its true observation v_t^i , i.e.,

$$|\hat{v}_t^i - v_t^i| \leq \epsilon, \quad (1)$$

where $\epsilon \in \mathbb{R}^+$ is defined based on the percentage of the range $R_{\mathcal{D}}$ that encompasses a sufficient proportion of percentiles for all values within the relevant data streams over the prior time period (i.e., a

modest one equal to $15\% \cdot R_{\mathcal{D}}$, and a strict one equal to $1\% \cdot R_{\mathcal{D}}$). We refer hereafter to as eq. 1 as the *error constraint* we set on s^i , and it is assumed identical to all nodes in the system.

3.2 Monitoring Strategy

The monitoring decision is guided by the current monitoring strategy, which balances communication efficiency with the need for accuracy. The primary objective is to reduce unnecessary transmissions while ensuring that C maintains an approximate but sufficiently accurate view of the observed data streams. Our system is built on the principle of forecasting the evolution of the monitored values to reduce communication. As long as the constraint in eq. 1 holds at time t , no update message (event) is needed for that round. The observations are stored locally in a buffer, with a maximum buffer length¹ of D which depends on the prediction models used but is constrained to a constant value.

If the observed value deviates significantly from what C currently knows (i.e., the constraint is violated), the node sends an update message. When a violation occurs on node s^i at time t , both the observed value v_t^i and the buffered values are sent to C . Then, C and node s^i utilize previous observations in the buffer to estimate new model parameters for predicting future values \hat{v}_t^i on node s^i . After sending an update message to C , the buffer is cleared to save memory for new observations.

Our proposed framework is a 2-layer hierarchical model where the nodes are all directly connected to the central coordinator C . In real-world systems, it can be easily extended to more hierarchical layers, by e.g., running our monitoring framework both at a cluster level of aggregation between the end-nodes and C and at the central coordinator C for a 3-layer hierarchy.

The communication benefit is measured by the *communication ratio* ρ , which is defined as the proportion of update messages sent to C , regardless of the information contained in those messages (i.e., one message could fit multiple values). For T being the number of timesteps in our studied period (assuming here that all nodes have the same monitoring period), it is calculated as:

$$\rho = \frac{Q}{T \cdot n}, \quad (2)$$

where Q is the number of updates sent by the current monitoring strategy during T . If we assume our system achieves a ρ as eq. 2, when taking into account the cost of protocol headers (i.e., not only the data itself but also the header in the packet), the achieved communication reduction is bounded by ρ together with the header size. In other words, the communication reduction in practice is less compared to ρ .

3.3 Prediction Models

Since the observations generated vary across nodes, we give different options for models employed to describe local observation behaviors at s^i and C . In real-world environments, our framework applies to generic models capable of making predictions, thus other models can be added as well to adapt to different data streams. While the specified models introduced later are effective, alternative models can also be utilized within the framework. We elaborate

¹The buffer size is limited to reduce memory usage and data transmission for better performance, but one could easily store all values that do not trigger an update.

here on how the combination of estimation of model parameters and monitoring strategy works in the following. Each node runs a prediction model independently, and all of the models are the same for each, thus for simplicity, we denote \hat{v}_t^i, v_t^i as \hat{v}_t, v_t here.

3.3.1 SIMPLE APPROXIMATION (SA). It produces a static value based on the last updated observation, assuming that the local observations remain static over time. In other words, the prediction \hat{v}_t does not change over time interval $t - t_{prev}$, where t_{prev} is the last time the node violated the constraint. The prediction model is given by:

$$\hat{v}_t = v_{t_{prev}}, \quad (3)$$

This model is trivial to implement, requiring no additional information to be exchanged between C and s^i . This model, referred to as Approximate Caching [28], is one of the baseline methods presented in [7]. In this study, we also implement it as the simplest model in our framework.

3.3.2 AUTO-REGRESSIVE MODEL (AR). The AR model can be employed in our system, with parameters that can be updated online as new observations become available while being computationally efficient to maintain. The predicted value \hat{v}_t at each node is based on a linear combination of past observations v_{t-j} , which can be expressed as an AR model of lag ℓ , defined as:

$$\hat{v}_t = \delta_t + \sum_{j=1}^{\ell} w_{j,t} \cdot v_{t-j}, \quad (4)$$

where $\mathbf{w} = (w_{1,t}, w_{2,t}, \dots, w_{\ell,t}, \delta_t)$ is the set of model parameters. The estimation of the model parameters \mathbf{w} is calculated using conditional Maximum Likelihood Estimation [29], which requires k samples for computation. As long as the local constraint in eq. 1 holds, the model parameters \mathbf{w} remain unchanged. When s^i breaks the constraint at time t , all the buffered values (of maximum size $D = k$) will be sent to C . Following this, both C and s^i will recalculate \mathbf{w} using the new data for time $t + 1$.

3.3.3 LEAST MEAN SQUARE FILTER (LMS). LMS is one of the most successfully applied adaptive filters [14], known for its low computational overhead and memory usage, making it well-suited for predictive tasks. Essentially, LMS takes each observation at time t as input and computes the prediction as a linear combination of the last ℓ observations, defined as:

$$\hat{v}_t = \sum_{j=1}^{\ell} \theta_{j,t} \cdot v_{t-j}, \quad (5)$$

where $\boldsymbol{\theta} = (\theta_{1,t}, \theta_{2,t}, \dots, \theta_{\ell,t})^T$ is the set of model weights at time t . Initially, $\boldsymbol{\theta}$ is set to zero and is updated iteratively to approach the optimal weights. After collecting ℓ observations, the model can start making predictions using the initialized weights. When a violation occurs at s^i , the error between the predicted value and the true value is computed as:

$$e_t = v_t - \hat{v}_t.$$

The goal is to minimize e_t as the cost function between the prediction and the true value. The weights are then updated in one iteration (which occurs when an update message is sent to C) as follows:

$$\theta_{j,t+1} \leftarrow \theta_{j,t} + \eta \cdot e_t \cdot v_{t-j},$$

where η is the step-size, which must satisfy the condition $0 \leq \eta \leq \frac{1}{\kappa E_t^2}$ to ensure convergence [17]. Here, κ is a constant associated with the bound E_t , which changes after each iteration as:

$$E_t = \frac{1}{\ell} \sum_{j=1}^{\ell} v_{t-j}^2.$$

This bound is used to compute the upper limit for the step-size η . Since our method introduces the error constraint, when $\boldsymbol{\theta}$ has not yet converged to the optimum, update messages are sent, and $\boldsymbol{\theta}$ is updated accordingly. This iterative process continues until the weights converge to the optimal values.

LMS requires fewer observations to compute the weights compared to other methods and is well-suited to track time variations in the statistics of the data streams, provided that these variations occur at a sufficiently slow pace. Predictions begin once ℓ observations are available, and updates to $\boldsymbol{\theta}$ also require only ℓ observations. This is in contrast to the “normal” mode as set in [31], which requires additional iterations for $\boldsymbol{\theta}$ to converge. As a result, the LMS implementation avoids the transmission overhead of these unnecessary updates.

3.3.4 PIECEWISE LINEAR APPROXIMATION (PLA). Instead of using previous values themselves as input for prediction (i.e., using the output as the input for the next time step), we can compute the PLA over the data by representing the observations in the data stream as a function of time, modeled by line segments with a size of ℓ points. The time duration of one line segment spans the range $[t - \ell, t)$, where the points on the segment are represented as $\langle t - \ell, v_{t-\ell} \rangle, \langle t - \ell + 1, v_{t-\ell+1} \rangle, \dots, \langle t - 1, v_{t-1} \rangle$. Let the set of timestamps $\{t - \ell, t - \ell + 1, \dots, t - 1\}$ be denoted as X , and the corresponding set of values $\{v_{t-\ell}, v_{t-\ell+1}, \dots, v_{t-1}\}$ as Y . By extending this line segment, we can estimate future predictions. The prediction at time t is expressed by the linear equation:

$$\hat{v}_t = a \cdot t + b, \quad (6)$$

where a is the slope of the best-fit line given by the covariance between X and Y divided by the variance of X , i.e.,

$$a = \frac{\text{cov}(X, Y)}{\text{var}(X)}.$$

The intercept b is calculated as:

$$b = \mathbb{E}(Y) - a \cdot \mathbb{E}(X),$$

where $\mathbb{E}(X)$ and $\mathbb{E}(Y)$ represent the expected values of X and Y , respectively. The remaining process follows the same steps as other prediction methods: when \hat{v}_t breaks the error constraint, $D = \ell$ observations are sent to C , and the parameters for a and b are recalculated on both sides.

4 DATA-AWARE MODEL SELECTION

In this section, we propose a data-aware model selection framework to adapt the prediction model to the diversity of data stream characteristics. Without prior knowledge of the observations, selecting an optimal model for the nodes can be challenging. Additionally, due to

the skewed distribution of data across distributed nodes, applying a single model type is not suitable for all nodes. Thus, we advocate for a data-aware model selection approach based on model scores, enabling automatic control in determining the appropriate model type and following the data streams continuously generated on the nodes. The selection is done by C using *node-wise* model selection over a list of m candidate models denoted as $\mathbb{M} = \{M_1, \dots, M_m\}$ simultaneously. Hence, s^i runs a personalized model type independently, without requiring alignment with other nodes, while being guided by messages from C regarding model type switching.

4.1 Model Score

$\forall M \in \mathbb{M}$ at time t , the competition of candidate models is calculated using score which is a trade-off between the score of monitoring accuracy acc_W and that of messages omission rate omiss_W :

$$\text{score}_{W,\alpha}(M, t) = \alpha \cdot \text{acc}_W(M, t) + (1 - \alpha) \cdot \text{omiss}_W(M, t),$$

where $0 \leq \alpha \leq 1$ is the control factor that controls the contribution of monitoring accuracy. Specifically, acc_W is calculated over the last W rounds as:

$$\text{acc}_W(M, t) = \frac{1}{W} \sum_{t \in \mathcal{N}} \left(1 - \frac{|\hat{v}_t - v_t|}{\epsilon} \right),$$

with \mathcal{N} being the set of timestamps included over the last W rounds. omiss_W is the fraction of omitted messages among all observations over the last W rounds, i.e.,

$$\text{omiss}_W(M, t) = 1 - \frac{Q_{W,M}}{W}$$

where $Q_{W,M}$ is the number of updates transmitted during the last W rounds when model M is used for prediction. Let us observe both acc_W and omiss_W lie in the interval $[0, 1]$ hence $\text{score}_{W,\alpha}(M, t)$ is bounded to the same interval. If we set $\alpha = 0$, the emphasis is on reducing communication overhead, and $\alpha = 1$ places all the importance on accuracy. Thus, tuning α enables the ability to take both communication and accuracy into account.

4.2 Node-wise Model Selection

C maintains the list \mathbb{M} and simulates them in parallel. Each sensor s^i is initialized with the execution of SA as the selected model M . After gathering at least W observations on s^i , when there is an update sent to C , C calculates score of all candidates in \mathbb{M} . Let $M_t \in \mathbb{M}$ be the model at round t with highest score, i.e., $M_t = \arg \max_{M \in \mathbb{M}} \text{score}_{W,\alpha}(M)$. Then the maximum score is compared with the selected model's score running on s^i . If

$$\text{score}_{W,\alpha}(M_t) > \text{score}_{W,\alpha}(M) + \xi$$

holds, where the parameter ξ is referred to as the model selection *criteria*, then a new model is selected. A model-switching message ("switch", M_t) is then sent from C to s^i . Since s^i retains its previous observations, it can estimate the parameters for the currently selected model. Refer to Algorithm 1 for more details. Such a selection process is done independently for each node and can scale up to multiple nodes. Moreover, we set an initialization phase (line 4) at the beginning of monitoring to collect enough observations, which takes τ time to finish. It produces t' as the timestamp representing the last time an update occurs. This ensures the fairness of

Algorithm 1: Forecast-based Error-Bounded Data-aware Model Selection

```

1 Node  $s^i \in \mathbb{S}$  executes:
   Input: absolute error upper-bound  $\epsilon$ , init. period  $\tau$ 
2    $M \leftarrow \text{SA}$  //  $M$  is an input in static model
3    $t' \leftarrow \text{Initialize}(M, \tau)$ 
4   for  $t \geq \tau$  do
5      $v_t \leftarrow \text{read}(t)$ 
6     if  $\text{Receive}(\langle \text{"switch"}, M_t \rangle)$  then
7        $M \leftarrow M_t$ 
8      $\hat{v}_t \leftarrow M.\text{predict}(t)$ 
9     if  $|\hat{v}_t - v_t| > \epsilon$  then
10       $\text{Send}(C, \langle \text{"update"}, L, t', t \rangle)$ 
11       $M.\text{update}(L)$ 
12       $L = \emptyset, t' \leftarrow t$  // clear buffer
13    else
14       $L \leftarrow L + \{v_t\}$  // store value in buffer

15 Coordinator  $C$  executes:
   Input: absolute error upper-bound  $\epsilon$ , candidate models
    $\mathbb{M}$ , criteria  $\xi$ , control factor  $\alpha$ , window size  $W$ ,
   init. period  $\tau$ 
16 for  $s^i \in \mathbb{S}$  do
17   for  $M \in \mathbb{M}^i$  do
18      $\text{Initialize}(M, \tau)$ 
19    $M^i \leftarrow \mathbb{M}^i[\text{default}]$  // set to default model
20 for  $t \geq \tau$  do
21   for  $s^i \in \mathbb{S}$  in parallel do
22     if  $C$  receives  $\langle \text{"update"}, L, t', t \rangle$  from  $s^i$  then
23       for  $M \in \mathbb{M}^i$  do
24          $M.\text{update}(L)$ 
25          $M_t = \arg \max_{M \in \mathbb{M}^i} \text{score}_{W,\alpha}(M)$ 
26         if  $\text{score}_{W,\alpha}(M_t) > \text{score}_{W,\alpha}(M^i) + \xi$  then
27            $\text{Send}(s^i, \langle \text{"switch"}, M_t \rangle)$ 
28            $M^i \leftarrow M_t$ 
29        $\hat{v}_t^i \leftarrow M^i.\text{predict}(t)$  // estimated value

```

each model's performance, eliminating the cold-start effect on the reported statistics.

4.3 Standard Solution

When distributed nodes receive entirely new data streams, we provide a standard system framework to compensate for the shortcomings arising from the lack of knowledge about the data distribution and characteristics. Since selecting candidate models also requires tuning efforts, it deviates from our objective of achieving automatic control. In other words, candidate models are predefined as a set of model types and parameter combinations, thereby eliminating the need for manual adjustments and adaptations. Thus, for heterogeneous datasets with varying data stream dynamics, the provided standard solution performs well without requiring tuning which is

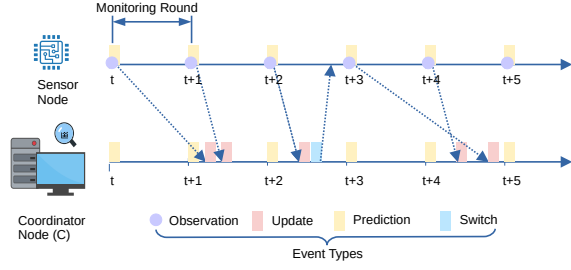


Figure 2: Example of different types of events arriving in different orders with one single node and coordinator C .

essential when data streams are completely new. In the experimental study section, detailed settings and evaluations are provided to demonstrate the resulting benefits and we will further show the strength of the chosen candidate models in various scenarios.

4.4 Reliability and Event Consistency

We discuss herebelow how our framework maintains stability and handles failures in data transmission.

Reliable communication. A reliable and stable connection is assumed between C and each node. We note that perfect synchronization between nodes and C is not required by our system. A reasonable assumption is that, due to the lightweight nature of our framework, the time spent in taking a monitoring decision on a particular node is negligible in comparison to the length of the monitoring round. Also, the typical latency allows in practice several communication round-trips between C and the nodes.

Types of events. New observation events are generated at the beginning of each monitoring round on the nodes. If the error constraint is violated at time t on node s^i , then an update is sent from s^i to C triggering what we refer as an *update event*. The update event corresponds to sending the message $\langle \text{“update”}, L, t', t \rangle$ (line 11), where L is the buffer of s^i , t' is the timestamp of the last update event which occurred on node s^i and t denotes the current timestamp. Then upon reception of an update event, C replaces \hat{v}_t^i for node s^i at time t by the exact current value v_t^i (the last element of L) when there is no failure, determined by verifying whether t corresponds to the current monitoring round. For ease of reading, consistency detection is abstracted away from the pseudo-code.

If a delayed or disordered update occurs, *out-of-order* events arise. As shown in Figure 2, updates may arrive later than scheduled (e.g., update of time t arrives at time $t + 1$ but before $t + 1$'s update, or update of $t + 3$ arrives after $t + 4$'s update), however, C knows whether to wait or process immediately the event by checking t' and t included in the event message; disorders are hence effectively handled. Moreover, due to the difference in scale between the time of reception of update events after the start of a monitoring round (requiring computation of monitoring decisions and sending of event messages) and the length of a monitoring round, such events are considered rare. Thus, they are not within the primary foci of

this study while out-of-order event handling has extensively been addressed in prior work [34]. Consistent with previous studies [8, 23, 24, 33], we assume a single instance of the central coordinator C , without considering its replication. Therefore, failures of C are beyond the scope of this study.

5 EVALUATION

In this section, we evaluate the experimental results on different datasets using our proposed methods under the assumption of reliable communication with no failure occurs. First, we evaluate how different prediction models perform in communication-saving and their accuracy. Then, we choose candidate models of the same model type in one dataset using data-aware model selection. Next, we move on to test the framework of standard model parameter settings in four datasets. Finally, we explore the trade-off between communication overhead and monitoring accuracy.

5.1 Experimental Setup

5.1.1 Datasets. We evaluate our approach on a variety of datasets with different characteristics. All experiments use timestamp sequences starting from 0 and increasing by 1 at every data point. Range $R_{\mathcal{D}}$ is an empirical knowledge we collect from each dataset \mathcal{D} which represents the possible fluctuation of the data streams. We have set the value of $R_{\mathcal{D}}$ in relation to \mathcal{D} so that it covers the 97.2% percentiles of all values within \mathcal{D} , hence discarding some outliers with extremal values. Then ϵ is calculated as a proportion of $R_{\mathcal{D}}$. The properties of the datasets we used are listed in Table 2.

- Ericsson [13] is 4 hours of hardware CPU usage retrieved during 8 runs of a load testing procedure in an Evolved Packet Core testing infrastructure.
- Geolife [37] is GPS trajectories generated from vehicles within the scope of the (Microsoft Research Asia) Geolife project. We calculate the vehicular speed based on the longitude, latitude, and timestamps from the raw data.
- IntelLab [21] is collected from 54 IoT sensors in IntelLab, from which we select 47 sensors for temperature readings.
- ACSF1 is from UCR Time Series Classification Archive [10], which contains power consumption from home appliances across 10 classes. In this study, we select class 3 and aggregate all the time series data (training and test datasets). Each node represents a subset of data from class 3.

For clarity, we denote by $R_{\mathcal{E}}$, $R_{\mathcal{G}}$, $R_{\mathcal{I}}$ and $R_{\mathcal{A}}$ (corresponding respectively to Ericsson, Geolife, IntelLab and ACSF1) when referencing a specific $R_{\mathcal{D}}$ of one of the datasets. The values of each are listed in Table 2.

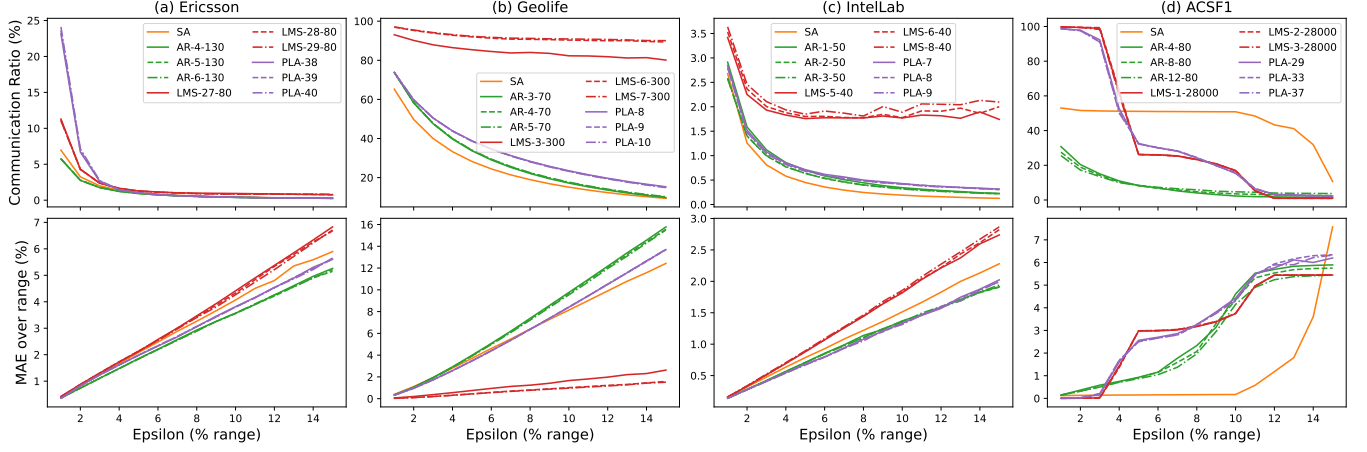
5.1.2 Parameters. We test various configurations and choose top performers based on empirical results; the values of the parameters ℓ, k, κ are listed in the legend of Figure 3. We set $W=100$ for all datasets in our data-aware system. Parameter settings for the standard solution for data-aware experiments are listed in Table 3.

5.1.3 Evaluation metrics. FEDAMON is evaluated for two metrics among all scenarios :

- Communication ratio: ρ is the number of update messages Q sent to C over the total duration of time T of all nodes n as described in § 3.2.

Table 2: Datasets with type of statistics, no. of nodes, total duration T , standard data range $R_{\mathcal{D}}$, $\epsilon = 5\% \cdot R_{\mathcal{D}}$, and model criteria ξ .

Dataset \mathcal{D}	Statistic	# Nodes	T	Standard Data Range $R_{\mathcal{D}}$	$\epsilon (\times 5\% \cdot R_{\mathcal{D}})$	ξ
Ericsson	CPU usage	720	1972	$R_{\mathcal{E}} = 97$	4.85	0.01
Geolife	Vehicle speed	100	54743	$R_{\mathcal{G}} = 120$	6.0	0.029
IntelLab	Sensor Temperature	47	299950	$R_{\mathcal{I}} = 44.6$	2.23	0.015
ACFS1	Power consumption	10	2870	$R_{\mathcal{A}} = 13$	0.65	0.01

**Figure 3: Forecast-based model performance in four datasets. Here, AR-4-130 indicates that $\ell = 4$ and $k = 130$, LMS-3-300 indicates that $l = 3$ and $\kappa = 300$, while PLA-8 indicates that $\ell = 8$.**

- Mean Absolute Error (MAE) over $R_{\mathcal{D}}$: MAE is the difference between the reported value on C and v_t^i , note that the reported value is different from \hat{v}_t^i since when the constraint is broken, \hat{v}_t^i is replaced by v_t^i .

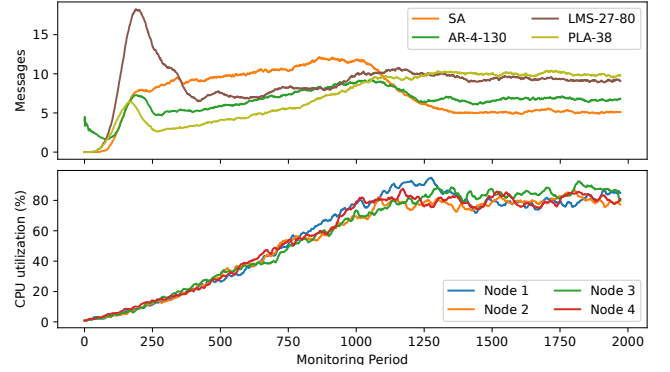
5.2 Single Model Prediction Performance

We start our evaluation by measuring the communication cost saved in each of the 4 datasets. Figure 3 shows ρ achieved by SA, AR, LMS and PLA together with the measurements in terms of MAE over $R_{\mathcal{D}}$ with different parameter settings. We have tested various configurations and choose top performers based on preliminary empirical results. We report ρ for varying ϵ values. More specifically, we show results for $1\% \cdot R_{\mathcal{D}} \leq \epsilon \leq 15\% \cdot R_{\mathcal{D}}$ for all datasets.

Regarding ρ , the performance of our forecast-based method is evident in Ericsson, IntelLab, and ACSF1 across all models. When ϵ exceeds $7\% \cdot R_{\mathcal{D}}$, the best model for each dataset achieves a ρ of less than 20%. However, other models still achieve a communication efficiency below 20% when ϵ is $15\% \cdot R_{\mathcal{D}}$. In Figure 3a and 3d, AR stands out as the most communication-efficient prediction model, and in Figure 3b and 3c, SA outperforms all other models. Considering the

Table 3: Standard model parameters for data-aware system.

Parameter	AR	LMS	PLA
ℓ	3,4,6,8	6,28	8,29,39
k	100	-	-
κ	-	100	-

**Figure 4: Messages sent (upper plot) over time across all nodes of Ericsson dataset ($\epsilon = 5\% \cdot R_{\mathcal{E}}$) using a subset of models from Figure 3 (a) and original data streams from four selected nodes (lower plot).**

significantly smaller resource overhead of the SA algorithm, its performance on those datasets is quite remarkable. We observe that ϵ not only bounds MAE over $R_{\mathcal{D}}$ but can reduce it to half of $R_{\mathcal{D}}$ from Figure 3a, 3c and 3d. Furthermore, increasing ϵ results in greater communication savings; however, the MAE over $R_{\mathcal{D}}$ also increases. This highlights a trade-off between communication efficiency and error, which we will examine in the following discussion.

We analyze the performance of each model over time by examining the number of messages sent across all nodes (excluding

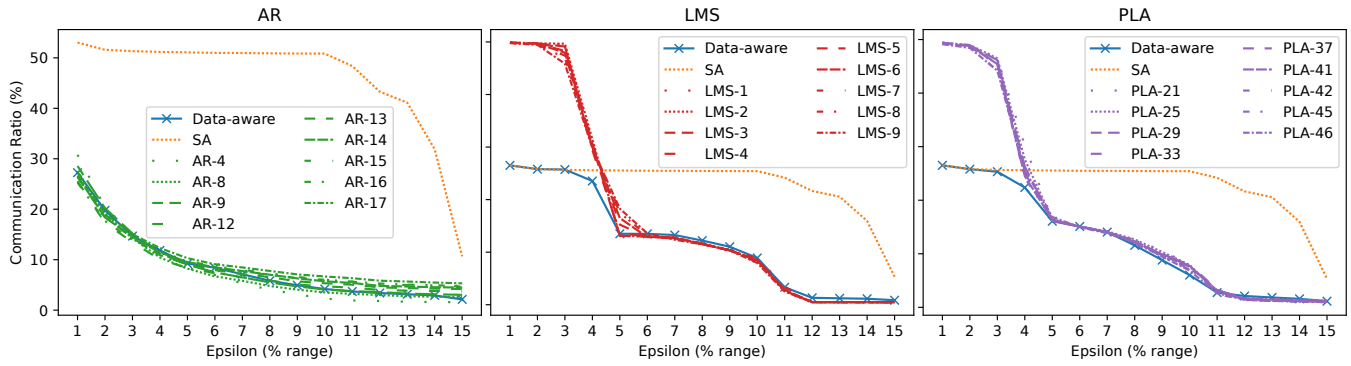


Figure 5: Comparison between the selected model and candidate models (same type) on ACSF1 ($\alpha = 0, k = 80, \kappa = 280000$).

the initialization phase). We select a parameter setting from AR, LMS, and PLA, consistent with Figure 3a together with SA, under Ericsson with $\epsilon = 5\% \cdot R_{\mathcal{E}}$. As we can see from Figure 4, PLA-38 initially outperforms when $t \in [200, 1000]$, but it exhibits the weakest performance toward the end of the period analyzed. Similarly, SA sends the most messages when $t \in [400, 1000]$, but it is the most communication-efficient at the end. The differences among models mainly stem from the temporal evolution of data streams, as shown in Figure 4 using selected original data streams from various nodes in the Ericsson dataset. Therefore, selecting a fixed model does not ensure consistent superiority over time, highlighting the necessity of our data-aware model selection approach.

5.3 Data-aware System Evaluation

5.3.1 *Communication Focus.* We continue to explore the performance of the selected model in ρ when having candidate models with the same model type but different parameter settings. Specifically, we evaluate the performance under ACSF1 with AR-only, LMS-only, and PLA-only with $\alpha = 0$. From Figure 5, the results show the data-aware method can always guarantee ρ close to the best parameter setting. Moreover, from PLA in Figure 5, the selected model outperforms all the others when ϵ is smaller than $10\% \cdot R_{\mathcal{A}}$.

Standard Solution: We run one SA, four ARs, two LMSs, and three PLAs to evaluate the generalization of our data-aware system with standard settings in four datasets with $\alpha = 0, \xi = 0.01$. As we can see from the previous discussion, the parameters are highly dependent on each dataset, we set standard model parameters (listed in Table 3) for all datasets to avoid manual model selection under situations where there is a lack of knowledge of the characteristics of the dataset. The model types and corresponding parameters are chosen from those showing consistently strong performance across all datasets. Specifically, AR generally outperforms the other models, so we select four of its better variants. PLA and LMS show weaker performances overall, and we select three and two configurations for them. As shown in Figure 6, the average performance of our data-aware method surpasses that of all individual models in terms of ρ when ϵ is below $11\% \cdot R_{\mathcal{D}}$. In the best-case scenario, when ϵ is 2%, it achieves a +3.59% absolute increase (+17% relative improvement) compared to AR. While our selection involves empirical

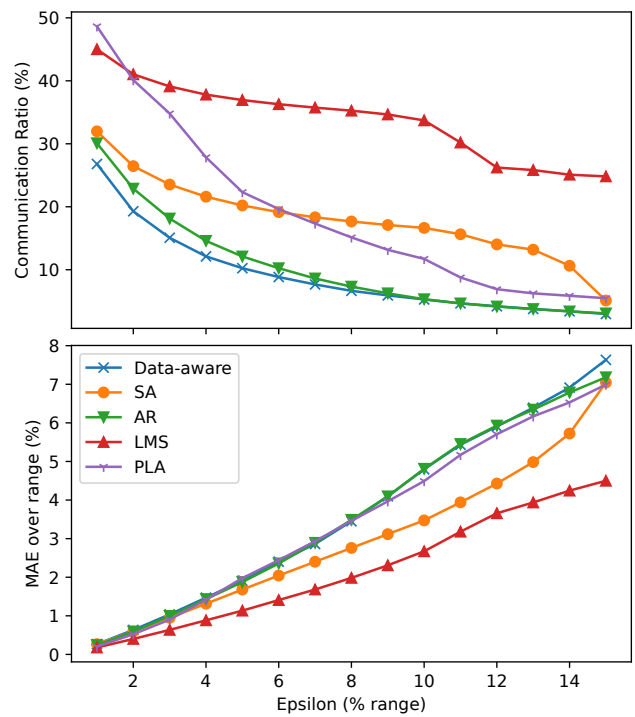


Figure 6: Average performance on selected vs. candidate models with standard parameters over all datasets ($\alpha = 0, \xi = 0.01$).

judgment over quantitative results, it is guided by systematic testing and clearly shows the effectiveness of the data-aware approach consisting in adapting and switching models.

5.3.2 *Accuracy Focus.* When nodes send more messages to C , monitoring accuracy is increased at the same time. We explore the trade-off between communication and accuracy by setting the ϵ as $15\% \cdot R_{\mathcal{D}}$. Because ϵ bounds our system’s error to a smaller range when it is small, and when ϵ is larger, it gives more variety to adjust the communication and accuracy based on model performance. We use the same model parameters as § 5.2 for each dataset and run the data-aware selection algorithm. Figure 7 shows the effectiveness of

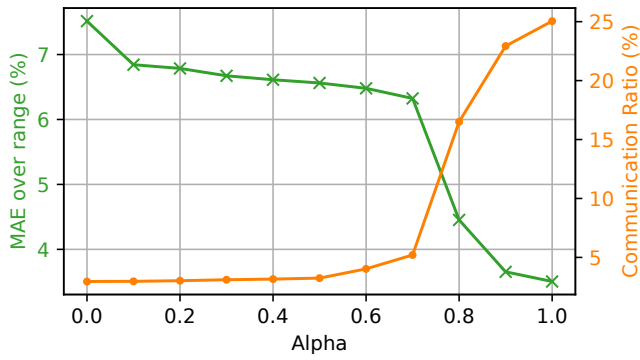


Figure 7: The trade-off of the data-aware system between ρ and MAE over range on four datasets ($\epsilon = 15\% \cdot R_D$).

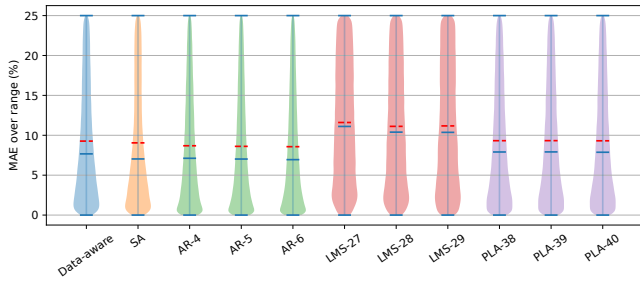


Figure 8: MAE over range on Ericsson, and the red dashed lines represent the mean values ($\epsilon = 25\% \cdot R_E, \alpha = 0.7$).

α controlling the trade-off. More specifically, when α is 0, it gives all the focus on communication, thus ρ is minimal. At the same time, MAE decreases linearly as α grows, i.e., our system places more importance on accuracy rather than communication cost.

Moreover, the reduction of +1% of MAE (13% relative improvement) is achieved by setting $\alpha = 0.5$ without sacrificing communication overhead (compared to $\alpha = 0$). The robustness of our data-aware system in absolute error is shown in Figure 8 with $\epsilon = 25\% \cdot R_E$ and the same settings as § 5.2 under Ericsson when $\alpha = 0.7$. The distribution of error is comparable with SA and both mean and median is under 10% of R_E even $\epsilon = 25\%$, which shows the effectiveness of controlling the error with α .

5.3.3 Occurrence of each Model. We continue our investigation on how each model is chosen over time: model’s occurrence is reported in Figure 9. The evolution is studied on Ericsson with $\epsilon = 25\% \cdot R_E, \alpha = 0.4$, the candidate models are one SA, three ARs with $k = 130$, three LMSs with $\kappa = 80$, and three PLAs. The entire duration is divided into eight time slots, with time increasing along the x-axis from left to right. It shows at slot 0, it begins with SA, and as time passes, other models are gradually selected. At slot 2, AR models become the predominant choice, while at time slots 4 and 5, PLA models dominate the currently selected model. This aligns with the model trends shown in Figure 4, demonstrating that data-aware system can effectively adapt through time-dependent data variation.

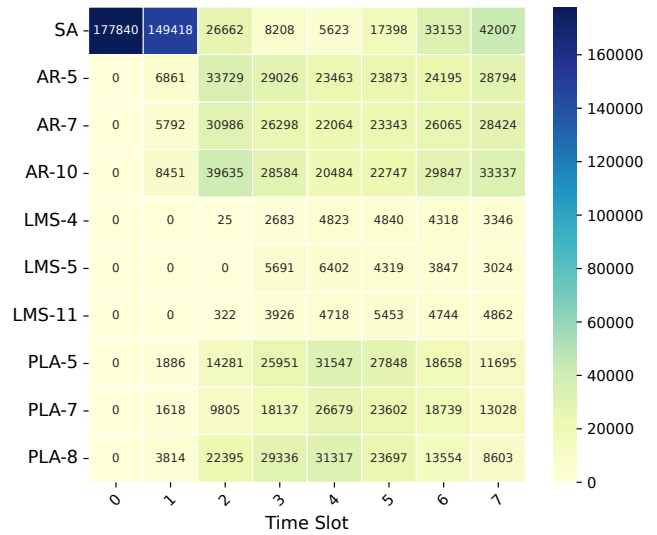


Figure 9: Occurrence of each model over time on Ericsson ($\epsilon = 25\% \cdot R_E, \alpha = 0.4$); total duration is divided into 8 timeslots.

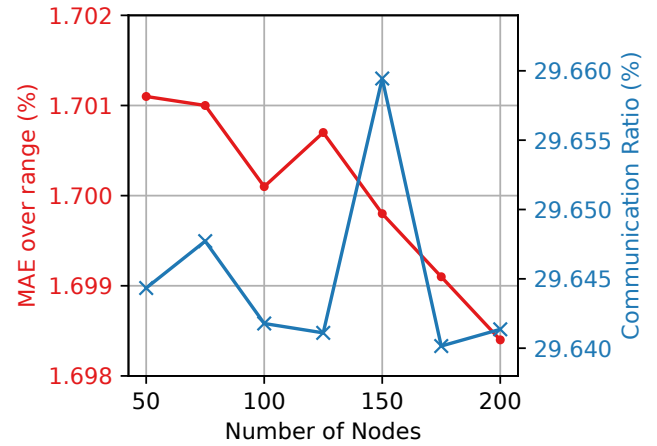


Figure 10: Impact of varying no. of nodes on performance using standard parameters on Geolife ($\epsilon = 5\% \cdot R_E, \alpha = 0$).

5.3.4 Scalability. Our framework is designed to scale effectively with an increasing number of nodes in large distributed systems. As previously demonstrated across four datasets, with the number of nodes ranging from 10 to 720, our framework consistently performs well. Additionally, we examine how varying the number of nodes impacts performance on the same dataset. The MAE and communication are shown in Figure 10 when the number of nodes changes. We simulate different nodes on Geolife with $\epsilon = 5\% \cdot R_E, \alpha = 0$, the candidate models are the same from the standard model parameters as listed in Table 3. The MAE over the range remains stable at approximately 1.7%, while communication overhead is around 29.65%. Therefore, adjusting the number of nodes impact very little the performance and our framework does maintain its effectiveness when scaled up.

6 CONCLUSIONS

In this paper, we propose FEDAMON, a forecast-based error-bounded monitoring framework for continuously monitoring the values generated from distributed nodes on a central node with data-aware model selection for reducing the communication overhead in large distributed systems. We aimed for finding a trade-off between the communication cost and monitoring accuracy. Our experiments evaluation shows that FEDAMON sends only 10% of the updates of the baseline monitoring, while maintaining less than 2% of average error across all monitored streams. Moreover, it is particularly suited for distributed applications where one aims to track all values of nodes without much prior knowledge. In contrast to assigning fine-calibrated models for a specific dataset, FEDAMON performs well when using standard candidate models for different datasets with up to 17% improvement in communication overhead with identical guarantees on maximum error which validates the generalization of our data-aware model selection framework. Moreover, the trade-off between communication overhead and monitoring accuracy is controlled by our data-aware model selection, achieving a 13% improvement in average monitoring error without sacrificing additional communication cost. Our evaluation results are promising and point to the potential of our framework for large-scale and efficient monitoring of distributed systems.

Our directions for extending our work include: (1) a parameter-free solution for a fully automatic control for our standard data-aware framework, and (2) more types of prediction models that are lightweight yet efficient to be utilized in our framework.

REFERENCES

- [1] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. 2016. Next generation 5G wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 18, 3 (2016), 1617–1655.
- [2] Ali M Alakeel et al. 2010. A guide to dynamic load balancing in distributed computer systems. *International Journal of Computer Science and Information Security* 10, 6 (2010), 153–160.
- [3] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. 2009. Energy conservation in wireless sensor networks: A survey. *Ad hoc networks* 7, 3 (2009), 537–568.
- [4] Jahongir Azimjonov and Ahmet Özmen. 2021. A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways. *Advanced Engineering Informatics* 50 (2021), 101393.
- [5] Brian Babcock and Chris Olston. 2003. Distributed top-k monitoring. In *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*. ACM, 28–39.
- [6] Ho-Leung Chan, Tak-Wah Lam, Lap-Kei Lee, and Hing-Fung Ting. 2012. Continuous monitoring of distributed data streams over a time-based sliding window. *Algorithmica* 62, 3-4 (2012), 1088–1111.
- [7] David Chu, Amol Deshpande, Joseph M Hellerstein, and Wei Hong. 2006. Approximate data collection in sensor networks using probabilistic models. In *22nd International Conference on Data Engineering (ICDE)*. IEEE, 48–48.
- [8] Graham Cormode. 2013. The continuous distributed monitoring model. *ACM SIGMOD Record* 42, 1 (2013), 5–14.
- [9] Graham Cormode, Minos Garofalakis, S Muthukrishnan, and Rajeev Rastogi. 2005. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *Proc. of the 2005 ACM SIGMOD International Conference on Management of Data*. ACM, 25–36.
- [10] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. 2018. The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.
- [11] Gabriel Martins Dias, Boris Bellalta, and Simon Oechsner. 2016. A survey about prediction-based data reduction in wireless sensor networks. *ACM Computing Surveys (CSUR)* 49, 3 (2016), 1–35.
- [12] Romaric Duvignau, Vincenzo Gulisano, Marina Papatriantafyllou, and Vladimir Savic. 2019. Streaming piecewise linear approximation for efficient data management in edge computing. In *Proc. of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC)*. 593–596.
- [13] Romaric Duvignau, Marina Papatriantafyllou, Konstantinos Peratinos, Eric Nordström, and Patrik Nyman. 2019. Continuous Distributed Monitoring in the Evolved Packet Core. In *Proc. of the 13th ACM International Conference on Distributed and Event-based Systems (DEBS)*. 187–192.
- [14] Behrouz Farhang-Boroujeny. 2013. *Adaptive filters: theory and applications*. John Wiley & sons.
- [15] Nikos Giatrakos, Antonios Deligiannakis, Minos Garofalakis, Daniel Keren, and Vasilis Samoladas. 2018. Scalable approximate query tracking over highly distributed data streams with tunable accuracy guarantees. *Information Systems* 76 (2018), 59–87.
- [16] Dimitrios Giouroukis, Alexander Dadiani, Jonas Traub, Steffen Zeuch, and Volker Markl. 2020. A survey of adaptive sampling and filtering algorithms for the internet of things. In *Proc. of the 14th ACM International Conference on Distributed and Event-based Systems (DEBS)*. 27–38.
- [17] Simon S Haykin. 2002. *Adaptive filter theory*. Pearson Education India.
- [18] Yichuan Jiang. 2015. A survey of task allocation and load balancing in distributed systems. *IEEE Transactions on Parallel and Distributed Systems* 27, 2 (2015), 585–599.
- [19] Grigorios Kakkavas, Adamantia Stamou, Vasileios Karyotis, and Symeon Papavassiliou. 2021. Network tomography for efficient monitoring in SDN-enabled 5G networks and beyond: Challenges and opportunities. *IEEE Communications Magazine* 59, 3 (2021), 70–76.
- [20] Jan Körner, Samuel Bach, Albin Karlsson, Linus Sundkvist, and Romaric Duvignau. 2024. Efficient Monitoring of CPS and IoT Systems: A Deployment Guide for Empirical Evaluations. In *2024 13th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 1–6.
- [21] Intel Berkeley Research Lab. 2004. Intel Lab Data. <https://db.csail.mit.edu/labdata/labdata.html> Accessed: 2024-11-06.
- [22] Yann-Aël Le Borgne, Silvia Santini, and Gianluca Bontempi. 2007. Adaptive model selection for time series prediction in wireless sensor networks. *Signal Processing* 87, 12 (2007), 3010–3020.
- [23] Alexander Mäcker, Manuel Malatyali, and Friedhelm Meyer auf der Heide. 2015. Online Top-k-position monitoring of distributed data streams. In *2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 357–364.
- [24] Alexander Mäcker, Manuel Malatyali, and Friedhelm Meyer auf der Heide. 2016. On competitive algorithms for approximations of Top-k-position monitoring of distributed streams. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 700–709.
- [25] Samuel Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. 2003. The design of an acquisitional query processor for sensor networks. In *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*. 491–502.
- [26] Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. 2009. Sketch-based modeling: A survey. *Computers & Graphics* 33, 1 (2009), 85–103.
- [27] Chris Olston, Jing Jiang, and Jennifer Widom. 2003. Adaptive filters for continuous queries over distributed data streams. In *Proc. of the 2003 ACM SIGMOD International Conference on Management of data*. 563–574.
- [28] Chris Olston, Boon Thau Loo, and Jennifer Widom. 2001. Adaptive precision setting for cached approximate values. *ACM SIGMOD Record* 30, 2 (2001), 355–366.
- [29] Hossein Pishro-Nik. 2014. *Introduction to probability, statistics, and random processes*. Kappa Research, LLC Blue Bell, PA, USA.
- [30] Mohammad Abdur Razzaque, Chris Bleakley, and Simon Dobson. 2013. Compression in wireless sensor networks: A survey and comparative evaluation. *ACM Transactions on Sensor Networks (TOSN)* 10, 1 (2013), 1–44.
- [31] Silvia Santini and Kay Romer. 2006. An adaptive strategy for quality-based data reduction in wireless sensor networks. In *Proc. of the 3rd international conference on networked sensing systems (INSS 2006)*. TRF Chicago, 29–36.
- [32] Izchak Sharfman, Assaf Schuster, and Daniel Keren. 2007. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems (TODS)* 32, 4 (2007), 23.
- [33] Mingwang Tang, Feifei Li, and Yufei Tao. 2015. Distributed online tracking. In *Proc. of the 2015 ACM SIGMOD International Conference on Management of Data*. 2047–2061.
- [34] Juliane Verwiebe, Philipp M Grulich, Jonas Traub, and Volker Markl. 2023. Survey of window types for aggregation in stream processing systems. *The VLDB Journal* 32, 5 (2023), 985–1011.
- [35] Diana Yacchirema, David Sarabia-Jácome, Carlos E Palau, and Manuel Esteve. 2018. System for monitoring and supporting the treatment of sleep apnea using IoT and big data. *Pervasive and Mobile Computing* 50 (2018), 25–40.
- [36] Ke Yi and Qin Zhang. 2013. Optimal tracking of distributed heavy hitters and quantiles. *Algorithmica* 65, 1 (2013), 206–223.
- [37] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin* 33, 2 (2010), 32–39.